VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOCATION-AWARE DATA TRANSFERS SCHEDULING FOR DISTRIBUTED VIRTUAL WALKTHROUGH APPLICATIONS

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE | ING. JAROSLAV PŘIBYL
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POLOHOVĚ ZÁVISLÉ PLÁNOVÁNÍ PŘENOSU DAT PRO APLIKACE PROCHÁZENÍ DISTRIBUOVANOU VIRTUÁLNÍ SCÉNOU

LOCATION-AWARE DATA TRANSFERS SCHEDULING FOR DISTRIBUTED VIRTUAL WALKTHROUGH APPLICATIONS

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE — ING. JAROSLAV PŘIBYL
AUTHOR

VEDOUCÍ PRÁCE — Prof. Dr. Ing. PAVEL ZEMČÍK
SUPERVISOR

BRNO 2013

Abstrakt

Důležitou součástí aplikací procházení distribuovanou virtuální scénou je proces plánování přenosu dat. Jeho hlavním úkolem je zajištění efektivního přenosu dat a maximální kvality renderovaného obrazu. Největší vliv na kvalitu renderované scény mají omezení síťového připojení. Tyto omezení lze redukovat pomocí multi-resolution reprezentace dat scény, určováním priorit stahování jednotlivých částí scény, a přednačítáním dat. Pokročilé metody pro určování priorit a přednačítání části scény jsou založeny na predikci pohybu uživatele vycházející z matematického popisu jeho pohybu. Tyto metody jsou schopny predikovat následující pozici uživatele jen v krátké vzálenosti od jeho aktuální polohy. V případě náhlých, ale pravidelných změn směru pohybu uživatele jsou tyto metody nedostatečné co do přesnosti i délky predikce. V této práci je navrhnut komplexní přístup k řešení plánování přenosu dat splňující i tyto požadavky. Navrhované řešení využívá predikci pohybu uživatele založenou na znalostech k určení priority stahování dat i předstahování částí scény. Provedené experimenty nad testovacími daty ukazují, že navržené schéma plánování přenosu dat umožňuje dosažení vyšší efektivity přenosu dat a vyšší kvality renderovaného obrazu během průchodu testovací scénou.

Abstract

Data transfers scheduling process is an important part of almost all distributed virtual walkthrough applications. Its main purpose is to preserve data transfer efficiency and rendered image quality. The most limiting factors here are network restrictions. These restrictions can be reduced using multi-resolution data representation, download priority determination and data prefetching algorithms. Advanced priority determination and data prefetching methods use mathematic description of motion to predict next position of a user. These methods can accurately predict only close future positions. In the case of sudden but regular changes in user motion direction (road networks), these algorithms are not sufficient to predict future position with required accuracy and at required distance. In this thesis a systematic solution to data transfers scheduling is proposed which solves also these cases. The proposed solution uses next location prediction methods to compute download priority or additionally prefetch data needed to render a scene in advance. Experiments show that compared to motion functions the proposed scheduling scheme can increase data transfer efficiency and rendered image quality during exploration of tested scene.

Klíčová slova

Průchod distribuovanou virtuální scénou, predikce následující pozice, databáze trajektorií, pohybová funkce, Markovovův řetězec, přednačítání dat.

Keywords

Distributed virtual scene walkthrough, next location prediction, trajectory database, motion function, Markov chain, prefetching.

Citace

Ing. Jaroslav Přibyl: Location-aware Data Transfers Scheduling for Distributed Virtual Walkthrough applications, disertační práce, Brno, FIT VUT v Brně, 2013

Location-aware Data Transfers Scheduling for Distributed Virtual Walkthrough applications

Prohlášení

I declare that this dissertation thesis is my original work and that I have written it under supervision of Prof. Dr. Ing. Pavel Zemčík. All sources and literature that I have used during elaboration of the thesis are correctly cited with complete reference to the corresponding sources.

....
Ing. Jaroslav Přibyl
August 20, 2013

Poděkování

I would like to thank to Prof. Dr. Ing. Pavel Zemcik for his scientific lead and valuable advices, to my wife and my family for her patience and encouragement, and to my past and current colleagues from UPGM for their great support.

Contents

Chapter 1

Introduction

Imagine a mobile application which renders a 3D scene such as foreign city or large landscape. The data for rendering the scene is downloaded from network via mobile data connection. A user of such application is exploring these places, moving towards the points of interest or is moving all around. In this thesis, a new method is proposed for increasing rendered image quality and data transfers efficiency for the users of such applications.

The initial purpose of distributed virtual walkthrough applications was to realize a virtual tourism task which allows users to visit places of interests without physically entering them (virtual city walkthrough, virtual museum etc). The main requirements are realistic rendering and informational richness of an explored scene. These requirements increase the amount of transmitted and processed data.

Distributed virtual walkthrough applications, compared to general distributed virtual environments, assume continuous user motion. Today's most popular and widely used distributed virtual walkthrough applications are e.g. Google Street View (see Figure 1.1) or intelligent navigations such as AUDI A4, BMW, Waze or Sony. Other applications are e.g. social virtual networks such as Second Life or Blue Mars.

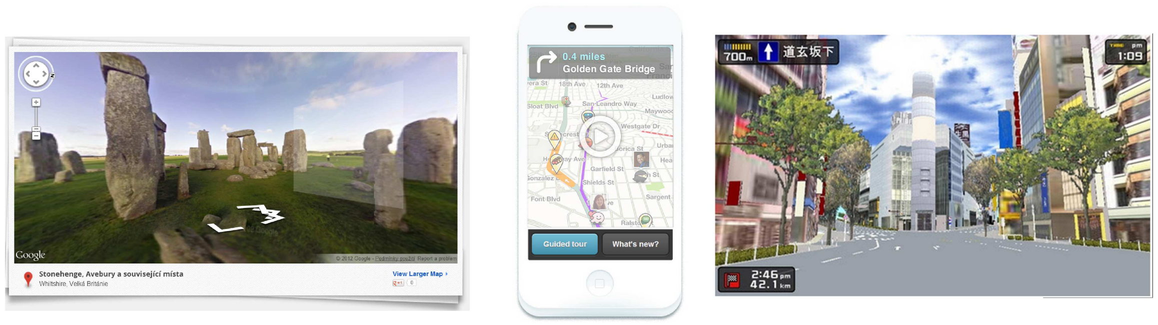


Figure 1.1: Google Street view[1] (left), Waze[2] (center) and Sony 3D car navigation system[3] (right) are examples of modern distributed virtual walkthrough applications.

Advances in computing and graphic performance of current mobile and embedded devices, the explosive growth of market with these devices and various digital media data archives created commercially or community contributed, can further increase the poten-

[1]http://google.com/streetview
[2]http://www.waze.com/
[3]http://www.itmedia.co.jp/lifestyle/articles/0404/06/news073.html

tial and usage of distributed virtual walkthrough applications.

Compared to desktop computers, the content provided by these devices can be associated with richer context, such as location, weather, traffic, etc. The most important hardware feature of these devices is GPS module for determining user location. Knowledge of user location is the key feature of distributed virtual walkthrough applications designed to run on mobile devices.

Typical distributed virtual walkthrough applications exploiting the knowledge of user location are e.g. augmented reality devices such as LifeClipper (historic sites guide), mobile augmented reality application Nokia City Lens, or project AIDA (see Figure 1.2).

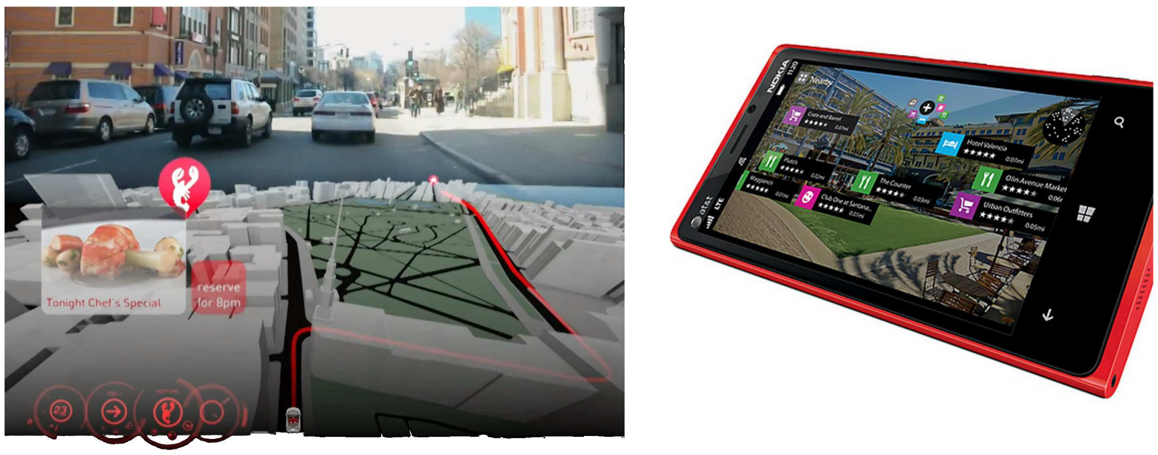


Figure 1.2: AIDA - predictive 3D visualization and navigation system with augmented reality[4] (left), and Nokia City Lens[5] - augmented reality system visualizing close restaurants, point of interests and other points of interest (right).

AIDA is a navigation system which becomes familiar with both the driver and the city. Instead of focusing solely on determining routes to a specified target, the system analyses driver behaviour to identify a set of goals the driver would like to achieve. These are e.g. information about business and shopping districts, tourist and residential areas, as well as real-time event information related to traffic information such as jams or accidents, etc.

The main bottleneck of distributed virtual walkthrough applications is network connection. It suffers from restrictions such as low bandwidth or higher latency especially on wireless networks, so the requested data cannot be transferred in time. Data transfers scheduling algorithms can reduce the impact of these restrictions with the help of multi-resolution data representation, priority determination and data prefetching algorithms so they can increase both quality of rendered scene and data transfers efficiency. Rendered image quality is defined as the ratio between the amount of available data and the amount of data which is needed to render the scene during its exploration. Data transfer efficiency is defined as the ratio between the amount of data downloaded and how this data increases the rendered image quality.

Current scheduling algorithms for distributed virtual walkthrough applications widely use mathematic description of motion (motion functions) to predict next motion of each individual user. The predicted position can be used to compute download priority or to

[4]http://senseable.mit.edu/aida/
[5]http://betalabs.nokia.com/trials/nokia-city-lens-for-windows-phone

prefetch scene parts in advance. Unfortunately, these methods predict accurately only near future positions, and their accuracy decreases also in the case of sudden changes in user motion direction. For networks with higher latency, low bandwidth or just for fast moving user a prediction method with higher accuracy and capable to predict farther positions is needed to keep both data transfers efficiency and rendered image quality as high as possible. From another point of view, for some applications the rendered image quality can be a much more important parameter than the data transfer efficiency. In this case, scheduling algorithms can download a large amount of data in advance but probably only a small subset of them will be used for rendering especially in the case of false prediction.

This thesis proposes a systematic solution for scheduling of data transfers for distributed virtual walkthrough applications. It uses next location prediction methods to increase both data transfers efficiency and rendered image quality. The proposed solution is based on two key insights. First, next location prediction methods have much higher prediction accuracy compared to motion functions. Second, next location prediction methods can be adaptively constructed according to the multi-resolution data representation. This feature allows the scheduling algorithm to prefetch missing data (scene parts) at specified resolution as is required by a rendering algorithm.

This thesis is organized as follows. First, a brief review of current state of distributed virtual walkthrough applications and related background with formulation of thesis objectives is given in Chap. 2. Then a state-of-the-art of streaming of content of 3D scenes over network is given in Chap. 3, and a survey of scheduling and prefetching mechanisms for distributed virtual walkthrough applications is summarized in Chap. 4. Further, a streaming and rendering framework for which a new location-aware data transfers scheduling algorithm is proposed is given in Chap. 5. The location-aware prediction scheme which is used by the proposed scheduling scheme is described in Chap. 6, and the scheduling scheme itself it defined in Chap. 7. Finally, results of experimental evaluation of the proposed location-aware data transfers scheduling scheme are shown in Chap. 8 followed by conclusions in Chap. 9.

Chapter 2

Motivation

The data transfers scheduling process is an important part of almost all distributed virtual walkthrough applications. Its main purpose is to preserve maximum data transfers efficiency and rendered image quality during scene exploration.

In distributed virtual walkthrough applications, the rendered image quality is defined as a ratio between the amount of available data and the amount of data which is needed during walkthrough to render the explored scene. Data required for rendering the scene during a walkthrough is precisely determined by the rendering algorithm used.

The ideal scheduling algorithm is defined so that all data necessary to render current view are downloaded into local storage right before they will be needed by the rendering algorithm to render the current view. Unfortunately, the network restrictions can significantly delay transfers of the requested data. Higher details then will not be available in time for rendering. In such a case, the rendering algorithm renders the scene only at coarser resolution (see Figure 2.1).

In high bandwidth networks, scene parts in close neighbourhood of a user can be downloaded in advance so they will be available in local cache for rendering future views. All the data which should be downloaded in advance is determined by data prefetching algorithms.

Data prefetching algorithms cannot exactly identify the scene parts needed for rendering future views especially in the case that the prefetching is based on motion prediction. Consequently, data downloaded in advance will not be necessarily used for rendering. This behaviour decreases overall data transfers efficiency in case the motion prediction was wrong.

Data transfer efficiency is defined as ratio between the amount of downloaded data and how this data contributes to rendered image quality during scene walkthrough. Consequently, data transfers scheduling algorithm is closely related to the rendering algorithm used, its characteristics and scene data representation. This relation help the scheduler to determine scene parts needed for rendering the scene at arbitrary user positions.

2.1 Multi-resolution approach

Level-of-detail algorithms is a well known and widely used approach to reduce hardware requirements for rendering of large scenes. The basic idea is to reduce the amount of triangles or texture data rendered at each frame without visible degradation of the rendered image quality.

Suitable level-of-detail algorithm and data reduction metric depend on the requirements

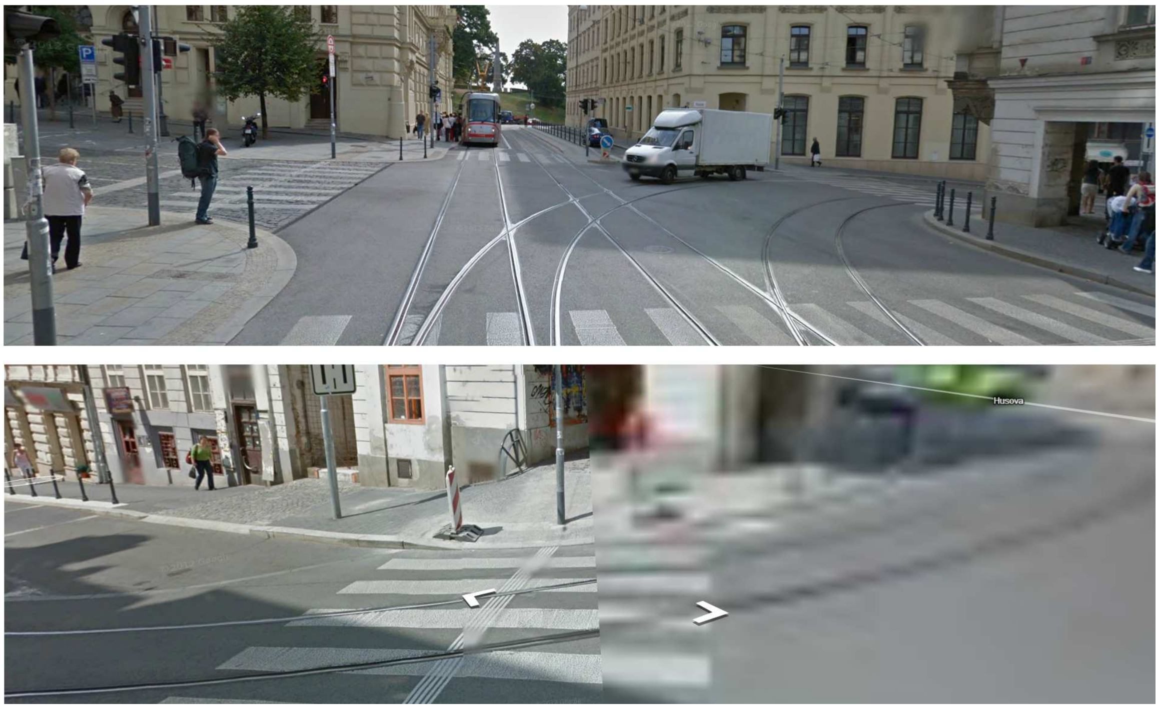


Figure 2.1: Fully loaded view in Google Street View (top) and decreased detail when moving through the crossroad to the right-hand side(bottom).

of concrete application and on data representation of the scene (e.g. heightmaps for terrains, triangle meshes with a large amount of triangles, etc.). The level-of-detail algorithms operates with the scene data at various resolutions (various level-of-detail). During rendering the most appropriate level-of-detail is selected or reconstructed to render the current frame. Such data representation is called multi-resolution data representation (see Figure 2.2).

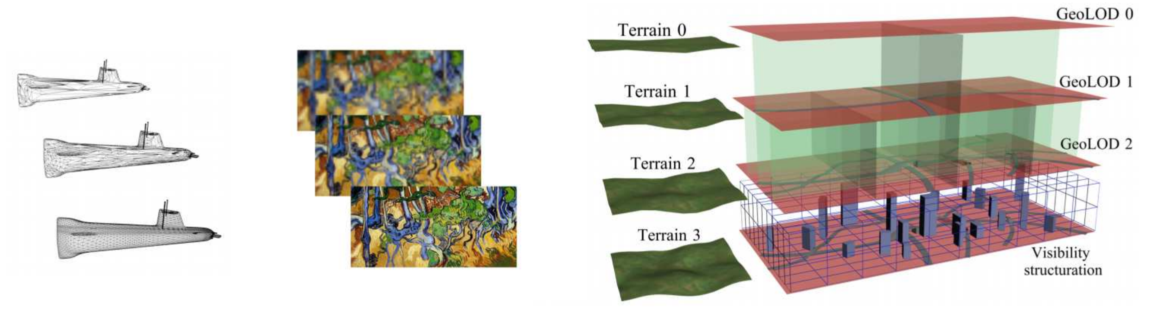


Figure 2.2: Multi-resolution geometry (left), multi-resolution textures (center) and more complex representation of multi-resolution data (right)^6.

The multi-resolution data representation is the fundamental requirement for effective data streaming over network. The multi-resolution form of data enables better scheduling and higher control of the transfer. For example, it is not physically possible to perceive all the details of an explored scene for a fast moving user. Therefore, the amount of transferred data needed for rendering the scene at the required quality (resolution) should be much smaller compared to a static or a slow moving user.

2.2 Communication and data transfers

Communication and data transfers in distributed virtual walkthrough applications are widely based on classic client-server model. In this model, each client manages its local cache and renders the scene. The server holds the whole database with multi-resolution scene representation. The advantages of the client-server model are easy data consistency preservation, higher safety and better control over the scene data (e.g. for commercial reasons). The disadvantage of the client-server model is higher amount of I/O requests at the server if large number of users are simultaneously connected to the server. The large number of connected users can increase the overall system response time.

The main bottleneck of distributed virtual walkthrough applications is network connection. It is affected by restrictions such as high latency and low bandwidth. Distributed virtual walkthrough applications aim to minimize impact of these restriction to keep the rendered image quality and data transfers efficiency high.

To reduce a low network bandwidth the multi-resolution data representation is needed. First much coarser resolution of the scene is downloaded, followed by finer resolutions successively as they are required by the rendering algorithm. In the case of low network bandwidth or a fast moving user, the client can receive and render only coarser resolution of the scene. Finer resolutions of the scene data will not be transmitted except when the user slows down or stops its motion.

The negative impact of low bandwidth and high latency should be further reduced with the help of priority determination and data prefetching methods.

Download priority determination algorithms can compute download priority of particular scene parts based e.g. on user motion, area of interest determination or other visibility determination algorithms. The goal of the priority determination algorithms is to ensure the downloaded data will be available as long as possible during the time they are needed for rendering. It means that the scene parts assumed to be used for longer time during rendering will have higher priority. This behaviour can increase both data transfers efficiency and the rendered image quality.

Data prefetching algorithms on the other hand can identify scene parts which will be needed for rendering in near future. The ideal data prefetching algorithm will download scene parts in advance so that these scene parts will be available as they become requested by the rendering algorithm. The ideal prefetching algorithm should also ensure all the downloaded data will be used for rendering thus leading to maximal data transfers efficiency and rendered image quality.

2.3 Prediction

Current scheduling algorithms for distributed virtual walkthrough applications widely use mathematic description of motion (motion functions) to predict next position of each individual user. The predicted position can be used to compute download priority or to prefetch scene parts in advance.

Motion functions realize the short time prediction. The mathematic description of motion can be exploited as a fast and simple prediction with minimal memory and storage requirements. Unfortunately, methods based on motion functions are accurate only when

[6] Jean-Eudes Marvie, Pascal Gautron, Pascal Lecocq, Olivier Mocquard, and François Gérard. Streaming and synchronization of multi-user worlds through http/1.1. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 111–120, New York, NY, USA, 2011. ACM.

predicting near future positions. Their accuracy also decreases in the case of sudden but regular changes in user motion direction (see Figure 2.3).

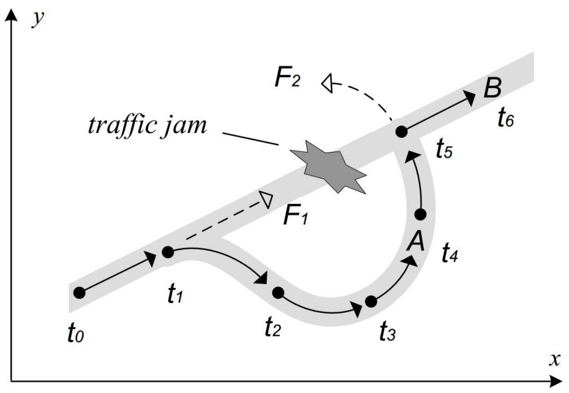


Figure 2.3: If some object frequently visits B at time t_6 after A at time t_4 and it is currently passing A at t_4, it can be said that the object is likely to be on B at t_6 instead of F_2[7].

In networks with higher latency, low bandwidth or just for a fast moving user a prediction method with higher accuracy and capable to predict farther positions is needed to keep both data transfers efficiency and rendered image quality as high as possible.

Prediction methods satisfying the accuracy and prediction length demands are represented by long term prediction methods so called next location prediction methods. These methods uses knowledge of past behaviour of a user (or users) to predict future locations. User behaviour here is described e.g. as a sequence of visited places (e.g. home → work → shop → home) or by sequences of visited cells within wireless mobile networks.

Next location prediction methods are more suitable to be used in multi-resolution environments as they can accurately predict users locations farther away from their current positions (see Figure 2.4). The distant locations are related to coarser levels-of-detail.

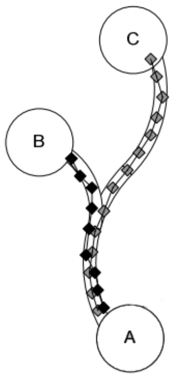


Figure 2.4: Long term prediction methods describe movement of a user as a sequence of visited locations (A, B, C). Motion functions predict next position of a user from its current motion parameters such as speed or motion direction.

[7]Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 70–79, Washington, DC, USA, 2008. IEEE Computer Society.

Usually, the term „motion prediction“ used in literature and through this thesis refers to prediction based on mathematical description of motion. The term „movement prediction“ on the other hand is widely used in literature to describe a prediction with higher level of abstraction such as places of interests (e.g. home, work, shopping mall, etc.). Therefore, motion parameters of a user (e.g. speed, angular velocity or motion direction) are not the most important parameters for the movement prediction methods.

2.4 Objectives

The main objective of this thesis is based on two key insights. First, next location prediction methods have much higher prediction accuracy compared to motion functions. Second, next location prediction methods can be adaptively constructed according to the multi-resolution data representation. This feature allows the scheduling algorithm to prefetch missing scene parts at specified resolution as will be required by the rendering algorithm.

The main goal of this thesis is to design a systematic solution for scheduling of data transfers for distributed virtual walkthrough applications which is based on location-aware prediction scheme. The proposed solution keeps both data transfers efficiency and rendered image quality as high as possible.

A combination of a Markov chain based predictor and a k-state predictor are originally adopted as the next location prediction methods for distributed virtual walkthrough applications. Both the Markov chain based predictor and k-state predictor predicts next user location based on user's current movement pattern. The Markov chain based predictor runs at server. A local movement pattern is used to identify all the users with the same movement pattern at the server. All the matching patterns over all users are used to construct the Markov chain based predictor. The k-state predictor models behaviour of each individual user. A local movement pattern is used to found all the matching movement patterns stored locally by each individual user.

Unfortunately, the next location prediction methods have low prediction rate (in other words prediction quantity) compared to motion functions. It means that it is not always guaranteed that a prediction query returns a prediction result. This can happen if the user behaviour is unknown. The unknown behaviour can happen in the case of insufficient information about past user movements (typically a new user) or in the case of change in user behaviour. Prediction result can be also marked as not confident if, for example two opposite movement directions have similar probability. If a probability distribution of following particular locations has too high variance then it is not reasonable for data transfers scheduling algorithm to prefetch any data in advance. Therefore such not confident prediction result represented by the most probable location is used only for computing priorities of missing data for rendering the current view. If the prediction scheme did not return any prediction result, this thesis proposes to extend the next location prediction methods with a recursive motion function to overcome the low prediction rate problem.

Data transfers in distributed virtual walkthrough applications are supported with multi-resolution data representation. In this thesis is also proposed a variable sampling of user's movement (represented by GPS trajectory) which creates movement patterns with different granularity. The variable sampling method can increase scalability of the proposed scheduling algorithm so it can schedule to download both distant coarse data representation as well as close detailed ones as is required by the rendering algorithm used.

It should be noted that it is not the main goal of this thesis to find the minimal form of data representation suitable for streaming over network. Instead, it is experimentally proved that it is possible to exploit the next location prediction methods to increase data transfers efficiency and the rendered image quality compared to motion functions.

Consequently, the main contribution of this thesis is the proposed complex prediction scheme evaluated in pre-defined multi-resolution virtual environment. The proposed prediction scheme can be characterized as follows:

- exploits next location prediction methods
- is adaptive according to multi-resolution data representation
- solves the case when no trajectory pattern is found
- evaluates prediction queries within the client-server environment

The scheduling algorithm based on the top of this prediction scheme is designed so that the prediction results is used to keep both data transfers efficiency and rendered image quality high. This behaviour is application dependent and can be changed e.g. so that the data transfers efficiency will be more important parameter because of pre-paid data limits on mobile data connections.

Chapter 3

Streaming graphics over network

This chapter describes state-of-the-art of methods for streaming content of 3D graphic scenes over network. Two basic approaches exist to distribute content of 3D virtual scenes in client-server applications [128]. The first approach is complete scene replication whereas the second approach is progressive data transmission.

The complete scene replication transfers content of the whole scene to clients so the scene is completely available at local storage before an application starts [18], [16], [103]. This is advantageous because it keeps transmission process simple.

For some applications, the size of data representation of a scene can be too large to be downloaded completely in time which is acceptable by users. Therefore, complete scene replication is widely used in applications which render only small standalone 3D models (see Figure 3.1) e.g. for web based graphics [147].

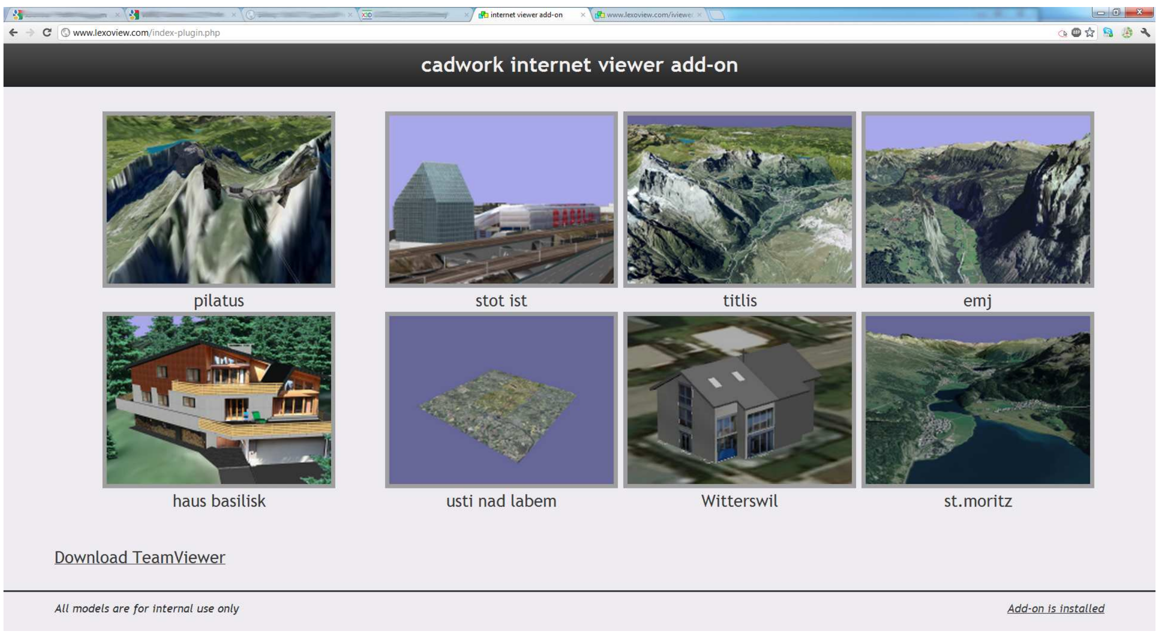


Figure 3.1: Cadwork viewer plugin for downloading and rendering a variety of 3D models[6].

The progressive data transmission [39],[123],[26],[85],[118],[19] assumes that a user can see only a small part of the whole scene. As the user moves within the scene, the missing

[6]http://www.lexoview.de

parts of the scene will be downloaded.

This approach has the advantage that for rendering the scene it is sufficient to download only a small amount of data compared to the complete replication approach. It can speed-up start of applications and eliminate transfers of data which is not necessary for rendering (typically invisible scene parts).

To determine visible parts of the scene, visibility determination algorithms can be used. These algorithms are also useful when optimizing rendering of large scenes. The following chapter addresses usability of such algorithms within distributed virtual walkthrough applications.

3.1 Visibility determination

Visibility determination has been a fundamental problem in computer graphics since the very beginning of the field [6]. Three kinds approaches to culling invisible objects exist (see Figure 3.2). Back-face culling algorithms avoid rendering geometry that faces away from a viewer, view-frustum culling algorithms avoid rendering geometry that is completely outside a view-frustum and occlusion culling avoid rendering primitives that are occluded by other primitives.

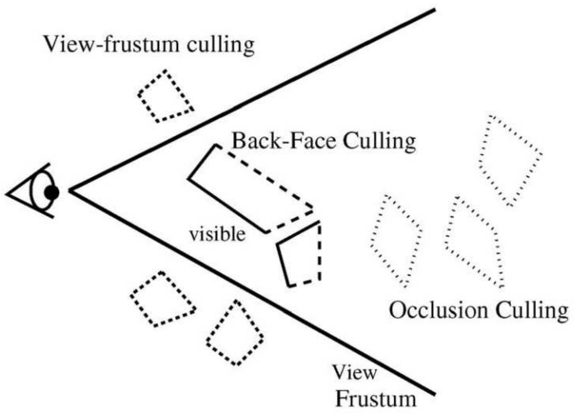


Figure 3.2: Three types of visibility culling techniques[7].

3.1.1 View frustum culling

View-frustum culling is the most common method used to determine visibility of particular objects in a scene. These objects can be tested one by one against a view-frustum and are marked as visible or invisible accordingly[122], [139], [110], [85], [60].

View-frustum culling can be used also in terrain streaming and rendering algorithms[116] [112], [35], [74], [152], [10] to determine visibility of particular terrain blocks. This approach is a compromise between testing visibility of all terrain triangles and between rendering and transferring of whole terrain block (or in general an object) even though only a small part of the block is visible.

[7]D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431, July 2003.

3.1.2 Occlusion culling

Occlusion culling methods are hardware accelerated methods invented to determine scene parts occluded by other primitives named occluders. Such occluded geometry is not visible event though it is inside a viewer's view-frustum. Evaluation of occluded geometry against occluders is computationally time-consuming task. Another issue is that this visibility determination approach is usable only for specific types of scenes such that they contain a lot of occluded geometry such as urban or indoor scenes [20], [114]. The main contribution of these methods is that they can reduce the geometry needed for rendering [32].

There is no benefit from application of occlusion culling methods in terrain rendering applications [55] or with fly-over applications where cost for evaluation of occluded parts can be larger than rendering of the scene including also the potentially occluded parts [88].

In centralized network applications, only server can determine visibility of scene parts, because it holds the whole scene representation. If a lot of clients are connected to a remote server, then the server computation power will become a bottleneck, because it must perform visibility computation simultaneously for all connected clients. The solution to this problem is a division of whole scene into virtual cells for which the visibility will be preprocessed using cell-to-cell or cell-to-object visibility relationship [85], [150].

Optimized occlusion culling for urban scenes is addressed in [150]. A scene is divided into cells so that cell-to-object visibility can be precomputed. The main contribution is the occluder fusion which shrinks sets of small occluders to a bigger entities (see Figure 3.3). Evaluation of occluded geometry by larger occluders is faster.

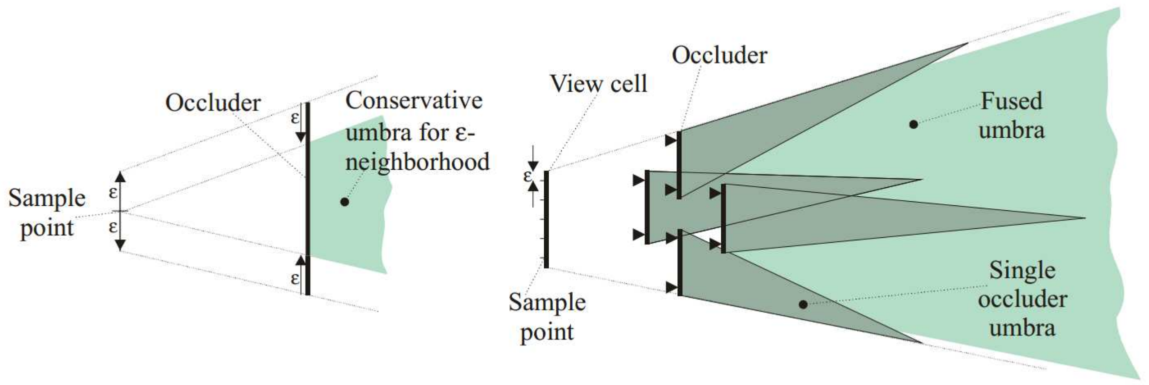


Figure 3.3: (left) an occluder after shrinking it by ε, the umbra from a point sample provides a good conservative approximation of umbra of the original occluder in a neighbourhood of radius ε of the sample point, (right) fused umbra from 5 point samples is constructed from intersection of individual umbrae[8].

In [85] a hybrid method is proposed for visibility determination. This method combines advantages of cell-to-cell visibility with cell-to-object visibility to reduce the number of objects referenced from each cell. It increases efficiency of visibility information streaming between the objects inside each cell and enables prefetching mechanisms to be utilized (see the subsection 4.1.2 for more details). Such a hybrid visibility determination is further used within networked virtual environments described in[93],[15] and[86].

[8]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, UK, 2000. Springer-Verlag.

From the streaming point of view, visibility relationships between cells should be downloaded first and right before any download request take its place.

3.1.3 Area of interest determination

A different approach to visibility determination is area of interest (AOI) method [39]. Area of interest is defined as a circle area around a viewer. Size of such area is important for both streaming and rendering algorithms.

In [52] only the geometry of those objects are transferred which are inside a viewer area of interest.

Distributed virtual walkthrough system CyberWalk [94], [26] generalizes the area-of-interest concept proposed in[39] to both viewers and objects (see Figure 3.4). This generalization aims to minimize the amount of data needed to be downloaded from server to clients.

A viewer scope (see Figure 3.4) determines, how far a viewer can see whereas an object scope indicates how far the object can be seen. The larger the scope of the object is, the more important the object is and the sooner it can be seen (i.e. downloaded to client cache). The object can only be seen by the viewer only when their scopes overlaps. Hence, not visible objects to the viewer do not need to be transferred to the client.

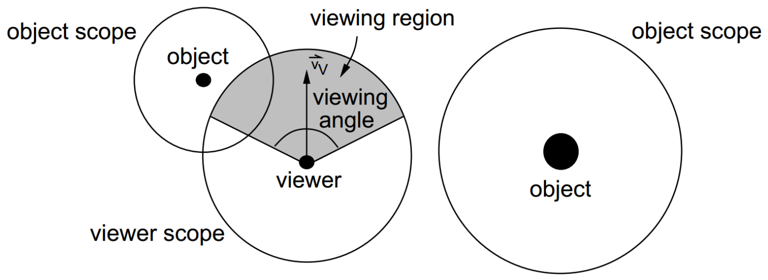


Figure 3.4: Object scope and viewer scope. The larger the object, the larger its scope[9].

If the number of objects (overall amount of geometry) covered by the viewer scope becomes too large to be downloaded at requested resolution in time, a method which divides the viewer scope into several parts with different download priority is proposed in [146](see Figure 3.5). Priority of each part is determined from distance between given part and the viewer's position, and on its presence in view-frustum of the viewer. Priority determination of particular segments must satisfy condition such that $Q_{1a} > Q_{1b} > Q_{2a} > Q_{2b} > ... > Q_{na} > Q_{nb}$.

Visibility determination algorithms can significantly reduce the amount of streamed and rendered data, but there are a variety of applications where the visibility determination will not be sufficient. Therefore, visibility determination algorithms should be used in a combination with other methods for further reducing the amount of streamed and rendered data. These methods are multi-resolution geometry representation and level-of-detail (LOD) algorithms [79].

[9]Beatrice Ng, Antonio Si, Rynson W.H. Lau, and Frederick W.B. Li. A multi-server architecture for distributed virtual walkthrough. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '02, pages 163–170, New York, NY, USA, 2002. ACM.

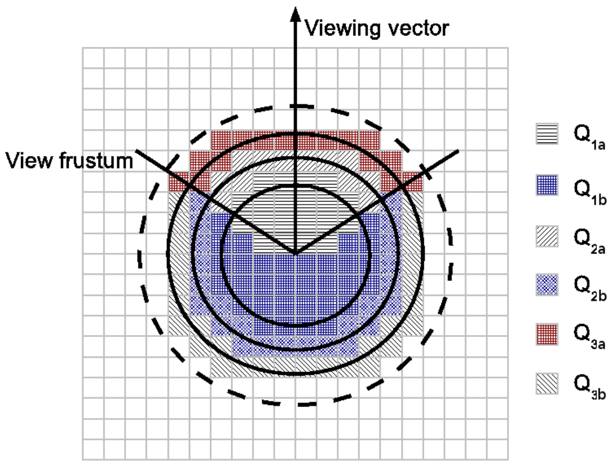


Figure 3.5: The SMLAOI concept divides AOI to parts with different download priority[10].

3.2 Streaming of 3D geometry

Level-of-detail algorithms are classic technique exploiting multi-resolution data representation to bridge a scene complexity and hardware capability by regulation of the amount of detail used to render a scene [30], [79].

Traditional representation of geometry of 3D objects is based on triangular meshes [79]. Three basic level-of-detail approaches can be used to transmit this geometry representation over network: discrete level-of-detail, continuous level-of-detail and continuous view-dependent level-of-detail. The main difference between them is how particular levels-of-detail are created and how they are used during rendering.

3.2.1 Discrete LOD

Discrete LOD principle consists of creating several instances of a given object with decreasing geometric detail (see Figure 3.6). Creation of these particular instances is done off-line in a preprocessing step and is controlled by selected error metrics. Generation of discrete LOD representation is independent from viewer line of sight, because it is not known in the preprocessing step [79]. Appropriate level-of-detail for each object is selected during rendering so it corresponds to the error metrics used (see Figure 3.7).

One of the first papers dealing with utilization of discrete LOD for optimizing 3D scene transfers in client-server environment was [123]. Its basic idea is that elementary data transmission units are represented by particular levels-of-detail instead of single object with full detail.

The scene is organized into a tree structure, whilst only its lists contain geometry of objects in the form of discreet LOD (see Figure 3.8). Only missing LODs are downloaded as they are requested by a rendering algorithm. As position and viewing direction of a user have changed, selection of LODs changes and all the missing LODs are scheduled to be downloaded immediately.

[10]Wei Wang and Jinyuan Jia. An incremental smlaoi algorithm for progressive downloading large scale webvr scenes. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 55–60, New York, NY, USA, 2009. ACM.

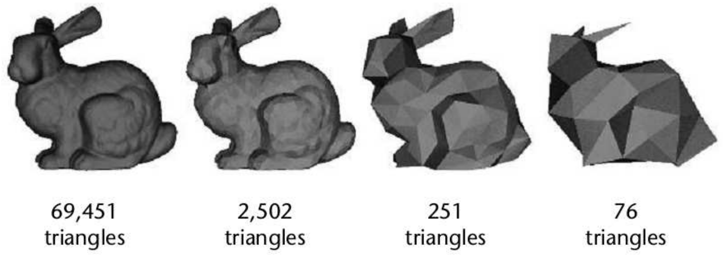


Figure 3.6: Simplification of complex object[11].

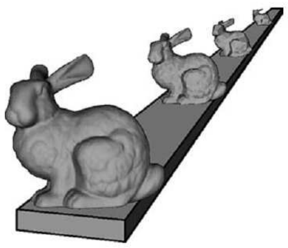


Figure 3.7: Level-of-detail used to reduce rendering cost of distant, or unimportant geometry[11].

Toward effective culling of scene parts that are not visible from current viewpoint is proposed a hierarchical scene organization [30] (see Figure 3.9). A hierarchical scene organization becomes a fundamental scene representation for all modern rendering libraries and tool-kits for rendering 3D graphics, for instance Open Scene Graph [117] or Inventor [143].

Processes in distributed virtual environments are described using standardized languages such as Vrml97 [17] and later designed X3D language [144], [14]. They standardize scene hierarchy, objects description, their behaviour, interaction and rendering. Discrete LOD is one of the standardized entities with precisely defined behaviour.

A disadvantage of the discrete LOD method is the redundancy during transferring of particular levels-of-detail of each object. It is because each higher level-of-detail contains all previous coarser representation extended with new detailed geometry information. Despite that redundancy, discrete LOD algorithms are widely used because they can be simply embedded into any streaming and rendering system.

3.2.2 Continuous LOD

Continuous LOD algorithms encode each object with continuous spectrum of detail during preprocessing whereas discrete LOD algorithms use only several resolutions. An advantage of continuous LOD representation is that it is possible to extract particular resolution from the spectrum during rendering.

Objects in the virtual scene can be represented using arbitrary number of polygons depending on selected error metrics. Continuous change in detail of each object enables a

[11]David P. Luebke and Benjamin Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, UK, 2001. Springer-Verlag.

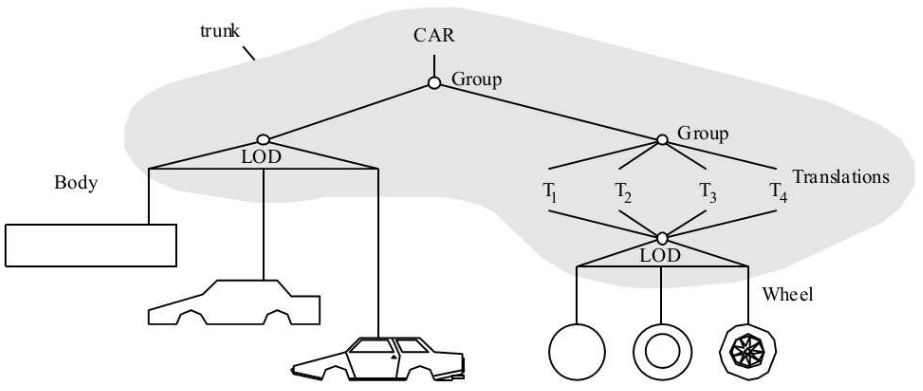


Figure 3.8: Car scene graph with discrete LOD. Geometry is placed only in lists of the tree. Rest of the tree is only a control structure with transformations and rendering states[12].

possibility to control the overall amount of triangles (to preserve constant frame rate) as well as the amount of consumed system resources [80].

Continuous LOD algorithms are also widely used for streaming 3D geometry over network [52], [139], [94], [23], [26], [75], [76], [3], [86], etc. They can significantly eliminate redundant data compared to the discrete LOD algorithm except some additional overhead which is needed to encode objects in the form of progressive representation. A disadvantage of the progressive algorithms is the higher algorithmic and computational complexity which is necessary to decode each object to requested resolution during rendering.

Two widely used approaches to represent 3D object using continuous LOD exists: progressive meshes [53] and wavelets [131].

Progressive meshes

The progressive mesh (PM) representation is efficient, lossless, continuous-resolution representation of arbitrary triangle meshes. In PM form, each mesh $\hat{M}$ is stored as a much coarser mesh M_0 together with a sequence of n recorded vertex-split operations (see Figure 3.10) that indicates how to incrementally refine mesh M_0 back to mesh $\hat{M}$ so that $\hat{M} = M_n$. Each vertex-split operation adds one vertex to current mesh.

An initial mesh $\hat{M} = M_n$ can be simplified into a much coarser mesh M_0 by applying a sequence of successive edge-collapse (ecol) operations so that:

$$(\hat{M} = M_n) \xrightarrow{ecol_{n-1}} \ldots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0, \tag{3.1}$$

, where $\xrightarrow{ecol_x}$ is an edge-collapse operation applied onto mesh M^{x+1}. Each performed edge-collapse operation generates one vertex-split operation. Thus the process described by the equation (3.1) is invertible and an arbitrary mesh can be represented using a much coarser mesh M^0 together with a sequence of n vertex-split operations as follows:

$$M^0 \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1} \ldots \xrightarrow{vsplit_{n-1}} (M^n = \hat{M}), \tag{3.2}$$

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

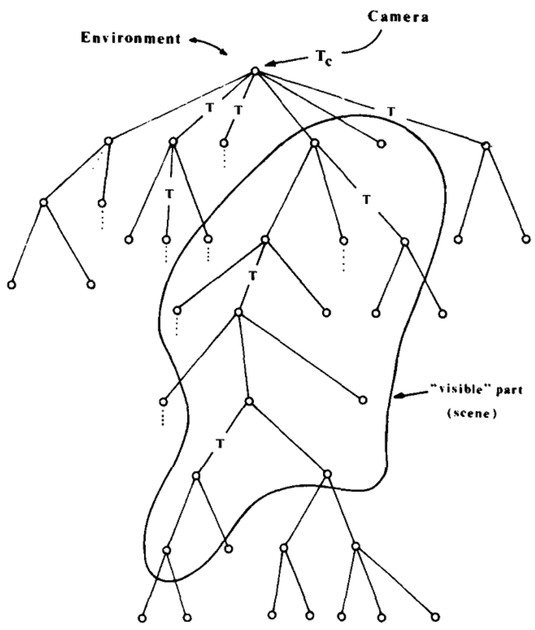


Figure 3.9: Structure of environment with transformation nodes T and visible part contour as a possible result of clipping. Each child node represent either transformation or pointer to more detailed part of scene[13].

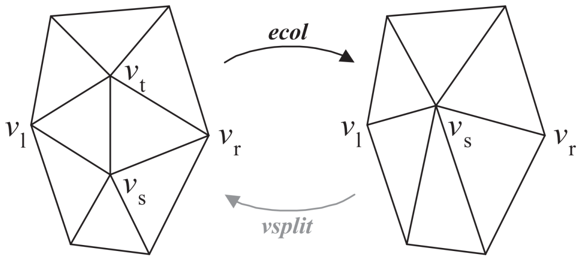


Figure 3.10: Illustration of edge collapse and vertex split operations[14].

A sequence of meshes $M_0, M_1, ..., M_n$ can be created, where each successive mesh contains one additional vertex. Thus each object can be represented using arbitrary number of triangles.

Description of objects represented by progressive meshes in network environment is standardized using VRML/X3D languages [51], [40], [82], [86]. In [51] each object represented by progressive mesh is stored in a separate file. Requested vertex-split records are then downloaded on demand.

Two new entities are defined for VRML/X3D in work [40]. The first entity is named *progIndexedTriSet* and extends the standard *IndexedFaceSet* entity. The second entity is named *ProgLOD* and contains additional data such as coordinates, indices, normals,

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

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

etc. These VRML/X3D extensions do not specify how the data should be streamed and rendered. Such a proposal is presented in [82], where the progressive records are compressed and extended with colour information which is not usually resolved by other algorithms.

Progressive mesh algorithm in its basic form (without any optimizations) is used to stream geometry of 3D objects in [86]. Here, a complex system for streaming VRML/X3D representation of planetary sized scene is described without any care about reducing the amount of transferred data.

Wavelets

Wavelet based representation of arbitrary meshes [131], [67] has more compact form for streaming compared to the progressive meshes.

Creation of the wavelet based representation from polygonal meshes is similar to the progressive meshes. An initial object is simplified to a much coarser representation and is stored together with a sequence of missing detail. These missing details are wavelet coefficients. Such representation is ready to be streamed over network [135], [63], [3].

Let M_n be a triangular representation of an object at resolution n. This object can be represented in various resolutions by a sequence of meshes $M_0, M_1, ..., M_n$, where M_0 is a base mesh (see Figure 3.11(a)) at much coarse resolution and mesh M_n is the original object at full detail.

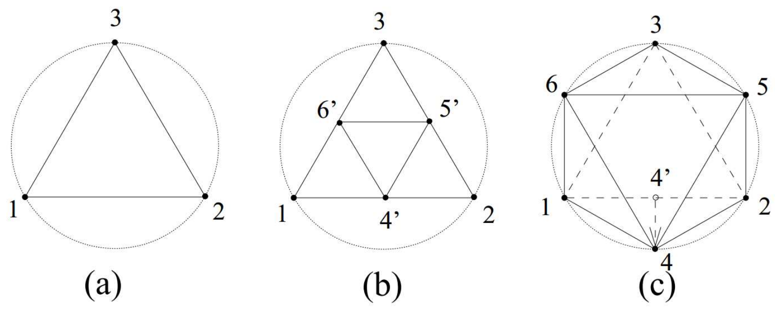


Figure 3.11: (a) A base mesh M_0 as a coarse approximation of a circle by a single triangle, (b) mesh obtained by splitting the base mesh M_0, and (c) M_1 as level 1 approximation of the circle[15].

For example, the base mesh $M_0(1, 2, 3)$ (see Figure 3.11a) is a much coarse representation of the circle object. To obtain one level higher resolution of the circle object, the triangle defined by vertices $(1, 2, 3)$ is divided into the four sub-faces. This introduces three new vertices $(4', 5', 6')$ at edge midpoints (see Figure 3.11b).

These three new vertices are translated to fit the approximated circle and are renamed to $(4, 5, 6)$ as is shown in Figure 3.11c). Wavelet coefficients (missing details) between the mesh M_0 and the mesh M_1 can be expressed as displacement between vertices 4, 5 and 6 from vertices 4', 5' and 6' respectively.

Wavelet decomposition of a mesh M_n produces the base mesh M_0 and sets $W_0, W_1, ..., W_{j-1}$ of wavelet coefficients. Each set W_i contains all wavelet coefficients at level i, which represent missing details between mesh M_i and mesh M_{i+1}.

[15]Mohammed Eunus Ali, Rui Zhang, Egemen Tanin, and Lars Kulik. A motion-aware approach to continuous retrieval of 3d objects. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 843–852, Washington, DC, USA, 2008. IEEE Computer Society.

The value of the wavelet coefficients plays the most important role when streaming them over network. The higher the value of the coefficients is, the higher its contribution to final object appearance. Final detail of transferred objects can be affected by transferring only coefficients with selected values (e.g. only high values).

3.2.3 View-dependent LOD

View-dependent LOD extends the continuous LOD algorithms with view-dependent simplification criteria to dynamically select the most appropriate LOD for the current view [80].

Many authors use view-dependent progressive meshes for streaming and rendering of large standalone 3D models [54], [141], [130], [140], [154], [155]. These algorithms allow selective refinement of arbitrary parts of given object e.g. silhouette (see Figure 3.13) or geometry inside view frustum (see Figure 3.12).

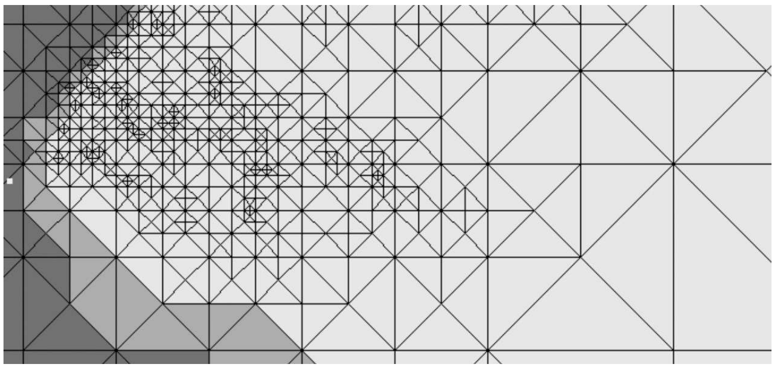


Figure 3.12: View-dependent LOD for terrain rendered using ROAM algorithm where the dark shaded region is outside view frustum, the middle shaded areas are on the border and the light shaded area is inside view frustum[16].

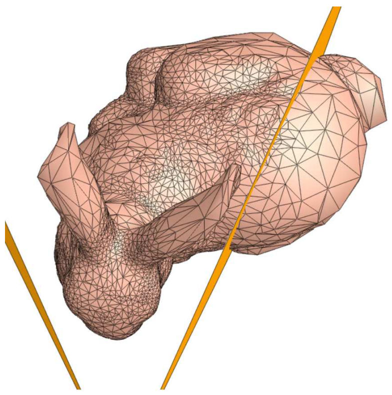


Figure 3.13: View-dependent rendering of Stanford bunny[17].

[16]Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. Roaming terrain: real-time optimally adapting meshes. In *Proceedings of the 8th conference on Visualization '97*, VIS '97, pages 81–88, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[17]Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 189–198, New York,

Two suitable view-dependent representations of 3D objects which support selective refinement exists: vertex trees [54], [38] and hierarchy of mesh update operators [100]. The hierarchy of mesh update operators such as half-edge collapses (see Figure 3.14) is advantageous, because it reduces storage cost by using only half as many nodes as the vertex hierarchies and is independent of vertex data.

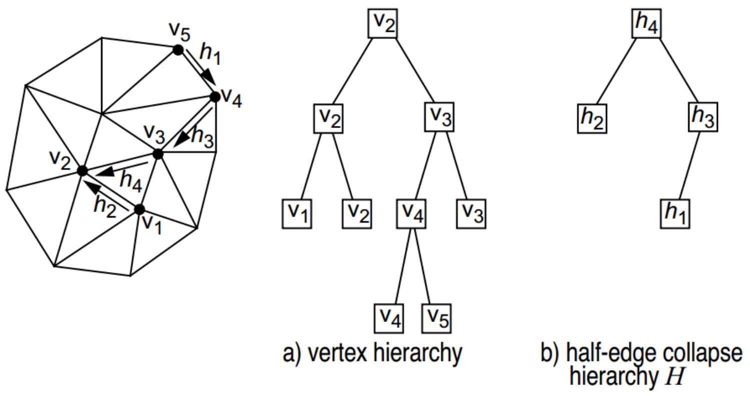


Figure 3.14: The principal difference between vertex trees and hierarchy of mesh update operators[18].

The view-dependent LOD algorithms are practically used to optimize rendering of large stand-alone models [54], [78], [38], [100], [151] as well as to optimize transmission of these models over network.

Authors of [141] and [69] were focused on view-dependent transmission of large triangle meshes over network, where the vertex trees are reconstructed from progressively received updates on client so that the view-dependent rendering can be performed locally on the partially received vertex trees.

Southern at all [130] proposed a framework based on edge collapse and vertex split operations. They extend these operations with spatial information represented by a visibility sphere. This information is used to determine whether an operation is visible and needs to be performed during rendering (see Figure 3.15).

Gueziec et al. [51] presented a framework for progressive transmission of geometry for not only VRML models. They propose a flexible LOD storage scheme named progressive multi-level mesh. In this scheme, the algorithm assigns a level value to each vertex and triangle as edges continuously collapses. During each edge-collapse, it also assigns to each removed vertex a representative which is the remaining vertex on the collapsed edge. The algorithm can traverse the representation locally so that the view-dependent transmission and rendering is possible.

Parallelization between a client and a server for view-dependent rendering and transmission of progressive mesh representation is presented in [155]. It proposes to run the client and the server processes in parallel, controlled by rendering time to cover a network latency. If the round-trip-time (RTT) between client and server is long, it manages to overlap the network latency for one frame with a rendering time for multiple frames.

NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[18]Renato Pajarola and Christopher DeCoro. Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):353–368, May 2004.

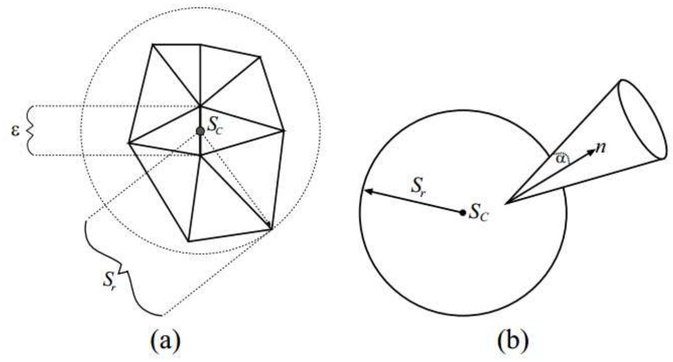


Figure 3.15: (a) determination of attributes of a visibility sphere. S_r represents radius of the sphere originating from centroid of base points of the region S_c. The error term ε indicates a distance between the two points which would be collapsed during an edge collapse operation. (b) A visibility sphere consists of the sphere (representing the atomic operation in space), and the cone of associated normals (for deter-mining whether the sphere is forward facing in real-time)[19].

In a classic sequential approach (see Figure 3.16) a typical frame on the client started first to send the view parameters to the server, goes through the waiting to get the mesh updates from the server, and ends as soon as the selected mesh will be updated and rendered. The problem is that the network overhead and the traversal time on the server side can be too large to guarantee the interactivity on the client. The proposed multi-frame predictive parallel strategy forces the client to send changed view parameters to the server each frame.

A common approach to send a refinements over a network is to let the server decide an order of refinement operations based on view-dependent criteria. Unfortunately, computational cost of making such ordering prohibits of such sender-driven approach from scaling to large number of clients. In [25] a different approach is presented based on receiver-driven protocol. In this protocol, a receiver decides sending order and explicitly requests refinements, while the sender simply sends the requested data. First, the algorithm assigns each vertex a unique identification number so that the receiver can explicitly request correspondent vertex-split operations. Second, the receiver (client) has to efficiently determine the importance of a vertex split operation based on partially received mesh.

A system for streaming geometry of terrain and buildings of a city model is described in [118]. Each building is encoded using a hierarchical representation called PBtree [119]. This representation encodes each building as a set of 2D coordinates of all its walls, its height and altitude. The PBTree representation also supports LOD by merging and simplification in-between several PBTrees.

View-dependent LOD algorithms suffer from an additional mesh reconstruction overhead and from evaluation of missing details at clients. Such evaluation must be performed upon each change of viewing parameters just during rendering. Preserving consistency of the rendered scene and its representation on server is also not trivial.

[19]Richard Southern, Simon Perkins, Barry Steyn, Alan Muller, Patrick Marais, and Edwin Blake. A stateless client for progressive view-dependent transmission. In *Proceedings of the sixth international conference on 3D Web technology*, Web3D '01, pages 43–50, New York, NY, USA, 2001. ACM.

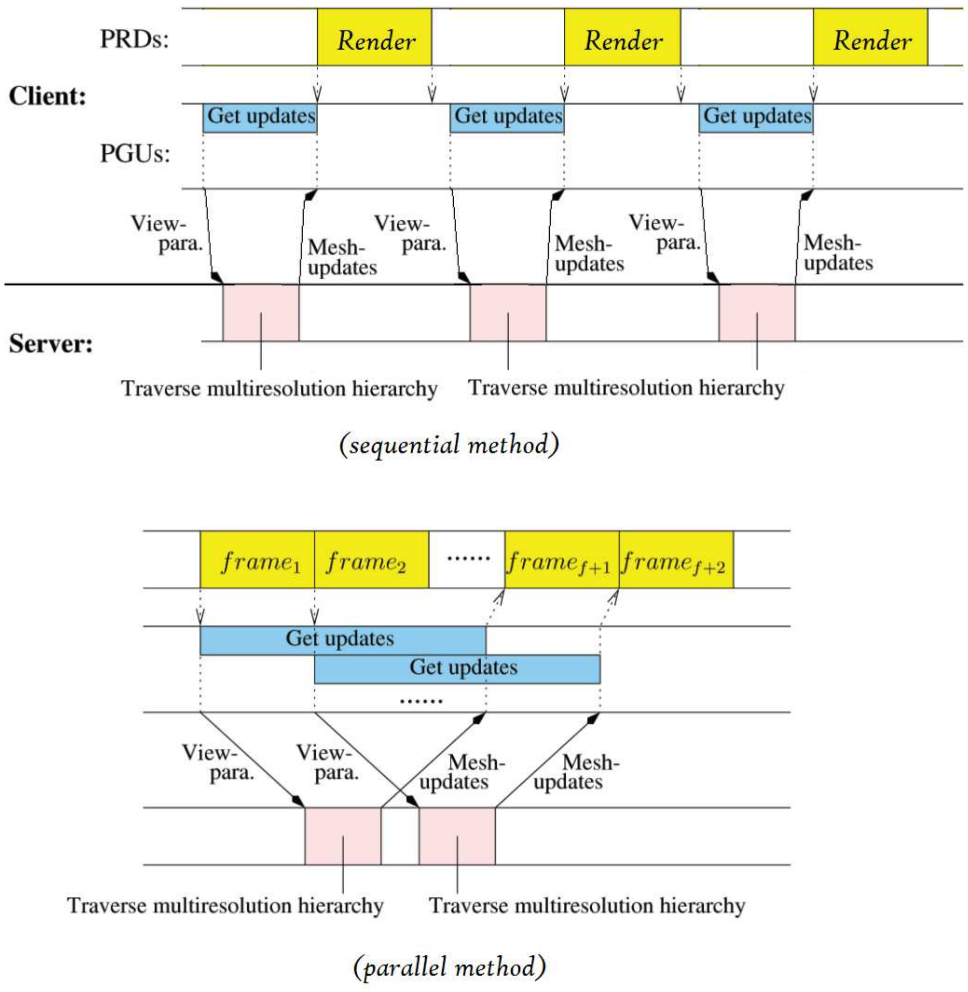


Figure 3.16: Comparison between the sequential approach (left) and the parallel approach (right), where PRD stands for render procedure and PGU stands for get updates procedure[20].

3.2.4 Image based methods

Image based methods can speed-up rendering of large scenes [62] using approximation of complex geometry by 2D images mapped onto quadrilateral or simplified geometry.

Billboards

Billboarding is the basic method for simplifying representation of 3D objects. Each billboard replaces an object or a group of objects with a single 2D image no-matter from which direction a viewer is looking at it. Each billboard simply just always faces a camera (see Figure 3.17). The billboard objects can be used for example to represent trees [12].

The billboarding idea is extended in [36] to billboard clouds. Each billboard cloud represents a geometry as a set of alpha-textured planes (see Figure 3.18.) organized so that they best approximate a simplified geometry.

[20]Zhi Zheng, Prakash Edmond, and Tony Chan. Interactive view-dependent rendering over networks. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):576–589, May 2008.

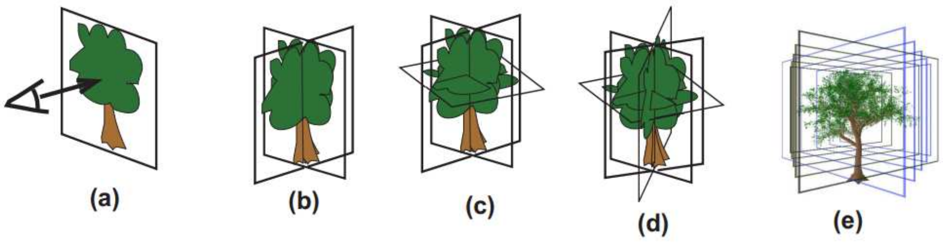


Figure 3.17: (a) billboard, (b) cross billboard, (c) 3-cross billboard, (d) 4-cross billboard, (e) billboard slices representing foliage[21].

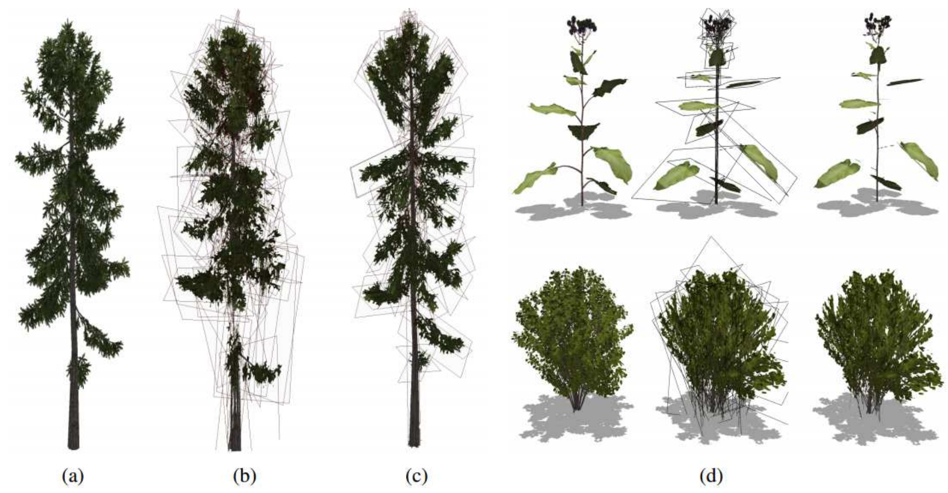


Figure 3.18: (a) original model, (b, c) billboard cloud approximation of the original model with different clustering of the geometry, (d) other approximations using billboard cloud[22].

Impostors

Another image-based representation of 3D objects are impostors. Simplest impostor is represented as a planar impostor [81], [121]. It basically consist of a planar geometry (e.g. a quadrilateral) with an alpha texture applied onto it.

The main advantage of it is that it can be fast created at runtime and also fast rendered because of its simple geometry. On the other hand, for parallax movements the impostor needs to be frequently updated. Other problems are discontinuities on the border between geometry and planar impostors, and also the visibility between a scene and flat impostors must be resolved satisfactorily [120].

In [48] and [59] a layered impostors method is proposed as an extension of planar impostors. It solves some planar impostors limitations, namely parallax movements of a viewer, correct visibility and seamless integration of them into a scene. The idea of layered impostors consists in usage of multiple image layers at different distances from the viewer. Each image layer represents certain depth range [61].

Another approach to replace geometry by images is the texture depth meshes method [33],

[21]A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J. C. Torres, editors, *Proc. of Eurographics (Short Presentations)*, Interlaken, Switzerland, 2000.

[22]Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Comput. Graph. Forum*, 24(3):507–516, 2005.

[127], [149]. This method is used for rendering objects which are relatively close to a viewer such that they just cannot be rendered using billboards. The basic idea is to triangulate a rendered image with respect to depth discontinuities (such as object silhouettes) based on z-buffer values (see Figure 3.19.). This technique provides good results for both parallax movements and also the visibility. The texture depth meshes method needs only a little additional geometry compared to the planar impostors.

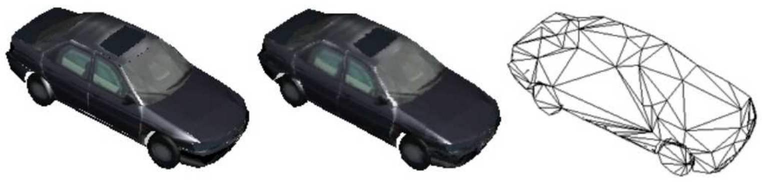


Figure 3.19: From left to right: original model, the texture depth mesh (TDM) representation and finally the mesh of the TDM[23].

Particular methods differ mainly in how the impostor is created, how it is rendered and how the visual inaccuracies are eliminated. A detailed survey of the image-based methods reducing the scene complexity is presented in [61] and [105].

Client-server systems which use impostors and billboard algorithms to represent distant objects are presented in [105], [98]. The main disadvantage of the image-based representation of 3D objects is that the transmitted image-based representation can be useless if a viewer is approaching them, because the close object needs to be rendered using full-resolution triangle mesh which have to be downloaded from scratch then.

Parallax mapping

Modern image-based approach to represent the geometry of complex objects is the parallax mapping method [66] or better the parallax occlusion mapping [137]. The parallax occlusion mapping reduces the geometry of objects by encoding their surface details into a texture. The surface is practically stored as a heightmap representation of the replaced geometry. During rendering of the replaced geometry, its detailed surface is reconstructed from associated heightmap in pixel shader (see Figure 3.20).

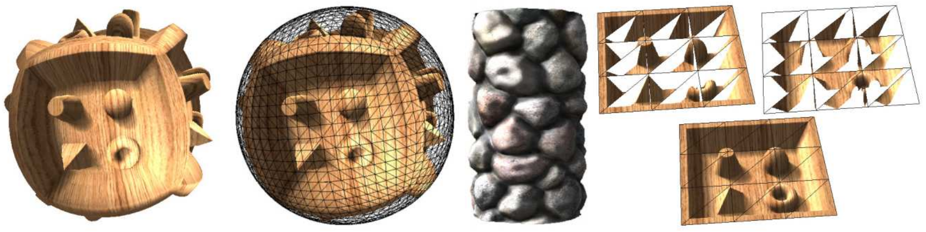


Figure 3.20: Examples of parallax occlusion mapping including corresponding geometry[24].

Parallax occlusion mapping can be potentially used in networked applications, because it can reduce the amount of data used to represent single 3D objects (see Figure 3.21).

[23]Stefan Jeschke. Accelerating the rendering process using impostors, March 2005.

[24]C. Dachsbacher and Tatarchuk N. Extended parallax occlusion mapping with accurate silhouette computation. In *I3D Poster*, 2007.

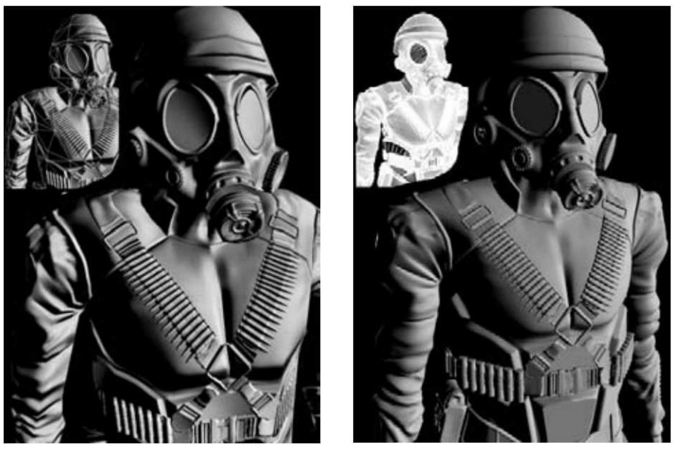


Figure 3.21: (left) A 1.100 polygon game soldier character with parallax occlusion mapping (cca 14Mb), (right) A 1.5 million polygon soldier character with diffuse lighting (45Mb)[25].

The soldier represented using the parallax occlusion mapping contains 1100 polygons with memory requirements: vertex buffer(79Kb), index buffer(6Kb) and 3Dc compressed 2048×2048 texture (13Mb), total cca 14Mb. The original soldier is the object composed of 1500000 polygons with memory requirements: vertex buffer(31Mb), index buffer(14Mb), total cca 45Mb. Several resolutions of the 3Dc compressed texture can be created in a preprocessing step and streamed from the server to the client with continuously increasing resolution. A disadvantage of such approach is texture data redundancy when streaming particular texture level-of-details (for details see the section 3.4).

Image based rendering

Image based rendering (IBR) systems are effective solution to solve rendering of complex 3D scenes mainly on mobile devices. As they have much less computing and rendering power they cannot render scenes at competitive quality at interactive frame rate compared to desktop computers. The IBR methods depends mainly on display resolution, compared to the classic rendering pipeline, where rendering performance is limited by throughput [22].

Communication model of image-based rendering systems is usually based on client-server architecture [22], [73], [13], [157]. A scene is rendered at the server from current current view and streamed to the client which renders the received image as impostor [22], using MPEG-4 standards [97], [70] or using panoramic images, which consist of creating a panorama representation of the scene from several images taken by rotating the camera around a fixed point [13].

In [22] an image-based rendering system is described based on client-server architecture. Here, the server renders a scene into an image and sends it to the client. The client uses McMillan's 3D warping method to render the received image [90]. The McMillan's warping method uses depth images (containing both colour and depth information at each pixel) together with a viewing 3×4 matrix which describes the view setup of the depth image. Resulting warped image must be created using only a single depth image. Practical consequence of this approach is that as the output image viewing matrix is changing, the

[25]Natalya Tatarchuk. Practical dynamic parallax occlusion mapping. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

output image start suffering from occlusion artefacts caused by revealing parts of the original 3D scene, which were occluded in the used depth image (see Figure 3.22). he solution to this problem is an extension of the McMillan's warping algorithm by layered depth images [125], but at the expense of increased size of the image rendered at server (the size will grows linearly with the number of layers).

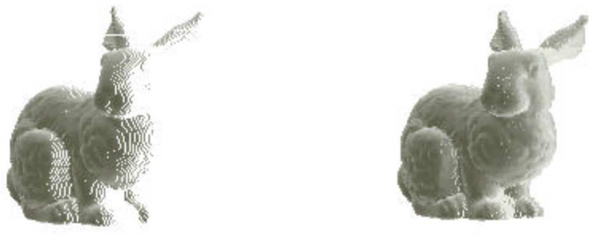


Figure 3.22: (left) bunny model with occlusion artefacts, (right) layered depth images as a solution to the occlusion artefacts[26].

A hybrid image-based rendering system was proposed by the authors of paper [95]. Their system renders all distant parts of a terrain model at a server using impostors whilst the terrain around the viewer is rendered using classical rendering algorithms.

Utilization of IBR techniques is not so wide because computing and graphic power of contemporary mobile devices is already relatively high.

3.2.5 Geometry data compression

This section contains a brief review of progressive compression techniques, because they can effectively further reduce the amount of geometry data streamed over network.

Concept of the progressive compression was introduced for the first time by Hoppe in his progressive meshes [53]. Compression of progressive records is achieved at a high cost, since positions of split vertices are explicitly encoded. A lot of methods were then introduced to increase a compression ratio by applying vertex split and edge collapse operations on sets of independent vertices [138], [102]. Another approach to improve compression ratio is presented in [31]. Instead of edge collapses, their approach consists in removing a vertex and re-triangulating the hole left by previous deletion operation.

Observing that geometry data is larger than connectivity data, more recent research has focused on geometry-driven compression algorithms [44], [107]. These works are based on space subdivision based on kd-tree [44] or more efficient octree cells [107]. A comprehensive survey of these methods can be found in [106] and [4].

In general, geometry-driven algorithms outperform connectivity-driven algorithms in terms of lossless compression ratio, but at the expense of quite poor results at intermediate resolutions (see Figure 3.23), which is the reason why they are not suitable for progressive transmission over networks.

Additional mesh properties such as colours or texture attributes can often be of greater size than other data flows within a mesh and can carry a great part of relevant information in applications such as scientific visualizations. Very few progressive mesh coders allow the

[26]Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 231–242, New York, NY, USA, 1998. ACM.

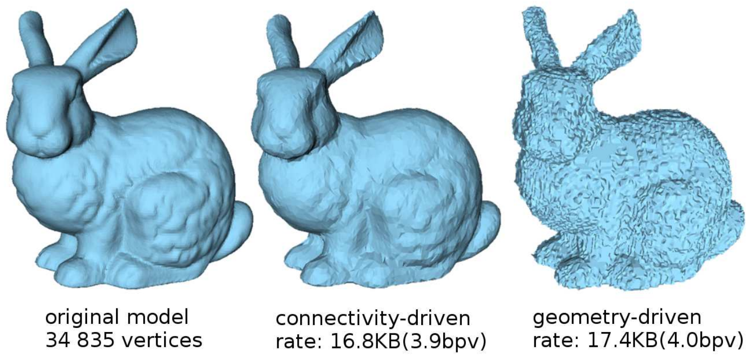


Figure 3.23: Original model and results of progressive decoding at similar bit-rates for state-of-the-art connection-driven algorithm and geometry-driven one[27].

encoding of these information together with geometry except [29]. Here, a geometry-based method is used together with the kd-tree space decomposition.

Another work [82] proposed a new connectivity-driven progressive encoding of a mesh and extent it with support for colours associated with vertices. It also proposed a X3D framework working with such mesh encoding which is suitable for remote progressive scientific visualization of large coloured meshes.

3.3 Streaming of terrains

This section describes state-of-the-art in terrain streaming algorithms. In general, terrain streaming is closely related to terrain data representation and rendering algorithm. The most common terrain data representations are triangle irregular networks (TIN) and heightmaps. The terrain created from TIN can be streamed and rendered as a general 3D geometry with the help of algorithms described in the previous section 3.2. The terrain represented by heightmaps must be streamed and rendered differently. A comprehensive survey of interactive terrain rendering of semi-regular multi-resolution models can be found in [101].

Terrain data is stored at a remote server and are usually progressively transferred to clients on demand. Time interval between some download request is sent to the server, and when the transfer of the requested data is finished can be relatively long compared to accessing local data solely. The size of the time interval is proportional to the network restrictions such as latency or network bandwidth.

3.3.1 Triangle irregular networks

Hoppe [55] uses a progressive meshes to simplify a terrain geometry (represented by TIN) for fast rendering (see Figure 3.24).

He partitioning a terrain model into blocks. Starting from the top detailed level, it simplifies each block by applying a sequence of edge collapse transformations. To avoid

[27]Adrien Maglo, Guillaume Lavoué, Celine Hudelot, Ho Lee, Christophe Mouton, and Florent Dupont. Remote scientific visualization of progressive 3D meshes with X3D.In ACM, editor, *International Conference on 3D Web Technology (Web3D)*, July 2010.

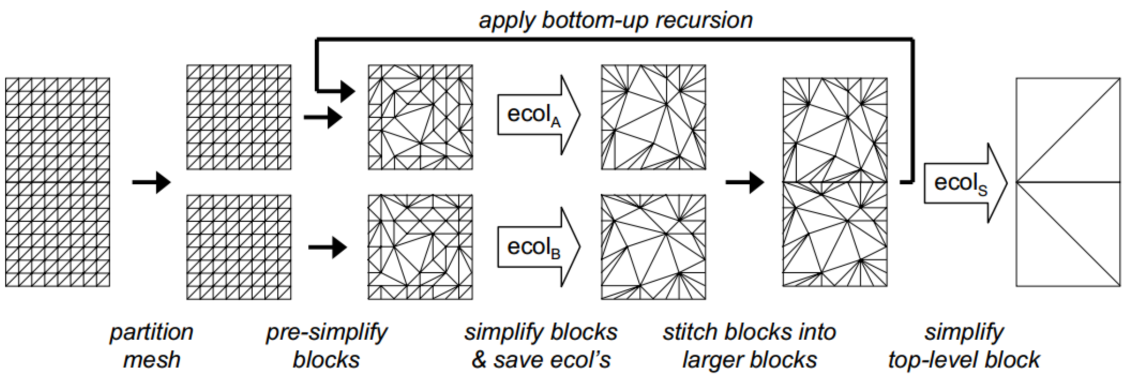

apply bottom-up recursion
ecol$_A$
ecol$_B$
ecol$_S$
partition mesh
pre-simplify blocks
simplify blocks & save ecol's
stitch blocks into larger blocks
simplify top-level block

Figure 3.24: Steps in the hierarchical block-based simplification of a terrain done in a pre-process[28].

refinement dependencies between adjacent blocks, the algorithm constraints edge collapse operations to leave all boundary vertices untouched. The simplification process terminates when an approximation error of next best edge collapse exceeds a user-specified threshold for current level. Then simplification sequences of edge collapse operations are saved to disk and resulting simplified meshes are stitched together 2x2 at a time. Then the same process is repeated for the next higher level using these larger blocks.

Methods based on progressive meshes enables the encoded terrain data to be transferred over network with only small overhead using a progressive streaming approach (see the section 3.2.2), with optionally compressed progressive records (see the section 3.2.5).

Instead of working with blocks of terrain, a batched dynamic adaptive meshes (BDAM) [28] method divides an input space into binary trees of right triangles. Each right triangle represents a terrain patch which is composed of larger group of triangles. These patches can be computed off-line in a preprocessing step. The BDAM method simplifies pairs of triangles inside each patch by performing edge collapses based on quadric error metric [45]. The solution to stream the data over network can be based on progressive meshes, but the work did not propose any solution for this.

The BDAM algorithm is used also in [19]. Here, a new parallel simplification algorithm runs effectively on current multi-core CPU and GPU. It has the advantage that it does not require the use of the regular grid so it can be applied to general 3D models.

Mesh simplification is done by successive contractions, or collapses of its edges. To control the error induced by these collapses, the quadric error metric (QEM) method was selected because it is naturally well adaptable to parallelism [46]. Both refinement operations as well as simplification of the model must support parallel execution. The only operation performed during a refinement process is vertex split. This is why a point can never see its valence (number of neighbour vertices) decreasing but only increasing, each time one of its neighbours is split in turn. Therefore, only valence v_R of a point R at the time of its creation (R is a point which remains after an edge collapse) is required by a decoder: during decompression, a point R will be candidate to a split when its valence reaches v_R.

To enable streaming, it is necessary that order of points during the refinement process

[28]Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 35–42, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

on client side is identical to that of the simplification. To ensure this consistency, points after each simplification stage (collapse batches) are reordered.

Binary structures describing decompression of a model can be compressed by any entropy algorithm such as gzip. Unfortunately, no solution is proposed to handle textures in both terms of streaming and compressing them.

3.3.2 Heightmaps

Pajarola [99] developed a restricted quad-tree triangulation of terrains, which are represented by heightmaps. It also proposes a progressive transmission of terrain data over network. The input heightmap is divided into blocks. Each block is triangulated by traversing a quad-tree hierarchy so that each block is rendered using a single triangle strip. This procedure is not optimal for GPU, because it requires re-meshing the quad-tree and re-sending generated geometry back to GPU for every frame as a viewer change its view or position.

He also proposed a progressive transmission of particular height points. The restricted quad-tree traversal builds up a progressive mesh (defined on vertex insert operation) as a sequence of vertices which can be transmitted successively. It has the disadvantage that the traversal must be done on the server to identify the vertices to transmit.

Kim and Ra [68] proposed a networked visualization algorithm combined with wavelet-based compression. In their approach, a restricted quad-tree mesh model is updated in real time by wavelet coefficients decoded from a compressed bit-stream. The terrain elevation grid is divided into blocks and the wavelet coefficients are pre-computed and stored on the server as hierarchical block-based tree of bit-streams for each block (see Figure 3.25).

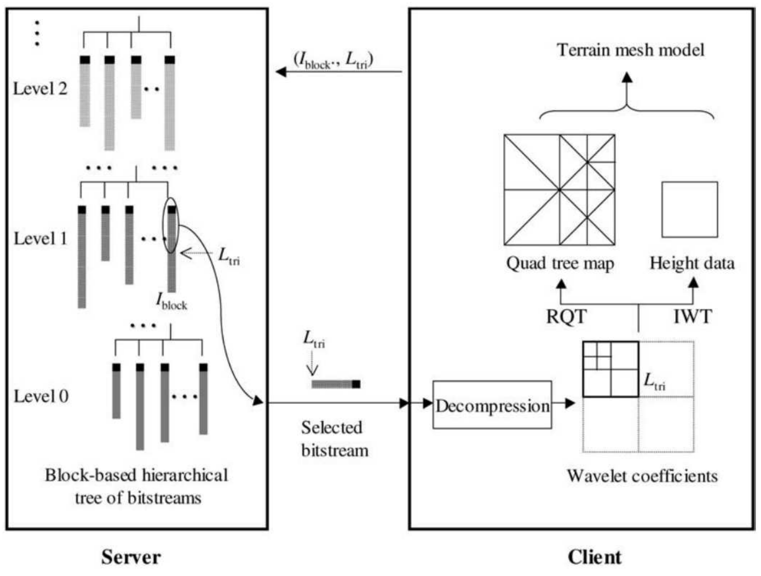


Figure 3.25: Server-client data streaming relationships in[29].

In [47] a geometry compression scheme is presented for restricted quad-tree meshes. The idea behind it is to build a triangle strip from generalized information such as triangle types (determined by edges across which a path enters and leaves each triangle), windings, and height values. A compression factor of $8-9$ is achieved by employing such generalized strip

[29]Jin Kook Kim and Jong Beom Ra.7 A real-time terrain visualization algorithm using wavelet-based compression. *Vis. Comput.*, 20(2):67–85, May 2004.

representation of quad-tree meshes to incrementally encode vertex positions. It allows significant reduction of bandwidth requirements when rendering large digital elevation models (which have to be paged from disk). The decompression can be effectively done on GPU. The triangle strips are pre-computed and compression is performed in the pre-processing step. Unfortunately, no solution is proposed to stream such compressed data over network.

TerraVision system II [116] streams and visualizes large terrain datasets using VRML. It exploits the multi-resolution approach, where the global digital elevation model (GDEM) of a terrain and the texture map are divided off-line to a regular grid of square tiles at different resolutions. These tiles are then stored together with other additional information into particular files (see Figure 3.26).

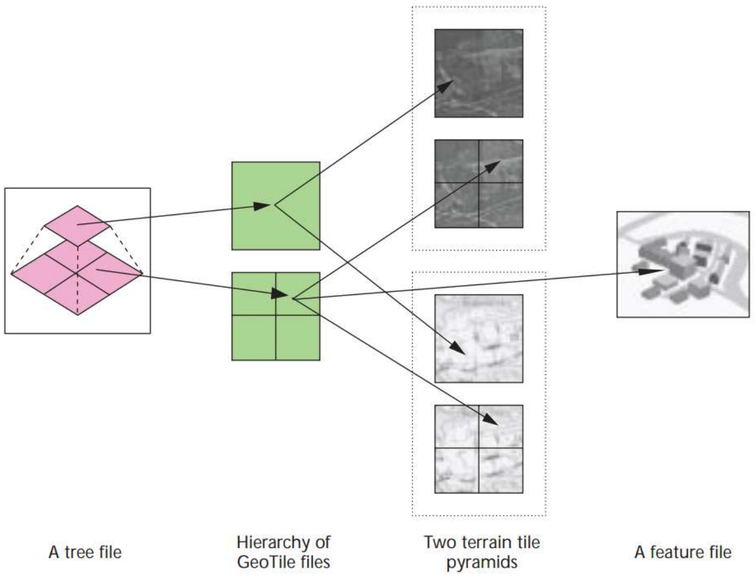


Figure 3.26: Unidirectional arcs represent in-line links to files over network. Each tree file contains a definition of hierarchy of references to geotile files. Each geotile contains reference to a heightmap, a texture and to additional information such as buildings, streets, names etc[30].

A newly proposed VRML node called *QuadLOD* efficiently manages loading and unloading of higher levels-of-detail. In effect, the tree files are the glue that holds the geotiles in the quad structure. The tree file initially downloads a single geotile. As a viewer needs more detail, the initial geotile is replaced with its four children (at higher-resolution) and so on. This behaviour is implemented by the *QuadLOD* node and can be easily applied to general VRML principles. The system induces the data redundancy and no solution is proposed to ensure continuity between different levels-of-detail of adjacent geotiles.

Another terrain streaming approach is proposed in [112]. Here, a full terrain heightmap is divided into small regular tiles which are streamed and managed adaptively. The adaptivity consists in capability of the rendering algorithm to render the scene on any kind of device (including slow hand-held as well as desktop workstations) by exploiting device capacity to draw as much triangles as possible for a target frame rate and memory limitations.

Each visible terrain tile is rendered using a set of precomputed triangle strips. Each

[30]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, March 1999.

triangle strip is created with different density (see Figure 3.27). The rendering system renders a specified area around an moving user whilst missing tiles are downloaded or loaded from cache (see Figure 3.28). It always keeps vertices data of all visible tiles around a viewer in memory and it only selects the precomputed triangle strip with required number of triangles. It allows to control the overall number of triangles only by selecting the strip-masks with different density, so that it can render the terrain on any kind of devices including also slow hand-helds.

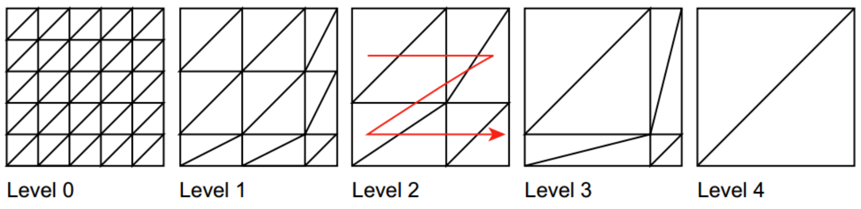


Figure 3.27: Each triangle strip is precomputed and renders covered height points using different number of triangles[31].

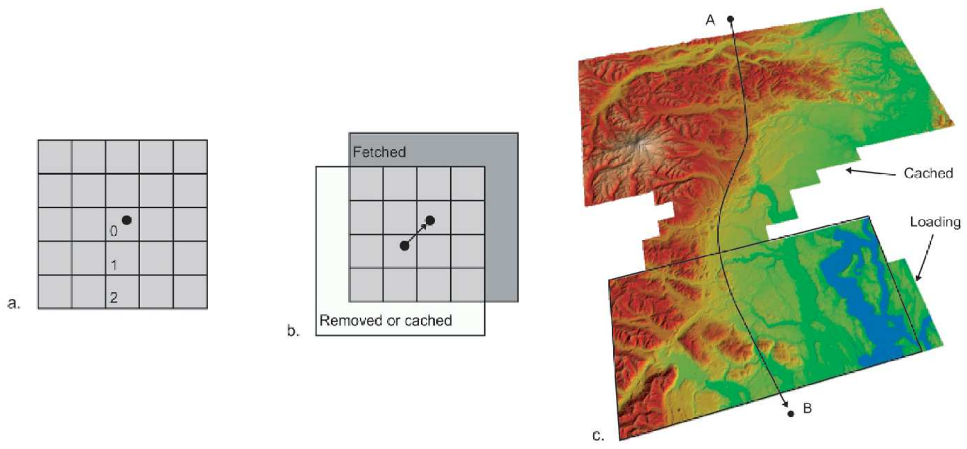


Figure 3.28: Tiles management and caching: a) a square render area made of three belts centered on a viewpoint, b) square area preservation on viewpoint move and c) illustrates caching all tiles stored in memory are rendered (only the black rectangular area is rendered in normal setup)[31]

A disadvantage of this algorithm is that the distant tiles or tiles with low importance contain the full resolution of height data even they are not always fully utilized because the strip with less number of indices can be selected for those tiles. The terrain tiles are also transferred only at the highest resolution without considering the rendering requirements. Moreover, the terrain triangulation is not watertight and is resolved only with the well known shadow plane projection algorithm.

Streaming of a terrain heightmap is proposed also in [152]. Here, the hierarchical representation of input heightmap is created as in geometrical mip-mapping algorithm [34]

[31] Joachim Pouderoux and Jean-Eudes Marvie. Adaptive streaming and rendering of large terrains using strip masks. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 299–306, New York, NY, USA, 2005. ACM.

by sampling from the input heightmap with step 2^n, where n is current level-of-detail in the hierarchy. First only coarser data in the large range of terrain around a user is transferred. Further, the detailed data in the user view frustum is transmitted. Since coarser height data is already present in the client cache, only missing height data will be transferred from server to the client.

During rendering the terrain, the detailed data must be restored from the basic coarse data and from received increments. Such approach can reduce redundancy between successive levels-of-detail but at the expense of reconstruction of them by the client (in the case they are not yet in the client cache).

A generic solution for streaming of heightmaps for a planetary scale terrain renderer is proposed in [74]. Here the sample height map is divided into a complete and uniform tree of blocks (each block is regular fixed size part of terrain heightmap) and samples of these blocks are organized in a structure with increasing resolution (see Figure 3.29).

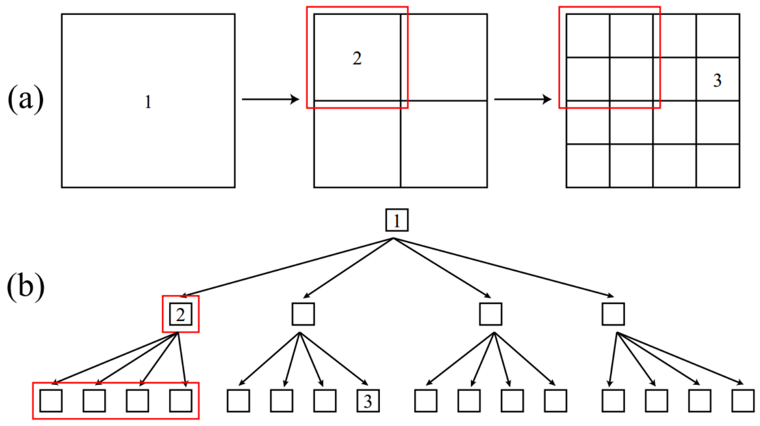


Figure 3.29: Construction of a three-level quad-tree of blocks: (a) Successive uniform subdivisions of the terrain (red frames cover the same terrain area). Block 1 is the root and covers the entire terrain. (b) Corresponding tree of blocks[32].

The main idea is similar as in [152] so that samples within each block are shared across all LODs instead of duplicating them (see Figure 3.30).

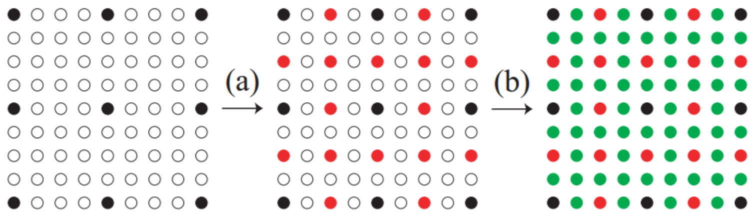


Figure 3.30: Block refinements of basic block with 9 height samples (these points come from parent), (a) Red samples (second LOD) are interleaved between black ones (first LOD), (b) Green samples (third LOD) are added and the array is now full[32].

[32]Raphael Lerbour, Jean-Eudes Marvie, and pascal Gautron. Adaptive streaming and rendering of large terrains: A generic solution. In *The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2009*. Vaclav Skala, feb 2009.

All height samples from the much coarser level and all height samples from the next finer levels are serialized in a coarse to fine order and are stored on server. An advantage is no redundancy when incrementally streaming the terrain blocks, but at the expense of costly data structure operations at client. The same principle is used also in the system for streaming large complex scenes trough HTTP [86].

Extension of the BDAM algorithm - C-BDAM (compressed batched dynamic adaptive meshes) [49] is based on the idea, that controlling triangle shapes during simplification to reduce triangle counts is no longer of primary importance regarding the modern GPU. Instead of working with irregular triangle networks the regular grid is used rather and the mesh simplification/refinement operators are replaced by digital signal processing operations. Instead of using edge collapse and vertex split operations, the C-BDAM algorithm uses a wavelet analysis for coarsening of the heightmap and a wavelet synthesis for refinement of it.

In [10] a similar approach is described as in C-BDAM [49]. Here, the used wavelet schema can be constructed using a parallel single-pass compression process which can be decoded by clients even with low computational power.

Wavelet based encoding of terrain details is used also in [118]. Here, the terrain data in a suitable form for streaming is represented as a much coarse polyhedron with a set of wavelet coefficients which can be interpreted as local deformations of the input terrain shape. Wavelet based solutions are advantageous for their compression efficiency and intrinsic ability to multi-resolution visualization and streaming with low complexity of a reconstruction algorithm.

A hybrid algorithm to stream only a portion of large terrain around a viewer is described in [95]. The proposed algorithm combines rendering of heightmap around the viewer with impostors. The terrain is divided by regular grid to equal sized height tiles so that each tile covers a square area including $(2^n + 1)x(2^n + 1)$ height values, where n is an integer value greater than zero. Each tile is then associated with single quad-tree. The quad-tree stores much coarse height tile at its root. The quad-tree is recursively subdivided and each child node adds a set of terrain points which were missing in the previous levels.

The hybrid part of the algorithm is based on the idea, that distant parts of the terrain can be rendered at the server (panorama) and transferred back to the client which effectively renders them as impostor (see Figure 3.31).

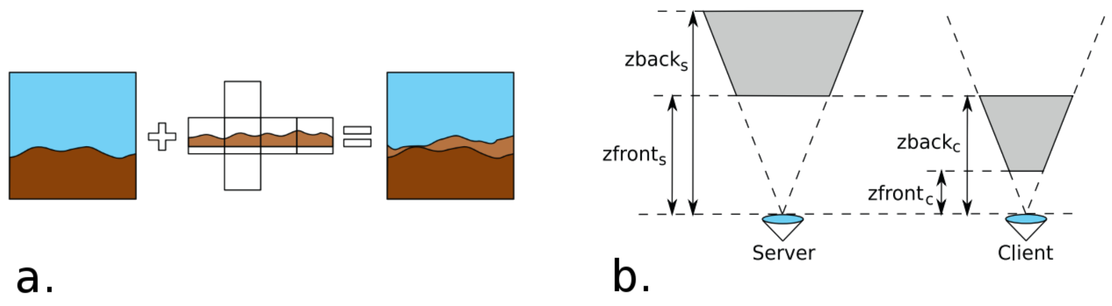


Figure 3.31: (a) Synthesis of nearby terrain rendered at client and the panorama rendered at server, (b) splitting the view volume as a terrain to be rendered and a panorama[33].

[33]José M. Noguera, Rafael Jesús Segura, Carlos J. Ogáyar, and Robert Joan-Arinyo. Navigating large terrains using commodity mobile devices. *Computers & Geosciences*, 37(9):1218–1233, 2011.

When streaming terrain data, the server sends only height points for requested quad-tree nodes. Until they arrived to the client, the coarser data is used to render the terrain. Update of the panorama images is based on perspective projection principle so that the changes in projection of distant points for small moves of a viewer are almost unnoticeable. Therefore, updating strategy of the panorama images is based on estimation of the error incurred in a rendered scene when the viewer moves but the panorama would not be updated. A detailed description of a client-server architecture which extends the previous hybrid algorithm with multi-client environments is presented in [96].

Suarez et al. [132] introduced a virtual globe framework. They render a planetary scale terrain using a simple quad tree of chunks of heightmap, where he gaps between the tiles at different LOD are resolved simply by skirts (see Figure 3.32).

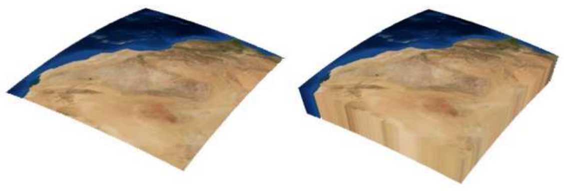


Figure 3.32: (left) A terrain chunk without skirts, (right) with skirts[34].

In [35] a terrain tiles streaming system transmits a very low resolution heightmap for entire terrain dataset at the beginning. As a viewer moves, higher detailed data is streamed to a client depending upon the viewer position. When requesting some tile, the last transferred resolution of the tile is checked and only residues between the requested high resolution tile and the super-sampled version of the low resolution tile is transmitted to the client. The residue is compressed using a progressive transform codec (PTC) [83] before transmission.

In [57] the authors pointed out that when using a 4-8 hierarchy, the rectangular tiles associated to each diamond could be also compressed using standard 2D image compression methods. The work proposes an efficient method for incorporating compression of both height and texture information within the BDAM algorithm.

3.4 Streaming and rendering of textures

Utilizing large amount of textures is not usually the main point of interest of published multi-resolution rendering techniques. But the amount of texture data can be more than one order of magnitude larger (typically terrain heightmaps with high resolution orthographic textures) compared to the amount of geometry data.

Large textures processing was first studied by Williams [148]. He introduced texture level-of-detail, represented by images with increasingly reduced resolution arranged into a pyramid. Starting with the finest level, each coarser level represents an image using one quarter of the texels from previous finer level (half the number of texels in each dimension). Per-pixel rendering with is accomplished by projecting the pixels into the mip-map space using texture coordinates and camera transformations. The rendered pixels are coloured

[34]José P. Suárez, Agustín Trujillo, Manuel de la Calle, Diego D. Gómez-Deck, and Jose M. Santana. An open source virtual globe framework for ios, android and webgl compliant browser. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, COM.Geo '12, pages 22:1–22:10, New York, NY, USA, 2012. ACM.

using a variant of trilinear interpolation of eight texels taken from two adjacent levels from the mip-map hierarchy.

Reddy at al. [116] utilized a multi-resolution approach, where the global digital elevation model of terrain and also the texture map are divided off-line to a regular grid of square tiles at different resolutions. These tiles are then stored together with other additional information into separate files (see Figure 3.26 and Figure 3.33). The rendering algorithm efficiently manages loading and unloading of the higher levels-of-detail from these files.

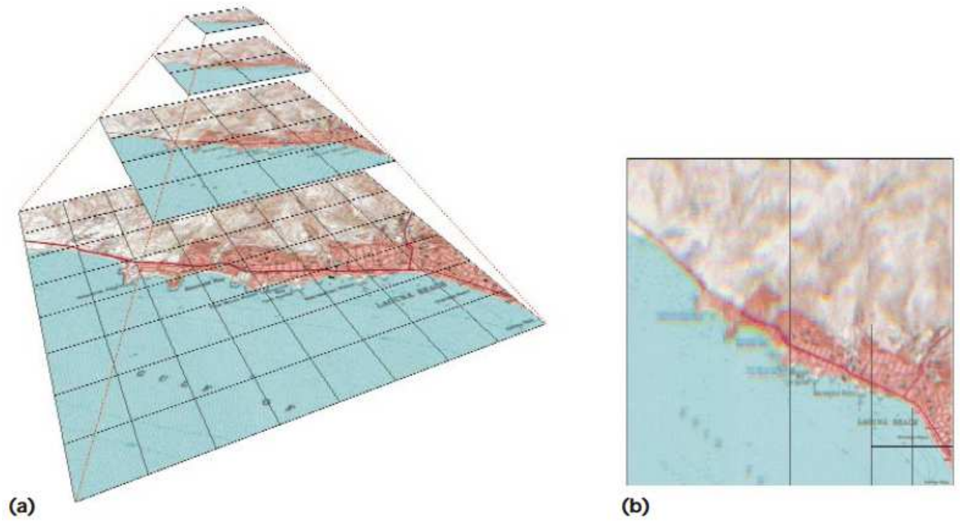


Figure 3.33: An example image pyramid showing (a) four different resolutions of the original image, where each level is segmented into 128 × 128 pixel tiles, and (b) how this structure can be used to alter the image resolution in different regions[35].

Dollner et al. [37] used memory-mapped texture files for textured terrain visualization. Their method utilizes a tree of texture patches in conjunction with a hierarchical model of the terrain geometry (see Figure 3.34). During rendering, the hierarchical terrain model is simultaneously traverses together with the texture tree of patches, selecting terrain and texture data according to a user-defined visual error requirements.

Ulrich [142] combined a quad-tree of mip-map tiles, called chunks, to handle huge textured terrains. The texture and the geometry chunks are generated in a preprocessing step. A single texture is defined for each chunk. Its resolution is determined by screen-space error metric at run-time. At the pre-processing step, it can be guaranteed that a projected texel size at run-time will not exceed 1 texel per pixel.

Cignoni et al. [28] demonstrated an ability to display of large textured terrains in real-time. They used a quad-tree texture hierarchy and bin-tree of triangle patches over triangle irregular mesh (TIN). Triangle patches are constructed off-line and are selectively refined from scratch each frame. The textures are organized in a quad-tree of square tiles. Their rendering algorithm traverses the texture quad-tree and corresponding patches in the geometry bin-tree until defined space error tolerance is reached.

In [56] a texture mip-map quad-tree (TMQ) is used to represent large textures for terrains. Each node of the TMQ corresponds to a rectangle region from the original texture,

[35]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, March 1999.

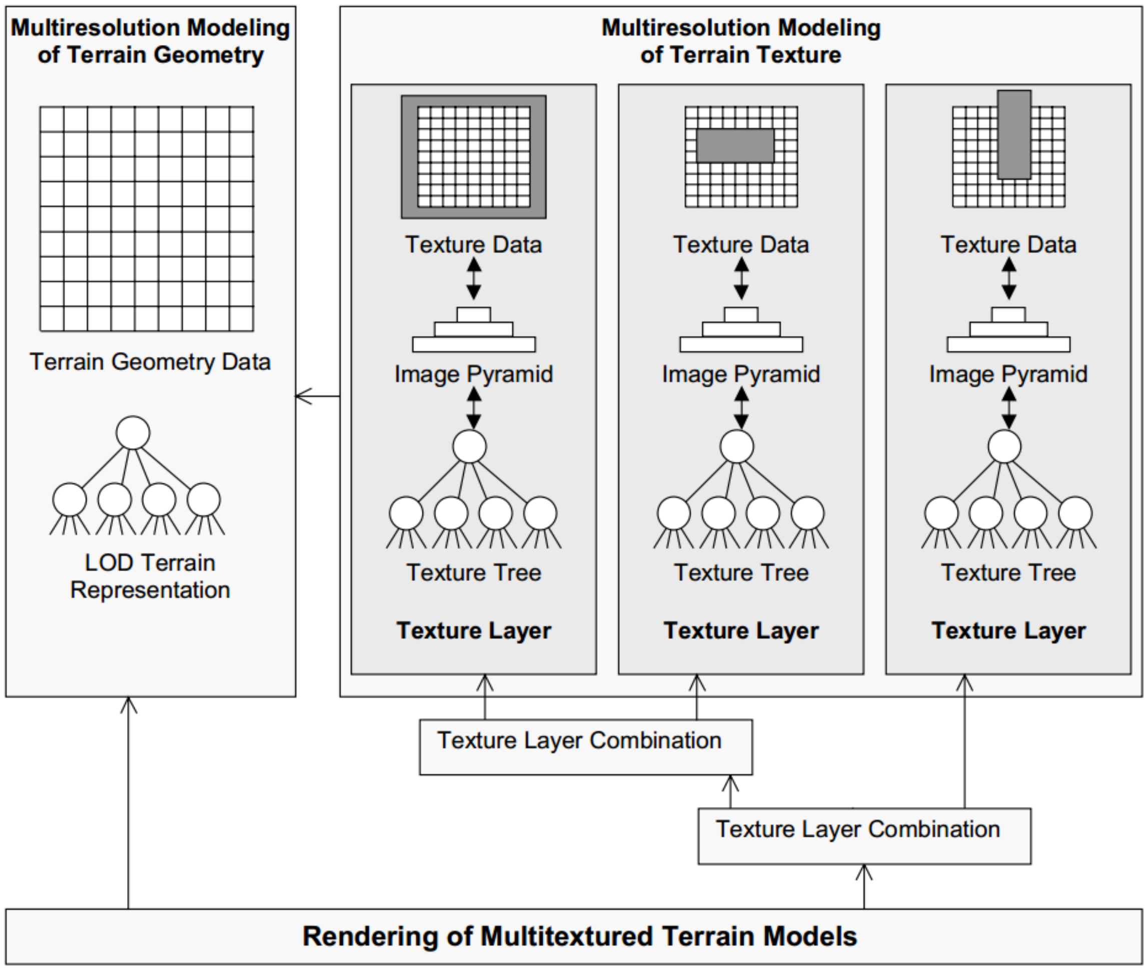

Multiresolution Modeling of Terrain Geometry
Terrain Geometry Data
LOD Terrain Representation
Multiresolution Modeling of Terrain Texture
Texture Data
Image Pyramid
Texture Tree
Texture Layer
Texture Data
Image Pyramid
Texture Tree
Texture Layer
Texture Data
Image Pyramid
Texture Tree
Texture Layer
Texture Layer Combination
Texture Layer Combination
Rendering of Multitextured Terrain Models

Figure 3.34: Conceptual model of multiple texture layers. Each texture layer contains a texture tree which is in conjunction with the terrain LOD [37]. The work distinguish between four texture layers: colour texture layer, luminance, topographic layer with shading information and finally the visibility layer which specifies the visibility of other texture layers[36].

and each sub-image in a region is further organized into a mip-map (see Figure 3.35). All nodes in the TMQ has the same resolution, but cover different sized triangle regions. All triangles within each region share the same texture no matter how far they are from a viewpoint. This causes aliasing in closer views. Therefore, for each texture assigned with nodes is generated a mip-map structure, which is further compressed to reduce redundancy.

Most multi-resolution texture algorithms use a quad-tree of textures, where all tiles have the same number of texels and particular children cover one fourth of the area covered by their parent. Selecting adjacent tiles where texels per unit area differ by factor of four (factor of two in each direction) can produce visual discontinuities. In [57] a novel texture hierarchy is introduced to handle the texture level-of-detail for extremely large surfaces. It provides twice as many levels-of-detail compared to the conventional quad-tree-style

[36] Jürgen Döllner, Konstantin Baumann, and Klaus Hinrichs. Texturing techniques for terrain visualization. In *Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, VISUALIZATION '00, pages –, Washington, DC, USA, 2000. IEEE Computer Society.

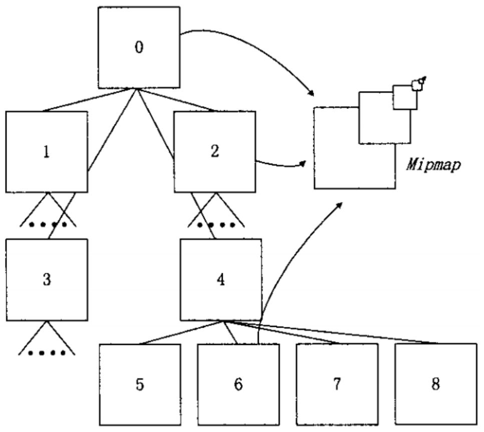


Figure 3.35: The first three levels of texture mip-map quad-tree (TMQ). Each node covers one quarter of its parent and is pre-filtered into a mip-map and compressed[37].

refinement schemes such as mip-maps. The texture hierarchy is based on 4-8 refinement of raster tiles, in which texture grids in effect rotate 45 degrees for each level of refinement (see Figure 3.36).

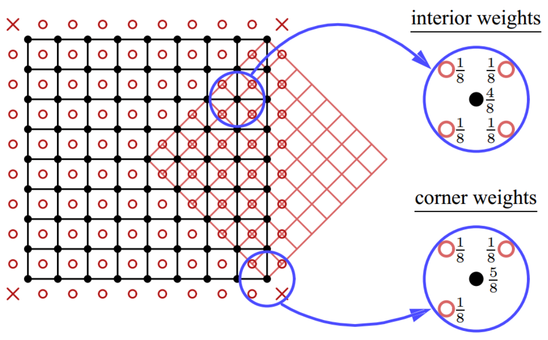

interior weights
$\frac{1}{8}$ $\frac{1}{8}$ $\frac{4}{8}$ $\frac{1}{8}$ $\frac{1}{8}$
corner weights
$\frac{1}{8}$ $\frac{1}{8}$ $\frac{5}{8}$ $\frac{1}{8}$

Figure 3.36: Low-pass filtering is performed by collecting both cell-centered values (hollow dots) and vertex-centered values (solid dots) from the four children of a tile (diamond). One child is highlighted, and weight masks for interior and corner cases (which potentially needs data from adjacent tiles) are given[38].

To generate one level coarser texture tile, half of each of its four children tiles are needed. The filtering works as follows. New vertex-centered values will be stored as new values in a coarser tile, and are recomputed using weighted averages from an old cell- and vertex-centered values obtained from its children. This low-pass per-pixel view-dependent filtering is twice as close to an ideal cut-off frequency for an average pixel. It allows rendering systems to avoid complexity and performance costs of per-pixel blending between texture levels-of-detail compared to the quad-tree based solutions.

The general approach to handle large scale texture data is virtual clipmaps method

[37]Wei Hua, Huaisheng Zhang, Yanqing Lu, Hujun Bao, and Qunsheng Peng. Huge texture mapping for real-time visualization of large-scale terrain. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, pages 154–157, New York, NY, USA, 2004. ACM.

[38]Lok M. Hwa, Mark A. Duchaineau, and Kenneth I. Joy. Adaptive 4-8 texture hierarchies. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 219–226, Washington, DC, USA, 2004. IEEE Computer Society.

presented in [134] as an extension of mip-maps. It also utilizes a factor-of-four texture pyramid and i allows arbitrarily large out-of-core textures to be handled. This algorithm utilizes the fact that the complete mip-map pyramid is rarely used during rendering a single view, allowing large textures to be used, because much of the pyramid can be clipped away. Recently, implementation of these principles is possible with programmable GPU [89] and is known as virtual textures, megatextures or sparse virtual textures.

All textures used in the scene are stored (baked) into a single large texture. The texture coordinates of all textured objects in the scene are transformed into the large texture. The large texture is further partitioned into tiles and arranged in a pyramidal structure called texture atlas, thus containing all texture mip-maps (see Figure 3.37).

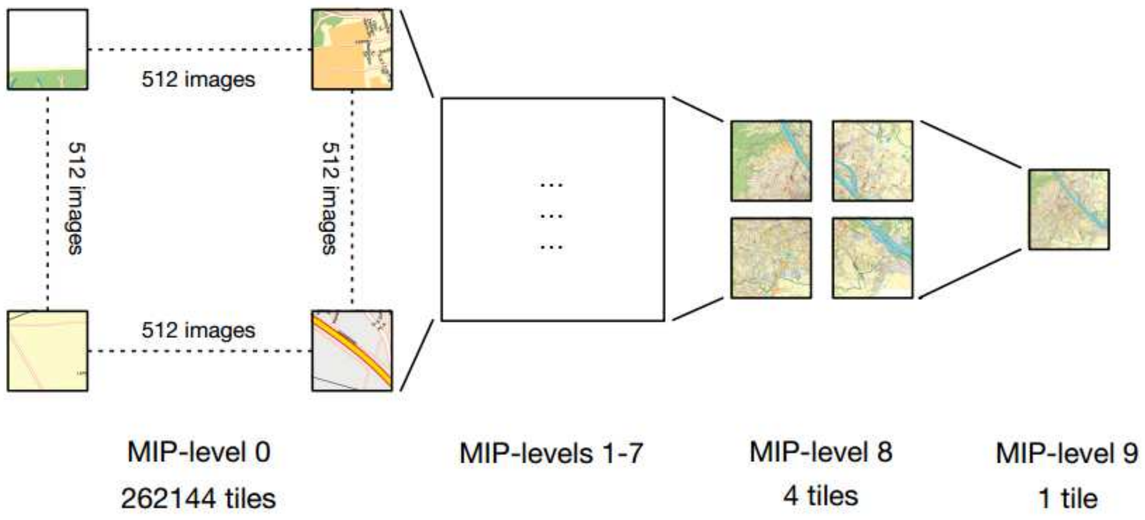


Figure 3.37: An example of texture atlas with ten mip levels of the single large virtual texture[39].

During rendering, a texture tile at appropriate level-of-detail is determined for each pixel and all the tiles needed for rendering the current frame are uploaded to a single physical texture object allocated at GPU. The texture coordinates of the underlying geometry have to be remapped from the large virtual texture space to the small physical texture on GPU (see Figure 3.38) according to address of the uploaded texture tiles in the physical texture.

The main advantages of the virtual texture algorithm are that it is no more necessary to clip the scene geometry according to the texture tiles, the consumed physical texture memory stays always fixed no matter what size of the virtual texture is, and finally it has per pixel precision.

The disadvantage is that the identifiers of missing texture tiles have to be downloaded from GPU to CPU to select and upload the requested tiles to the physical texture by CPU. The time for uploading the missing tiles to GPU is large, but this task can be done using hardware supported DXT compression for the best performance. Current high end graphics accelerator AMD Radeon 7970 supports hardware accelerated creation of texture atlases and hardware accelerated determination of missing texture tiles with its Partially Resident Textures technology [5].

[39]Albert Julian Mayer. Virtual texturing. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, October 2010.

[39]Albert Julian Mayer. Virtual texturing. Master's thesis, Institute of Computer Graphics and Algorithms,

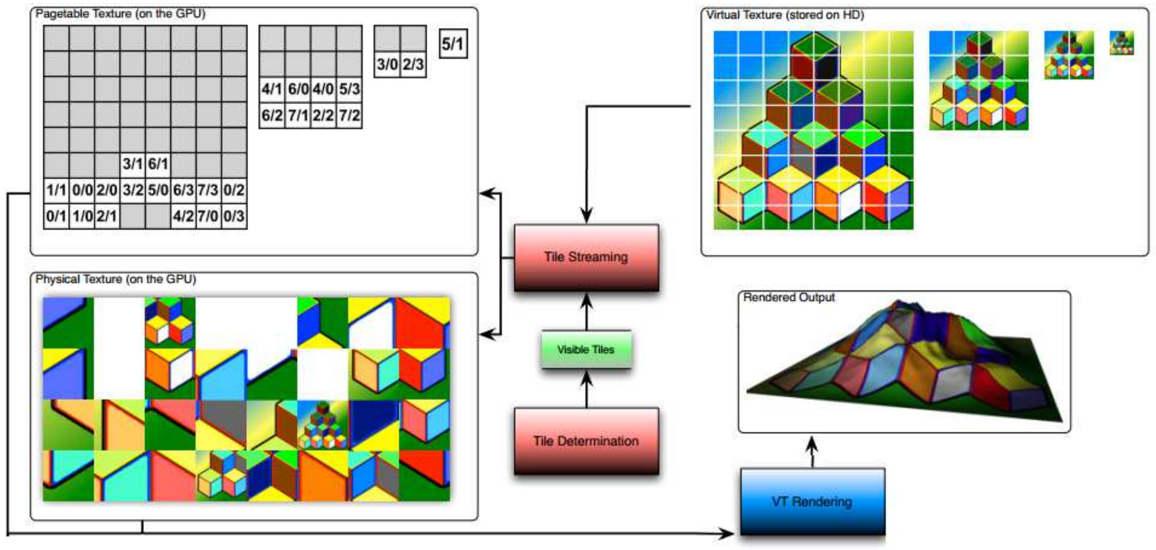


Figure 3.38: The page-table texture contains addresses of loaded virtual texture tiles in the physical texture on the GPU[39].

3.4.1 Streaming of texture data

Rendering algorithms can exactly determine the texture tiles needed to be downloaded from a remote server to clients. Handling of the large amount of textures is usually supported with quad-tree structures of texture tiles. Streaming systems request texture tiles from the remote server starting from the much coarse resolution and continue to the finer ones as they are requested by the rendering algorithm.

The main difficulty of current texture streaming systems is redundancy of transferred texture data. Consider the widely used quad-tree structures. All tiles at the finest level of the quad-tree compose together the original full resolution texture. Coarser texture tiles from higher levels of the quad-tree hierarchy contain only redundant data as they can be obtained from the finest level.

Instead of transferring particular texture mip-map levels represented by tiles from different quad-tree levels, it will be more effective to enable the progressive transferring of the texture data. Progressive transferring of textures gets more important in the case, that it is possible to download only mip-map levels needed to produce the best visual representation for given viewpoint. This can be achieved successfully e.g. with the virtual textures algorithm, which is per-pixel precise.

Many of today image compression methods propose progressive transmission mechanisms, including row interlacing (GIF), scan-based progressive encoding (JPEG),and hierarchical progressive encoding [126], [58]. The hierarchical progressive encoding schemes based on zero-tree techniques enable scalable progressive transmission based on resolution or peak signal-to-noise (PSNR). However, the resulting reduced redundancy comes at the expense of computational cost for decoding.

Another alternative is S3 texture compression (S3TC), which provides good compression ratio (6:1). It is directly supported in hardware even on old graphic accelerators. This technique is very useful to reduce system bus transfers as well as consumed GPU memory.

Unfortunately the S3TC has no progressive like feature concerning mip-maps. Moreover the compression ratio is worse than standard loose image compression techniques such as jpeg so that they are not the best choice for streaming over network.

Marvie et al. [84] presented a new progressive texture map (PTM) format that encodes mip-map levels of a texture map into a compact and progressive way which allows transferring of mip-maps level by level with minimal redundancy. The PTM consists in storing only a portion of each level, whilst the rest can be reconstructed using few additional data and previous coarser mip level (see Figure 3.39).

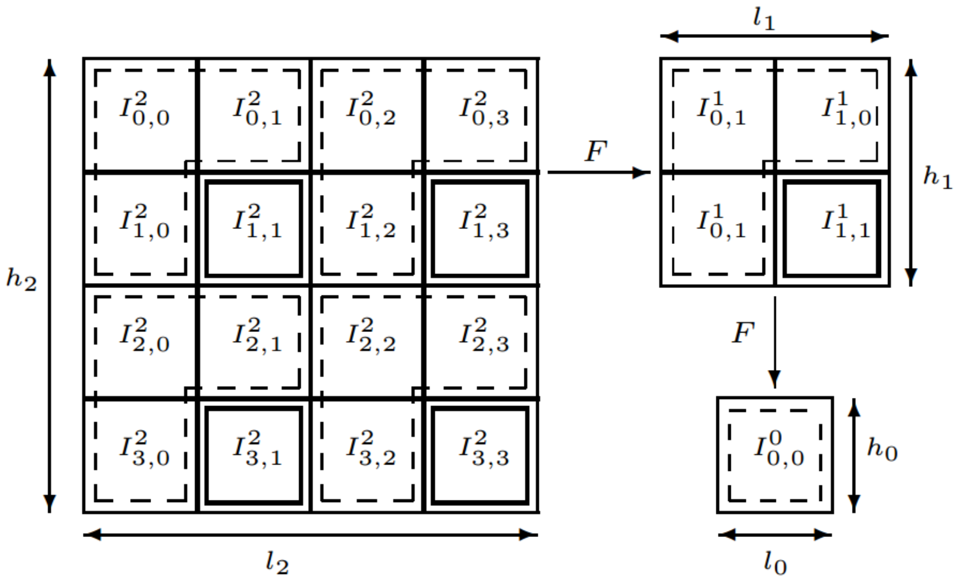


Figure 3.39: Three lower levels of a square PTM. For each level, the dotted lines surround pixels that are transmitted from server to a client (partial mip-map level) and the bold lines surround pixels that are computed on the client side. F is a low-pass filter that transforms a level n into a level $n-1$protect[40].

The low pass filter F is a 2×2 matrix applied to each colour component of the source mip-map level and is defined as follows:

$$F = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix} \tag{3.3}$$

A colour component $I^{n-1}_{i,j}$ of a texture pixel $P^{n-1}_{i,j}$ of mip-map level $n-1$ is obtained using the equation 3.4, where $I^n_{i,j}$ is intensity of colour component of pixel $P^n_{i,j}$ of the mip-map level n having a width and a height equal to l_n and h_n respectively.

$$I^{n-1}_{i,j} = \frac{1}{4} \sum_{\substack{0 \le l \le 1 \\ 0 \le m \le 1}} I^n_{i \cdot 2+l, j \cdot 2+m} \begin{cases} \forall i,j \in \mathbb{Z} \\ i < h_{n-1} \\ j < l_{n-1} \end{cases} \tag{3.4}$$

The PTM is used to stream textures over network also in [86] (see Figure 3.40).

[40]Jean Eudes Marvie and Kadi Bouatouch. Remote rendering of massively textured 3D scenes through progressive texture maps. In *The 3rd IASTED conference on Visualisation, Imaging and Image Processing*, volume 2, pages 756–761. ACTA Press, Sept 2003.

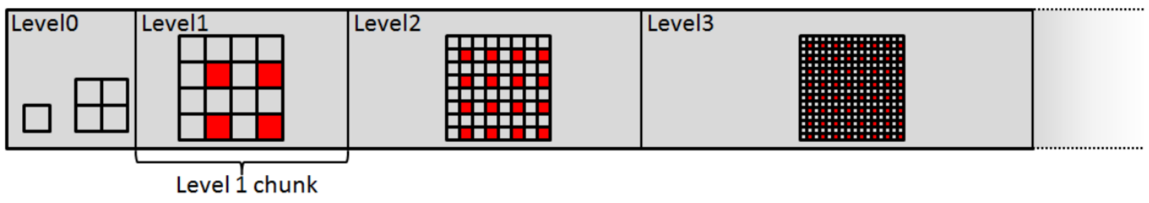


Figure 3.40: Progressive textures are encoded by chunks representing partial mip-map levels, and lossless reconstruction of missing texels (in red) at a client[41].

Small mip-map resolutions are stored in the first chunk of data, whilst the higher resolutions are stored in the next chunks. Each chunk is compressed using lossless ZIP compression method. A disadvantage of the progressive texture maps is that the lossless compression must be used, because the original uncompressed data is necessary to reconstruct the consecutive mip-map levels. Therefore, the amount of transferred data using lossy compression with sending the complete mip-map levels is similar as for the PTM, but the PTM algorithm has additional computation overhead at the client side to reconstruct the finer mip-maps.

[41] Jean-Eudes Marvie, Pascal Gautron, Pascal Lecocq, Olivier Mocquard, and François Gérard. Streaming and synchronization of multi-user worlds through http/1.1. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 111–120, New York, NY, USA, 2011. ACM.

Chapter 4

Scheduling and Prefetching Mechanisms for Walkthrough Applications

This chapter describes state-of-the-art of scheduling and data prefetching algorithms used in distributed virtual walkthrough applications. The data prefetching is an important part of complex process of scheduling data transmission over network.

The most frequently used scheduling and data prefetching algorithms are based on visibility determination (e.g. view-frustum culling, area of interest (AOI) determination, potential visible sets, etc.), motion functions, or combination of both approaches.

4.1 Visibility determination

Visibility determination algorithms can significantly reduce the overall amount of transferred data because it is not necessary to transmit the hidden geometry. The basic visibility determination algorithm widely used in walkthrough applications is well known view-frustum culling. It is based on very simple idea such as only the objects in view-frustum of a viewer will be downloaded. Other widely used visibility determination methods are Area o Interest (AOI) determination and potential visible sets (PVS) determination.

4.1.1 Area of interest

Scheduling of data transmission in distributed virtual walkthrough applications is widely solved using AOI determination algorithms. The AOI is defined as a circular area around a viewer [123], [52], [111], [146], [27].

A complex strategy for rendering in distributed virtual environments is presented in [123]. Here, several techniques including level-of-detail, progressive refinement and graceful degradation are combined together to deliver the requested data just in time over network to the rendering process.

Instead of downloading a complete scene content, it is sufficient to transfer only the data in a spherical area around a user which represents its AOI (see Figure 4.1). The fundamental idea is to consider static levels-of-detail as a unit of network transmission. Only those levels-of-detail of objects that are needed for rendering needs to be available

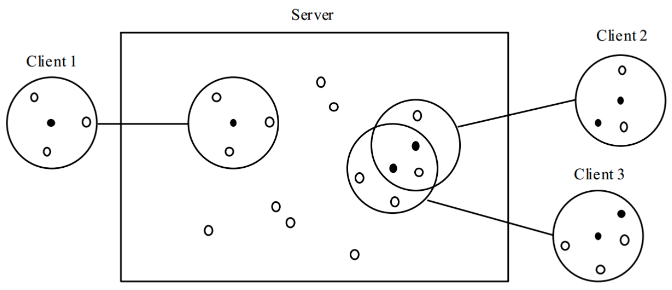
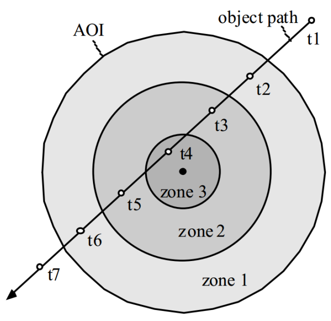


Figure 4.1: A distributed geometry database - a server stores geographically dispersed objects (small white circles). View of each client (user) is limited to an AOI (large circles). If the AOIs overlap, clients can see each other (small black circles)[42].

locally in the client cache. As the user moves the selection of LODs changes.

To compensate delay introduced by transferring the missing LODs, a prefetching mechanism is used such as when LOD selection algorithm selects certain level-of-detail, the prefetching algorithm requests the next finer level instead (see Figure 4.2).



time	request	display
t1	LOD1	-
t2	LOD2	LOD1
t3	LOD3	LOD2
t4	-	LOD3
t5	-	LOD2
t6	-	LOD1
t7	-	-

Figure 4.2: A prefetching strategy. As an object moves through the AOI (see arrow), it traverses multiple zones indicating which LOD of the object is displayed. The next finer LOD is always requested one zone in advance to compensate network delay[42].

Since the objects approached will be displayed with increasing resolution, the distance of them from the viewer must be sufficient regarding the network latency. If the prefetching algorithm fails or until a requested LOD is received, an available coarser LOD is displayed instead. This strategy assumes high frame to frame coherence. The prefetching algorithm also fails if the viewer or objects are moving too fast so an appropriate LOD cannot be downloaded in time.

Chim et al. [27] introduced a storage caching and prefetching mechanism which allows virtual objects from a remote database server to be cached in a local storage of a client at various resolutions. The database sever contains objects encoded as progressive meshes and are streamed to the local storage on clients on demand. Each object in a scene has its own object scope. The object scope is similar to the AOI. Each object scope defines an area in which the object will be visible for a viewer (see Figure 4.3).

The viewer has also its own scope called viewer scope. Practically, it defines the viewer's depth of sight. The viewer can see an object only when their scopes intersects. To determine

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

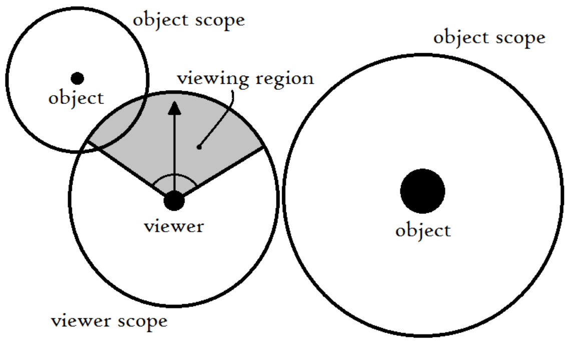


Figure 4.3: Organization of the virtual world[43].

an appropriate resolution of visible objects for rendering, the authors propose a new function for optimal resolution estimation. Optimal resolution of an object model can be defined according to a visual importance of the object to the viewer. The visual importance is defined as a percentage of its maximum resolution, i.e. the number of progressive records, and is determined according to the object distance from the viewer and an angular distance of the object from viewing direction, i.e. line of sight (see Figure 4.4).

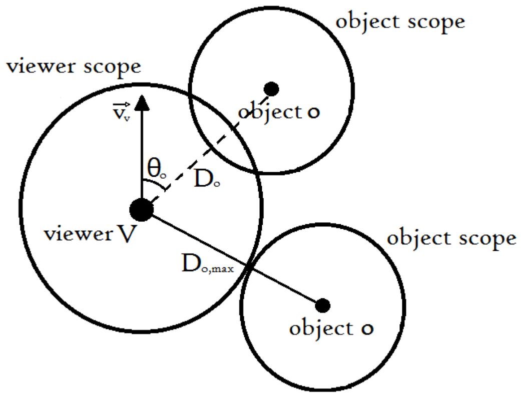


Figure 4.4: Visual importance of an object o to a viewer V in a virtual environment[43].

The visual importance of the object o can be defined with the following formula:

$$I_o = (\frac{D_{o,max} - D_o}{D_{o,max}})^2 \dot{e}^{-K_o|\theta|}, 0 \leq D_o \leq D_{o,max} \tag{4.1}$$

where $D_{o,max}$ is defined as a sum of radii of a viewer scope and an object scope, D_o indicates current distance of the object from the viewer, θ_o is the angular distance $(-\pi \leq \theta_o \leq \pi)$ of the object o from the viewer's line of sight, i.e., its viewing direction $\vec{v}_v$ and K_o is a constant for adjusting decrement rate of object o due to increase in θ_o. The optimal resolution of the object o is then defined as I_o fraction of the number of progressive records.

[43]Jimmy H. P. Chim, Mark Green, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. On caching and prefetching of virtual objects in distributed virtual environments. In *Proceedings of the sixth ACM international conference on Multimedia*, MULTIMEDIA '98, pages 171–180, New York, NY, USA, 1998. ACM.

To enable prefetching, the server maintains a separate profile for each viewer V, containing a set of historical movement vectors $\{\overrightarrow{m}_1, \overrightarrow{m}_2, ..., \overrightarrow{m}_{n-1}\}$. When the viewer V moves to a new position, the new movement vector m_n is calculated. Each vector is calculated from corresponding viewer position and orientation, containing the moving direction and the moving distance.

The server attempts to predict th next movement vector and transmits such progressive records that would be needed if the viewer V will be at the predicted position. These predicted progressive records are transferred in addition to the needed records for renderable objects at the current position. Three different schemes are proposed to compute weights of the members of the set of historical movement vectors: mean, window, and exponential weighted moving average (EWMA). For details see the section 4.2.

Different AOI method was proposed by Hesina at al. [52]. They optimally utilized network bandwidth by downloading only the exact portion of geometry which is necessary for rendering the current view. Their solution is based on progressive geometry data structures (see Figure 4.5) and on the observation such that relative importance of each object is proportional to its size in final image [42].

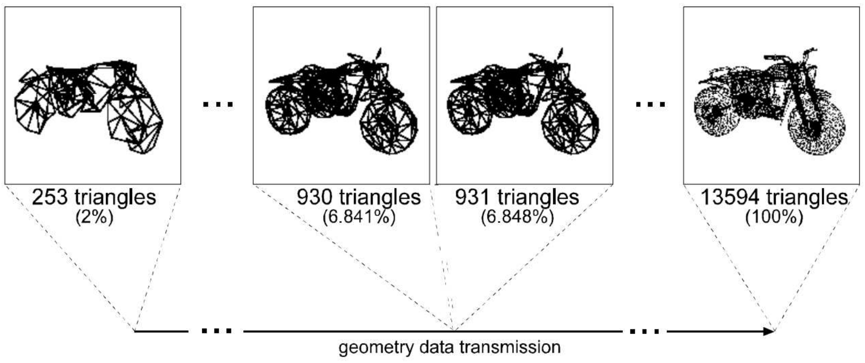


Figure 4.5: Smooth levels-of-detail allow progressive transmission of an object and display it at desired resolution[44].

The importance of given object can be expressed by the number of polygons with the following formula:

$$p(x) = \frac{A(x)^{\alpha} \cdot P}{\sum\limits_{i} A(i)^{\alpha}} \tag{4.2}$$

where $p(x)$ is the selected number of polygons for the object $O(x)$; P is the number of polygons determined by rendering algorithm to preserve rendering cost of the current frame with the given set of objects; $A(i)$ is the screen-size of the object $O(i)$ and α is the correction factor ($0 < \alpha \leq 1$), which express the fact that visual quality does not linearly depend on the number of polygons.

As each client must know in advance which objects or which parts of them should be downloaded, a larger AOI radius is considered for prefetching. For each object in the prefetching AOI, the amount of geometry that should be downloaded must be determined. The prefetching algorithm assumes that the user has travelled toward each object in the

[44]Gerd Hesina and Dieter Schmalstieg. A network architecture for remote rendering. In *Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications*, DIS-RT '98, pages 88–, Washington, DC, USA, 1998. IEEE Computer Society.

prefetching AOI by a constant distance. From an increased screen size the desired number of polygons $\overline{p}(x)$ of each object is computed as in previous formula:

$$\overline{p}(x) = \frac{\overline{A}(x)^{\alpha} \cdot P}{\sum_i A(i)^{\alpha}} \tag{4.3}$$

where $\overline{p}(x)$ is the desired number of polygons for the object $O(x)$ and $\overline{A}(i)$ is the increased screen-size of $O(i)$.

The amount of geometry which should be downloaded for each object can be calculated as a difference between the desired and the available number of polygons and is further weighted by available bandwidth expressed as the number of polygons downloaded per time unit:

$$DL(x) = \frac{(\overline{p}(x) - avail(x)) \cdot \overline{P}}{\sum_i \overline{p}(i) - avail(i))} \tag{4.4}$$

where $DL(x)$ is the number of polygons to be downloaded for the object $O(x)$, $avail(x)$ is the number of locally available polygons for the object $O(x)$ and $\overline{P}$ is the polygon budget per time unit.

Park et al. [104] assumed that considering the possibility of rapid rotation of user movement, ignoring angular distance is more reasonable for determining download priority of objects. Hence, they distinguish between walking mode (expecting rapid rotations) and moving mode.

Moreover, they proposed user-based prefetching exploiting object access priority generated from spatial distance and individual user interests, because the spatial relationship fails to determine which types of objects are more important for the individual users. By combining the interest score and the popularity score of each object with spatial relationship, the prefetching accuracy can be increased. Access priority of the object O by the user A is determined as follows:

$$AP(A, O) = \alpha DS(A, O) + (1 - \alpha)[\beta IS(A, O) + (1 - \beta)PS(O, W)] \tag{4.5}$$

where DS(A,O), IS(A,O) and PS(A,W) represent the distance score, the interest score and the popularity score respectively. The α and β are modulation parameters, where α controls degree of spatial relationship in the calculation of access priority of the objects, and β determines which one contributes more to access priority between the interest and the popularity. The prefetching is based on considering larger area of interest around the user as in [52].

An architecture for just-in-time data replication in network virtual environments focused on quality of service is presented in [111]. Data replication is controlled as in a pull network model, where a data flow is controlled by clients. This requires, that each client has acquired the description of the virtual environment before any navigation process starts.

The scene-graph is composed of nodes which are defined as a pairs of $geometrynodeURL$ and $sizeofgeometryfile$. Download time of each node can be estimated based on its known data size and on available bandwidth. This estimation model and user navigation are combined together so they define the prediction model for achieving requested QoS. Prefetch manager keeps a table with time penalties (calculated from estimated client network bandwidth) for all not downloaded nodes in user AOI. Final size of the user's AOI is computed so that the largest (in terms of data size) object in the scene can be downloaded in time for being rendered.

Two methods are used to estimate the time $T_{visible}$ when geometry of a node becomes visible. The first method uses magnitude of a viewpoint speed vector while the second method takes into consideration direction of a viewpoint motion:

$$T_{download} = \frac{size(object)}{bandwidth} \tag{4.6}$$

$$T_{buffer} = T_{response} + T_{param} \tag{4.7}$$

$$I.\ T_{visible} = \frac{(\|\overrightarrow{VC}_{obj}\| - R_{visible\_area})}{\|\overrightarrow{v}\|} \tag{4.8}$$

$$II.\ T_{visible} = \frac{(\|\overrightarrow{VC}_{obj}\| - R_{visible\_area})}{\langle \overrightarrow{v}, \frac{\overrightarrow{VC}_{obj}}{\|\overrightarrow{VC}_{obj}\|} \rangle} \tag{4.9}$$

where $\overrightarrow{v}$ is speed of the viewpoint and $\|\overrightarrow{VC}\|$ is viewpoint-object vector.

Speed estimation used in the second $T_{visible}$ equation employs a moving average predictor, which is described in detail in the section 4.2. The advantage of the second method is that the prefetching is more optimized thus reduce wasting of bandwidth. The buffering time T_{buffer} is composed of response time of server and a parameter proportional to standard deviation of instant traffic measurements.

Replication policy is defined so that whenever a time to get to a node is less than its download time plus buffer time (accounting for variation in connection bandwidth), a geometry will be requested (see Figure 4.6):

$$T_{visible} < T_{download} + T_{buffer} \tag{4.10}$$

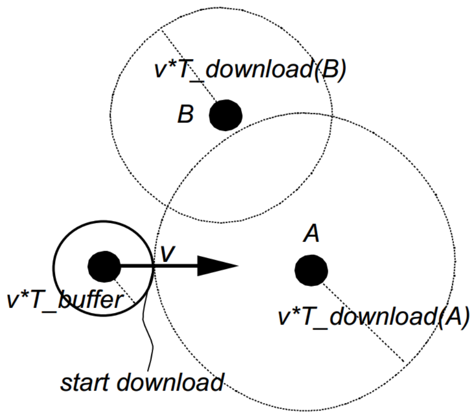


Figure 4.6: Prediction for just-in-time data replication. The black dots are visible areas of objects A, B and of the viewer[45].

Figure 4.6 illustrates download mechanism for the first replication method. The replication condition was represented by circles proportional to the time computed using the

[45]G. Popescu. On scheduling 3d model transmission in network virtual environments. In *Proceedings of the Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, DS-RT '02, pages 127–, Washington, DC, USA, 2002. IEEE Computer Society.

equations 4.6 to 4.8. The download starts when the circle A or B overlap the viewpoint circle.

Object transferring priority determined from its position in an AOI is presented in [75]. Here, the important objects are the ones whose scopes intersect with the viewer scope. The viewer scope is divided to three regions with different transferring priority of objects inside them as is shown in Figure 4.7a.

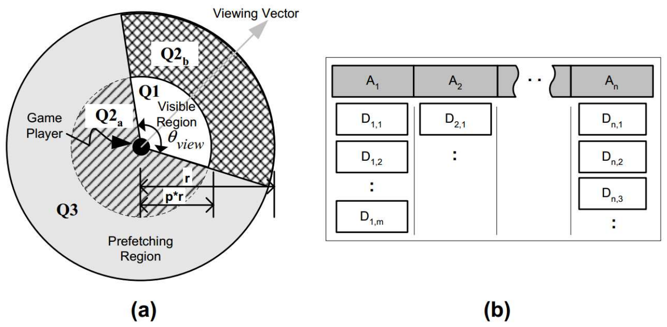


Figure 4.7: (a) the viewer scope partitioning and (b) prioritized content delivery queues[46].

The region Q_1 is visible region with highest download priority. Q_2 is potential visible region, composed of Q_{2a} and Q_{2b}. All objects inside Q_2 are not actually visible to a viewer, but will become visible if the viewer moves forward or turns around. Objects from this region can be transferred to clients right after all objects from region Q_1 are already transferred. Finally, Q_3 is a prefetching region such that objects inside it can be pre-fetched only if all objects from the previous regions are already downloaded.

The content delivery queues shown in Figure 4.7b are determined as follows. Objects inside each queue are first sorted by their angular distances A_n from viewer line of sight and then, they are sorted by their distances $D_{n,m}$ from the line of sight. Scheduling algorithm here first selects a query from the left-most vertical sub-queue (lowest A_n) to the right-most one (highest A_n), one by one. Objects from each sub-queue are selected to transfer starting from the top to bottom.

In [146] a new prefetching scheme called scalable multi-layer area of interest (SMLAOI)is proposed to solve the case the number of objects covered by a viewer AOI becomes too large to be downloaded satisfactorily. This scheme divides the area of interest around the viewer to several concentric circles. Each circle is divided into two sectors, the first Q_a with 120 degrees and the another one Q_b with 240 degrees (see Figure 4.8).

The scene management algorithm divides the whole scene into a grid of 2D cells and determines a list of overlapped virtual objects for each cell. The virtual objects are progressively represented and are described using the VRML/X3D extensions as is described in [40]. Such representation of virtual world and its objects is suitable for the SMLAOI algorithm, because the download priority of content of all cells can be set as $Q_{1a} > Q_{1b} > Q_{2a} > Q_{2b} > ... > Q_{na} > Q_{nb}$, where n is the number of circles (layers).

[46]Frederick W. B. Li, Rynson W. H. Lau, and Danny Kilis. Gameod: an internet based game-on-demand framework. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, pages 129–136, New York, NY, USA, 2004. ACM.

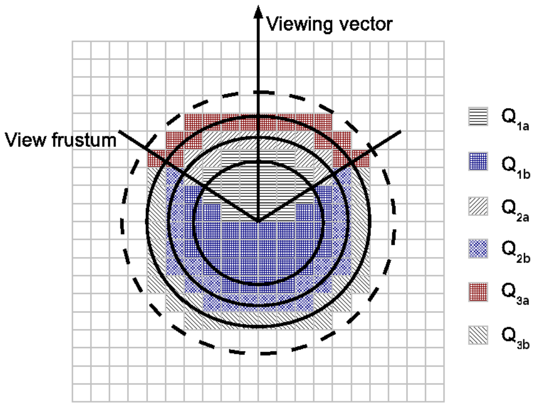


Figure 4.8: The SMLAOI concept divides the AOI to parts with different download priority[47].

4.1.2 Potential visible sets

Marvie et al. [85] presented VRML-X3D extension describing precomputed cell-to-cell, cell-to-object and hybrid visibility relationships in context of progressive transmission. A virtual scene is composed of cells. Each cell (VRML97 build-in node named ConvexCell) is delimited by a convex volume whose boundaries are described using a set of convex polygons.

During navigation, the current cell holding a viewpoint determines the geometry which is needed to be downloaded and displayed. Each cell contains its own geometry and a list of all potentially visible cells (see Figure 4.9b). During navigation, the set of potential visible cells for the current cell is downloaded. First, the cells completely outside of a view frustum are culled from rendering based on their convex volumes. Second, the objects inside the remaining cells are also culled by the view frustum. Only the objects that are present in the view frustum are rendered.

To handle indoor scenes (cell-to-cell visibility) together with outdoor scenes such as streets with lamps (cell-to-object visibility), the authors developed a hybrid visibility approach. It has the advantage, that it can eliminate large amount of references to objects when cell-to-object visibility is used (see Figure 4.9a). The large amount of references appears because each cell references a lot of the same objects.

The proposed hybrid visibility approach divides outdoor scenes (for example streets with lamps) into cells and references these cells instead of individual objects (see Figure 4.9c).

The pre-fetching scheme consists of three steps. The first fetching step is performed when a navigation starts or when viewpoint is moved to different cells. In this step, the content of the current cell and also the information about its potential visible sets are downloaded. In the second step, content of the adjacent cells to the current cell and their potential visible sets are downloaded asynchronously. Once te content of the first adjacent cell is downloaded, the third step can start pre-fetching. The third step relies on motion prediction to find the next adjacent cell that will be visited but whose content are not yet downloaded. The motion prediction algorithm uses ray-casting to determine the next adjacent cells so that the ray is computed from the two most recent viewpoints.

[47]Wei Wang and Jinyuan Jia. An incremental smlaoi algorithm for progressive downloading large scale webvr scenes. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 55–60, New York, NY, USA, 2009. ACM.

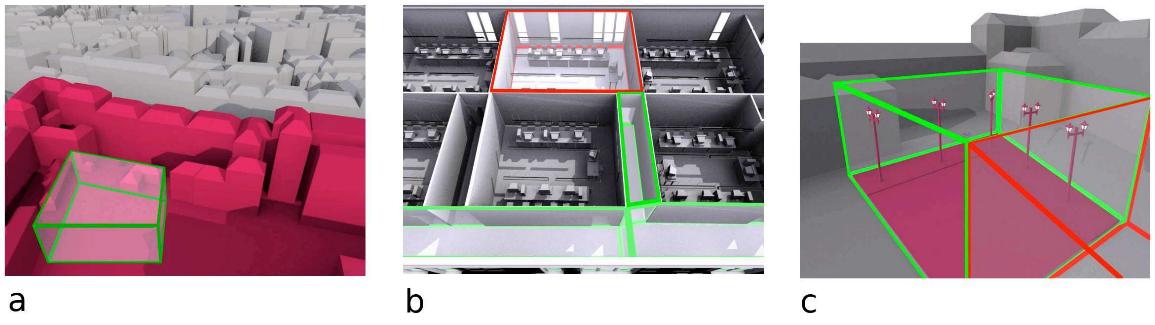


Figure 4.9: (a)cell-to-geometry relationship, the green volume is the evaluated cell and the red marked geometry represent visible objects, (b) cell-to-cell visibility relationships, the green cells are potentially visible from the red one, (c) hybrid visibility relationship, where the lamps are objects visible from the evaluated red cell and the green cell composes the potential visible cell from the red one[48].

Later, Marvie et al. in [86] proposed a complex planetary rendering system. They used the previous hybrid visibility algorithm [85], and data streaming supported by various methods for each layer of the planet (see Figure 4.10). The scheduling and prefetching

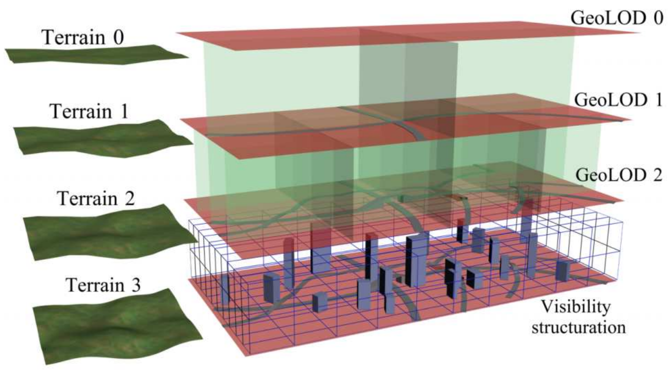


Figure 4.10: Altitude of the viewpoint drives management of the levels-of-detail. At ground level a more optimized representation of the world based on visibility relationships between cells of a uniform grid is used. The terrain itself is managed separately using dedicated streaming techniques based on HTTP/1.1 chunk reads[49].

mechanisms here are controlled by visibility determination between the scene nodes equally as is described in [85].

A method, where the scene is represented using a graph, where the nodes are view-cells (parts of streets) and te edges denote adjacencies among these cells is proposed in [20] (see Figure 4.11). The pre-fetching algorithm is transformed into graph partitioning problem where the partitions might overlap with each other.

[48]Jean-Eudes Marvie and Kadi Bouatouch. A vrml97-x3d extension for massive scenery management in virtual worlds. In *Proceedings of the ninth international conference on 3D Web technology*, Web3D '04, pages 145–153, New York, NY, USA, 2004. ACM.

[49]Jean-Eudes Marvie, Pascal Gautron, Pascal Lecocq, Olivier Mocquard, and François Gérard. Streaming and synchronization of multi-user worlds through http/1.1. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 111–120, New York, NY, USA, 2011. ACM.

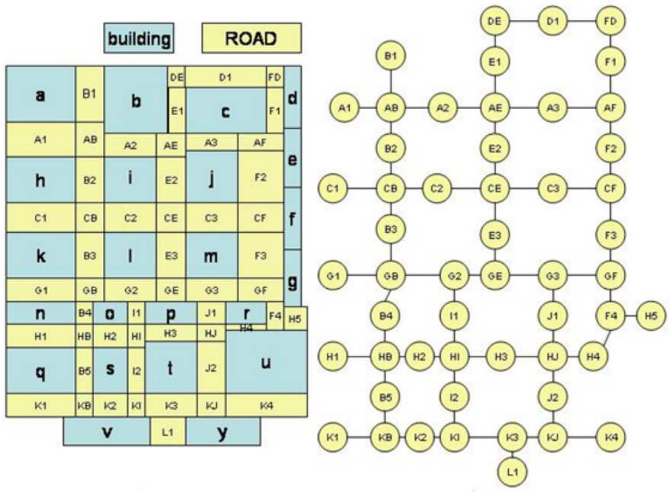


Figure 4.11: View cells (parts of streets) to graph mappping[50].

The method they proposed is developed for virtual walk through urban environments, where buildings are occluders so that a viewer navigating along the streets may only see only a small part of the whole scene. The graph is partitioned into prefetch groups using two following criteria: high probability of transitions from the source and destination nodes into the same group, and nodes in which the viewer stays for short time should pull their adjacent nodes into the same group. The authors experimented with simulated annealing method to minimize the maximum transition cost or minimize the popular path cost. The experiments are performed only with synthetic input data sets. Therefore, practical application of this method has not been proven. It is also difficult to develop an efficient heuristic so the global optimization can help local optimization or vice versa.

Data prefetching for image based rendering systems on mobile platforms is proposed in [13]. Here, a scheduling and buffering mechanism for on-time image delivery is specified for remote virtual environment exploration applications. The main idea of the buffering scheme here is to have all the possible viewpoints that the user can visit in next step available already in client cache memory (see Figure 4.12).

The server decides which viewpoints will be requested by the rendering engine based on the last received position updates. The server application also determines the priority of all possible viewpoints. The viewpoint with the highest priority will be used to render the scene on the server. The scheduling mechanism uses a fixed look ahead (LA) approach which requests images that may be needed in LA steps in advance. Each step is defined as the time it takes to move from one viewpoint to another. The scheduling mechanism at the server must guarantee, that the scene image rendered on the server will arrive to the client before its deadline. The deadline is the time when an image is no more needed for rendering. The server must keep the following equation valid:

$$T_s \leq D - (T_r + T_c + Ti) \tag{4.11}$$

where T_s is the time the request stay in the scheduler queue at server, T_r is the time to

[50]Safak Burak Cevikbas, Gürkan Koldas, and Veysi Isler. Prefetching optimization for distributed urban environments. In *Proceedings of the 2008 International Conference on Cyberworlds*, CW '08, pages 291–297, Washington, DC, USA, 2008. IEEE Computer Society.

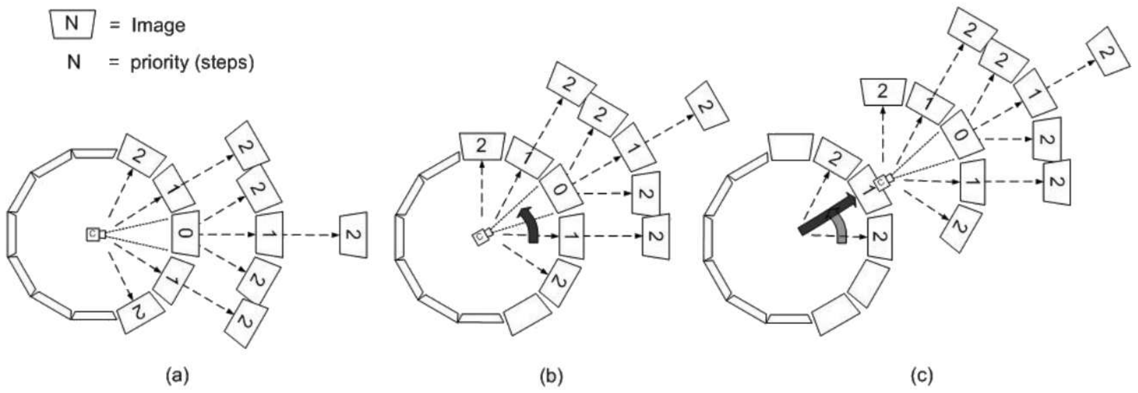


Figure 4.12: (a) Shows all of possible user movements and steps necessary to reach the images. Both client and server maintain an updated map such as the one in the picture. (b) The user rotated the viewpoint to the left. Observe that priorities assigned to each image have changed. These changes must be reflected in the scheduler queue by updating deadlines of each image. (c) In this picture, the user moved forward after a rotation to the left. Again, priorities have been modified[51].

render one frame at server, T_c is the time to prepare warping and compression at server and T_i is time in what the image arrives to the client.

4.2 Motion functions

In general, motion prediction methods can be roughly classified into the short time and the long time prediction methods. The short time prediction methods so called motion functions can predict the next motion of a viewer based on its current motion parameters. That is why these methods have accurate results when predicting locations close to the viewer.

As was described in the section 4.1.1, Chim et al. [27] introduced the prefetching method for DVE based on the knowledge of he set of weighted movement vectors $\{\overrightarrow{m}_1, \overrightarrow{m}_2, ..., \overrightarrow{m}_{n-1}\}$. When the viewer V moves to a new position, the new movement vector m_n is calculated. Three different schemes was proposed to compute the weights of the members of the set of historical movement vectors: mean, window, and exponential weighted moving average (EWMA), see Figure 4.13.

In the mean scheme, the next movement vector m_{n+1} is predicted as the average of previous n movement vectors. The predicted vector is computed as $\hat{m}_{n+1} = \frac{1}{n}\sum_{i=1}^{n}\overrightarrow{m}_i$. The intermediate vectors are not necessarily needed for the prediction, but only the running sum of suffices $\hat{m}_{n+1} = \frac{(n-1)\hat{m}_n + \overrightarrow{m}_n}{n}$, where n is the number of movement vectors and $\hat{m}_n$ is previously predicted vector.

In the window scheme, each viewer is associated with a window of size W. The next movement vector is predicted as the average of W most recent movement vectors. The mean scheme can be used to predict the next movement vector only if $n < W$, otherwise

[51]Azzedine Boukerche and Richard W. Nelem Pazzi. Scheduling and buffering mechanisms for remote rendering streaming in virtual walkthrough class of applications.In *Proceedings of the 2nd ACM international workshop on Wireless multimedia networking and performance modeling*, WMuNeP '06, pages 53–60, New York, NY, USA, 2006. ACM.

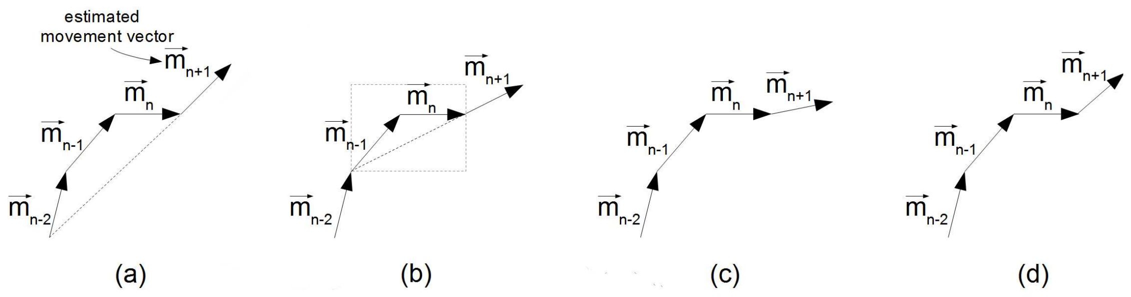


Figure 4.13: Prediction of next moving direction: (a) mean, (b) window, (c) EWMA with low α, (d) EWMA with high α[52]

the predicted vector will be computed as $\hat{m}_{n+1} = \hat{m}_n + \frac{1}{W}(\vec{m}_n - \vec{m}_{n-W})$.

To avoid the need of the moving window, and to fast adapt to changes in the viewer motion, the EWMA scheme assigns weight α to each previous movement vector so the recent vectors have higher weights. The new parameter α represents exponentially decreasing weight (the most recent vector has the weight of 1, the previous vector has the weight of α, the next previous vector has the weight of α^2 and so on). The larger the alpha is, the more the previous vectors affect prediction of the next movement vector:

$$\hat{m}_{n+1} = \frac{1}{S_n}\sum\nolimits_{i=1}^{n} \alpha^{n-i}\vec{m}_i \tag{4.12}$$

where $S_n = \sum_{i=1}^{n} \alpha^{n-i} = \frac{1-\alpha^n}{1-\alpha}$. As $n \to \infty$, $S_n \to \frac{1}{1-\alpha}$. Therefore, the movement prediction can be approximated by $(1-\alpha)\sum_{i=1}^{n} \alpha^{n-i}\vec{m}_i$. Incrementally, $\hat{m}_{n+1}$ can be approximated by $\alpha\hat{m}_n + (1-\alpha)\hat{m}_n$.

The EWMA prediction needs to be further corrected with residuals. Each residual of a movement vector is denoted as $\vec{e}_n = \hat{m}_n - \vec{m}_n$. To catch non-stationary changes in directions (pure rotations), residual of angle between $\hat{m}_n$ and $\vec{m}_n$ is considered instead. This angle can be expressed as $\phi_n = arg(\hat{m}_n) - arg(\vec{m}_n)$, where $arg(\vec{m}_n)$ is argument of the vector $\vec{m}_n$ in a complex plane so that $\vec{m}_n$ can be predicted by rotating $\hat{m}_n$ through an angle of $-\phi_n$. This corresponds to a multiplication by $e^{-i\phi_n}$ in the complex plane.

Since the $\hat{m}_{n+1}$ is predicted, the ϕ_{n+1} must be predicted as well. Thus, $\phi_{n+1} = \alpha\phi_n + (1-\alpha)\phi_n$, and $\vec{m}_{n+1} = \hat{m}_{n+1}e^{-i\phi_{n+1}}$.

A comprehensive solution for virtual walkthrough applications which includes progressive multi-resolution caching of a mesh geometry and the prefetching technique based on prediction of a viewer movements such as in [27] is proposed in [26]. The caching mechanism here tries to maintain at least a minimum resolution of a scene to provide at least coarse scene rendering.

Teler [139] presented a new 3D scene streaming approach for remote walkthroughs where a user on a client machine interactively navigates through the scene which resides on a remote server. The proposed streaming algorithm selects object representation to send based on an integral of a benefit measure along a predicted path of movement. The movement prediction is done by the server and is based on the assumption that once the

[52]Jimmy H. P. Chim, Mark Green, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. On caching and prefetching of virtual objects in distributed virtual environments. In *Proceedings of the sixth ACM international conference on Multimedia*, MULTIMEDIA '98, pages 171–180, New York, NY, USA, 1998. ACM.

user started a particular type of motion, she is likely to continue it in near future. This approach does not consider any previous positions of the user.

Scheduling of 3D model transmission in network virtual environments is studied in [110]. The virtual environment here is composed of 3D geometry objects encoded as progressive meshes. Both a viewer and the objects have defined their own view scopes as in [27] (see the section 4.1). The server receives updates of the viewpoint position and schedules transferring of missing geometry for the current view area. The proposed streaming algorithm operates with a visual error of the objects rendered at the client to optimize their transferring. The authors evaluated quality of the visual perception of each object using accuracy of its 3D representation and its importance in the rendered scene with the following formula:

$$Err_{obj} = \frac{(1 - \frac{S_c}{S_{max}})^3 (1 - \frac{D_{obj}}{R_{AOI}}) R_{obj}}{\sum_{k=1}^{N} (1 - \frac{D_k}{R_{AOI}}) R_k} \tag{4.13}$$

for $D_{ojb} < R_{AOI}$, and where S_c is the size of the object in bytes at current resolution, S_{max} is the total size of the object, R_{AOI} is the radius of the viewing area, R_{obj} is the radius of the object's bounding sphere, D_{oj} is the distance between the viewpoint and the object and N is the number of objects in the viewing area. The importance of all objects depends on distance between them and the viewpoint.

Two scheduling policies was proposed to evaluate the visual quality. First, a greedy sub-optimal scheduling policy is selected for transferring the data blocks which improves most the visual quality at the client at each time step. The second policy uses a viewpoint motion model similar to the well known random way point, which is widely used model for mobility management in wireless communications networks. This model assumes that a user moves at constant speed, with a direction angle determined as follows: a direction remains the same with probability pb, or is picked randomly from interval $(0, \phi_{max}]$ with probability $(1 - pb)$. The probability can be expressed as $pb = \frac{speed}{k}$, where *speed* is speed of the viewpoint and k is parameter controlling randomness of the motion.

A client-server architecture, where clients send at regular time intervals a position update messages to a server is proposed in [73]. The server renders the scene according to the updated positions from a client and sends rendered image of the scene back to the client. Two prefetching schemes are proposed by the authors within this architecture. The first approach is a locality-based technique. It assumes that the position of the user changes only a little in comparison with possible changes in orientation. In this case, the server renders and sends the rendered images with larger field of view (FOV) to minimize the impact of network restrictions. The second prefetching approach is based on a motion prediction such the server predicts next user motion based on its motion history. The rendered scene image is send along a predicted path. The prediction itself is based on the linear average window technique [27].

The localization-based approach with larger FOV increases network traffic, but guarantees a much higher hit-ratio for certain movement patterns. The experiments show, that the linear motion prediction almost always outperforms the localization-based approach in terms of hit-ratio for circular movement patterns, random walk pattern, rotation ratio or step size changes.

A motion-aware approach for efficient evaluation of continuous queries on 3D object databases is described in [2]. Multi-resolution representation of 3D objects utilizes wavelet based approach, where details are encoded by wavelet coefficients with higher values. These coefficients are retrieved from the server based on „speed to resolution“ mapping function.

This function simply maps higher wavelet coefficients to fast moving objects and conversely maps both low and hight coefficients for slow moving objects. As a user moves, new download queries are sent to the data server. The server sends only the data which is not already retrieved by the previous queries (see Figure 4.14).

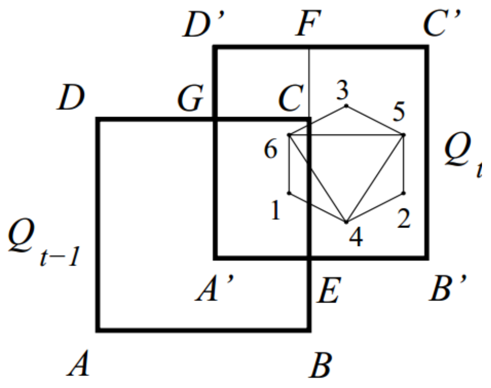


Figure 4.14: Continuous data retrieval initiated by queries Q_{t-1} and Q_t[53].

Each query retrieves all the vertices that fall inside its frame and also all the vertices from the neighbour frames which are connected with the query frame. For example, for the query frame Q_{t-1}, the client retrieves vertices $\{1, 6\}$ which are inside it and the neighbour vertices $\{3, 4, 5\}$ that connect to 1 and or 6.

To overcome the high retrieval latency caused by communication between the client and the server for each query frame, the authors propose a predictive buffer management scheme. The proposed buffer management scheme uses a state estimation based on motion-prediction to determine probability distribution of data (wavelet coefficients) to be accessed. State s_t of the moving client at time t is defined by positions of the client from h most recent timestamps, i.e. $s_t = [p(t), p(t-1), ..., p(t-n)]^T$, where p(t) is a position vector. Prediction of the future state at time $t+1$ is $s_{t+1} = As_t$, where A is a transition matrix, called one step predictor.

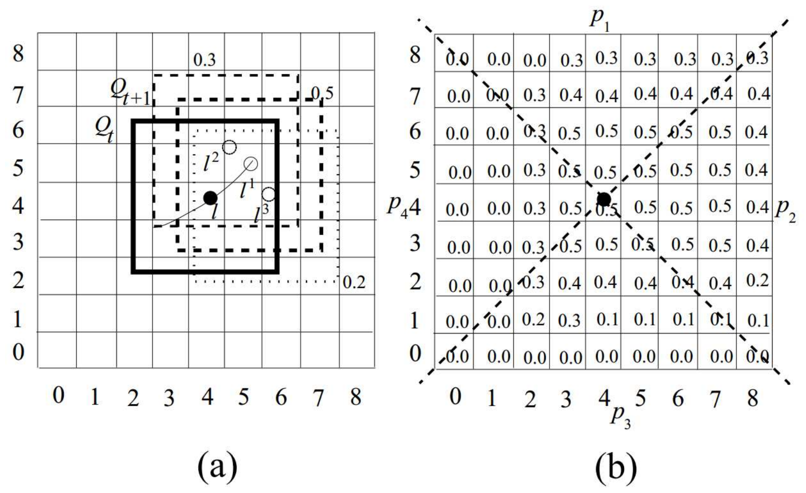
The transition matrix can be used also for multi-step predictions - the predicted state at time $t+i$ can be calculated as $s_{t+i} = A^i s_t$. The transition matrix can be calculated by recursive motion function (described hereinafter) which supports linear and a broad class of non-linear motion patterns. The total space is divided into grid cells and the probability is evaluated for each cell of that space.

The prediction does not provide any confidence estimation of the prediction results. Hence, after predicting a future state, expected confidence of the predicted state is estimated by calculating error covariances of that state. If $\hat{s}_t$ is the predicted state and s_t be the true state, then error of the prediction is $e_t = s_t - \hat{s}_t$. Similarly, prediction error of the state $t+i$ is $e_{t+i} = A^i(s_t - \hat{s}_t)$. From this, a covariance matrix P_t, which is a measure of uncertainty of the predicted state $\hat{s}_t$ can be defined as $P_t = E[e_t e^T]$. Then, probability of predicted state $\hat{s}_t$ is estimated and conditioned on all prior estimates. Hence, the probability of a state can be estimated using a normal probability distribution as follows:

$$P(s_t) \sim N(E(s_t), P_t) = N(\hat{s}_t, P_t) \tag{4.14}$$

[53]Mohammed Eunus Ali, Egemen Tanin, Rui Zhang, and Lars Kulik. A motion-aware approach for efficient evaluation of continuous queries on 3d object databases. *The VLDB Journal*, 19(5):603–632, October 2010.

where the predicted value $\hat{s_t}$ is mean of the probability distribution and a variance for this prediction is obtained from singular values of P_t. The total data space is divided into grid cells. Probability of visiting these cells is calculated rather than determining probability for each point of the space (see Figure 4.15).



	0	1	2	3	4	5	6	7	8
8	0.0	0.0	0.0	0.3	0.3	0.3	0.3	0.3	0.3
7	0.0	0.0	0.3	0.4	0.4	0.4	0.4	0.4	0.4
6	0.0	0.0	0.3	0.5	0.5	0.5	0.5	0.5	0.4
5	0.0	0.0	0.3	0.5	0.5	0.5	0.5	0.5	0.4
4	0.0	0.0	0.3	0.5	0.5	0.5	0.5	0.5	0.4
3	0.0	0.0	0.3	0.5	0.5	0.5	0.5	0.5	0.4
2	0.0	0.0	0.3	0.4	0.4	0.4	0.4	0.4	0.2
1	0.0	0.0	0.2	0.3	0.1	0.1	0.1	0.1	0.1
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4.15: (a) Motion prediction for the next time-stamp and (b) probabilities of visiting different cells[53].

Figure 4.15a shows current position l and query frame Q_t at time t and the next query frame Q_{t+1} predicted at l^1, l^2, and l^3, with probabilities 0.5, 0.3, and 0.2, respectively. Similarly, next queries with future timestamps such as Q_{t+2} can be used to calculate the probabilities of visiting them.

Finally, probability of following the four directions (determined by the cells inside the two dotted lines - see Figure 4.15b) can be obtained by summing up all the cell probabilities related to those specific directions and normalizing them. Computed probabilities for the different directions are used to determine a data block which will be transmitted to the client's buffer.

4.2.1 Indexing and managing of moving objects

A lot of methods for indexing and managing information of objects moving in one, two or three dimensional spaces in spatio-temporal databases exists [145], [72], [136], [2], [24]. These methods are important for several emerging applications including traffic supervision, flight control, mobile computing, etc.

Assuming conventional databases, data remains constant unless it is explicitly modified. Capturing continuous movement with such a behaviour would entail performing very frequent updates or recording outdated position thus leading to inaccurate data. A solution to this problem is, rather than storing the updated positions, to store functions of time that express the positions of the objects. Then updates are necessary only when parameters of the stored functions have been changed [145]. Additionally, the current motion parameters such as position and velocity can be stored together with the motion function.

[53]Mohammed Eunus Ali, Egemen Tanin, Rui Zhang, and Lars Kulik. A motion-aware approach for efficient evaluation of continuous queries on 3d object databases. *The VLDB Journal*, 19(5):603–632, October 2010.

The database system can deduce the object position for all queries involving time t. The database uses stored motion function, motion parameters stored at the same time as the motion function and current motion parameters of given object to predict its position.

Widely used spatio-temporal prediction schemes assume only linear movement [145], [72]. Specifically, $o(t) = o(t_o) + v_o(t - t_o)$, where t_o is last time-stamp when object o issued an update, $o(t_o)$ is location of object o at time t_o, and v_o denotes its current velocity (constant since t_o). Both o and v_o are d-dimensional vectors (where d is dimensionality of input data space) since they capture the information of o on all axes. An update is necessary whenever v_o (i.e. speed or direction of motion) changes.

Unfortunately, the motion of objects is not always linear and piece-wise linear segments cannot be effectively applied for prediction (see Figure 4.16). The non-linear model [1] can be more accurate compared to the linear ones in some cases. In [1] a quadratic function is used to describe the object motion as follows:

$$o(t) = o(t_o) + v_o(t - t_o) + \frac{a_o}{2}(t - t_o)^2 \tag{4.15}$$

where a_o is acceleration vector of the object o. Although this model captures linearity as a special case (and therefore has higher applicability), it cannot represent the motion curve shown in Figure 4.16.

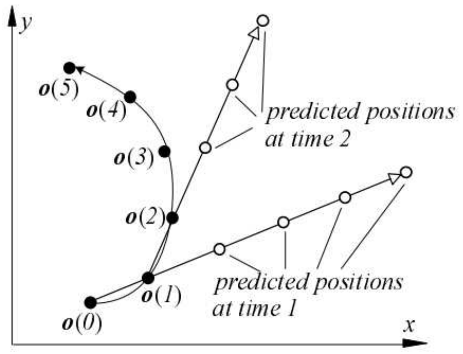


Figure 4.16: Failure of linear prediction (white dots)[54].

Figure 4.16 shows that the linear prediction of the object o moving along a curvature during 6 timestamps is very imprecise. Consider a prediction query issued at the time 1 which asks for locations of the object o at next four timestamps. The linear prediction model uses the position $o(0)$ and $o(1)$ to predict the four next positions (white dots). The predicted positions deviate from the real ones significantly. Similar observations hold for a predictive query issued at time-stamp 2, where prediction is based on positions $o(1)$ and $o(2)$.

Prediction of moving objects with unknown motion patterns is presented in [136]. Here, a new recursive motion function which supports a broad class of non-linear motion patterns is proposed. The function does not presume any a-priori movement model but can predict motion of each object by examining its recent locations.

Let o be the object whose motion type is unknown. Given current locations of the object o at h most recent timestamps, the objective is to derive a motion function which

[54]Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 611–622, New York, NY, USA, 2004. ACM.

correctly captures all these locations and predicts future trajectory of the object o.

The proposed recursive motion function is defined as follows:

$$o(t) = C_1 \cdot o(t-1) + C_2 \cdot o(t-2) + ... + C_f \cdot o(t-f) \tag{4.16}$$

where $C_i(1 \leq i \leq f)$ is a $d \times d$ constant matrix, $o(t)$ is location of object o at time t, and f is a system parameter called retrospect. This equation can express an extensive number of simple and complex movement types (by varying C_i), including polynomials, ellipses, sinusoids, etc.

Motion state s_o(t) of an object o at time t is defined as a vector $\{o(t), o(t-1), ..., o(t-f+1)\}$ enclosing its location at f most recent timestamps ($s_o(t)$ is a $(d \cdot f) \times 1$ vector).

The equation 4.16 can be transformed to equivalent matrix form:

$$s_o(t) = K_o \cdot s_o(t-1) \tag{4.17}$$

where K_o is a constant $(d \cdot f) \times (d \cdot f)$ motion matrix for object o. The following notations will be used: k_{ij} is element of K_o at $i-$th row and $j-$th column $(1 \leq i, j \leq d \cdot f)$, k_{i*} is $i-$th row, i.e., a $1 \times (d \cdot f)$ vector, and $o(t).x_i$ is $o(t)$ coordinate on $i-$th dimension.

For simple case of $d = 2$ and $f = 2$ the equation 4.17 becomes:

$$\begin{bmatrix} o(t).x_1 \\ o(t).x_2 \\ o(t-1).x_1 \\ o(t-1).x_2 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} o(t-1).x_1 \\ o(t-1).x_2 \\ o(t-2).x_1 \\ o(t-2).x_2 \end{bmatrix} \tag{4.18}$$

In general $k_{ij} = 0$ for $i \geq d+1$ and $i \neq j+d$, and $k_{ij} = 1$ for $i \geq d+1$ and $i = j+d$.

Motion matrices indicate whether there are dependencies among dimensions. In general, motion type is axis-independent if and only if:

$$k_{ij} = 0, \forall 1 \leq i \leq d \text{ and } (j \bmod d) \neq (i \bmod d) \tag{4.19}$$

Another crucial observation is that all motion matrices should satisfy translation rule:

$$s_o(t) = K_o \cdot s_o(t-1) \rightarrow s_o(t) + c = K_o \cdot (s_o(t-1) + c) \tag{4.20}$$

where translation vector c is a $(d \cdot f) \times 1$ vector in the form $c = (c_1, c_2, ..., c_d, ..., c_1, c_2, ..., c_d)$ (i.e. $c_1, c_2, ..., c_d$ repeated f times), for arbitrary constants $c_1, c_2, ..., c_d$.

In general, the translation rule indicates that if K_o captures a trajectory o_1, it also expresses any other trajectory o_2 translated from o_1. The equation 4.20 imposes some important constraints on elements of motion matrix, i.e., the translation rule implies:

$$c = K_o \cdot c \tag{4.21}$$

where c is any translation vector. For arbitrary values of d and f, the equation 4.21 holds if and only if:

$$\begin{aligned} k_{ij} + k_{i(j+d)} + k_{i(j+2d)} + ... + k_{i(j+f+d)} = 1 \text{ if } j = (i \bmod d), \text{ and} \\ k_{ij} + k_{i(j+d)} + k_{i(j+2d)} + ... + k_{i(j+f+d)} = 0 \text{ otherwise} \end{aligned} \tag{4.22}$$

The last important property of K_o is, based on motion state $s(T_c)$ at current time T_c, that state $T_c + t$ (i.e., t timestamps later) can be efficiently computed as:

$$s_o(T_c+t) = K_o^t \cdot s_o(T_c) \tag{4.23}$$

The prediction task is now reduced to find correct K_o (i.e. determining corresponding movement type) after which parameters of a motion are automatically finalized by $s_o(t-1)$.

Let $l_o(t)$ is actual location of object o at time t. Given $l_o(T_c-h+1), l_o(T_c-h+2), ..., l_o(T_c)$ at the h most recent timestamps, the goal is to decide a function $o(t)$ that minimizes summed squared distances (ssd) between computed (by o) and actual locations:

$$ssd = \sum_{t=T_c-h+1}^{T_c} |l_o(t) - o(t)|^2 \tag{4.24}$$

where $|l_o(t)-o(t)|^2 = \sum_{i=1}^{d}[l_o(t).x_i - o(t).x_i]^2$ and $l_o(t).x_i$ is coordinate of $l_o(t)$ on its $i-$th dimension.

For instance, in linear model a process of minimizing ssd is equivalent to find the best fitting line, which can be computed using least squared error method. The objective of the recursive motion is to decide the optimal K_o.

K_o involves $d^2 \cdot f$ unknowns, which constitute the first d rows from k_{1*} to k_{d*}. The h locations define $h-f+1$ motion states (where f is the retrospect), or specifically:

$$\begin{aligned}
s_o(T_c-h+f) &= \{l_o(T_c-h+f), l_o(T_c-h+f-1), ..., l_o(T_c-h+1)\} \\
s_o(T_c-h+f+1) &= \{l_o(T_c-h+f+1), l_o(T_c-h+f), ..., l_o(T_c-h+2)\} \\
&\qquad \cdots \\
s_o(T_c) &= \{l_o(T_c), l_o(T_c-1), ..., l_o(T_c-f+1)\}
\end{aligned} \tag{4.25}$$

Therefore, the optimal K_o should satisfy the following $h-f$ equations:

$$s_o(t) = K_o \cdot s_o(t-1) \text{ for } T_c-h+f+1 \le t \le T_c \tag{4.26}$$

The equations are solved corresponding to single row (from k_{1*} to k_{d*}) at a time. It suffices to elaborate the solution for k_{1*} (i.e., an $1 \times (d \cdot f)$ vector) as the extension to the other rows is straightforward.

Consider, $d = f = 2$, where each equation in the set 4.26 is in the form of 4.18. The extracted k_{1*} part of the formula 4.18 is as follows:

$$l(t).x_1 = k_{11} \cdot l(t-1).x_1 + k_{12} \cdot l(t-1).x_2 + k_{13} \cdot l(t-2).x_1 + k_{14} \cdot l(t-2).x_2 \tag{4.27}$$

The above equation can be written into the following form which also holds for general d and f (recall that $s(i-1)$ is a $(d \cdot f) \times 1$ vector):

$$l(t).x_1 = k_{1*} \cdot s(t-1) \tag{4.28}$$

Since a similar formula is obtained from all $(h-f)$ equations in 4.26, the $h-f$ (linear) equations for the $d \cdot f$ variables in k_{1*} are obtained and are organized into a matrix form (notice that k_{1*} now appears on the right of the multiplication sign):

$$\begin{bmatrix} s(T_c-1)^T \\ s(T_c-2)^T \\ \cdots \\ s(T_c-h+f)^T \end{bmatrix} \cdot k_{1*} = \begin{bmatrix} l(T_c).x_1 \\ l(T_c-1).x_1 \\ \cdots \\ l(T_c-h+f+1).x_1 \end{bmatrix} \tag{4.29}$$

where the superscript T stands for transpose. Let's denote:

$$S = \begin{bmatrix} s(T_c - 1)^T \\ s(T_c - 2)^T \\ \cdots \\ s(T_c - h + f)^T \end{bmatrix}, \text{ and } l = \begin{bmatrix} l(T_c).x_1 \\ l(T_c - 1).x_1 \\ \cdots \\ l(T_c - h + f + 1).x_1 \end{bmatrix} \tag{4.30}$$

where S is a $(h-f) \times (d \cdot f)$ matrix and l a $(h-f) \times 1$ vector. A robust solution of the linear equation set $S \cdot k_{1*} = l$ can be obtained using singular value decomposition (SVD) [113]. The SVD decomposes S into $U \cdot W \cdot V^T$, where U is a $(h-f) \times (d \cdot f)$ column-orthogonal matrix, $W = [\text{diag}(w_1, w_2, ..., w_{d \cdot f}]$ is a $(d \cdot f) \times (d \cdot f)$ diagonal matrix with positive elements $w_1, w_2, ..., w_{d \cdot f}$ on the diagonal, and V is a $(d \cdot f) \times (d \cdot f)$ orthogonal matrix (i.e., $V \cdot V^T = I$, the identity matrix). Thus, k_{1*} is solved as:

$$k_{1*} = V \cdot [\text{diag}(\frac{1}{w_1}, \frac{1}{w_2}, ..., \frac{1}{w_{d \cdot f}})] \cdot (U^T \cdot l) \tag{4.31}$$

If $h - f > d \cdot f$ the SVD produces a k_{1*} that minimizes $|S \cdot k_{1*}|^2$, so that the equation 4.24 can be rewritten to:

$$ssd = \sum_{i=1}^{d} \sum_{t=T_c-h+1}^{T_c} |l_o(t).x_i - o(t).x_i|^2 \tag{4.32}$$

Similarly, the solution of each $i-$th row k_{i*} of K_o minimizes the ssd, thus the overall K_o minimizes ssd.

4.3 Next location prediction

Long time prediction methods so called next location prediction methods assume that there is always a certain regularity in user movement patterns so the movement is not completely random. Movement of a user is described by a sequences of visited locations instead of current motion parameters.

Human trajectories show a high degree of temporal and spatial regularity, following simple and reproducible patters [50]. Analysis of location histories can be roughly classified, according to the manner by which data is modeled, into two general approaches: state-space models and data mining methods.

Next location prediction methods has several other potential applications such as evaluation of geo-privacy mechanisms, location-based services or design of location-aware handover management in wireless networks, where the mobility is usually described by a spatial trajectories represented e.g. by sequence of GPS coordinates. A detailed survey of computing with spatial trajectories is given in [153].

4.3.1 Virtual environments

In [71] the advantages of the next location prediction methods are described for on-line gaming environments. The author proposed a hybrid method, where a combination of a short term prediction method and a long term prediction method is used to reduce latency.

The short term prediction is represented by hybrid mouse motion predictor which combines linear model for slow mouse motion and an elliptic model for mouse motion at high velocity [21].

The long term prediction method uses a statistical approach to predict next user position. A virtual environment is divided into regular zones (square cells), which form basic units for statistical prediction. Given a boundary location x, where a user enters a zone, the probability that the user will leave the zone at a particular boundary location y can be evaluated from statistics collected for each zone. Probability distribution function is modelled using a Gaussian mixture (see Figure 4.17).

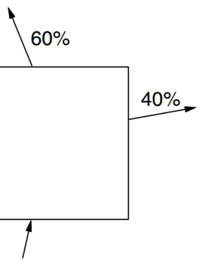


Figure 4.17: Statistical prediction of exit positions depends on enter positions[55].

Based on the computed probability values, the neighboring zone(s) which the user will likely visit next can be estimated. The author further proposes to construct a path tree for the current user's zone by accumulating statistics from zone to zone (see Figure 4.18).

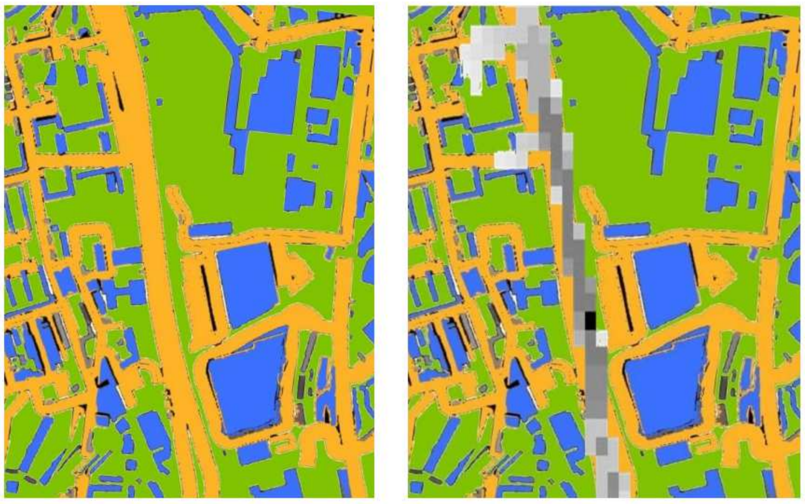


Figure 4.18: An example path tree: left image shows part of the Central London while the right image shows a superimposed path tree, starting from current user position (the black zone)[55].

The path tree in Figure 4.18 shows probability of various paths which are reachable from the current user location. The problem with the path tree is how to compute a prediction error. With a single predicted path, euclidean distance between actual user location and a predicted location can be used. With many potential paths each with a different probability, it is no longer straightforward matter to compute the accuracy. For example, a user is currently in a zone that shows an exit to the left with a 60% probability

[55]Rynson W. Lau and Addison Chan. Motion in games. chapter Motion Prediction for Online Gaming, pages 104–114. Springer-Verlag, Berlin, Heidelberg, 2008.

and an exit to the right with a 40% probability. If the user turns left, should it be said that the prediction has a 100% accuracy or 60% accuracy? The author considers only exits with the highest probability when measuring accuracy of this method.

The path tree allows a prediction of more than one cell ahead but for the expense of decreasing prediction accuracy as the prediction length increases. Because the transition statistics are collected only from zone to zone, the information about continuous movement is lost. Multi-resolution characteristic of the scene data is also not considered.

4.3.2 State-space models

State-space models attempt to capture variation in spatial sequences. Bhattacharya et al. [11] used a simplest Markov model, which assumes that a process S is a time-invariant Markov chain defined as:

$$\begin{aligned} P_r[V_k = v_k | V_1 = v_1, \cdots, V_{k-1} = v_{k-1}] \\ = P_r[V_k = v_k | V_{k-1} = v_{k-1}] \\ = P_r[V_i = v_i | V_{i-1} = v_{i-1}] \end{aligned} \tag{4.33}$$

for any arbitrary choice of k and i, where V_i is a stationary stochastic process such that V_i assumes that the value $v_i \in V_i$ is an event such that a user is in cell v_i.

One-step transition probabilities between two adjacent cells $P_{i,j} = P_r[V_k = v_j | V_{k-1} = v_i]$ are estimated from movement history database.

The authors further evaluate an entropy of the proposed simplest Markov chain model as well as higher order Markov chains to show that the higher the order of the Markov chain is the lower the improvements in model richness is. For an universal model, they estimate appropriate order dictated by input sequence using LZ78 compression algorithm. In practice, a natural limitation can appear such that quantity of data available for analysis becomes the main limiting factor for the order of the Markov-chain model.

Ashbrook et al. [8] proposed a system that automatically clusters GPS data taken over a period of time into frequently visited locations (POI, places) at multiple scales. Each place is defined by its GPS coordinates. A significant place can be identified by a time a user stay at that place. The idea behind it is that users usually stops for longer time at significant or important places. Based on source data, the authors define that staying for ten minutes at the same position determines the significant place. A variant of k-means clustering algorithm is used because close significant places should be considered as single significant location or cluster (see Figure 4.19).

These significant locations can obscure prediction opportunities on smaller scales. Hence, the authors proposed a concept of sub-locations. The sub-locations are determined for each cluster using the same k-means algorithm but with varying radius. As the locations is known, each GPS trajectory can be encoded as a sequence of visited places or at higher level of abstraction as a sequence of visited locations. Hence, each location has assigned an unique ID so that each GPS trajectory can be described as a sequence of these IDs. Finally, a Markov model is created based on these sequences of locations, with transition probabilities estimated from the travelling frequencies between two different locations.

The k-means algorithm has several drawbacks for detecting the significant locations. First, it needs to know the number of clusters before clustering begins. Second, all points are included in the final clustering results, which makes the results quite sensitive to noise and third, the k-means algorithm is non-deterministic i.e. the final clustering depends on the initial random assignment of places to clusters [156]. Therefore, instead of k-means a

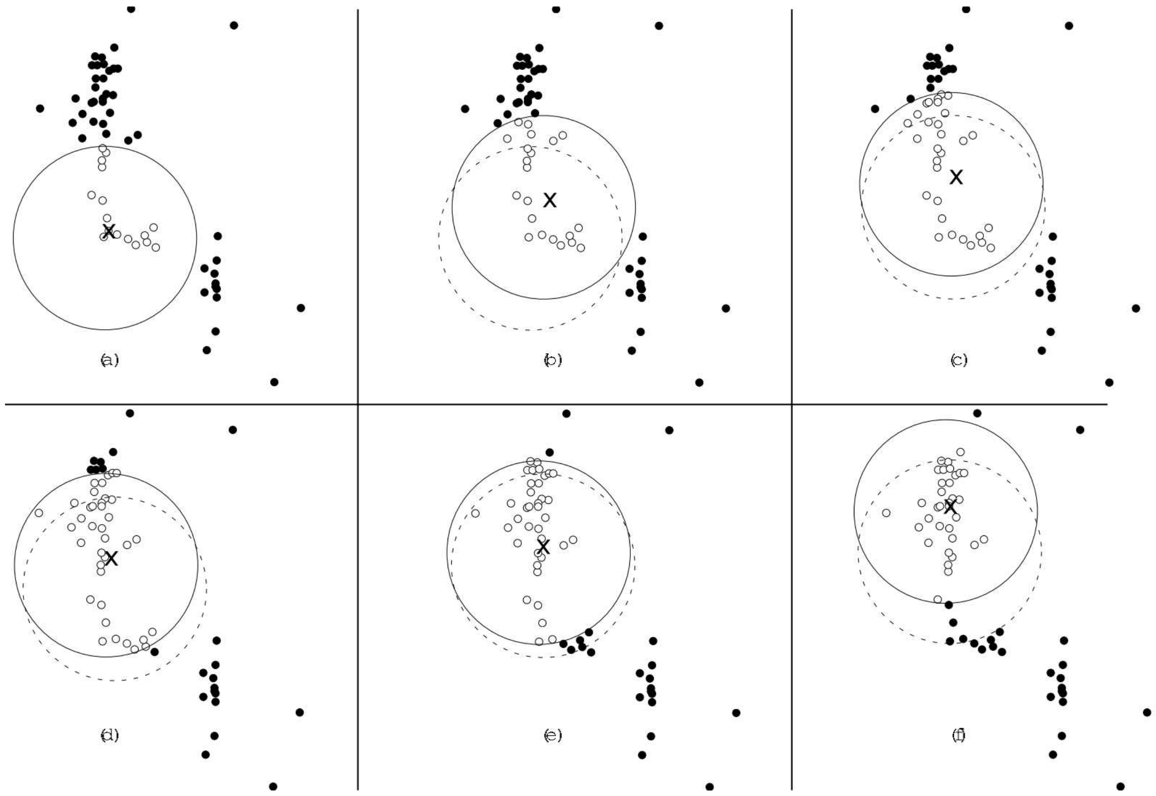


Figure 4.19: Illustration of location clustering algorithm. The X denotes center of a cluster. The white dots are points from a cluster, and the dotted line shows location of the cluster in the previous step. At step (e), the mean has stopped moving, so all of the white points will be part of this significant location[56].

density based algorithm will be more suitable for clustering significant locations for Mobility Markov Chain (MMC) algorithm [43]. Here, a density-joinable DJ-cluster [156] is used to overcome the k-means drawbacks. Behaviour of each individual user is modelled as a discrete stochastic process in which probability of moving to another state (significant location or place of interest) depends only on the previous visited state and probability distribution of transitions between these two states. The Mobility Markov Chain is composed of a set of states $P = \{p_1, ..., p_k\}$ and a set of transitions $t_{i,j}$. Each state corresponds to a significant location which has assigned a semantic label such as „work“, „home“, etc. Each transition $t_{i,j}$ represents a probability of moving from state p_i to state p_j.

A n-MMC algorithm, based on order-m Markov chain is further applied to increase prediction accuracy. The transition probabilities are computed for sequences of n last visited locations (see Figure 4.20).

Recently a Mixed Markov-chain model (MMM) [7] method is proposed for user next location prediction. The authors noted, that the discrete-time Markov chain and the Hidden Markov models cannot model behaviour of all users with single universal model. Therefore, they proposed to classify trajectories into patterns corresponding to groups of similarly moving users. Each pattern does not show an individual behaviour, but it shows a common behaviour among similar users. In this setup, the next location prediction is based on the

[56]Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7(5):275–286, October 2003.

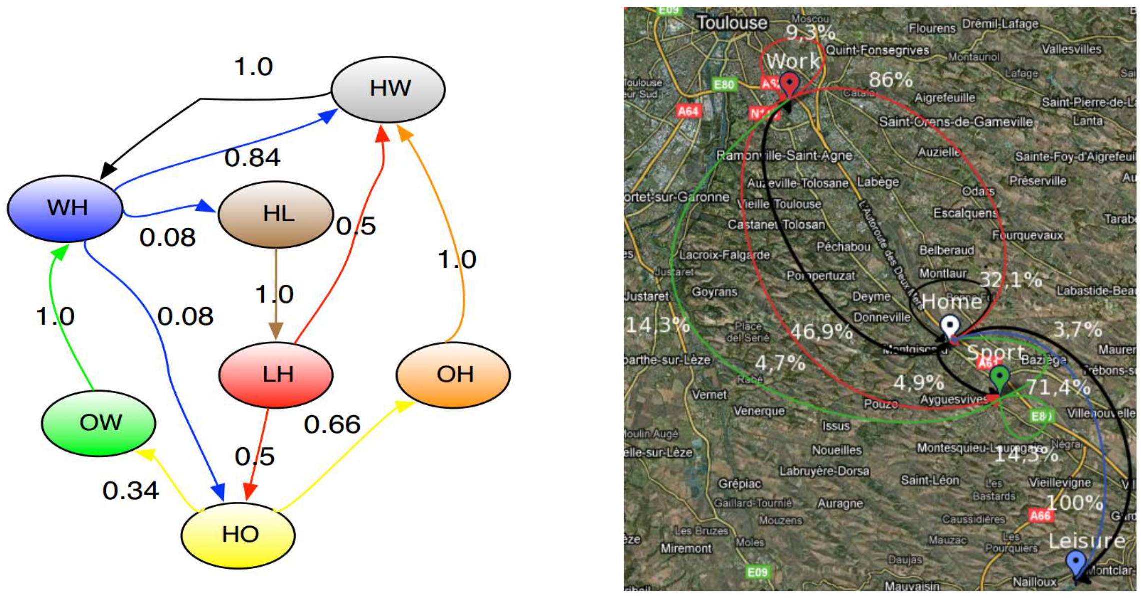


Figure 4.20: (left) Graphical representations of 2-MMC model („W“, „H“, „L“ and „O“ stand for Home, Work, Leisure and Others locations respectively) for single user, (right) and of 1-MMC model[57].

Markov model which belongs to a group of individuals with similar mobility behaviour. A specific Markov model is generated for each group. As the MMM method has higher prediction rate compared to the Markov chain based prediction and Hidden Markov model, the authors did not evaluate its accuracy, which is usually a much more important parameter.

Mathew et al. [87] constructed Hidden Markov Model (HMM) to predict future user locations. Input history trajectories are clustered to locations based on their characteristics (i.e. temporal period in which they occurred) so that these locations can be later used to train the HMM. The location characteristics are used as unobservable parameters. The history database is clustered with selected characteristics to different clusters so that each cluster is used to train different HMM. „Continuous“ GPS trajectories within each cluster are discretized using hierarchical triangular mesh [129] to efficiently train the HMMs.

In general, the HMMs based methods succeed only when dealing with uncertainty [115] (i.e. they generalize well), but they suffer from higher training complexity and slow relearning [109].

4.3.3 Data mining methods

The data mining methods explore association rules and frequent patterns from databases [92], [65]. Trajectories are defined as an ordered sequences of time-stamped locations, which are further analysed by variations of a-priori algorithms [91].

Data mining method for predicting future locations of moving objects from movement data is presented in [92]. Here, the association rules from trajectories database are extracted in a form such that they are transformed from original continuous domain of moving object positions into a discretized domain of edges of a superimposed grid. Prediction is realized

[57]Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, MPM '12, pages 3:1–3:6, New York, NY, USA, 2012. ACM.

by a matching function which selects a best association rule which matches partial object trajectory. The on-line matching of partial trajectories through the database of movement rules is executed very fast, but at the cost of expensive and burdensome computations for discovering the association rules and frequent trajectories.

Jeung et al. [65] presented a novel hybrid prediction model for moving objects. The authors addressed a problem how to discover trajectory patterns and how to manage a potentially large number of mined trajectory patterns to answer predictive query effectively. Their hybrid prediction algorithm provides accurate results for both distant time queries and non-distant time queries. If a prediction query did not return any candidate trajectory, the recursive motion function (see the section 4.2.1) will be used instead.

The main goal of the data mining techniques is to maximize confidence computed from the previous occurrences and they often do not consider spatial neither temporal distance. This is a reason why state-space models generalize the matching process much better. The other general disadvantage of the trajectory pattern mining methods is the expensive and burdensome computations necessary to discover association rules and their confidences. Hence, these methods are not applicable in highly dynamic systems with the large amount of new trajectories frequently arriving.

4.4 Terrain environments

Data prefetching algorithm for terrain streaming applications is proposed in [35]. The terrain is represented by heightmap tiles sampled at different resolutions which are stored at server. At the beginning, the system transfers only a very low resolution heightmap of the entire terrain. As a viewer moves, the detailed terrain tiles are streamed to the client device depending upon then viewer position. When requesting some tile, the last transmitted resolution of the tile is checked and only the residue between the high resolution tile and the super-sampled version of the low resolution tile is streamed to the client.

Closer tiles have higher detail compared to distant tiles (see Figure 4.21). As view frustum of the viewer moves, newer higher resolution tiles are streamed to the client device. Only tiles inside or intersecting the view frustum are streamed to the client.

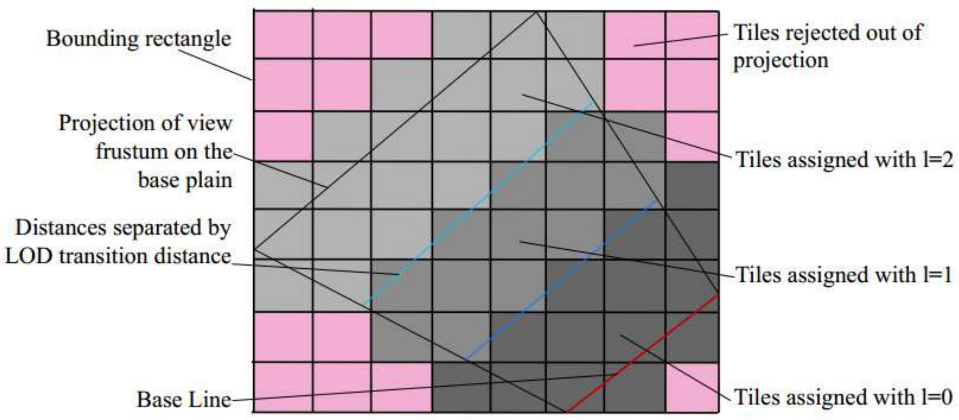

Bounding rectangle
Projection of view frustum on the base plain
Distances separated by LOD transition distance
Base Line
Tiles rejected out of projection
Tiles assigned with l=2
Tiles assigned with l=1
Tiles assigned with l=0

Figure 4.21: View frustum culling and LOD assignment for terrain tiles[58]

[58]Soumyajit Deb, Shiben Bhattacharjee, Suryakant Patidar, and P. J. Narayanan. Real-time streaming and rendering of terrains. In *Proceedings of the 5th Indian conference on Computer Vision, Graphics and*

The client uses a motion prediction method for prefetching data from server in advance. The motion parameters for the predictor are determined from past viewer positions. Practically, motion parameters from last five frames are averaged to generate motion parameters for the next frame. Assuming constant rate of change in acceleration, a next position vector P_{i+1} can be expressed using the following formula:

$$P_{i+1} = P_i + c_1(t_{i+1} - t_i)v_i + c_2(T_{i+1} - t_i)^2(2a_i - a_{i-1}) \tag{4.34}$$

where c_1 and c_2 are constants, a_i is acceleration, v_i is velocity and P_i is position vector in i-th frame which need t_i time to render. The constant acceleration is not a good assumption for rotational motion. The time t_{i+1} must be estimated from older known frame times by averaging them. Once future position of the viewer is known, the terrain data corresponding to the estimated position can be pre-fetched.

Royan at al. [118] proposed a networked environment based on hierarchical level-of-detail models which utilizes view-dependent progressive streaming of geometry and texture data. They use two hierarchical LOD models separately for terrain data and for groups of buildings. To reduce the impact of round trip delays a client-side LOD selection process must update a local 3D model with a minimum number of server requests.

Without any details, they introduces a notion of observer lookahead and some special cases of motion (e.g. flight simulation). Their algorithm also assumes that rotations are relatively slow. When streaming 3D buildings, the authors expect the amount of transferred data per second will not be higher than 10kB, which is well bellow bandwidth of most broadband Internet connections. In this case, the main factor affecting perceived latency is the view-dependent LOD selection method and resulting client-server communication handling.

Speculative prefetching of terrain tiles is presented in [124]. Here, a user view direction is predicted by fitting a spline through a list of last viewing parameters so the tiles that are predicted to become visible in near future can be pre-fetched in advance.

Bettio et al. [10] developed a networked terrain rendering engine which operates with data which is represented by heightmaps supported with the BDAM algorithm . The BDAM algorithm recursively partitions an input domain in a hierarchy of right triangles. The algorithm further utilizes wavelet based encoding to compress differences between terrain patches (diamonds) with different level-of-detail.

The wavelet coefficients from requested diamond patch are transmitted using HTTP/1.1 persistent connection with pipelining. The pipelining approach allows multiple HTTP requests to be written together to single open socket without waiting for the corresponding responses. The client receives the responses in the order in which they were issued. The pipelining can significantly improve response times especially in the case of small sizes of the requested patches as they are highly compressed.

Diamond patches which are needed to refine a diamond graph from the current point of view are requested first. Each such request is pushed in a priority queue. Only limited number of requests are issued based on the estimated network bandwidth. The prefetching is done in this stage by pushing a set of diamonds which are not scheduled for refinement, but are likely to be refined in near future to the queue. Since these diamonds are at the end of the requests queue, they are considered only if all previous requests have been completed.

Image Processing, ICVGIP'06, pages 276–288, Berlin, Heidelberg, 2006. Springer-Verlag.

Chapter 5

Streaming and rendering framework

The description of the scene rendering algorithm used and data representation is important part of this thesis, because the location-aware data transfers scheduling is tightly related to them. Therefore, the proof-of-concept implementation of the framework described in this chapter is necessary to prove the results of both the proposed prediction scheme and data transfers scheduling algorithm.

The distributed virtual environment is represented by a 3D scene composed of large terrain with high-resolution orthographic textures. The terrain is populated by buildings generated from cartographic data obtained from well known Open Street Map project database (see Figure 5.1).

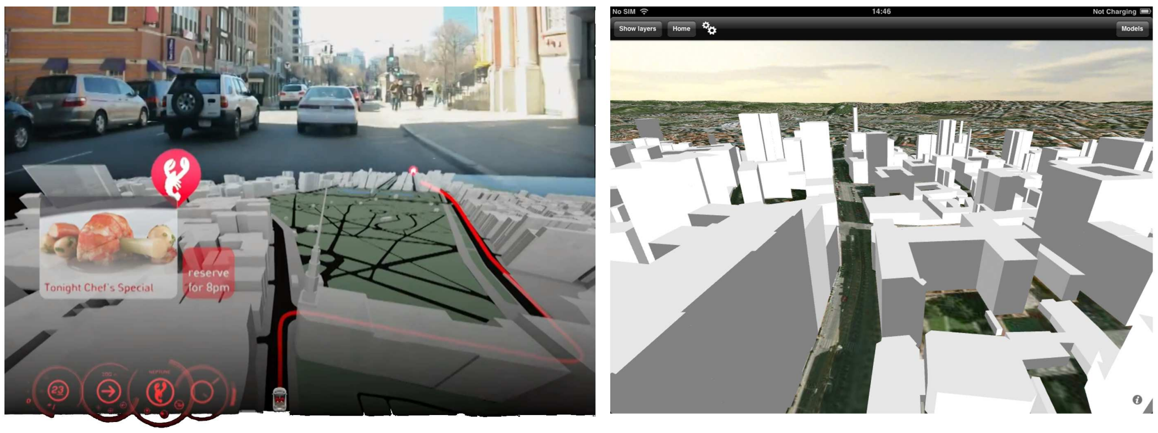


Figure 5.1: Output of the implemented virtual environment framework running on iPad (right) is similar to the AIDA augmented reality system (left).

The AIDA system analyses driver's behaviour to identify a set of goals the driver would like to achieve (e.g. information about business and shopping districts, tourist and residential areas, as well as real-time event information related to traffic information such as jams, accidents, etc.). This thesis utilizes the driver's behaviour to optimize streaming of the data needed for rendering the visualized 3D scene instead of identifying the goals of the AIDA system.

The described framework is primarily implemented to prove the results of the proposed prediction and scheduling schemes, but also to see that various applications of the schemes are possible. For example, the analysis of the driver's (or a user) behaviour can be used to decide which scene parts presented in local cache will be preserved on storage limited devices or thin clients.

This chapter briefly describes the rendering algorithm used, the client-server framework and data resources used to render the scene.

5.1 Client-server architecture

Communication between clients and a server in distributed virtual environments is usually based on a pull or a push model. In the pull model, each client sends requests to the server to obtain necessary data for rendering the current view. Hence, each client needs to have knowledge of the visualized scene structure and spatial data organization. In the push model, each client continuously sends its position or other motion data to the server. Based on this information, the server can identify data needed to render the current view and sends the data to client cache. This approach has the advantage, that no prior knowledge of the scene structure is required at clients. On the other hand, this approach is more CPU-demanding for the server. The server has also no knowledge of data already presented in clients cache.

This thesis uses a hybrid demand driven communication model such that both client and server can initiate transferring of scene data from server to clients (see Figure 5.2).

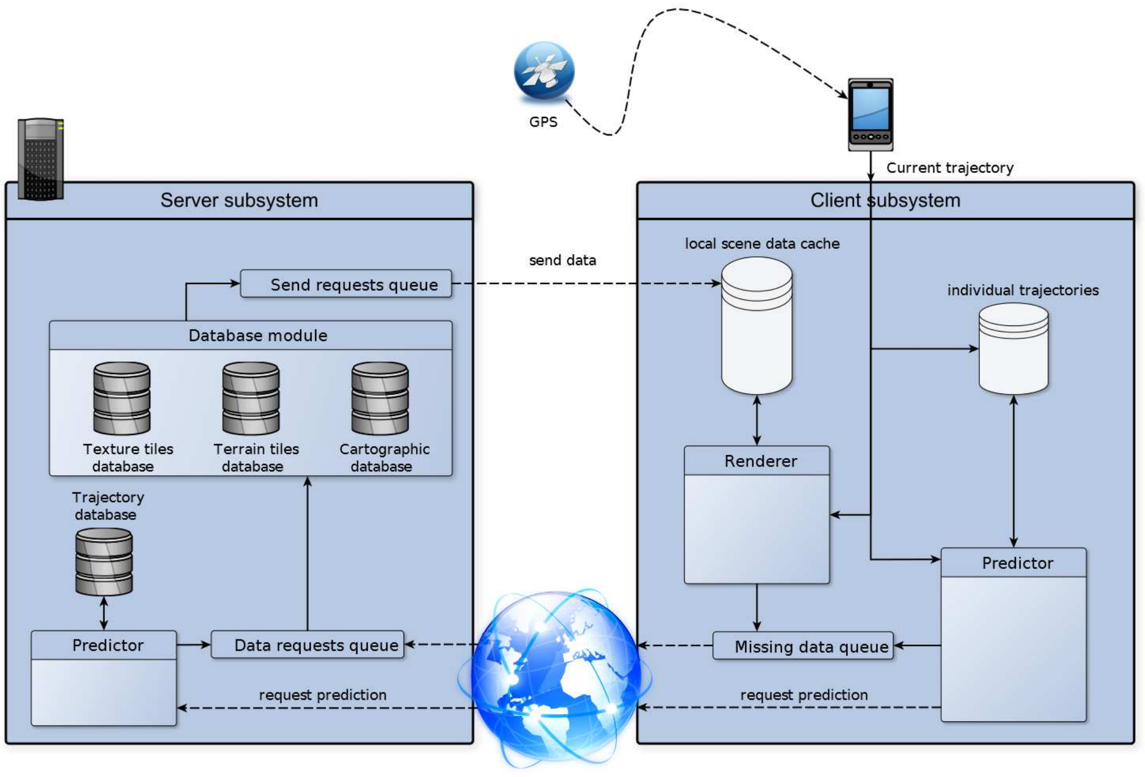


Figure 5.2: The proposed client-server framework with hybrid demand driven communication model.

5.1.1 Server sub-system

The server receives missing data requests from clients. These requests are accepted through an input data requests queue. Each request is evaluated by the server so that it selects appropriate data tiles from database module and sends them back to the clients through the output requests queue.

If prediction at the client did not succeed, then it will be performed by the server. The server performs the prediction based on different principles and with different data compared to the prediction done by clients. As the server contains trajectories from all users, it can predict movement of each user by matching its current movement pattern with movement patters from all users with the same behaviour (see next chapter for details).

If the predictor at the server succeeds, the priority of current missing data by client is computed and the scene data needed by the client to render the scene at the predicted position is pushed to the end of the send requests queue. The data which is needed by the client to render the scene at some predicted position is determined by server based on knowledge of client rendering algorithm. The server maintains a profile for each user to remember the scene parts already sent to each client. Hence, only the missing data related to the predicted position will be sent to the client.

The server is implemented as web server with the help of Boost asio library. The Boost asio library effectively uses threading and network sockets sharing to serve multiple simultaneous connections at a time. The database module uses MongoDB which is a noSQL document database. It contains all the scene data including terrain elevation tiles, terrain texture tiles and cartographic tiles. It also contains a trajectory database which can be used for the prediction tasks.

5.1.2 Client sub-system

Input of the client subsystem can be e.g. a mobile client with ability to determine its own geographic position using GPS hardware. The path of each mobile client is described by continuous sequence of geographic positions (GPS trajectory).

The current client position is further used as an input to a renderer which renders a scene viewed from the current position. The renderer determines also all the missing data (scene parts) which is needed to render the scene at current client position. The resulting missing data requests are pushed to a missing data queue. All the requests from this queue are sent to a remote server.

Each client can additionally predict its next position based on its current trajectory. The trajectory forms a movement pattern which can be used as an input to a local predictor. The local predictor determines scene parts potentially needed for rendering the scene at a predicted position. The scene parts which are needed to render the scene at the predicted position, but are not available in a local cache, are pushed at the end of the missing data queue. The predicted position is used also to compute priorities of current missing data to render the current view. If the local predictor does not answer a prediction query, the prediction will be performed at server.

The client application is implemented as a thick client running on both iPad device and desktop workstations (see Figure 5.3).

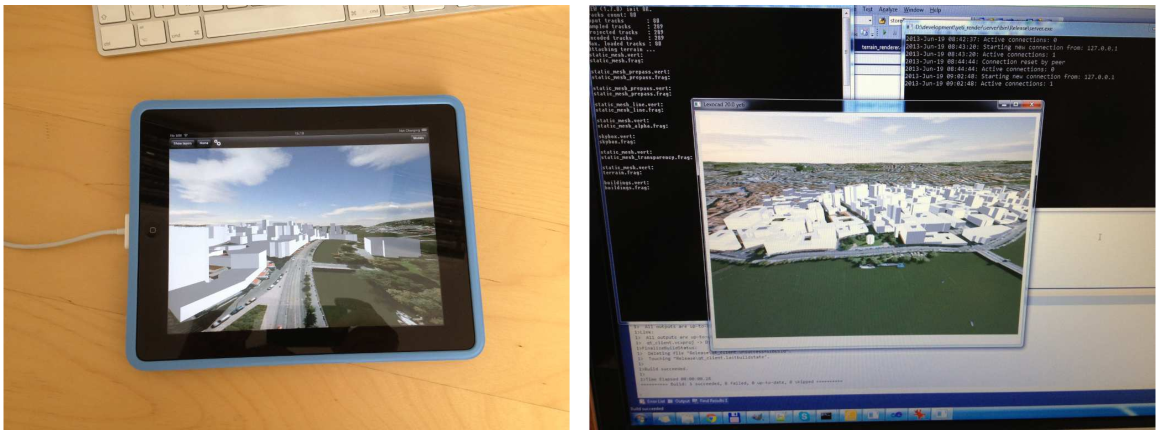


Figure 5.3: The client application runs on iPad with the help of Cocoa API (left). Qt framework is used on Windows, Linux and MacOSX systems (right).

5.2 Multi-resolution data representation

Multi-resolution data representation is a necessary form of input data for both fast rendering and efficient streaming of the scene parts over network. It allows streaming and rendering of the scene parts at coarser resolutions in the case of slow network connection, fast moving user or limited rendering capabilities of target devices. In the proposed framework, each client renders a scene which is composed of three layers: terrain geometry layer, terrain textures layer and cartographic layer (including only buildings).

5.2.1 Terrain geometry and textures layer

The terrain layer contains height-map tiles (obtained from ASTER global digital elevation model [64]) which are further organized into an elevation data pyramid (see Figure 5.4).

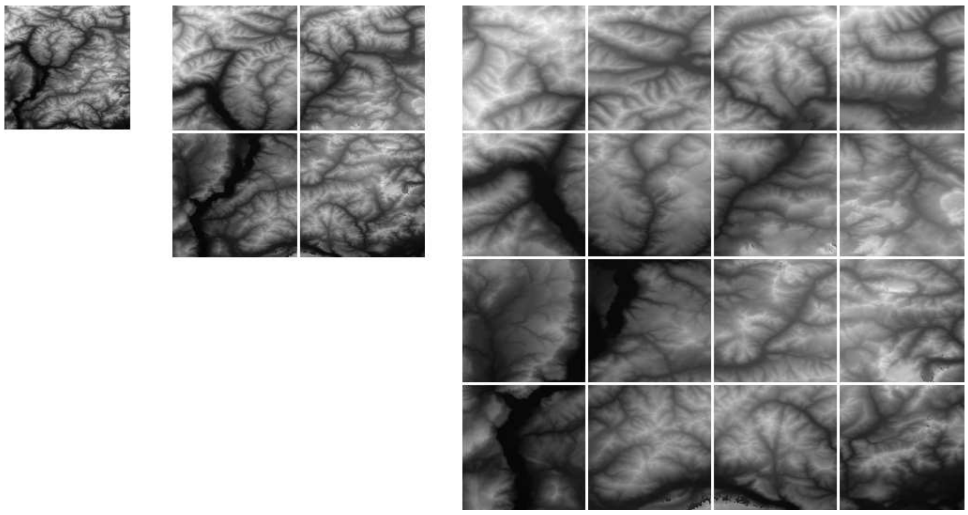


Figure 5.4: Each tile from coarser level (left) can be covered by its four children (center), continuing recursively (right) to the bottom of the pyramid. Each child tile covers one quarter of the area covered by its parent tile.

Each tile through all levels of the elevation data pyramid has resolution of 256 × 256 height points. The top level of the pyramid contains a single tile which covers the whole terrain at a much coarser resolution. The four child tiles cover the same area as their parent tile thus resulting in double resolution. Tiles at the bottom of the elevation data pyramid cover the whole terrain at the highest possible resolution.

The texture layer contains high resolution orthographic texture tiles which are mapped onto the terrain tiles. Each tile has the resolution of 256 × 256 pixels and is also included inside a texture data pyramid (see Figure 5.5) similarly to the terrain tiles.

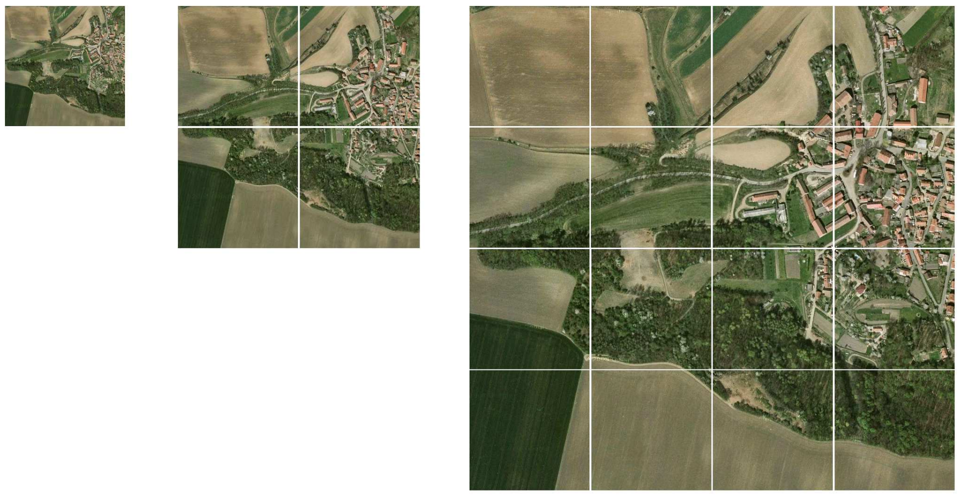


Figure 5.5: Part of the textures data pyramid which is created similarly as the elevation data pyramid..

As the elevation and textures data is obtained from real datasets, it is defined that each elevation tile is covered by an array of 8 × 8 texture tiles. Consequently, each 256 × 256 terrain tile is covered by a texture data with the resolution of 2048 × 2048 pixels.

5.2.2 Cartographic layer

The cartographic layer is composed of cartographic data obtained from the Open Street Map project database (see Figure 5.6). The OSM data is obtained as a single xml file which holds all the cartographic information within a retrieved area. This file is divided during a preprocessing step into regular square tiles, each covering only small part of the whole xml file. The area covered by each tile is determined by the size of the terrain tiles used at their finest resolution. This enables the streaming an the rendering system to request and render only the cartographic information in close neighbourhood of the client.

In fact, only declaration of buildings is used. The buildings from each tile are preprocessed so that corners of each building are annotated with altitude value to positioning the building onto the rendered terrain.

Each building is defined as a multi-polygon entity with single outer and zero or more inner members (corners). Other additional information such as roof types, floor levels or building height are also available, but not for all buildings.

The cartographic data contains also definition of streets (physical location, names, types, etc.), buildings (their outlines, nested outlines, roof types, height, floor levels, etc.) and many other information.

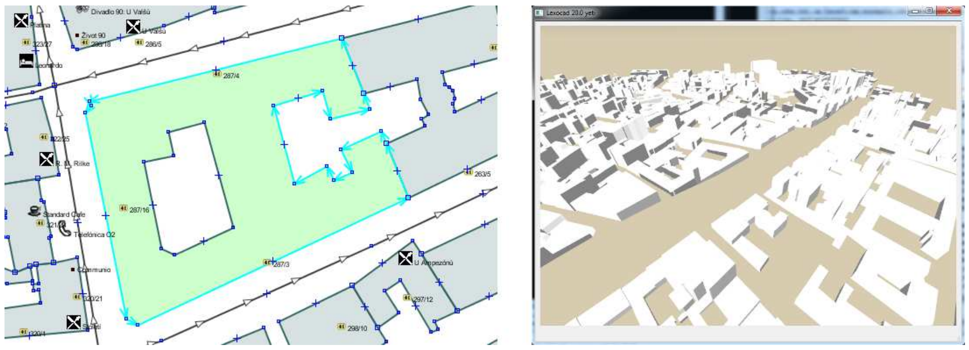


Figure 5.6: Visualization of the OSM database[59] (left) and 3D visualization of the buildings in the implemented framework (right).

This thesis supports rendering of 3D buildings only (see Figure 5.6) as they can greatly increase the 3D perception of the rendered scene. The buildings can be optionally used to visualize additional information from other data sources (e.g. business, shopping informations, etc.).

5.3 Terrain rendering algorithm

The terrain rendering part of the implemented framework is based on a new tile based geometry clipmap algorithm which is suitable for smooth and fast rendering of large textured terrains in client-server environment. This method is a side product of this thesis and will be shortly introduced here because the rendering algorithm is tightly related to the proposed scheduling scheme.

Traditional geometry clipmap algorithm renders the terrain as a set of nested regular grids centered about a viewer. As the viewer moves, the grids are incrementally updated with the possibility to crop the update if it is too costly within a frame. For the tile based streaming of large textured terrains, both the incremental update and the cropping mechanisms become ineffective. In our new tile based geometry clipmap algorithm, each grid of the clipmap is logically divided into a uniform array of square patches. This allows an independent update of the rendered terrain data as new tiles become locally available, selective cropping, and also effective toroidal update of the rendered terrain data on GPU. The proposed algorithm can fast render and fluently update large textured terrains streamed from a remote server even on commodity hardware including also mobile platforms.

5.3.1 Review of geometry clipmaps algorithm

The geometry clipmaps algorithm [77] as well as its GPU based implementation [9] assume coherent movement of viewer so that only small L-shaped region needs to be processed each frame. If the data needed for rendering particular levels of the clipmap is not immediately available then the update and cropping mechanisms become ineffective. For the GPU-based geometry clipmaps the cropping mechanisms is not straightforward to implement at all. The update and the cropping mechanisms have to respect how the girds are partitioned (see Figure 5.7).

[59]http://wiki.openstreetmap.org/wiki/Building

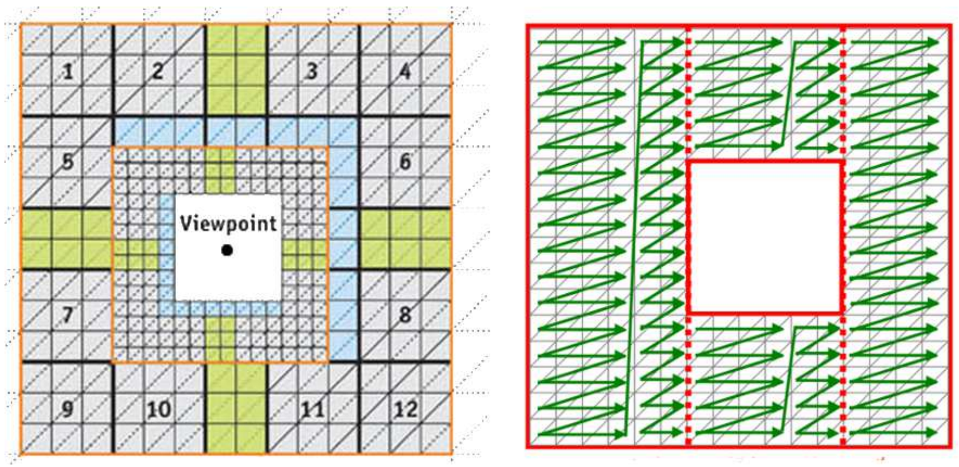


Figure 5.7: Partitioning a clipmap ring for GPU-based geometry clipmap[60] (left) and original geometry clipmap[61] (right)

As the tile by tile streamed terrain data is not available immediately within a single frame, the coherence between two updates is no more valid. It can be seen from Figure 5.7, the partitioning of the rings do not allow effective update of the related buffers on GPU as the tiles needed for rendering becomes available one by one.

The new tile based geometry clipmaps algorithm logically partitions each level of the clipmap into an uniform array of square patches. Such partitioning allows an independent update of elevation and texture buffers on GPU covered by these patches, selective cropping and it still keeps the effective toroidal update of the rendered data on GPU. The proposed algorithm allows the large textured terrain to be rendered at interactive frame rates on commodity hardware or even on mobile devices.

The geometry clipmap algorithm is not the optimal solution concerning the minimal amount of transferred elevation data. This is not a serious issue in our case where the amount of texture data is more than one order of magnitude larger than the amount of elevation points. The textures are handled similarly as in the texture clipmap algorithm which is close to level-of-detail treatment of images in texture mapping. Therefore, the streaming and rendering of texture tiles can be considered nearly optimal passing over the redundancy.

5.3.2 Resources management

Implementation of the new tile based geometry clipmaps algorithm follows the widely supported OpenGL ES 2.0 standard. This restriction permits the usage of advanced OpenGL features, but on the other hand all mobile devices supporting this standard can use the proposed algorithm without any modifications.

The tile based approach has two advantages over the GPU-based and original geometry clipmaps. The first is that the cropping mechanism can be easily handled and the second is that the opengl resources can be updated locally as new data becomes downloaded. The

[60] A. Asirvatham and Hugues Hoppe. *Terrain rendering using GPU-based geometry clipmaps*. Addison-Wesley, 2005.

[61] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, August 2004.

	Original	GPU-based	tile-based
Elevation data	in vertex buffer	in 2D vertex texture	in vertex buffer
Vertex buffer	incrementally updated by CPU	vertex texture update	independent update, CPU
Index buffer	generated every frame by CPU	constant	constant
Texture buffer	incremental update CPU	not resolved	independent update, CPU
Texture coordinates	constant	not resolved	constant
Watertight	zero-area triangles	blending	index buffer selection in constant time, CPU
Cropping	horizontal and vertical	no	variable per patch basis

Table 5.1: Comparison between original, GPU-based and the new tile-based geometry clipmaps method for terrain rendering.

effective update mechanisms is very important because the amount of elevation data and texture data is the main limiting factor against smooth rendering. Table 5.1 shows a comparison between the original geometry clipmaps algorithm [77], its GPU implementation [9] and our tile-based method.

The original geometry clipmaps consider only normals and colour maps with resolution twice a higher than resolution of geometry. The texture update is similar to the geometry update i.e. it is incrementally updated. For real orthographic texture data the incremental update of the texture buffer object on GPU is not effective. Our method uses a single vertex buffer object and a single texture object for each level of the clipmap. These two objects are the only updated resources on GPU (see Figure 5.8). As the viewer moves the patches which fall outside of a given grid are identified. In a parallel process an appropriate terrain data for such invalidated patches are prepared, and only the sub-parts of the vertex and texture buffers on GPU which are covered by them are updated. The stitching of boundaries between different levels of the clipmap is resolved at per patch basis. The constant texture coordinates and index buffers are precomputed at the beginning for a single level of the clipmap and are reused over all levels.

5.3.3 Formal definition of the patches

Each level of the clipmap is marked as L_i, where $i \in 0..N-1$, and N is number of levels of the clipmap. Each level L consists of a 2D uniform array of $M \times M$ patches $P_{j,k}$, where $j, k \in \langle 0..M-1 \rangle$ is the index of the patch in the array, and M must be even and dividable by four.

Each patch $P_{[j,k]}$ covers an appropriate part of both vertex buffer object and texture

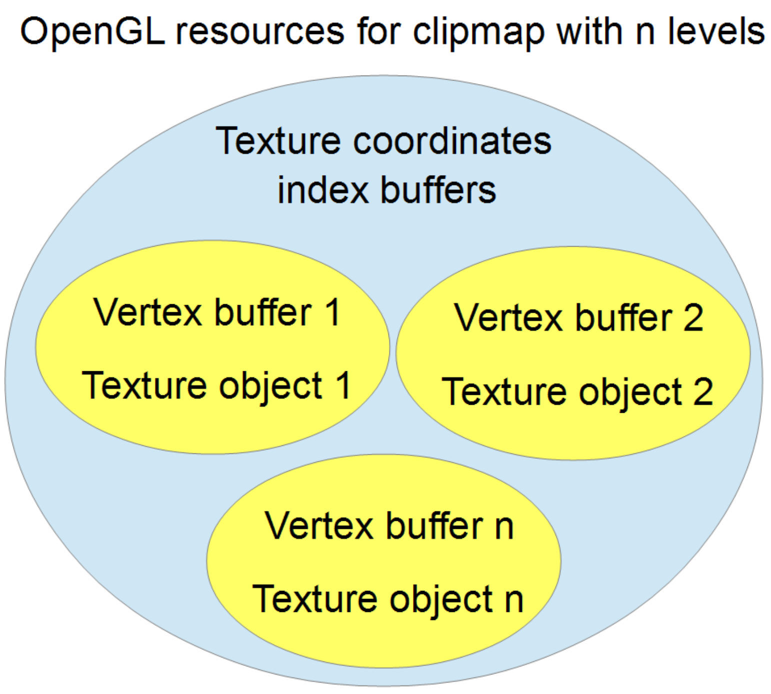


Figure 5.8: Only the yellow sub-sets needs to be updated as a viewer moves, the blue set is precomputed once at the beginning and is shared between all n levels of the clipmap.

object specified by the index $[j, k]$ (see Figure 5.9A- 5.9C). Each patch contains precomputed indices such that it renders the covered set of elevation points with appropriate texels sampled from the texture object.

Number of elevation points covered by each patch P is $D \times D$ and D must be odd. The boundary elevation points within each patch are the same between all adjacent patches. This behaviour is needed for independent updates and rendering of the terrain data covered by particular patches. The independence is also important because of stitching between patches from different levels. Similarly, the boundary texels covered by each patch are also the same between all adjacent patches because of the texture filtering (see Figure 5.9C.). The width of the border with repeated pixels depends on used texture filtering. The repeated elevation points and texels induce a redundancy which depends on the size of the array of patches $P \times P$, on the size of the patches and on size of the border of texels between adjacent patches.

5.3.4 Index buffers

The index buffers are precomputed before the application starts. Fourteen permutations of indices are generated for each patch for t-junction free transitions between patches from adjacent levels of the clipmap (see Figure 5.10).

The index buffers are computed only for a single level and are shared between all levels. As fourteen index buffers are generated for each patch, the overall number of index buffers generated within the tile-based rendering of the geometry clipmap is $14 \cdot M \cdot M$.

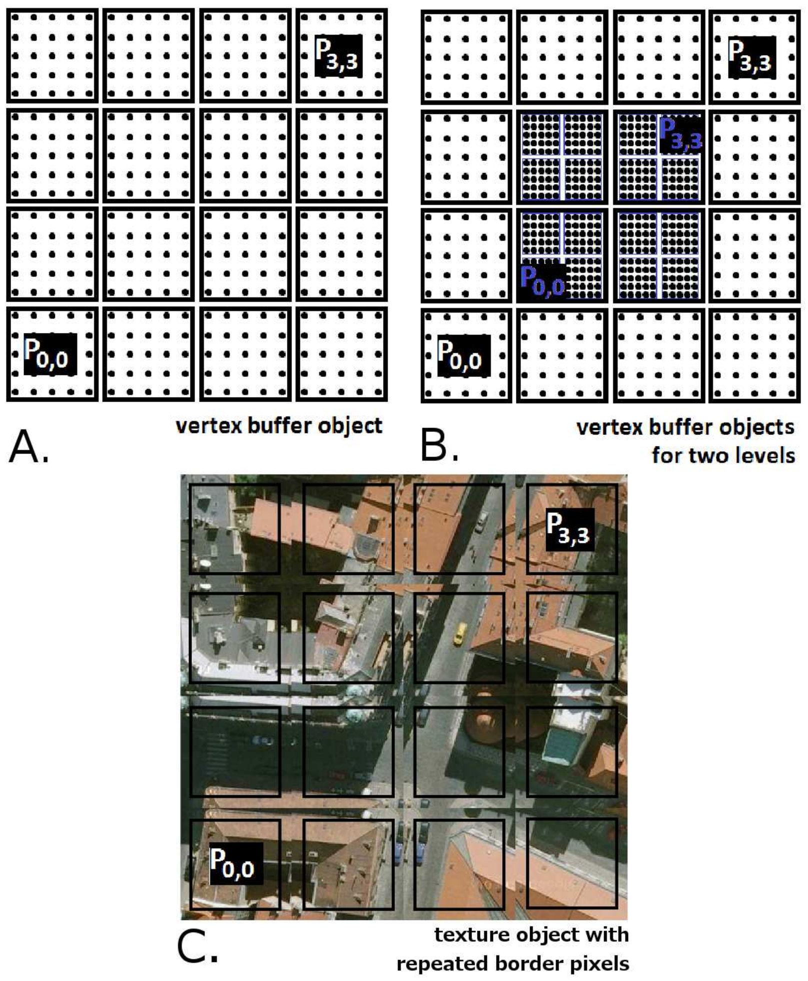


Figure 5.9: (A,C) Each patch with index $[j, k]$ covers an appropriate set of elevation points in vertex buffer object and appropriate texels in the texture object. (B) Example of two levels, the coarser level is rendered as a ring, the finest inner level is rendered using all patches. (C) Texels at border of each patch are the same between all adjacent patches because of filtering (here 8px border is used).

5.3.5 Tile-based geometry clipmap rendering

The clipmap is rendered as a set of concentric square rings centered around the viewer. Therefore, only a subset of the $M \times M$ array of patches is needed to render the rings except the finest level L_0, which is rendered completely (see Figure 5.9B.). The tile-based rendering is described using the algorithm 1 and consists of three steps.

In the first step, all N levels of the clipmap are examined from the much coarser level L_{N-1} to the level L_0 and position states of all the patches are determined. These position states are boolean values: *inside_ring* and *inside_hollow*.

In the second step, all N levels of the clipmap are examined from the much coarser level L_{N-1} to the level L_0 and the patches which are inside ring (*inside_ring*) are temporarily stated as to be rendered. Only the patches inside the hollow are further inspected. All four child patches from level L_{i-1} covered by a patch P from level L_i are tested if they can be

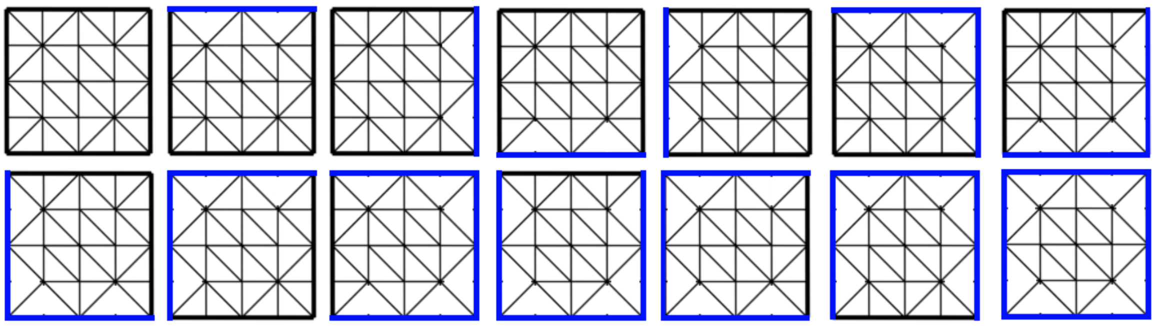


Figure 5.10: The fourteen permutations of indices precomputed for each patch.

rendered (are valid). If at least one of the four child patches is not valid, then the patch P from the level L_i will be forced to render even it is inside the hollow. We call this feature of our algorithm as selective cropping, because all the four child patches from level L_{i-1} will be forced not to render.

In the third step of the algorithm, an appropriate index buffer from the fourteen permutations will be selected to stitch the geometry of adjacent patches from two successive levels of the clipmap. The selection of the index buffer for a patch P is based on examining render states of the four adjacent patches of the patch P within the given level. The render states are boolean values: *render*, *force_render*, and *force_not_render*. Finally, if a patch has the *force_render* state set to true, then it will be rendered even it is inside the hollow. If the render state is *force_not_render*, then the patch will not be rendered even it is inside ring and is valid.

A the viewer is moving, the patches which fall outside current level boundaries are invalidated and the elevation and texture data covered by that invalidated patches must be updated with new data. As soon as the new data for an invalidated patch $P_{j,k}$ is prepared ($P.new\_data\_prepared$) by a prepare thread, the new data will be used to update the vertex buffer object (VBO) and texture object (TO) covered by the patch. The position of the update is determined from the $[j,k]$ index.

5.3.6 Preparing patches for the render rings

The viewer position in the scene is defined by [latitude, longitude] geographic coordinates. To identify the elevation data tiles which are necessary to prepare the patches, the viewer position is projected from the [latitude, longitude] geographic coordinates to a 2D space using the Mercator projection.

Let's assume, the position of a viewer in 2D space after the Mercator projection is $p = [x, y]$, where $x, y \in \mathbb{R}$. The viewer position p can be further scaled to level L_i using the following formula:

$$p_s = round(p) \backslash 2^i \tag{5.1}$$

where $p_s = [x_s, y_s]$ is the scaled point, $x_s, y_s \in \mathbb{Z}$ and i is index of the target level L_i. The level at the highest resolution has index $i = 0$.

The scaled point p_s can be used to determine an index I_c of the center patch in the scaled space as:

$$I_c = p_s \backslash (D - 1) \tag{5.2}$$

where D is number of elevation points covered by the patch in one dimension (see the subsection 5.3.3).

Algorithm 1 Adaptive clipmap rendering algorithm

```
vec2 frame_steady_pos = camera.getWorldPos2D();
for $level = N - 1$ to 0 do
    determine position states of all $M \times M$ patches
end for

for $L = N - 1$ to 0 do
    for $P = 0$ to $M \times M$ do
        if P.invalid & P.new_data_prepared then
            update VBO and TO
        end if
        if P.inside_ring then
            P.render = true
        end if
        if P.inside_hollow then
            if P.children.render != true then
                P.force_render = true
                P.children.force_not_render = true
            end if
        end if
    end for

    for $P = 0$ to $M \times M$ do
        determine index buffer for P
        P.render()
    end for
end for
```

Indices of boundary patches of the level L_i can be computed using the following formulas

$$\begin{aligned} I_{lo} &= I_c - \frac{M}{2} + \|I_c \bmod 2\| \\ I_{hi} &= I_c + \frac{M}{2} - 1 + \|I_c \bmod 2\| \end{aligned} \tag{5.3}$$

where I_{lo} is index of the bottom left boundary patch, I_{hi} is index of the top right boundary patch inside the level L_i and M is number of patches in a row within each level (see the subsection 5.3.3).

Indices of the boundary patches can be used to test if some patch is inside the level L_i i.e. set the position states of the patches. To complete the render rings, the inner patches inside the level L_i which fall inside the hollow of the ring at the level L_i (see Figure 5.11) need to be determined.

The equations 5.1- 5.4 are re-computed for level L_{i-1} to get indices of boundary patches at the finer level:

$$\begin{aligned} I_{lo_{up}} &= I_{c_{up}} - \frac{M}{2} + \|I_{c_{up}} \bmod 2\| \\ I_{hi_{up}} &= I_{c_{up}} + \frac{M}{2} - 1 + \|I_{c_{up}} \bmod 2\| \end{aligned} \tag{5.4}$$

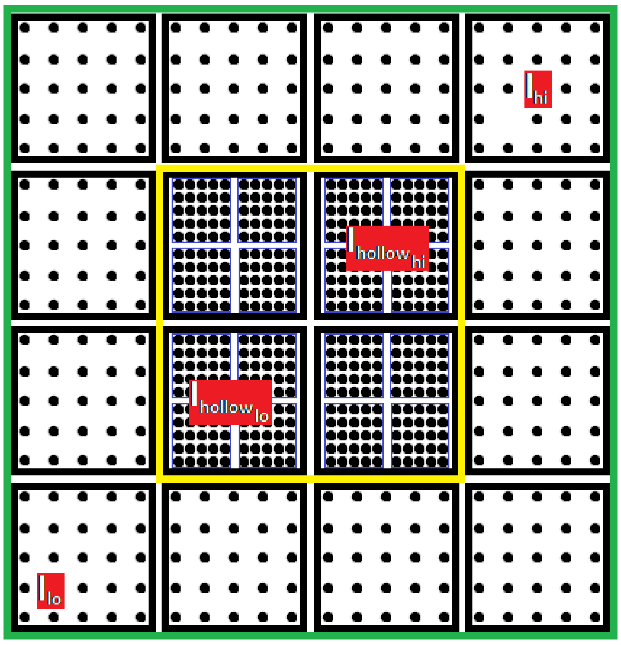


Figure 5.11: The green square bounds a level L_i, the I_{lo} and I_{hi} are indices of boundary patches of the level. The yellow square bounds the hollow inside the level L_i and the $I_{hollow_{lo}}$ and $I_{hollowhi}$ are indices of boundary patches of the hollow at level L_i.

where the subscript $I_{lo_{up}}$ and $I_{hi_{up}}$ are indices of boundary patches at level L_{i-1}. Indices of these boundary patches can be transformed to level L_i so that the transformed indices will index the boundary patches of the hollow at the level L_i:

$$\begin{aligned} I_{hollow_{lo}} &= I_{lo_{up}}/2 \\ I_{hollow_{hi}} &= I_{hi_{up}}/2 \end{aligned} \tag{5.5}$$

where the $I_{hollow_{lo}}$ is index of the left bottom patch and the $I_{hollow_{hi}}$ is index of the top right patch inside the hollow within level L_i.

The boundary patches is used to identify all the elevation tiles and texture tiles which are needed to prepare the levels for rendering. The index $I_{[x,y]}$ of a patch is used to compute a scale point p_s:

$$p_s = I_{[x,y]} * (D - 1) \tag{5.6}$$

where p_s is an elevation point placed in the bottom left corner of the patch. The server stream terrain elevation tiles with resolution of 256×256 elevation points. Then the terrain tile which contains the point p_s is computed as:

$$terain\_tile = p_s/256 \tag{5.7}$$

The texture tiles streamed from server have resolution of 256×256 texels and each texture tile covers an array of 32×32 elevation points. Then the appropriate texture tile which contains the point p_s is computed as:

$$texture\_tile = p_s/32 \tag{5.8}$$

From the bottom left and top right elevation points of the patch $P_{[x,y]}$, the terrain and texture tiles which are needed to prepare data for the patch are determined.

The indices of the outer boundary patches of current level L_i is used to test if all the patches within the level L_i cover valid elevation and texture data. If some patch covers a data which is outside current level, the toroidal update mechanism will prepare new data for the patch and the this new data are uploaded to the GPU at appropriate place at vertex and texture buffers.

5.3.7 Toroidal updates of the vertex and texture objects

During rendering, all the patches within each level are tested if they are still inside a level. If some patch $P_{[j,k]}$ with index I falls outside the I_{lo} and the I_{hi} boundaries, it is stated as invalid and is scheduled to a prepare thread. The prepare thread prepare data for the invalidated patch. The data which is needed to update the vertex and texture buffer objects is determined from the index [j,k] of the invalidated patch and the index [j,k] also specify the updated part of the pengl buffer objects. The new index I_{new} of the invalidated patch is computed using the following formula:

$$I_{new} = I_{lo} + ((M + [j,k] - (I_{lo} \bmod M)) \bmod M) \tag{5.9}$$

where the $(I_{lo} \bmod M)$ defines a patch with array index $[j_{lo}, k_{lo}]$ where current boundary patch with index I_{lo} is placed. Then the resulting $((M + [j,k] - [j_{lo}, k_{lo}]) \bmod M)$ realizes the toroidal update.

As the viewer moves, the patches which fall outside particular level boundaries are invalidated and new data are uploaded to the GPU buffer objects. The toroidal update has the advantage that the opengl buffers covered by the patches can be selectively updated so that the changes in the buffers are local and the rest of the buffers remains untouched.

Chapter 6

Prediction for location-aware data transfers scheduling

In this chapter the location-aware prediction scheme for data transfers scheduling algorithm is proposed. The location-aware prediction methods are based on the knowledge of users location history. Here, the knowledge database is composed of sequences of visited locations collected for each user. Analysis of history of these sequences of locations can be used to construct predictors which are able to predict next adjacent location of a moving user. Such kind of prediction is called next location prediction. Next location prediction methods are widely used in cellular telecommunication network for handover management, prevent call drop-off and increase quality-of-service.

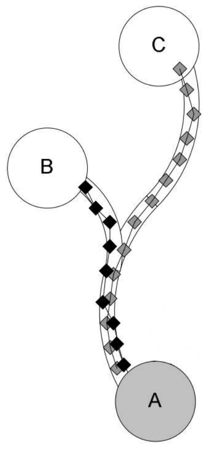
Almost all movement prediction algorithms rely on the fact that user movement is based on some pattern and it is not completely random [50]. This observation is beneficial for many applications, including also distributed virtual walkthrough applications.

In walkthrough applications, the mobility of users is usually described by spatial trajectories represented by the sequences of GPS coordinates. Such spatial trajectories can be directly used as an input of motion function based predictors. Unfortunately, the GPS trajectories cannot be used directly as an input of next location prediction methods. The reason of it is that instead of sequence of last visited positions the next location prediction methods use sequence of last visited locations to perform the prediction (see Figure 6.1a).

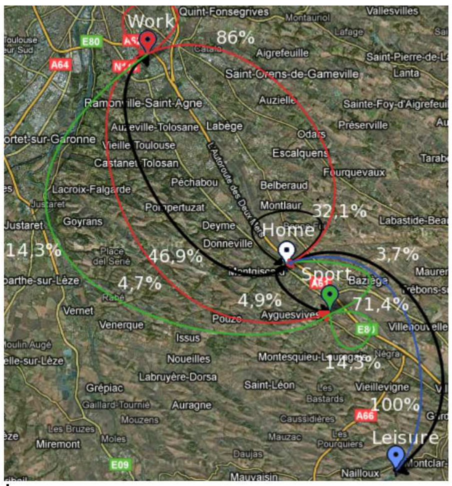
For example, the locations in telecommunication networks are determined by distribution of base transceiver stations (BTS) which form a cellular network, or by discovered places of interests in location-aware applications (see Figure 6.1b).

6.1 Knowledge database

In the preprocessing step, the knowledge database is created from encoded sequences of locations which are discovered from input GPS trajectories. This section describes how these locations can be discovered from the input GPS trajectories and how the discovered locations can be encoded to a form which is suitable for the later described next locations prediction methods.



a.



b.

Figure 6.1: The next location prediction methods use discovered locations „A“, „B“, „C“ and the motion functions based predictors use the black and gray squares for prediction (a). The probabilities of transition between discovered locations („Work“, „Home“, „Sport“ and „Leisure“) are used to decide the next most probably visited location.

6.1.1 Movement representation

The mobility of a user is defined using a GPS trajectory t which is represented by a continuous sequence of GPS coordinates:

$$t = [lat, lng]_0[lat, lng]_1...[lat, lng]_{n-1} \tag{6.1}$$

where the $[lat, lng]$ is a pair of latitude and longitude geographic coordinates, n is the length of the trajectory t and trajectory t starts at the position $[lat, lng]_0$.

The GPS trajectories can be directly used as input of motion function based predictors, but they cannot be used as input for next location prediction methods as they require sequences of visited locations.

These locations can be discovered from input GPS trajectories using various clustering mechanisms [8]. Current methods discover the visited locations mainly as places of interests (places with high density of visiting users or places where the users stay for a long time, etc.) as is shown in Figure 6.1b.

Once the clustering is finished, the input GPS trajectories are encoded into sequences of visited places of interest (e.g. $home \rightarrow work \rightarrow sport \rightarrow home$). This approach is not suitable to control data transfers scheduling for distributed virtual walkthrough applications because the density of discovered places of interests cannot cover prediction requirements during the continuous movement of a user. Therefore, instead of using the clustering methods, the locations are determined based on the scene data representation and on the rendering algorithm used.

6.1.2 Locations determination

The locations here are determined from the rendering algorithm used and data representation. As was specified earlier, the scene is rendered using a set of concentric square rings centered around the user (see the subsection 5.3.5). Each render ring requires the data tiles at specified resolution from the elevation and textures data tiles pyramid and from cartographic database.

Each location is defined as a square bounded area called location tile $l_{[x,y]}$, where $x, y \in \mathbb{Z}_{\geq 0}$ are indices to a discrete 2D space of location tiles. The borders of each location tile are determined from the smallest data tile representation. In the described rendering framework, the borders of the location tiles are determined from the area covered by single texture tile (see Figure 6.2).

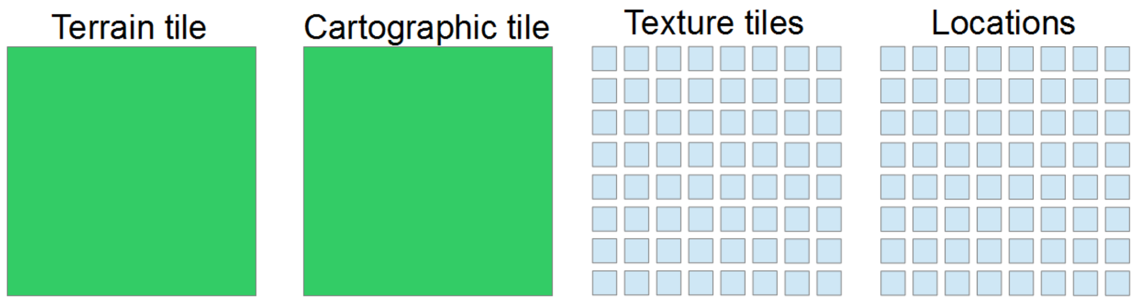


Figure 6.2: Comparison of sizes of terrain (elevation) tile, cartographic tile, texture tile and location tiles (from left to right). Each terrain tile and cartographic tile covers an array of 256×256 elevation points and each texture tile as well as each location tile covers 32×32 elevation points.

The size of the array of covered elevation points by each terrain and texture tile is determined from the source data as they are from real datasets. The texture tiles are used as a definition of the location tiles size because they are the smallest element among other tile types (elevation and cartographic tiles). It means that each elevation or cartographic tile is covered by a set of 8×8 texture tiles. The usage of the texture tiles for this task assures that when a set of required elevation or cartographic tiles is changed at some level, then the set of texture tiles has been changed as well.

6.1.3 Trajectories projection

As the size of the location tiles (covered area of elevation points) can now be determined from the size of the texture tiles, the input GPS trajectories can be transformed to a sequence of location tiles.

Each GPS trajectory is projected using Mercator projection to the discrete 2D space of location tiles so that each $[lat, lng]$ pair is projected to some location tile $l_{[x,y]}$. Afterwards, each GPS trajectory t can be encoded using a continuous sequence of location tiles:

$$T = l_{[x,y]_0} l_{[x,y]_1} ... l_{[x,y]_{m-1}} \tag{6.2}$$

where T is the projected trajectory, $l_{[x,y]_0} l_{[x,y]_1} ... l_{[x,y]_{m-1}}$ is a sequence of location tiles, m is length of the projected trajectory and $x, y \in \mathbb{Z}_{\geq 0}$. Length m of the projected trajectory T is not necessarily the same as length n of the input GPS trajectory t because more than one geographic coordinates from the GPS trajectory can be projected to the same location tile.

The projected trajectory is continuous which means that every two successive locations $l_{[x,y]_k}$ and $l_{[x,y]_{k+1}}$ are adjacent in the 2D space of location tiles so that

$$[x, y]_{k+1} - [x, y]_k = [x_{k+1} - x_k, y_{k+1} - y_k] \tag{6.3}$$

where $-1 \leq x_{k+1} - x_k \leq 1$ and $-1 \leq y_{k+1} - y_k \leq 1$.

The same input GPS trajectories are further projected into the 2D space of location tiles, but with resolution of the space decreased by power of 2 (see Figure 6.3). The data tiles assigned to these locations are taken from the higher levels (which contain coarser resolution of the data tiles) of the elevation and texture data pyramid.

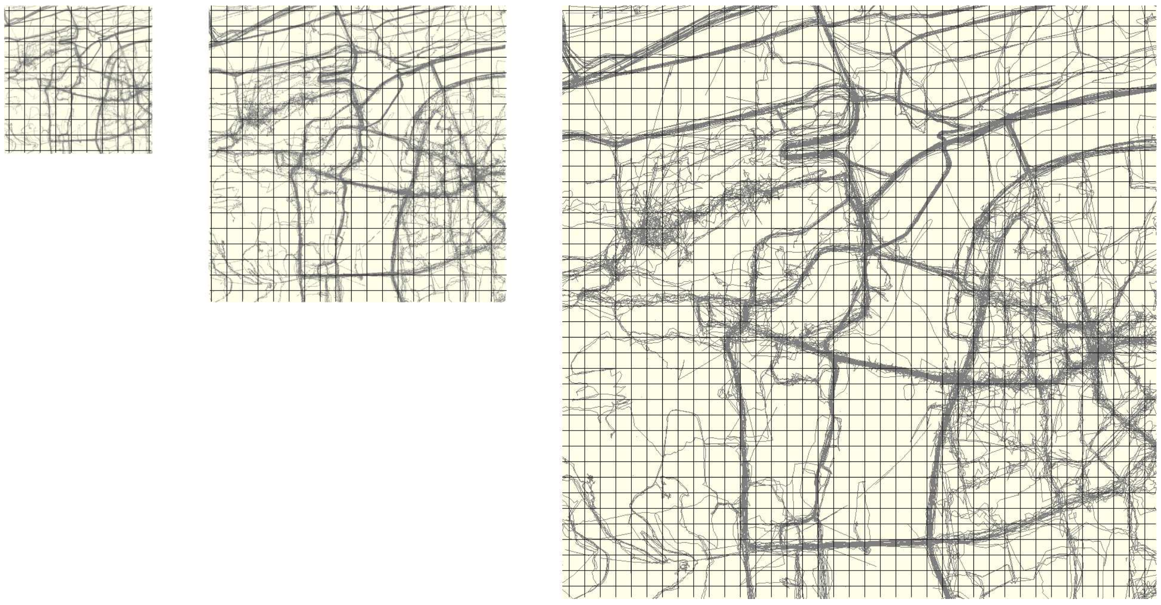


Figure 6.3: Subset of input GPS trajectories projected to location tiles at various resolution levels (increasing by power of two from left to right).

The idea of the projection at various resolution levels has an important advantage concerning the prediction. It allows the next location prediction methods to be performed at selected resolution thus allowing to compute priority or prefetch data tiles at particular resolution (see the section 7).

6.1.4 Trajectories encoding and storage

Once the projection is finished, each GPS trajectory t is represented by a set of trajectories T created at various resolution (see Figure 6.4).

Each trajectory T is further encoded by chain code of eight directions [41] (see Figure 6.5) to efficient storage, and fast evaluation of prediction queries.

For example, the red trajectory starting from left in Figure 6.4, will be encoded as a sequence of directions $2 \rightarrow 0 \rightarrow 0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 4$ at the finest resolution and as a sequence $0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 4$ at the coarser resolution.

Trajectories encoded in the form of eight directions have to be stored in a form which will be suitable for further described next location prediction methods. The general approach to solve this problem is local movement patterns.

A local movement pattern is a general solution to solve satisfactorily movement prediction during a long time movement run. Instead of remembering the movement from the beginning, the local movement pattern allows to exploit only several past locations which are potentially more important for the prediction.

This thesis proposes that the local movement patterns will be created as follows. A local

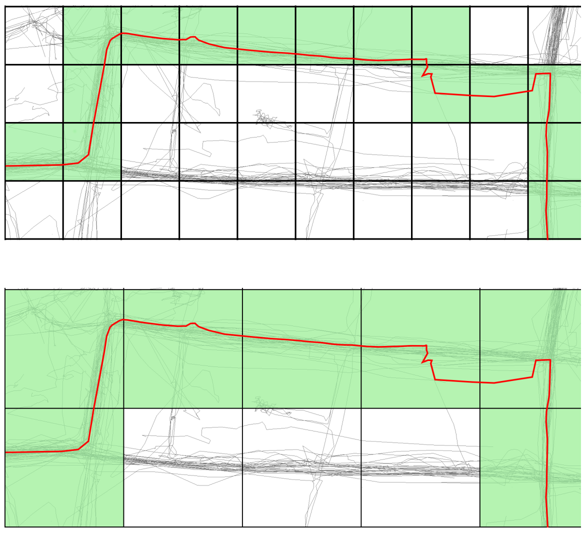


Figure 6.4: Example trajectory (red) projected into the 2D space of location tiles at finest resolution (top) and at one level coarser resolution (bottom).

7	0	1
6		2
5	4	3

Figure 6.5: Codes for eight possible movement directions to adjacent locations from current (center) location.

movement pattern M^T is computed for each location tile of the trajectory T. The local movement pattern $M^T(k)$ for location tile $l_{[x,y]_k}$ from trajectory T is defined as a sequence of 9 successive location tiles from the location tile $l_{[x,y]_k}$

$$M^T(k) = l_{[x,y]_{k+1}} l_{[x,y]_{k+2}} \dots l_{[x,y]_{k+9}} \tag{6.4}$$

where $0 \leq k < m$ and m is length of the trajectory T (see Figure 6.6).

Concerning the prediction accuracy and the results (see the section 8) it is sufficient to remember nine successive location tiles followed by the user from each tile of the trajectory T.

For storage and evaluation it is more efficient to store the sequence of nine successive directions followed from a location tile $l_{[x,y]_k}$ instead of the sequence of nine successive location tiles. The sequence of the nine successive directions and its length can be efficiently encoded using only a single 32 bit integer value which is suitable for efficient storage and fast evaluation of the prediction queries.

Each direction is encoded using 3 bits. Then a sequence of 9 directions can be stored using 27 bits plus 4 bits to encode the length d of the sequence. The length is important because length of sequences for last nine location tiles is shorter than nine tiles. This value forms a local movement pattern defined for each location tile from the trajectory T. These local movement patterns are computed for each location tile for all input trajectories projected at all resolution levels.

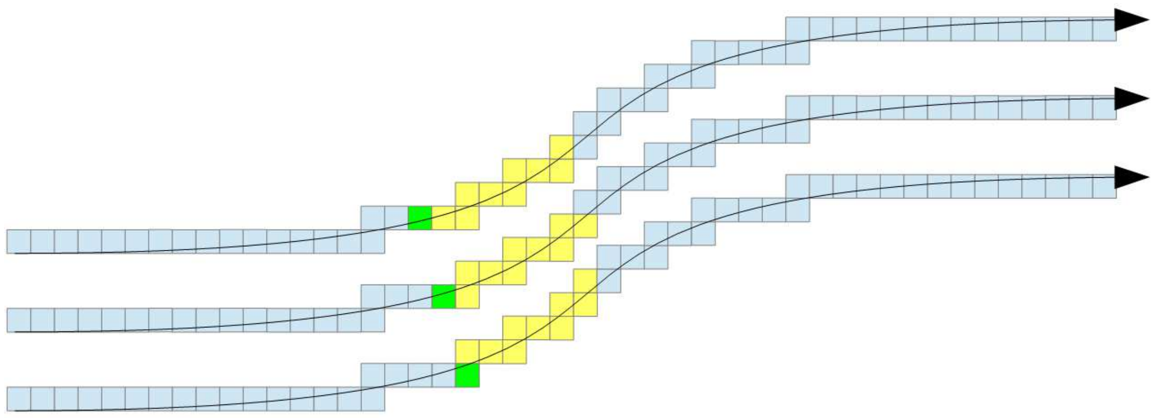


Figure 6.6: Three local movement patterns $M^T(k)$, $M^T(k+1)$ and $M^T(k+2)$ with predefined length of 9 are computed for the green location tiles.

The local movement patterns forms a knowledge database which is used as an input for the following NLP methods.

6.2 Next location prediction

The next location prediction methods suitable for distributed virtual walkthrough applications implemented within this thesis have to satisfy several requirements: efficient learning, fast adaptation to new behaviour of each individual user, high prediction accuracy and quantity, and fast evaluation of prediction queries. The most important requirement when streaming data for distributed virtual walkthrough applications is prediction accuracy, because each wrong prediction can decrease data transfers efficiency and rendered image quality.

Learning of NLP methods can be stated as efficient, if the knowledge database can be modified by adding or removing trajectories so that the prediction methods do not need to be completely relearned. Considering fundamental characteristics of different next location prediction methods [109], this requirement is met by both the Markov chain based predictor [8], [109], [43] and also by the K-state predictor [108].

6.2.1 Markov chain based predictor

A Markov chain is a sequence of random variables $X1, X2, X3, ..., Xn$ with the Markov property, namely that, given the present state, the future and past states are independent. More formally

$$Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = Pr(X_{n+1} = x | X_n = x_n) \tag{6.5}$$

where the possible values of X_i form a state space (countable set) of the chain.

The Markov chain based predictor used in this thesis as a next location prediction method is based on definition of Markov chain of order j. A Markov chain of order j is a process satisfying the following equation

$$\begin{aligned} Pr(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_0 = x_0) = \\ Pr(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_{n-j} = x_{n-j}) \text{ for } n > j \end{aligned} \tag{6.6}$$

The Markov chain of order j selects its next state depending upon j past states. Here, the state space of the Markov chain of order j is defined by all location tiles $l_{[x,y]}$ from the 2D discrete space of location tiles. Therefore, from current location tile $l_{[x,y]_n}$ the Markov chain of order j can select next location tile $l_{[x,y]_{n+1}}$ based on past j location tiles $l_{[x,y]_{n-j+1}}, l_{[x,y]_{n-j+2}}, \ldots, l_{[x,y]_n}$.

For current location tile $l_{[x,y]_n}$ the Markov chain operates with transition frequency counters which count an overall number of transitions from the current location tile $l_{[x,y]_n}$ to the eight adjacent tiles as is described by the chain code of eight directions (see Figure 6.7).

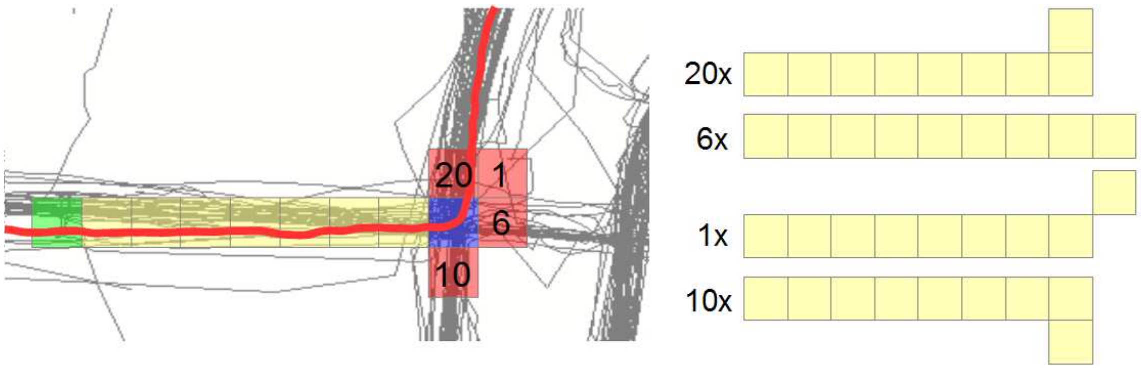


Figure 6.7: The blue one tile is current location tile $l_{[x,y]_n}$, whereas the green tile is the $(j+1)$th past location tile $l_{[x,y]_{n-j}}$ and for the red tiles the frequencies of transition from current location tile are counted.

The frequency counters computed for the current location tile $l_{[x,y]_n}$ are incremented by local movement patterns which are stored for the $(j+1)$th past location tile $l_{[x,y]_{n-j}}$. Let K is a set of local movement patterns such that

$$K = \{M^{T_0}(n-j), M^{T_1}(n-j), \ldots, M^{T_{m-1}}(n-j)\} \tag{6.7}$$

where T_k $(0 \leq k < m)$ is the trajectory from knowledge database from which the pattern $M^{T_k}(n-j)$ was taken, m is the count of local movement patterns stored for the location tile $l_{[x,y]_{n-j}}$, n identifies the current tile and j is order of the Markov chain predictor $(j = 8)$.

The frequency counters computed for the current tile $l_{[x,y]_n}$ are incremented only using a subset of patterns from the set K. Only those local movement patterns which satisfy the following condition can be used. As each local movement pattern contains nine successive directions, first eight of these directions can be applied from $(j+1)$th location tile $l_{[x,y]_{n-j}}$. If current tile $l_{[x,y]_n}$ is reached after applying these eight directions, the 9th direction can be used to increase the appropriate frequency counter (see the subsection 6.3.2 for more formal definition of the matching process).

The local movement patterns from the set K whose application leads to current location tile $l_{[x,y]_n}$ can be identified using simple bitwise operations as the local movement patterns are stored as single 32 bit integer values. Intuitively, this allows fast construction of the next location predictors by exploiting vector operations or parallelization.

After passing through all patterns of the set K, the direction with the highest value of its frequency counter should be selected as a prediction result. Unfortunately, this is not that simple. Instead, confidence of such selection has to be computed to ensure the prediction is reasonable. The confidence of the selected direction is computed as the ratio between the value of its frequency counter and the sum of frequency values from all directions

$$C(d_{sel}) = \frac{f(d_{sel})}{\sum_{i=0}^{7} f(d_i)} \tag{6.8}$$

where $C(d_{sel})$ is confidence of the selected direction d_{sel}, $f(d_{sel})$ is value of frequency counter of selected direction and $f(d_i)$ is value of frequency counter of direction d_i where the index i is related to the chain code of eight directions. Therefore, the confidence here is also the probability that the selected direction d_{sel} with the highest value of the frequency counter $f(d_{sel})$ will be followed by a moving user. The selected direction will be returned as a prediction result only if its confidence is greater than 90%:

$$C(d_{sel}) > 90\% \tag{6.9}$$

otherwise the prediction will be marked as not confident. The lower is the confidence threshold, the lower is the accuracy (see the section 8). If the confidence is not applied at all, the accuracy is decreased by 5-8% depending on which resolution of the location tiles was used for the prediction.

The confidence threshold is applied because the prediction accuracy is the most important parameter concerning the distributed virtual walkthrough applications character and also because the goal of this thesis is to keep both data transfers efficiency and rendered image quality high. If the confidence will be for example 50%, i.e. two directions should be returned as a prediction result, or if the confidence will be for example 12.5% i.e. all the directions can be returned as a prediction result, then it will be better not to return any prediction result (to keep the data transfers efficiency high).

Markov chain based predictor has a disadvantage that it cannot reflect fast to changes in habits [109] neither temporary changes (street closures because of road works, etc.) in behaviour of individual users. Consequently, this disadvantage can lead to confident but wrong prediction for relatively long time until the frequency counters reflect the changed behaviour.

6.2.2 K-state predictor

The concept of k-state predictor as a next location predictor inside a smart office building is introduced in [108]. It can be also successfully used with the trajectories projected into the 2D space of location tiles and encoded by the chain code of eight directions.

The k-state predictor is constructed as a simple finite automaton with k-states. Actually, only a 2-state predictor is used with eight contexts where each context represents one direction as is defined by the chain code of eight directions. The eight directions are used because of the tile based rendering the scene where a user will move within a regular grid of location tiles. The two states are a weak and a strong states (see Figure 6.8).

If the 2-state predictor is in strong state then the appropriate direction is returned as prediction result, otherwise no prediction is returned. The process of constructing the k-state predictor is similar to the Markov chain based predictor. Instead of increment frequency counters, the 2-state predictor switches between its contexts and states as the set K of local movement patterns is traversed. For example, consider a 2-state predictor is in undefined context at the beginning. After applying e.g. eight right directions from the local movement pattern shown in Figure 6.7 followed by single up direction, it will be switched to up context in the weak state. If the same pattern will be repeated (a user

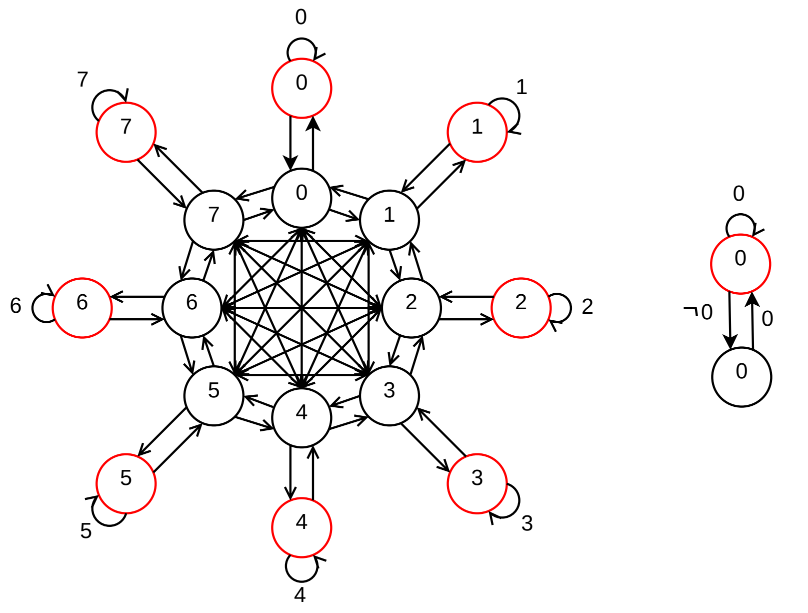


Figure 6.8: The 2-state predictor with eight strong (red) and eight week states. Each context of the predictor represent one direction from the used chain code of eight directions.

moves along the same way again) followed by the up direction again, the up context will be switched to the strong state and the up direction will be returned as a prediction result. This functionality can be easily extended by other dimension, such as time, day of a week, etc.

6.3 Proposed prediction scheme

This section describes the prediction scheme which incorporates the two Markov chain based and k-state predictors to the client-server architecture of the framework. The prediction scheme should achieve several goals:

- adaptivity to multi-resolution scene representation (perform the prediction with different granularity)
- evaluation of prediction queries in the client-server architecture
- ability to solve the case when no trajectory pattern is found

6.3.1 Adaptivity to multi-resolution scene representation

The adaptivity of the prediction scheme to the multi-resolution scene data representation is provided by projecting the input GPS trajectories to the 2D space of location tiles at various resolutions (see the section 6.1.3). The result of this approach is that each GPS trajectory t is represented by a set S of trajectories T:

$$S = \{T^0, T^1, \ldots, T^r\} \tag{6.10}$$

where T^0 is a trajectory obtained by projecting the input GPS trajectory t using Mercator projection to the 2D space of location tiles, T^1 is a trajectory obtained by projecting the input GPS trajectory t to scaled 2D space of location tiles by factor of 2^1, the T^R is a trajectory obtained by projecting the input GPS trajectory t to scaled 2D space of location tiles by factor of 2^r and r is number of available resolution levels.

Local movement patterns are created for all tiles from all projected trajectories from the set S and over all input GPS trajectories t. The Markov chain and k-state predictors can be then constructed from local movement patterns selected at a desired resolution. This allows the predictors to predict a next location tile at appropriate resolution.

During the scene walkthrough, a user forms a current GPS trajectory t_c. As the user moves, the trajectory t_c is continuously projected to the 2D space of location tiles thus forming the set S_c.

$$S_c = \{T_c^0, T_c^1, \ldots, T_c^r\} \tag{6.11}$$

For simplicity, consider only the current trajectory T_c without specifying the resolution level. Given a current location tile $l_{[x,y]_c}$ a current local movement pattern $M_c^{T_c}(c-j)$ can be extracted from the current trajectory T_c

$$M_c^{T_c}(c-j) = l_{[x,y]_{c-j+1}} l_{[x,y]_{c-j+2}} \ldots l_{[x,y]_{c-j+9}} \tag{6.12}$$

where c identify current tile, j is order of the predictor, the constant 9 is predefined local movement pattern length and $j < 9$.

Then input of the proposed prediction scheme is current local movement pattern $M_c^{T_c}(c-j)$ and output is a predicted location tile $l_{[x,y]_{c+1}}$.

6.3.2 Evaluation of prediction queries in the client-server architecture

The described Markov chain and k-state predictors have different characteristics concerning the adaptivity to changes in behaviour of users. Therefore, the 2-state predictor is used as a local predictor running at client devices whilst the Markov chain based predictor is used to model trends in behaviour of users with similar habits and it runs at server.

When constructing these predictors, the local movement patterns from the set K which is loaded for the location tile $l_{[x,y]_{c-j}}$ from local (2-state) or remote (Markov chain) knowledge database are matched with the current local movement pattern $M_c^{T_c}(c-j)$.

If the predefined length of the local movement patterns is 9 locations (or directions), then the maximum order allowed for the next location predictors is 8. For this value of the order, the equation 6.12 can be re-written as

$$M_c^{T_c}(c-8) = l_{[x,y]_{c-7}} l_{[x,y]_{c-6}} \ldots l_{[x,y]_c} l_{[x,y]_{c+1}} \tag{6.13}$$

It can be seen from the equation 6.13, that the next adjacent location tile $l_{[x,y]_{c+1}}$ to the current tile $l_{[x,y]_c}$ can be used as a prediction result for the predictors at order of 8. As the directions are used instead of the locations, the predicted direction can be computed from the difference between the location tiles $l_{[x,y]_{c+1}}$ and $l_{[x,y]_c}$. The prediction is further encoded by the chain code of eight direction.

The frequency counters and the states of the 2-state predictors can be initialized only by such local movement patterns from the set K which match the first eight locations of the current local movement pattern

$$M_c^{T_c}(c-8) = M^{T_k}(c-8) \tag{6.14}$$

$$l_{[x_c,y_c]_{c-7}} l_{[x_c,y_c]_{c-6}} \ldots l_{[x_c,y_c]_c} = l_{[x_e,y_e]_{c-7}} l_{[x_e,y_e]_{c-6}} \ldots l_{[x_e,y_e]_c} \tag{6.15}$$

From the above equations, it can be seen that the matching process compares the indices $[x_c, y_c]$ and $[x_e, y_e]$ between appropriate locations of the two patterns.

First, the 2-state predictor is constructed and evaluated based on the individual user trajectories stored in the local knowledge database. If the k-state predictor does not return any prediction result or the prediction result is marked as not confident, then the Markov chain based predictor at server will be further incorporated to the prediction process.

As the Markov chain based predictor needs the knowledge database (local movement patterns) of all users in the system, it will be constructed and evaluated at server. It receives the location tile $l_{[x,y]_{c-j}}$ from which the current local movement pattern $M_c^{T_c}(n-j)$ is determined and current location tile $l_{[x,y]_c}$, and it tries to search all the matching local movement patterns stored on server for constructing the Markov chain based predictor.

Prediction quantity of both predictors is relatively low because of lack of movement data for new users, users with new behaviour or low confidence of the predicted directions. The standard solution to this problem is prediction by partial matching and in this thesis it is further extended with application of recursive motion function.

6.3.3 Solving the case when no trajectory pattern is found

The case when no trajectory pattern is found can happen if the process of matching the set of local movement patterns K from knowledge database and the current local movement pattern $M_c^{T_c}(c-j)$ did not find any matched local movement pattern. The standard solution to this problem is a method called prediction by partial matching.

Prediction by partial matching

Prediction by partial matching (PPM) is based on shortening the length of the matched sequence of location tiles. The length of the matched sequence of location tiles is initially determined by the order of the predictors i.e $j = 8$.

Consider that the length of the matched sequence of location tiles is shortened by s locations. Then the equations 6.14 and 6.15 will be rewritten as follows

$$M_c^{T_c}(c-8, s) = M^{T_k}(c-8, s) \tag{6.16}$$

$$l_{[x_c,y_c]_{c-7+s}} l_{[x_c,y_c]_{c-6+s}} \dots l_{[x_c,y_c]_c} = l_{[x_e,y_e]_{c-7+s}} l_{[x_e,y_e]_{c-6+s}} \dots l_{[x_e,y_e]_c} \tag{6.17}$$

Both the local movement patterns from the set K and the current local movement pattern are shortened until the predictor did not return some prediction result or other conditions are met. Note that the set K contains still the local movement patterns loaded from the location tile $l_{[x,y]_{c-j}}$ i.e. $l_{[x,y]_{c-8}}$ and only the matched sequence of location tiles is shortened.

The disadvantage of this approach is that the shorter the sequence of the matched local movement patterns is, the less is the prediction accuracy. From the results, a minimal acceptable local movement pattern length can be determined so that the predictors have increased quantity, but still have high prediction accuracy (see the section 8).

If the next location predictors return a prediction result (predict a direction) by shortening the matched sequence of location tiles, such prediction is marked as not confident. Even such prediction result is not confident, it is still better than accuracy of motion function based predictors (see the section 8). Therefore, such prediction result can be used to compute priorities for downloading only missing data tiles for rendering the current view, but not for prefetching them in advance for rendering future views, because the goal of the scheduling algorithm in this thesis is to keep the data transfers efficiency high (see the section 8).

Recursive motion function

If both the Markov chain and 2-state predictors do not respond to a prediction query (i.e. no matching patterns were found), the proposed prediction scheme predicts next movement using state-of-the-art recursive motion function (RMF) [136].

Input of the RMF method is a sequence of past GPS coordinates so that it can predict next position based on motion function determined from these past GPS positions. The predicted GPS position is projected using Mercator projection to the discrete 2D space of location tiles at the finest resolution only. If the location tile determined from the projection does not differ from the current location tile, the RMF predictor is applied again to predict more steps ahead until a difference between current tile and the predicted tile occurs.

The RMF method has lower accuracy compared to the next location predictors even the prediction by partial matching is used, but it is still more accurate compared to the linear predictors. On the other side, the RMF method has high quantity compared to the next location predictors. As the prediction accuracy is the most important requirement concerning the distributed virtual walkthrough applications and also the goals of this thesis, the prediction result of the RMF method can be used only to compute priorities of currently missing data tiles at the finest resolution level.

6.3.4 Prediction scheme overview

The result of each next location predictor has three states. These three states are confident state, not confident state and empty state. Confident prediction can be used for both determining download priority of missing data tiles which are required for rendering the scene from the current user position and also for determining missing data tiles for the future user location (see chapter 7). Not confident prediction result can be used only for determining the download priority of missing data tiles needed for rendering the scene from the current user position.

The two described next location prediction methods give a confident result only if the confidence estimation of a selected direction is higher than 90% (for the Markov chain based predictor) or if the context of the 2-state predictor is in strong state and the shortening by the PPM method was not applied (see the section 8). If the next location predictors satisfy the conditions (confidence equals to 90% for the Markov predictor or the 2-state predictor is in strong state) but with applied shortenings by the PPM, the prediction is marked as not confident. Otherwise empty state is returned.

Only if the empty state is returned, the RMF method can be used for determining the download priority of missing data tiles which are required for rendering the scene from current user position at the finest resolution level only. At the coarser resolution levels, the RMF prediction result cannot be applied because of the low accuracy of the RMF method when predicting distant positions. In this case the download priority of current missing data tiles is determined from the distance between the current user position and centers of the missing data tiles needed for rendering the current view.

The prediction scheme defines that first the 2-state predictor will be evaluated at the client. If the state of its prediction result will be not confident or empty, then a server will be asked to do the prediction. In this case, the client additionally performs the RMF prediction and sends its result to the server together with current local movement pattern $M_c^{T_c}(c-j)$.

The server creates the Markov chain predictor based on the received current local movement pattern $M_c^{T_c}(c-j)$. If the prediction result is marked as confident, the server computes download priority of all the data tiles which are missing by the client for rendering the scene from the current user position. As soon as all these missing data tiles are downloaded, then the confident prediction result will be used to determine and download the missing data tiles which will be potentially required for rendering the scene if the user is moving in the direction determined by the predicted location tile $l_{[x,y]_{c+1}}$. If the prediction is not confident, then only the download priority of missing data tiles which are required for rendering the scene from the current user position is computed based on the predicted tile $l_{[x,y]_{c+1}}$. If the prediction state is empty, then the received result of the RMF method is used to compute the download priority of missing data tiles which are needed to render the scene from the current user position at the finest resolution level only.

The prediction scheme is evaluated at particular resolution levels whenever the user passes from the current location to some adjacent location at appropriate resolution level.

Chapter 7

Data transfers scheduling scheme

In this thesis, a data transfers scheduling algorithm based on the proposed prediction scheme is proposed to achieve effective data transfers with maximum rendering quality during the scene exploration within the framework.

First, the rendering algorithm determines which data tiles are missing for rendering the scene from current user position (current view). The scheduling algorithm then determines priority of download of these missing tiles based on the predicted location tile $l_{[x,y]_{c+1}}$.

As soon as all the missing data tiles required by the current view are downloaded, the scheduling algorithm uses the predicted location $l_{[x,y]_{c+1}}$ to schedule download of missing data tiles for the case the user will move in the direction determined from the current and predicted locations $l_{[x,y]_c}$ and $l_{[x,y]_{c+1}}$.

Within the client-server approach, both the client and the server can compute the download priority or schedule download data tiles in advance depending on where the prediction was done.

7.1 Rendering requirements

The current user position can be used to determine exactly all the data tiles from the three data layers which are needed to render the scene from current user position (see Figure 7.1) by the rendering algorithm described in the section 5.3.

In Figure 7.1 a user moves to the right location tile. The red point is current user position, the black square delimits the area of elevation points covered by single level of the clipmap for the current user position and the full red square marks current user location. The black square determines all the texture tiles (yellow squares), elevation and cartographic tiles (green tiles) which are required by the rendering algorithm to render the current level of the clipmap pyramid from the current user position. The light blue colour of terrain and texture data tiles indicates that the data tiles are not needed to render the given level of the clipmap. As the user moves to the right, two terrain (and also cartographic) tiles will be needed to render the level of the clipmap.

More precisely the missing data tiles for rendering the scene from the current user position are determined as follows. The rendering algorithm uses the current user position to determine current patch (see the subsection 5.3.3). Each patch covers a square bounded area of elevation point and each level of the clipmap is rendered using an array of patches. The four corner elevation points of each patch can be used to determine the data tiles needed to render the patch. By examining all the patches within a level of the clipmap, all

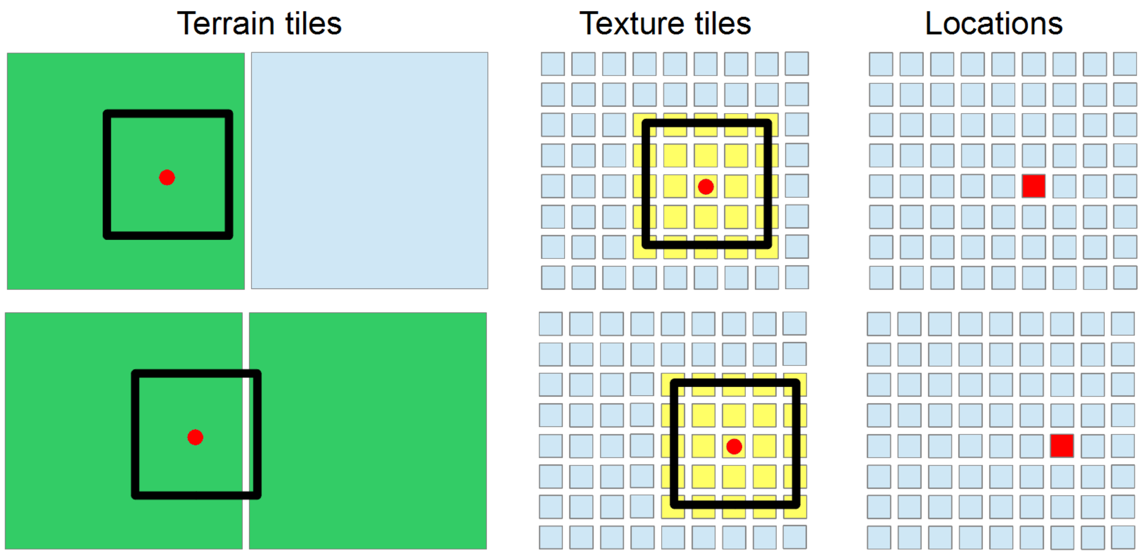


Figure 7.1: If the user moved to the right location tile, the new data tiles will be required for rendering the levels of the clipmap.

the data tiles for rendering the whole level can be determined. Practically, the size of each patch is usually smaller than the size (in terms of area of covered elevation points) of the location tiles (32×32 elevation points). The size of the patches depends on the rendering requirements and can be changed dynamically during rendering.

As the user continuously moves through the scene, the scheduling algorithm generates requests for downloading the data tiles which are not available in cache memory, but are required for rendering the scene from the current user position. The scheduling algorithm generates these requests for all the levels of the clipmap pyramid starting by generating the requests for downloading the data tiles at much coarser resolution level continuing to the finer resolutions.

7.1.1 View frustum

In the framework, the view frustum culling is used solely to increase the rendered scene frame rate. If a patch (see the subsection 5.3.3) from the given level of the clipmap is outside the view frustum, then it is not rendered, but it is still considered when the scheduling algorithm determines the missing data tiles for rendering the current view.

This behaviour is application dependent. In this framework it is expected that the scene can be viewed from the top view, where the frustum culling has no effect as well as from the first person view, where it can be considered when downloading the missing data tiles. If the view frustum culling will be incorporated into the scheduling scheme, the data tiles which are outside the current view frustum and are missing from local cache will not be downloaded.

The view frustum is not considered in the scheduling scheme neither in the prediction scheme. The reason for that is that the view frustum is likely related to the motion function based prediction. Therefore, it is not reasonable to use the view frustum for the next location prediction methods because its direction can change many times (this is still application dependent) until a user changes its location tile. Moreover, the information about the view direction for the input trajectories which are used to learn the predictors and test the scheduling method is not available.

7.2 Application of the prediction scheme

Output of the prediction scheme is represented by the predicted location tile $l_{[x,y]_{c+1}}$, by direction which is computed from the difference between the current location tile $l_{[x,y]_c}$, and the predicted tile $l_{[x,y]_{c+1}}$ with confidence of the prediction. This information is used to compute download priorities of data tiles needed for rendering the current view and to additionally prefetch all the data tiles potentially needed for rendering the future views if a user moves in the predicted direction.

7.2.1 Priority determination

For a fast moving user or slow network connection, it is not possible to transfer all the required data tiles for all levels of the clipmap in time. Even worse, some tiles required for rendering the current view will be downloaded late, so that they are no longer needed for rendering the current view but were downloaded.

To reduce such negative impact of the network restrictions, the priority of missing data tiles for the rendering current view can be computed so both the data transfers efficiency and the rendered image quality will be as high as possible. The data transfers efficiency is defined as the ratio between the amount of downloaded data tiles and how these data tiles contribute to the rendered image quality. Rendered image quality is defined as the ratio between the time for which each data tile is needed for rendering the current view and the time for which the tile is available i.e. present in local cache.

Therefore, the priority of missing data tiles for rendering the current view is determined using the predicted location tile $l_{[x,y]_{c+1}}$ as follows. The difference between the current location and the predicted location tile determines the predicted movement direction (more precisely the predicted direction is direct result of the prediction scheme). The predicted movement direction is used to compute a reference location $l_{[x,y]_r}$ which is the predicted location translated so that is lies outside the area covered by data tiles needed for rendering the current level of the clipmap (see Figure 7.2).

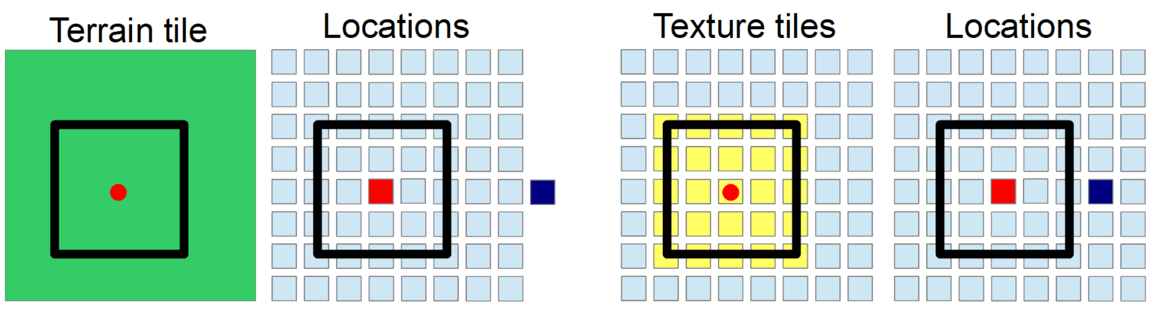


Figure 7.2: The red point is current user position. The black square delimits the area of elevation points covered by single level of the clipmap for the current user position. The red square marks current user location. The blue filled square is the reference location $l_{[x,y]_r}$ such that it lies outside the area of the data tiles needed for rendering the level of the clipmap.

The reference tile is determined individually for each type of the data tiles as is shown in Figure 7.2 (the reference location tile for the cartographic data is the same as for the terrain tile). The priority of the data tiles is then computed as a distance between center of the reference location tile $l_{[x,y]_r}$ and center of the missing data tiles. The less the distance is, the higher is the priority.

This priority determination is done for all levels of the clipmap (i.e. for all resolutions of the data tiles) if the predicted tile is for these levels available. If the prediction result is not available at some level, it cannot be decided where the used will continue within that level. I can be for example considered that the user will continue the current type of motion, but this should be rarely true especially at coarser levels. Therefore, if the direction of next movement of the user at the issued level cannot be determined, then rather than giving a higher priority to the distant tiles, the download priority is computed as a distance between the center of current location tile $l_{[x,y]_c}$ and centers of the missing data tiles. Then the smaller the distance is, the higher the priority.

7.2.2 Data prefetching

As soon as all the data tiles which are needed for rendering the current view are downloaded for all levels of the clipmap, then the data transfers scheduling algorithm can use the predicted location tile $l_{[x,y]_{c+1}}$ to prefetch additional data tiles in advance. The prefetching of the data tiles can be done only if the prediction is marked as confident.

As the current user position is used to determine the current patch within each level (i.e. render ring) of the clipmap, the predicted direction is used to determine a patch which is adjacent to the current patch along the predicted direction. The potentially missing data tiles can be then determined for the adjacent patch similarly as they are determined for the current patch (see the section 7.1).

For simplicity, consider the size of the patches equals to the size of the location tiles. Then the adjacent patch along the predicted direction equals to the predicted location (see Figure 7.3).

In Figure 7.3 the size of the patches equals to the size of the location tiles. The red square delimits the data tiles (single terrain and cartographic tiles and an array of 5×5 texture tiles) which are needed for rendering the current view from current patch (defined by the four red corner elevation points). The blue square delimits the data tiles which will be needed for rendering the level of the clipmap in the case the user enters the predicted patch (defined by the four blue corner elevation points). The green texture tiles will be prefetched if they are missing in local cache, the sets of terrain and elevation tiles remain the untouched.

The size of the patches is used to determine the missing data tiles for rendering the future view instead of the location tiles, because the patches are closely related to the rendering algorithm. If the location tiles are used to determine the future missing data tiles and size of the patches is significantly smaller than the size of the location tiles, then there is a high probability that the prefetched data tiles will not be used for rendering.

Priority of the missing data tiles which will be probably needed for rendering the scene from the predicted patch can be computed using the same principle as was described in the subsection 7.2.1. It means that the predicted location tile is moved along the predicted direction so that the reference tile $l_{[x,y]_r}$ lies outside the area covered by data tiles needed for rendering the current level of the clipmap from the predicted patch. The priority of the data tiles is then computed as a distance between the center of the reference location

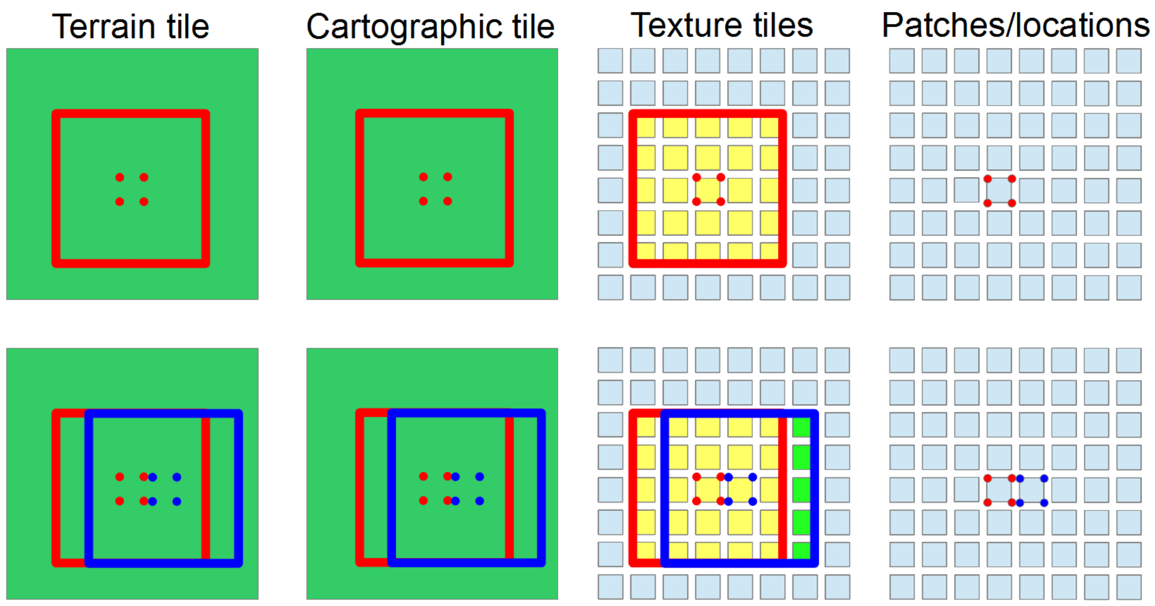


Figure 7.3: Here, the size of the patches equals to the size of the location tiles. The four red and blue points are used to determine the data tiles needed to render the current level of the clipmap.

tile $l_{[x,y]_r}$ and the center of the data tiles determined to be pre-fetched. The smaller the distance is, the higher the priority.

7.2.3 Data transfers scheduling scheme overview

Every data transfers scheduling algorithm including also the one proposed in this thesis always depends on the data representation and the rendering algorithm. The scheduling algorithm proposed here exploits the proposed location-aware prediction scheme to control download of data tiles. The goal of the scheduling algorithm is to use the predicted direction from the prediction scheme to keep both the data transfers efficiency and rendered image quality as high as possible. The prediction result determines download priority of missing data tiles for rendering the current view and also to prefetch data tiles in advance.

Chapter 8

Experimental evaluation

A proof-of-concept of the client-server streaming and rendering framework which is proposed in chapter 5 was implemented to support evaluation of the proposed location-aware data transfers scheduling algorithm. The client part of the framework can run on iPad, windows or Linux and renders the described virtual environment. It also exploits the proposed scheduling algorithm with the proposed prediction scheme to increase both quality of the rendered scene and data transfers efficiency.

The experiments concern evaluation of the prediction accuracy and quantity of the proposed prediction scheme and evaluation of the location-aware data transfers scheduling algorithm. The experiment are set so that various network restrictions are simulated and the rendered image quality and data transfers efficiency are evaluated for each experiment. Real trajectories are used to model behaviour of users.

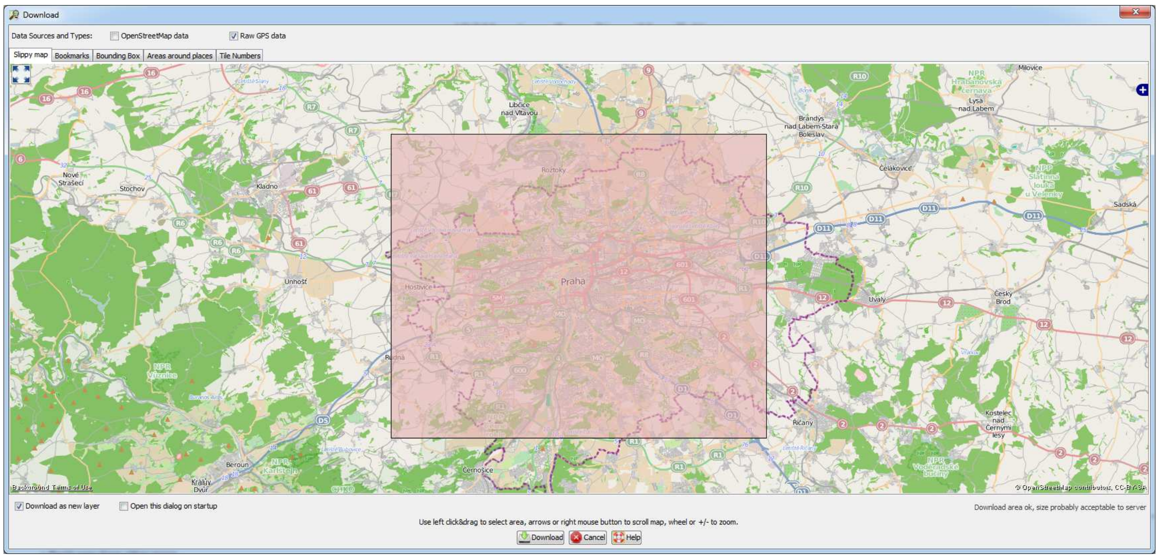
The purpose of the experiments is to prove that the proposed scheduling algorithm outperforms state-of-the-art methods for data transfers scheduling in distributed virtual walkthrough applications. The parameters which are measured are prediction accuracy and quantity of the proposed prediction scheme, and the effect of the prediction scheme to the rendered image quality and data transfers efficiency during walkthrough the environment.

As the general purpose of the scheduling schemes is to reduce the impact of network restrictions, the effect of the scheduling scheme is evaluated for various network bandwidth and latency. The second important factor which affect the rendered image quality and data transfers efficiency is the speed of a moving user. Therefore, the following experiments with the proposed scheduling algorithm are performed:

- Slow moving user, high bandwidth and high latency to prove the scheduling algorithm is efficient on fast networks, but with high latency.
- Fast moving user, high bandwidth and high latency to prove the scheduling algorithm is working as expected if large amount of data cannot be downloaded in time for rendering.
- Fast moving user, low bandwidth and zero latency to prove the scheduling algorithm is efficient also on slow networks with zero latency just to see the low bandwidth is handled by the scheduling algorithm also efficiency and outperforms the state-of-the-art scheduling scheme based on the recursive motion function method.

8.1 Input trajectories dataset

All the experiments were done with GPS trajectories obtained from the well known Open Street Map gpx database. The trajectories were downloaded from a rectangle area specified by two [latitude, longitude] coordinates. The low corner is $[49.9812545, 14.230042]$ and the high corner is $[50.182172, 14.617306]$. This area covers the city of Prague and its close neighbourhood (see Figure 8.1).



Data Sources and Types: ☐ OpenStreetMap data ☑ Raw GPS data

☑ Download as new layer ☐ Open this dialog on startup

Use left click&drag to select area, arrows or right mouse button to scroll map, wheel or +/- to zoom.

Download area ok, size probably acceptable to server

Figure 8.1: Size of the area (pink) from which the input trajectories are obtained.

The gpx trajectories dataset contains 2,648 trajectories recorded using various GPS devices. The GPS trajectories are collected by concrete users, but can be exported from the OSM database only as anonymous user. The trajectories cover local streets, highways and foot paths at various lengths.

As was specified in the section 6.1, the GPS trajectories are projected into the 2D space of location tiles so that each location can be associated with one texture tile. The area covered by each location tile (at the finest resolution) is defined as an array of 32×32 elevation points i.e. an geographic area with the size of $[0.000278, 0.000278]$ degrees per each location tile.

The number of resolution levels of the location tiles is set to 5. It is not reasonable to define more resolution levels because the length of the GPS trajectories projected to the 2D space of location tiles, which is scaled by factor higher than 2^5 will be too short.

Accuracy and quantity of the next location prediction methods is evaluated using 20-fold cross validation. Each trajectory set contains 132 GPS trajectories. The input trajectories are assigned to the sets using modulo 20 operation. The validation is performed so that 19 sets are used for learning both the Markov chain based predictor and also the k-state predictor and the remaining set is used to evaluate the prediction accuracy and quantity of the predictors. This process is repeated 20 times for each set and compute the resulting accuracy and quantity as an average value across results from all validation passes.

As the trajectories can be downloaded only as anonymous records, it is not possible to assign them to individual users. Instead, all the input trajectories are assigned to a single user and both the Markov chain based predictor and the K-state predictor are constructed with this assumption.

8.2 Prediction accuracy and quantity

The length of the local movement patterns is set to 9 location tiles (9 movement directions) which are followed (applied) from the $(j+1)$th location tile $l_{[x,y]_{c-j}}$ along the trajectory. For this length of the stored local movement patterns, the highest order j_{max} of the two defined next location predictors is $j_{max} = 8$, because the last location tile (movement direction) is reserved for constructing (frequency counters and context switching) the predictors.

The experiments over the input trajectories show that the selected length of the local movement patterns is sufficient, because when the length of a local movement pattern is shortened to 7 location tiles (or directions), the accuracy is decreased only negligibly and the quantity is slightly higher for both predictors (see Figure 8.2. and Figure 8.3.).

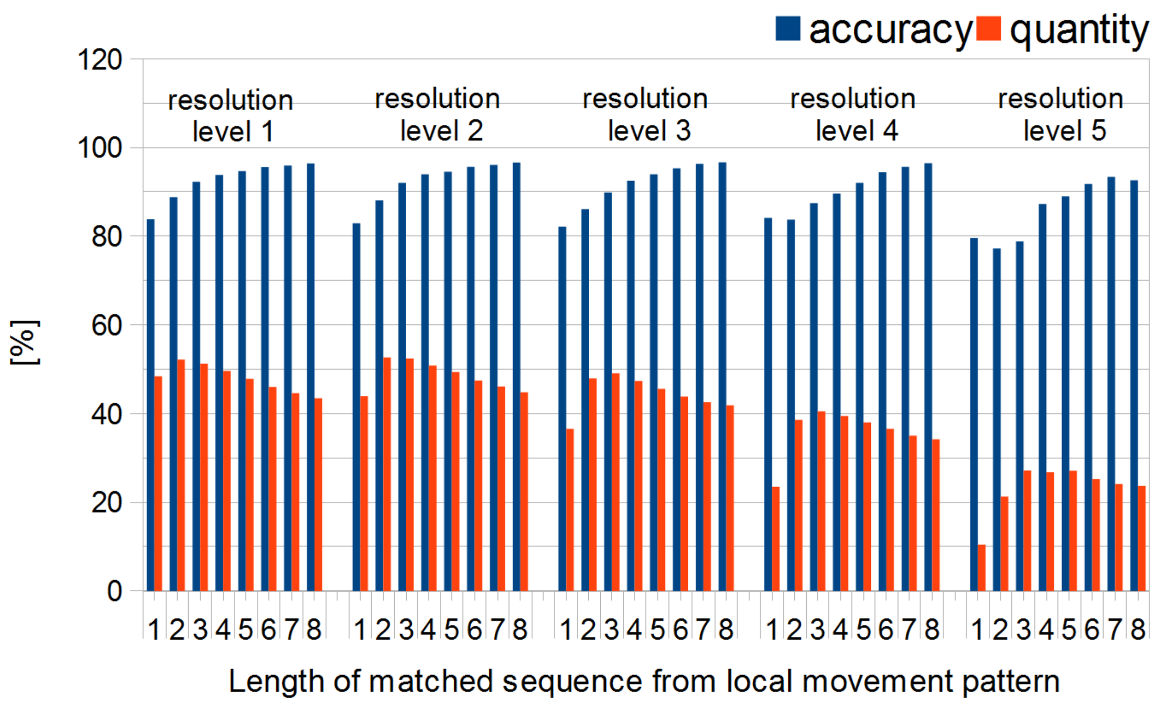


Figure 8.2: Comparison of accuracy and quantity of Markov chain based predictor for different lengths of local movement patterns and different resolution levels of the location tiles.

The quantity and accuracy in Figure 8.2 account only confident predictions. The significant quantity drops off for the shortest lengths of local movement patterns (shorter than 3 locations) is caused by increased amount of not confident predictions. The confidence threshold which was set to 90% and the low „selectivity“ of short local movement patterns are the reasons of the quantity drops off.

Accuracy and quantity were evaluated also for the 2-state predictor (see Figure 8.3) which is constructed from all local movement patterns gathered over all input trajectories. As can be seen from the results, the 2-state predictor has very low accuracy and increased quantity for the shortest lengths of the local movement patterns. Compared to the Markov chain based predictor, there is no drop off in prediction quantity for the shortest local

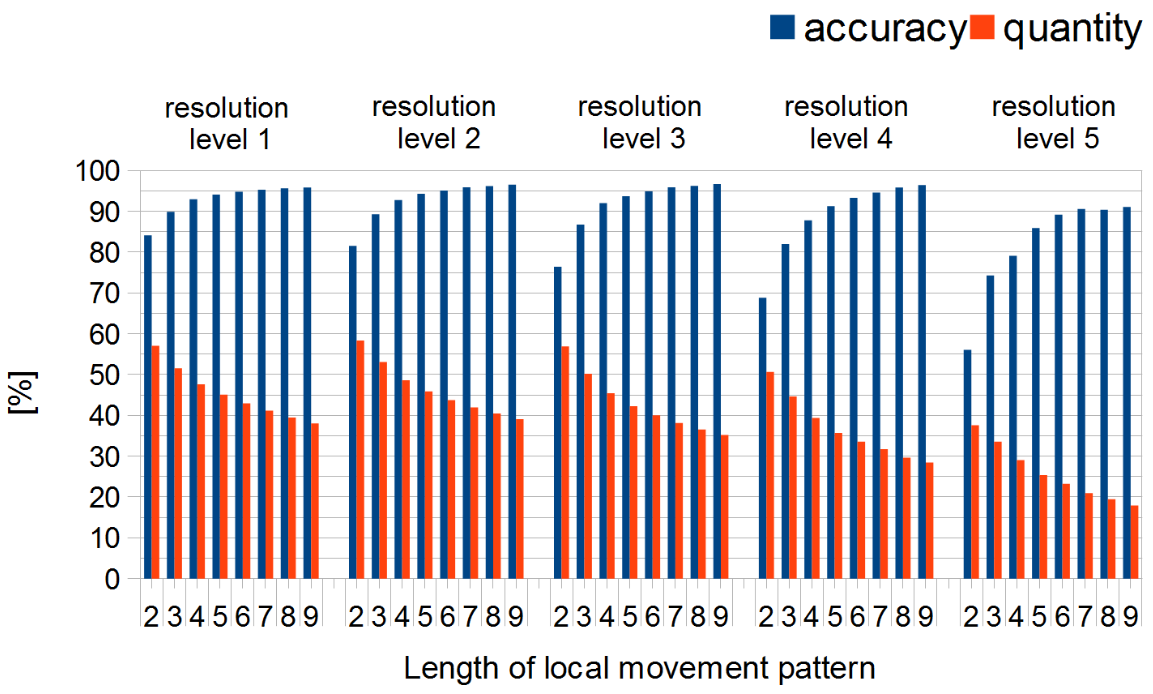


Figure 8.3: Comparison of accuracy and quantity of 2-state predictor for different lengths of local movement patterns and different resolution levels of the location tiles.

movement patterns. It is because the 2-state predictor misses the confidence estimation method which is used in the Markov chain based predictor. If the 2-state predictor has more states, then the quantity for the shortest patterns will be lower and the accuracy will be higher.

Based on this observation about the confidence, it can be said that the weaker the confidence estimation method is (e.g. the confidence threshold), the lower the accuracy.

In practice, the 2-state predictor can temporarily have low prediction quantity compared to the Markov chain based predictor especially for the new users or for users with changed behaviour.

As the accuracy is the most important parameter concerning the goals of this thesis, the length of the local movement pattern used through the next experiments is set to 7 locations. From the results, the accuracy of both the K-state predictor and Markov chain based predictor is higher than 95% for the first four resolutions of the location tiles and higher than 90% for the last resolution even though the length of the local movement pattern is set to 7 locations.

8.2.1 Prediction by partial matching

The proposed prediction scheme uses also prediction by partial matching for both Markov chain based and K-state predictors. Figure 8.4 and Figure 8.5 show effect of the PPM to accuracy and quantity of the next location predictors for all possible shortenings of the matched sequence of location tiles between current local movement pattern and the local movement patterns from the set K defined for location tile $l_{[x,y]_{c-j}}$, and for all five resolutions of the location tiles.

Initial length of the matched sequence of locations is 8 directions for local movement

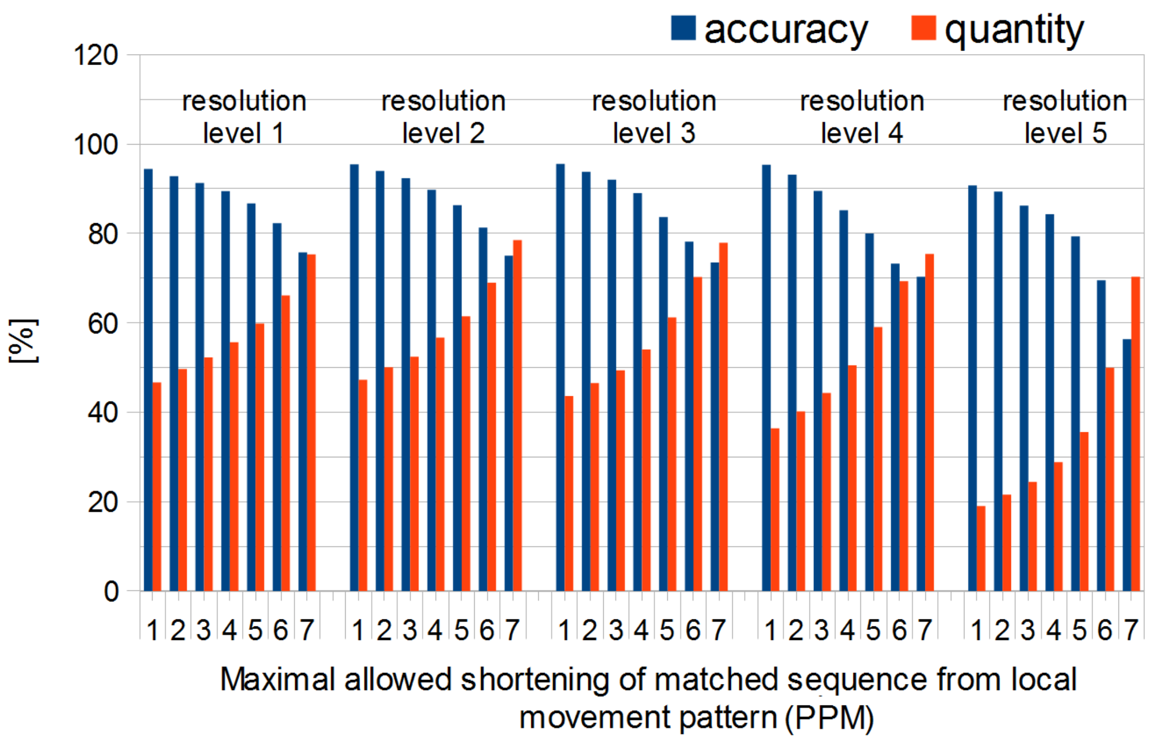


Figure 8.4: Comparison of accuracy and quantity of Markov chain based predictor for different lengths of the matched sequence of location tiles, and for different resolution levels of the location tiles.

patterns with length of 9 locations. The results show, that the higher the length of the shortening is, the lower the accuracy is but the higher the quantity of prediction is. Based on the results the similar increase in quantity is reached if the local movement patterns are shortened from 9 location tiles to 7 location tiles, or if the shortening length of the PPM is 2 locations for both predictors.

As the length of local movement patterns was set to 7 locations, then if the shortening by the PPM method was applied to return non-empty prediction, then the prediction will be marked as not confident. Such prediction result will be used to compute only priorities of data missing tiles which are needed for rendering the current view.

8.2.2 Recursive motion function and linear prediction

The recursive motion function algorithm uses six past GPS locations to create the predictor. It can be created only for the finest resolution of the location tiles, because of the character of this method.

The prediction accuracy of the recursive motion function is 55% and its quantity is 97%. Prediction accuracy of simple linear predictor is 39% and its quantity is 99%. It can be seen from the results, that even the prediction by partial matching uses the longest shortening of the matched sequence of location tiles, its prediction accuracy is still higher compared to the results of the recursive motion function. Therefore, the RMF method is used only if the next location predictors did not return any prediction results at all (no trajectory pattern is found).

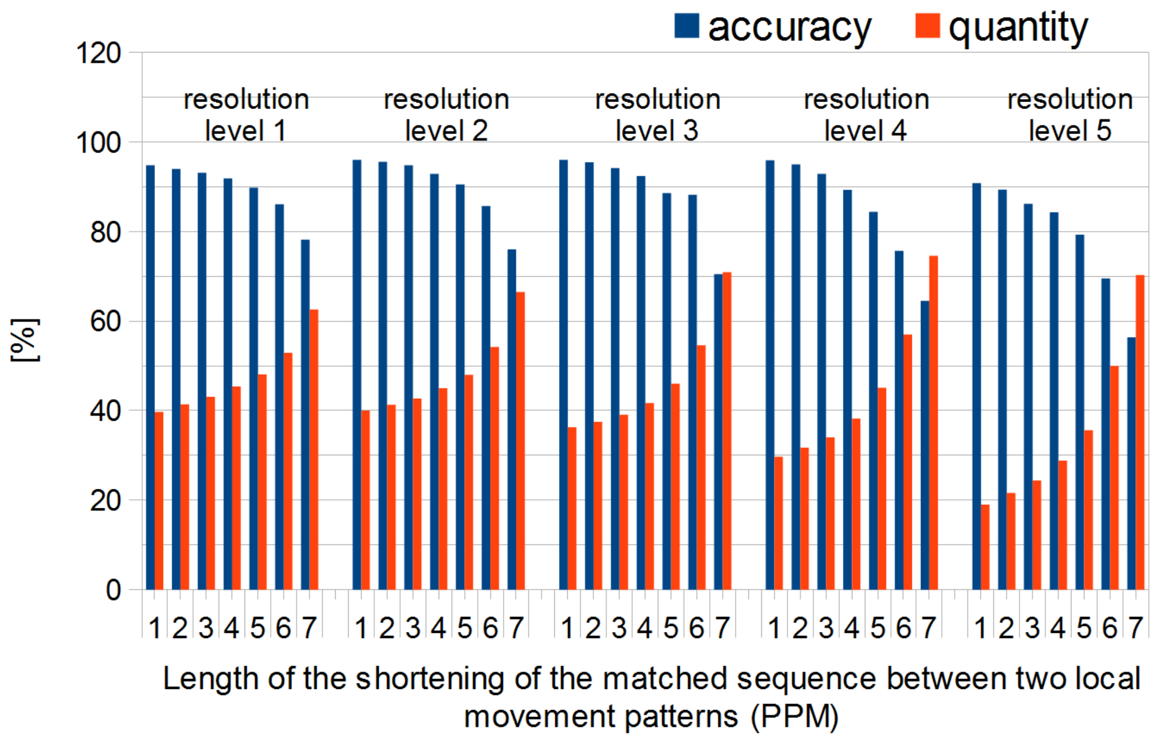


Figure 8.5: Comparison of accuracy and quantity of 2-state predictor for different lengths of the matched sequence of location tiles, and for different resolution levels of the location tiles.

8.3 Scheduling scheme evaluation

The goal of the proposed scheduling scheme is to keep both rendered image quality and data transfers efficiency as high as possible during walkthrough the described virtual environment.

The amount of time for which each data tile is rendered and the amount of time for which each data tile is available when it is needed for rendering are measured. These times are measured only for the texture tiles as they are the smallest piece of data which is needed for rendering the scene i.e. the scene cannot be rendered without textures. Moreover, the texture tiles data set is more than one order of magnitude larger than the elevation and cartographic data sets.

The rendered image quality is then defined as the ratio between the time the texture tiles are available for rendering and the time the texture tiles are needed for rendering. The data transfers efficiency is defined as the ratio between the amount of downloaded texture tiles and how these texture tiles contribute to the rendered image quality. The quality and efficiency are evaluated over all texture tiles within each level of the clipmap separately.

Several experiments were proposed to measure the impact of the location-aware data transfers scheduling scheme to the rendered image quality and data transfers efficiency. In these experiments, the parameters which are changed are motion speed of user, network latency and connection bandwidth.

In the experiments, the results of the proposed scheduling scheme are compared against a scheduling scheme based on the RMF prediction only and against a scheduling scheme which has 100% accuracy and quantity of the prediction. The 100% accurate predictor is represented by 2-state predictor learned from current trajectory so that it returns always confident prediction for each prediction query.

The experiments are performed on local area network with network latency and bandwidth emulated at linux server using „tc“ command with „netem“ kernel component. The experiments are performed over trajectories from first bundle of input GPS trajectories which are not used to learn the predictors and the results are averaged.

In the experiments, it is expected from relatively recent measurements of 3G mobile data networks [133] that average bandwidth of current 3G networks varies from approx. 1000kbps to 3000kbps and latency from approx. 50ms to 100ms depending on many conditions.

8.3.1 Settings of the rendered scene

The scene is rendered using the described rendering algorithm (see Figure 8.6). The clipmap has five levels (i.e. five resolutions of elevation and texture data tiles are required for rendering the scene) and renders also the cartographic layer (buildings). Each level is rendered using 16×16 array of patches, where each patch covers an area of 13×13 elevation points. Therefore, each level covers an area of 208×208 elevation points with texture mapped onto it with the size of 1536×1536 pixels.

The size of each texture tile is 256×256 pixels, which covers an area of 32×32 elevation points, each terrain tile covers an area of 256×256 elevation points and single cartographic tile is defined for each terrain tile (see the section 5.2 for more detailed description of the scene data). Based on the covered area by each level of the clipmap at most an array of 6×6 texture tiles, array of 2×2 terrain tiles and cartographic data tiles will be needed to render each level.

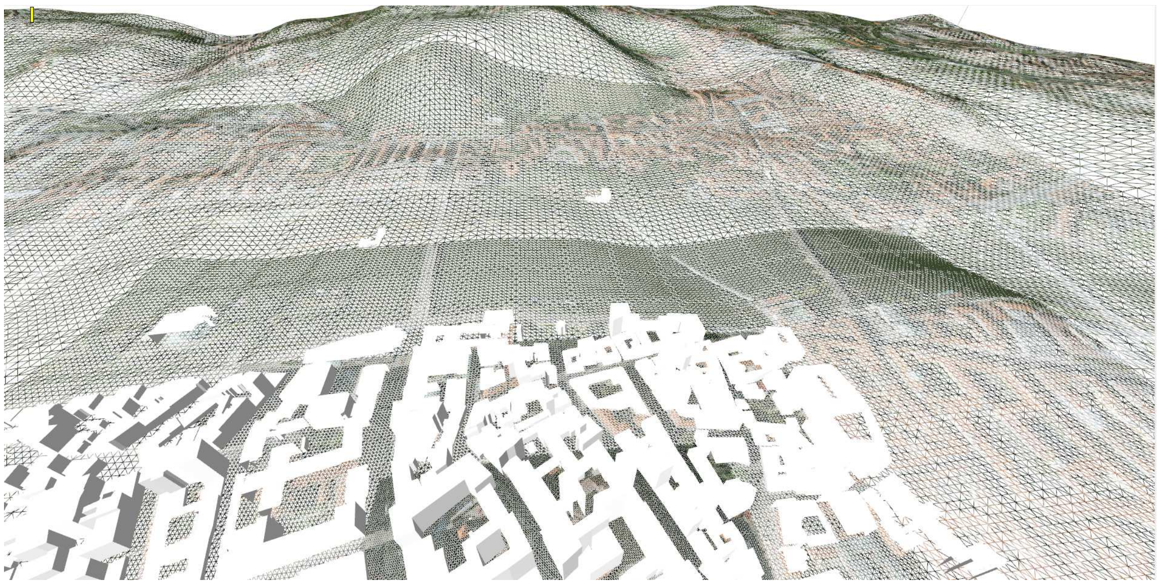


Figure 8.6: Streamed and rendered scene in wire-frame mode with the five levels where each level is rendered using an array of 16×16 patches.

8.3.2 Slow moving user, high bandwidth and high latency

In the first experiment (see Figure 8.7), the rendered image quality and data transfers efficiency are measured for motion speed which was set to 60km/h, network bandwidth to 2000kbps and latency to 100ms.

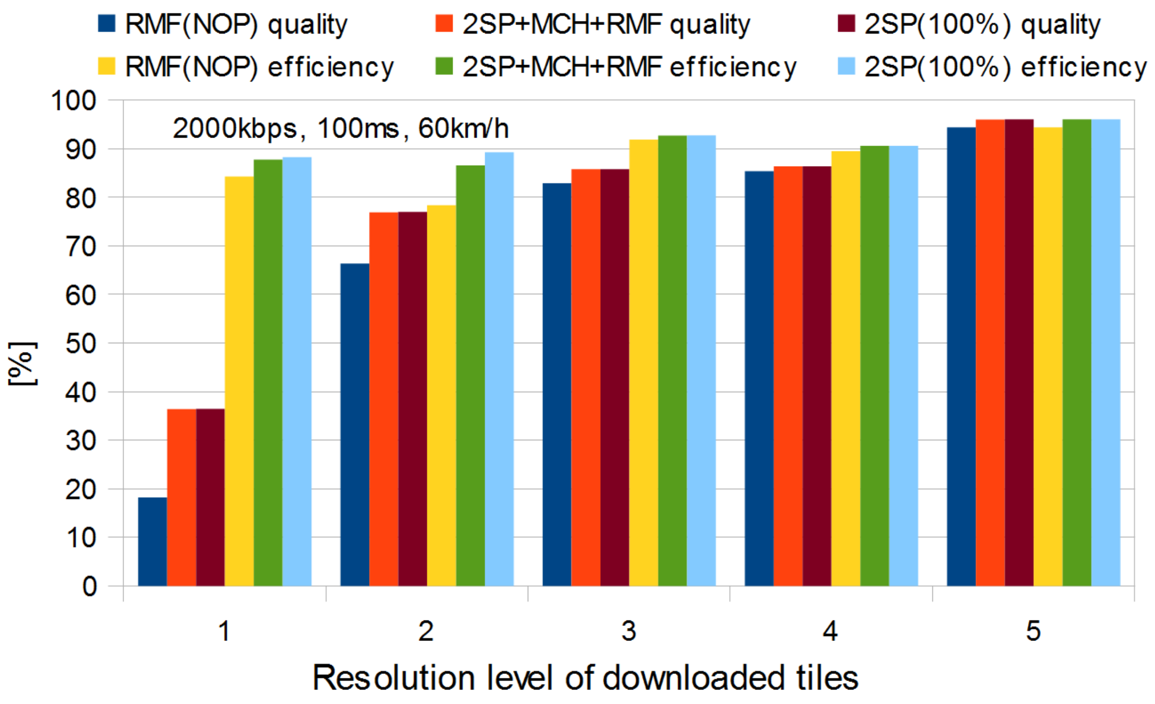


Figure 8.7: Comparison of rendered image quality and data transfer efficiency between the proposed (2SP+MCH+RMF) scheme, the RMF(NOP) scheme and the 2SP(100%) scheme for slowly moving user, high bandwidth and higher network latency.

The (2SP+MCH+RMF) is the declaration of the proposed scheduling scheme, the RMF (NOP) stands for the recursive motion function used only for the finest resolution level and no prediction (NOP) for the coarser resolutions, where the priority of missing data tiles for rendering the current view is determined from the distance between the current user position and the centers of the missing data tiles. The less the distance is, the higher the priority is. The 2SP(100%) stands for the 2-state predictor which has 100% accuracy and 100% quantity of prediction.

As the user is moving slowly and the bandwidth is high, all the data tiles for all levels of the clipmap can be downloaded so that the prefetching can download some data tiles in advance. The rendered image quality and data transfers efficiency is high for the three last levels for all three scheduling schemes. The quality of both (2SP+MCH+RMF) and the 2SP(100%) schemes is twice higher compared to the RMF(NOP) scheme.

The efficiency of all three schemes is similar over all the resolution levels. The efficiency of the RMF(NOP) is also high, because there is no data tiles prefetched in advance and the speed is sufficiently high so the requested tiles for rendering the current view are downloaded relatively fast.

The efficiency of the RMF(NOP) scheduling scheme is slightly higher for the first level compared to the second level, because the downloaded data tiles for the first level are more utilized during rendering.

The effect of the RMF on computation of download priority of tiles at the first resolution level is evaluated as well. Instead of using the RMF for the first resolution level, the priority of the data tiles is computed using the distance from the current user position and the centers of the missing data tiles. The rendered image quality drops down from the 18.2% to 3% and the data transfer efficiency drops down from 84% to only 23.4%. From this result, even the accuracy of the RMF is not as high as the accuracy of the next location prediction methods, it is still very useful for determining the download priority of the missing data tiles.

8.3.3 Fast moving user, high bandwidth and high latency

In the second experiment the motion speed is set to 130km/h which simulates a fast moving user on a highway. The results show (see Figure 8.8) that no data prefetching is applied here, because the complete set of data tiles over all levels of the clipmap was not downloaded during rendering.

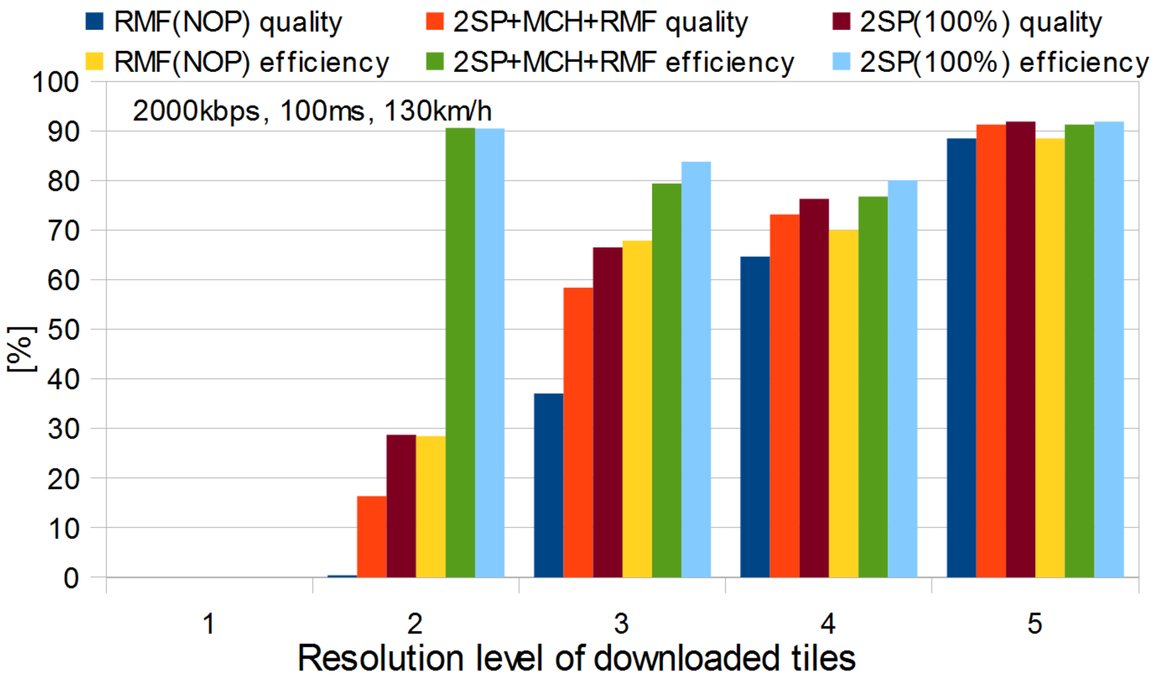


Figure 8.8: Comparison of the rendered image quality and data transfer efficiency between the proposed (2SP+MCH+RMF) scheme, the RMF(NOP) scheme and the 2SP(100%) scheme for fast moving user, high bandwidth and higher latency.

Here, the 2SP(100%) scheme has almost twice a higher quality compared to both (2SP+MCH+RMF) scheme for the second resolution level and is higher over all the resolution levels. The rendered image quality of the RMF(NOP) is very low for the second and third levels.

The efficiency of the (2SP+MCH+RMF) is significantly better compared to the RMF (NOP) scheme for the second resolution level. The efficiency of the RMF(NOP) scheme is very low for the second level because considering the quality of the RMF(NOP) scheme at the second level, the data tiles which were downloaded are not used for rendering. The reason that they are not used is that they are downloaded too late and are no more needed for rendering.

The efficiency of the (2SP+MCH+RMF) and the 2SP(100%) is similar over all resolution levels. From this results, it can be said that the proposed (2SP+MCH+RMF) scheme keeps both the rendered image quality and data transfers efficiency high considering the rendering algorithm and the rendered data.

8.3.4 Fast moving user, low bandwidth and zero latency

For the last experiment, the motion of user is set again to 130km/h, but with low bandwidth and zero latency. This experiment shows the effect to quality and efficiency of the proposed (2SP+MCH+RMF) scheduling scheme at low bandwidth networks and for fast moving users (see Figure 8.9).

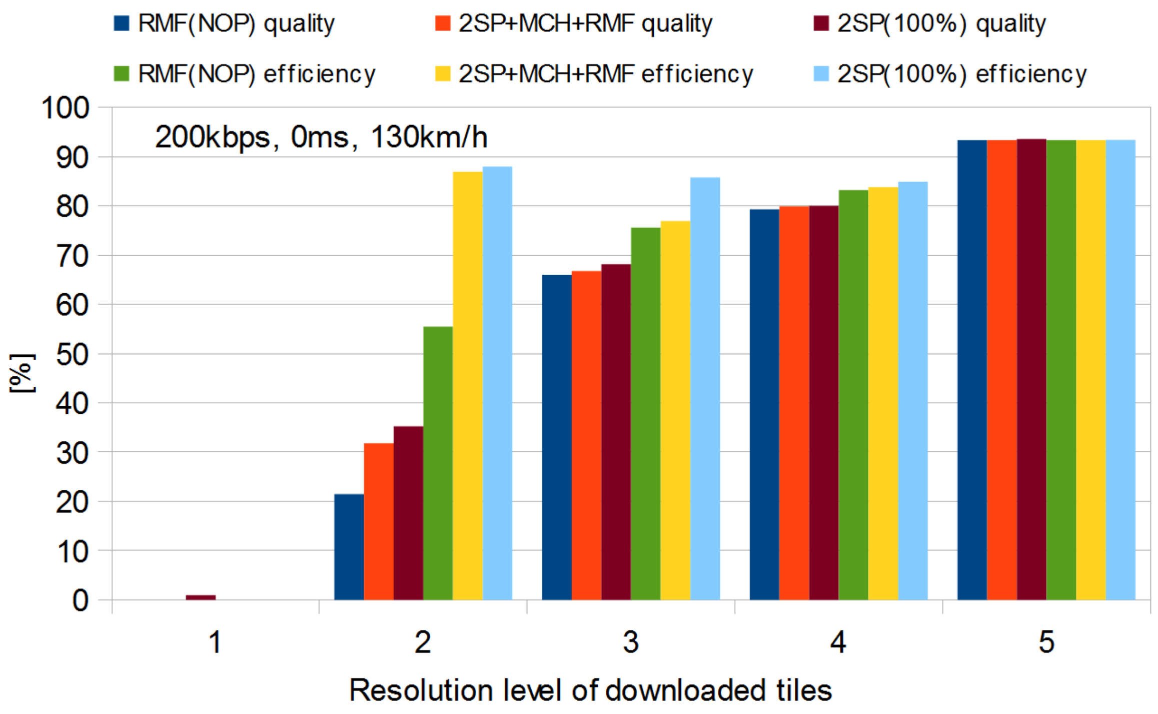


Figure 8.9: Comparison of the rendered image quality and data transfer efficiency between the proposed (2SP+MCH+RMF) scheme, the RMF(NOP) scheme and the 2SP(100%) scheme for fast moving user, low bandwidth and zero latency.

The results show that the 2SP(100%) scheme outperforms both the proposed (2SP+MCH +RMF) and the RMF (NOP) scheme. The quality of the proposed (2SP+MCH+RMF) scheme is much higher compared to the RMF(NOP) scheme at the second resolution level. The quality of all the three schemes is similar over the last three levels, because the bandwidth is sufficient to download the data tiles fast so the download priority plays a less important role in this case.

The data transfers efficiency of the (2SP+MCH+RMF) and the 2SP(100%) is much higher compared to the RMF (NOP) scheme for the second level. But the efficiency of the RMF(NOP) scheme is much better compared to the previous experiment. It is because the network delay is set to zero in this experiment. It has the consequences that the data tiles which started to be downloaded will be downloaded sooner compared to the second experiment. Therefore, they will be used for rendering for a longer time.

The terrain rendering algorithm is not tested within this thesis as the rendering performance of the proposed algorithm is not the main point of interests of this thesis. Anyway, it renders the scene at interactive frame-rates even on the iPad2 device. This allows the streaming and rendering framework to run on today's mobile devices. Please, see the attachment of this thesis for the screen-shots from the iPad device.

8.3.5 Summary of the scheduling scheme results

The results from the three experiments performed show that the proposed data transfers scheduling scheme corresponds to the expected results. It means the scheduling algorithm keeps both the data transfers efficiency and rendered image quality as high as possible. The results also show that the proposed scheduling algorithm significantly outperforms the scheduling scheme based on the state-of-the-art recursive motion function in all experiments. Compared to the ideal scheduling scheme, the proposed scheduling algorithm has very similar efficiency, but the ideal scheme outperforms the proposed algorithm in quality in some cases.

The contributions of the proposed scheduling algorithm which is based on the proposed prediction scheme are the following. First the increased quality is advantageous for users, because the scene is rendered in higher detail and second the efficiency of the data transfers allows to render the scene in higher detail with less transferred data which can be beneficial in case of pre-paid data limits.

Chapter 9

Conclusions

The goal of this thesis was to propose and experimentally evaluate the new location-aware data transfers scheduling algorithm which utilizes the proposed prediction scheme to increase both the rendered image quality and data transfers efficiency within the pre-defined multi-resolution client-server environment.

Nowadays, thanks to the sharp increase of popularity of mobile and embedded devices, the distributed virtual walkthrough applications receives an increased degree of interest. As the mobile data connections still suffer from significant network restrictions, the data transfers scheduling algorithms are the most important part of the distributed virtual walkthrough applications. As the mobile device can represent an individual behaviour of its user, the data transfers scheduling can exploit of such feature. Consequently, the main contribution of this thesis is the proposed complex prediction scheme which exploits the knowledge of individual behaviour of given user or behaviour of groups of users with the same habits to increase the rendered image quality and data transfers efficiency.

The proposed prediction scheme uses the combination of the Markov chain based predictor and the 2-state predictor as the next location predictors in the client server environment. The predictors are trained from real trajectories using local movement patterns. The local movement patterns are applied so that only the last location tile is used to construct the predictors. The local movement patterns can be used so that e.g. last two location tiles can be applied. In this case, the conditional probability of moving a user to location tiles which are adjacent to the first predicted tile can be computed. As the goals of this thesis is to keep the rendered image quality and data transfers efficiency high, the conditional probability is not computed, because it is not reasonable to perform the prediction more steps ahead with these requirements.

The experimental evaluation of the proposed location-aware data transfers scheduling algorithm outperforms the state-of-the-art scheduling scheme which is based on the recursive motion function. The data transfers efficiency of the proposed algorithm is very similar to the scheduling scheme which is based on the ideal predictor. Compared to the state-of-the-art scheduling method, the efficiency of the proposed method is about 60% bigger for the second resolution level for the fast moving user and high latency networks. The image quality of the proposed method is approximately two times bigger than the quality of the state-of-the-art method with the same setup, but it is also two times lower compared to the ideal prediction scheme. If the network restrictions are relatively negligible and the user is moving slow, the proposed method outperforms the state-of-the-art approach two times in quality. For low bandwidth networks, the increase of quality of the proposed method over the state-of-the-art method is 10% and the data transfer efficiency increases about 33%.

The significant increase of the efficiency and quality of the proposed method over the state-of-the-art method is advantageous, because the scene can be rendered in higher detail whereas the amount of transferred data is smaller compared to the state-of-the-art scheduling scheme. It can be beneficial in the case of pre-paid limits on data connections.

The advantages of the proposed scheduling scheme which exploits the next location prediction methods is that it is adaptive to behaviour of individual users. The disadvantage is that the next location predictors depend on the knowledge database which needs to be kept up to date. Especially the Markov chain based predictor evaluated at server needs GPS trajectories collected by all users of the system. This can be potentially problematic as the users of the system have to be asked to agree with that. On the other hand, the trajectories at the server can be stored as anonymous records.

Further research can be focused on exploitation of the temporal information within the prediction scheme. It can further increase the prediction accuracy which, in effect leads to the better data transfers scheduling. Another area which should receive a high degree of attention is how the predicted locations can be used to solve cache invalidation in devices with limited storage space.

Bibliography

[1] Charu C. Aggarwal and Dakshi Agrawal. On nearest neighbor indexing of nonlinear trajectories. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '03, pages 252–259, New York, NY, USA, 2003. ACM.

[2] Mohammed Eunus Ali, Egemen Tanin, Rui Zhang, and Lars Kulik. A motion-aware approach for efficient evaluation of continuous queries on 3d object databases. *The VLDB Journal*, 19(5):603–632, October 2010.

[3] Mohammed Eunus Ali, Rui Zhang, Egemen Tanin, and Lars Kulik. A motion-aware approach to continuous retrieval of 3d objects. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 843–852, Washington, DC, USA, 2008. IEEE Computer Society.

[4] Pierre Alliez and Craig Gotsman. Recent advances in compression of 3D meshes. In *In Proceedings of the Symposium on Multiresolution in Geometric Modeling*, pages 3–26, 2003.

[5] AMD Partially Resident Textures (PRT). AMD Radeon HD 7970 graphics. `http://www.amd.com`.

[6] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM.

[7] Akinori Asahara, Kishiko Maruyama, Akiko Sato, and Kouichi Seto. Pedestrian-movement prediction based on mixed markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 25–33, New York, NY, USA, 2011. ACM.

[8] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7(5):275–286, October 2003.

[9] A. Asirvatham and Hugues Hoppe. *Terrain rendering using GPU-based geometry clipmaps*. Addison-Wesley, 2005.

[10] Fabio Bettio, Enrico Gobbetti, Fabio Marton, and Giovanni Pintore. High-quality networked terrain rendering from compressed bitstreams. In *Proceedings of the twelfth international conference on 3D web technology*, Web3D '07, pages 37–44, New York, NY, USA, 2007. ACM.

[11] Amiya Bhattacharya and Sajal K. Das. Lezi-update: an information-theoretic framework for personal mobility tracking in pcs networks. *Wirel. Netw.*, 8(2/3):121–135, March 2002.

[12] Frédéric Boudon, Alexandre Meyer, and Christophe Godin. Survey on computer representations of trees for realistic and efficient rendering. Technical Report RR-LIRIS-2006-003, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumiere Lyon 2/École Centrale de Lyon, February 2006.

[13] Azzedine Boukerche and Richard W. Nelem Pazzi. Scheduling and buffering mechanisms for remote rendering streaming in virtual walkthrough class of applications. In *Proceedings of the 2nd ACM international workshop on Wireless multimedia networking and performance modeling*, WMuNeP '06, pages 53–60, New York, NY, USA, 2006. ACM.

[14] Don Brutzman. X3d: extensible 3d graphics for web authors. In *CGEMS: Computer graphics educational materials source*, CGEMS, pages 15:1–15:1. The CGEMS Project, 2008.

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

[16] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen. The simnet virtual world architecture. In *Proceedings of the 1993 IEEE Virtual Reality Annual International Symposium*, VRAIS '93, pages 450–455, Washington, DC, USA, 1993. IEEE Computer Society.

[17] Rikk Carey and Gavin Bell. *The annotated VRML 2.0 reference manual.* Addison-Wesley Longman Ltd., Essex, UK, UK, 1997.

[18] C. Carlsson and O. Hagsand. Dive a multi-user virtual reality system. In *Proceedings of the 1993 IEEE Virtual Reality Annual International Symposium*, VRAIS '93, pages 394–400, Washington, DC, USA, 1993. IEEE Computer Society.

[19] Fabien Cellier, Pierre-Marie Gandoin, Raphaëlle Chaine, Aurélien Barbier-Accary, and Samir Akkouche. Simplification and streaming of gis terrain for web clients. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, pages 73–81, New York, NY, USA, 2012. ACM.

[20] Safak Burak Cevikbas, Gürkan Koldas, and Veysi Isler. Prefetching optimization for distributed urban environments. In *Proceedings of the 2008 International Conference on Cyberworlds*, CW '08, pages 291–297, Washington, DC, USA, 2008. IEEE Computer Society.

[21] Addison Chan, Rynson W. H. Lau, and Beatrice Ng. Motion prediction for caching and prefetching in mouse-driven dve navigation. *ACM Trans. Internet Technol.*, 5(1):70–91, February 2005.

[22] Chun-Fa Chang and Shyh-Haur Ger. Enhancing 3d graphics on mobile devices by image-based rendering. In *Proceedings of the Third IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, PCM '02, pages 1105–1111, London, UK, UK, 2002. Springer-Verlag.

[23] Bing-Yu Chen and Tomoyuki Nishita. Multiresolution streaming mesh with shape preserving and qos-like controlling. In *Proceedings of the seventh international conference on 3D Web technology*, Web3D '02, pages 35–42, New York, NY, USA, 2002. ACM.

[24] Su Chen, Beng Chin Ooi, and Zhenjie Zhang. An adaptive updating protocol for reducing moving object database workload. *The VLDB Journal*, 21(2):265–286, April 2012.

[25] Wei Cheng and Wei Tsang Ooi. Receiver-driven view-dependent streaming of progressive mesh. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, pages 9–14, New York, NY, USA, 2008. ACM.

[26] Jimmy Chim, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. Cyberwalk: A web-based distributed virtual walkthrough environment. *IEEE Trans. on Multimedia*, 5:503–515, 2003.

[27] Jimmy H. P. Chim, Mark Green, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. On caching and prefetching of virtual objects in distributed virtual environments. In *Proceedings of the sixth ACM international conference on Multimedia*, MULTIMEDIA '98, pages 171–180, New York, NY, USA, 1998. ACM.

[28] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Comput. Graph. Forum*, 22(3):505–514, 2003.

[29] Gabriel Cirio, Guillaume Lavoué, and Florent Dupont. A Framework for Data-Driven Progressive Mesh Compression. In Springer, editor, *International Conference on Computer Graphics Theory and Applications (GRAPP), Lecture Notes on Computer Science*, May 2010.

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

[31] Daniel Cohen-Or, David Levin, and Offir Remez. Progressive compression of arbitrary triangular meshes. In *Proceedings of the conference on Visualization '99: celebrating ten years*, VIS '99, pages 67–72, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[32] Daniel Rákos. Hierarchical-Z map based occlusion culling. `http://rastergrid.com/blog/2010/10/hierarchical-z-map-based-occlusion-culling/`.

[33] Lucia Darsa, Bruno Costa Silva, and Amitabh Varshney. Navigating static environments using image-space simplification and morphing. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 25–ff., New York, NY, USA, 1997. ACM.

[34] W. H. De Boer. Fast Terrain Rendering Using Geometrical Mipmapping, 2000.

[35] Soumyajit Deb, Shiben Bhattacharjee, Suryakant Patidar, and P. J. Narayanan. Real-time streaming and rendering of terrains. In *Proceedings of the 5th Indian*

conference on Computer Vision, Graphics and Image Processing, ICVGIP'06, pages 276–288, Berlin, Heidelberg, 2006. Springer-Verlag.

[36] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Trans. Graph.*, 22(3):689–696, July 2003.

[37] Jürgen Döllner, Konstantin Baumann, and Klaus Hinrichs. Texturing techniques for terrain visualization. In *Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, VISUALIZATION '00, pages –, Washington, DC, USA, 2000. IEEE Computer Society.

[38] Jihad El-Sana and Amitabh Varshney. Generalized view-dependent simplification. *Comput. Graph. Forum*, 18(3):83–94, 1999.

[39] John Falby, Michael Zyda, David R. Pratt, and Randy L. Mackey. Npsnet: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers & Graphics*, 17(1):65–69, 1993.

[40] Efi Fogel, Daniel Cohen-Or, Revital Ironi, and Tali Zvi. A web architecture for progressive delivery of 3d content. In *Proceedings of the sixth international conference on 3D Web technology*, Web3D '01, pages 35–41, New York, NY, USA, 2001. ACM.

[41] Herbert Freeman. On the Encoding of Arbitrary Geometric Configurations. *IEEE Transactions on Electronic Computers*, EC-10:260–268, 1961.

[42] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 247–254, New York, NY, USA, 1993. ACM.

[43] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, MPM '12, pages 3:1–3:6, New York, NY, USA, 2012. ACM.

[44] Pierre-Marie Gandoin and Olivier Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. Graph.*, 21(3):372–379, July 2002.

[45] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[46] Michael Garland and Eric Shaffer. A multiphase approach to efficient surface simplification. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 117–124, Washington, DC, USA, 2002. IEEE Computer Society.

[47] Zihou Ge and Wenhui Li. Geometry compression method for terrain rendering with gpu-based error metric. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, pages 387–390, New York, NY, USA, 2011. ACM.

[48] Schaufler Gernot. Image-based object representation by layered impostors. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '98, pages 99–104, New York, NY, USA, 1998. ACM.

[49] Enrico Gobbetti, Fabio Marton, Paolo Cignoni, Marco Di Benedetto, and Fabio Ganovelli. C-bdam - compressed batched dynamic adaptive meshes for terrain rendering. *Comput. Graph. Forum*, 25(3):333–342, 2006.

[50] Marta C. González, Cesar A. Hidalgo R., and Albert-László Barabási. Understanding individual human mobility patterns. *CoRR*, abs/0806.1256, 2008.

[51] André Guéziec, Gabriel Taubin, Bill Horn, and Francis Lazarus. A framework for streaming geometry in vrml. *IEEE Comput. Graph. Appl.*, 19(2):68–78, March 1999.

[52] Gerd Hesina and Dieter Schmalstieg. A network architecture for remote rendering. In *Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications*, DIS-RT '98, pages 88–, Washington, DC, USA, 1998. IEEE Computer Society.

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

[54] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 189–198, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[55] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 35–42, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[56] Wei Hua, Huaisheng Zhang, Yanqing Lu, Hujun Bao, and Qunsheng Peng. Huge texture mapping for real-time visualization of large-scale terrain. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, pages 154–157, New York, NY, USA, 2004. ACM.

[57] Lok M. Hwa, Mark A. Duchaineau, and Kenneth I. Joy. Adaptive 4-8 texture hierarchies. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 219–226, Washington, DC, USA, 2004. IEEE Computer Society.

[58] ISO/IEC. *ISO/IEC FCD 15444-1, ITU-T Rec. T.800 Information technology - JPEG 2000 image coding system: Core coding system*, 2004.

[59] A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J. C. Torres, editors, *Proc. of Eurographics (Short Presentations)*, Interlaken, Switzerland, 2000.

[60] Tom Jehaes, Peter Quax, Patrick Monsieurs, and Wim Lamotte. Hybrid representations to improve both streaming and rendering of dynamic networked virtual environments. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, VRCAI '04, pages 26–32, New York, NY, USA, 2004. ACM.

[61] Stefan Jeschke. Accelerating the rendering process using impostors, March 2005.

[62] Stefan Jeschke, Michael Wimmer, Heidrun Schumann, and Werner Purgathofer. Automatic impostor placement for guaranteed frame rates and low memory requirements. In *Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 103–110. ACM, ACM Press, April 2005.

[63] Jens Jessl, Martin Bertram, and Hans Hagen. Web-based progressive geometry transmission using subdivision-surface wavelets. In *Proceedings of the tenth international conference on 3D Web technology*, Web3D '05, pages 29–35, New York, NY, USA, 2005. ACM.

[64] Jet Propulsion Laboratory. ASTER Global Digital Elevation Map. `http://asterweb.jpl.nasa.gov/gdem.asp`.

[65] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 70–79, Washington, DC, USA, 2008. IEEE Computer Society.

[66] Tomomichi Kaneko, Toshiyuki Takahei, Masahiko Inami, Naoki Kawakami, Yasuyuki Yanagida, Taro Maeda, and Susumu Tachi. Detailed shape representation with parallax mapping. In *In Proceedings of the ICAT 2001*, pages 205–208, 2001.

[67] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 271–278, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[68] Jin Kook Kim and Jong Beom Ra. A real-time terrain visualization algorithm using wavelet-based compression. *Vis. Comput.*, 20(2):67–85, May 2004.

[69] Junho Kim, Seungyong Lee, and Leif Bobbelt. View-dependent streaming of progressive meshes. In *Proceedings of the Shape Modeling International 2004*, SMI '04, pages 209–220, Washington, DC, USA, 2004. IEEE Computer Society.

[70] Fabrizio Lamberti and Andrea Sanna. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):247–260, March 2007.

[71] Rynson W. Lau and Addison Chan. Motion in games. chapter Motion Prediction for Online Gaming, pages 104–114. Springer-Verlag, Berlin, Heidelberg, 2008.

[72] Iosif Lazaridis, Kriengkrai Porkaew, and Sharad Mehrotra. Dynamic queries over mobile objects. In *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '02, pages 269–286, London, UK, UK, 2002. Springer-Verlag.

[73] S. Lazem, M. Elteir, A. Abdel-Hamid, and D. Gracanm. Prediction-based Prefetching for Remote Rendering Streaming in Mobile Virtual Environments. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 760–765, 2007.

[74] Raphael Lerbour, Jean-Eudes Marvie, and pascal Gautron. Adaptive streaming and rendering of large terrains: A generic solution. In *The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2009*. Vaclav Skala, feb 2009.

[75] Frederick W. B. Li, Rynson W. H. Lau, and Danny Kilis. Gameod: an internet based game-on-demand framework. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, pages 129–136, New York, NY, USA, 2004. ACM.

[76] H. Li, M. Li, and B. Prabhakaran. Middleware for streaming 3d progressive meshes over lossy networks. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(4):282–317, November 2006.

[77] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, August 2004.

[78] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 199–208, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[79] 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.

[80] David P. Luebke and Benjamin Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, UK, 2001. Springer-Verlag.

[81] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pages 95–ff., New York, NY, USA, 1995. ACM.

[82] Adrien Maglo, Guillaume Lavoué, Celine Hudelot, Ho Lee, Christophe Mouton, and Florent Dupont. Remote scientific visualization of progressive 3D meshes with X3D. In ACM, editor, *International Conference on 3D Web Technology (Web3D)*, July 2010.

[83] Henrique S. Malvar. Fast progressive image coding without wavelets. In *Proceedings of the Conference on Data Compression*, DCC '00, pages 243–, Washington, DC, USA, 2000. IEEE Computer Society.

[84] Jean Eudes Marvie and Kadi Bouatouch. Remote rendering of massively textured 3D scenes through progressive texture maps. In *The 3rd IASTED conference on Visualisation, Imaging and Image Processing*, volume 2, pages 756–761. ACTA Press, Sept 2003.

[85] Jean-Eudes Marvie and Kadi Bouatouch. A vrml97-x3d extension for massive scenery management in virtual worlds. In *Proceedings of the ninth international conference on 3D Web technology*, Web3D '04, pages 145–153, New York, NY, USA, 2004. ACM.

[86] Jean-Eudes Marvie, Pascal Gautron, Pascal Lecocq, Olivier Mocquard, and François Gérard. Streaming and synchronization of multi-user worlds through http/1.1. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 111–120, New York, NY, USA, 2011. ACM.

[87] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 911–918, New York, NY, USA, 2012. ACM.

[88] „Oliver Mattausch, Jiří Bittner, and Michael Wimmer“. „chc++: Coherent hierarchical culling revisited“. *„Computer Graphics Forum (Proceedings Eurographics 2008)“*, „27“(„2“):„221–230“, April „2008“.

[89] Albert Julian Mayer. Virtual texturing. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, October 2010.

[90] Leonard McMillan and Gary Bishop. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 39–46, New York, NY, USA, 1995. ACM.

[91] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 637–646, New York, NY, USA, 2009. ACM.

[92] Mikolaj Morzy. Mining frequent trajectories of moving objects for location prediction. In *Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '07, pages 667–680, Berlin, Heidelberg, 2007. Springer-Verlag.

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

[94] Beatrice Ng, Antonio Si, Rynson W.H. Lau, and Frederick W.B. Li. A multi-server architecture for distributed virtual walkthrough. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '02, pages 163–170, New York, NY, USA, 2002. ACM.

[95] José M. Noguera, Rafael Jesús Segura, Carlos J. Ogáyar, and Robert Joan-Arinyo. Navigating large terrains using commodity mobile devices. *Computers & Geosciences*, 37(9):1218–1233, 2011.

[96] José M. Noguera, Rafael J. Segura, Carlos J. Ogáyar, and Robert Joan-Arinyo. A scalable architecture for 3D map navigation on mobile devices. *Personal and Ubiquitous Computing*, pages 1–16, September 2012.

[97] Yuval Noimark and Daniel Cohen-Or. Streaming scenes to mpeg-4 video-enabled devices. *IEEE Comput. Graph. Appl.*, 23(1):58–64, January 2003.

[98] Antti Nurminen. Mobile, hardware-accelerated urban 3d maps in 3g networks. In *Proceedings of the twelfth international conference on 3D web technology*, Web3D '07, pages 7–16, New York, NY, USA, 2007. ACM.

[99] Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 19–26, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[100] Renato Pajarola and Christopher DeCoro. Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):353–368, May 2004.

[101] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *Vis. Comput.*, 23(8):583–605, July 2007.

[102] Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, January 2000.

[103] Igor S. Pandzic, Tolga K. Capin, Elwin Lee, Nadia Magnenat-Thalmann, and Daniel Thalmann. A flexible architecture for virtual humans in networked collaborative virtual environments. *Comput. Graph. Forum*, 16(3):177–188, 1997.

[104] Sungju Park, Dongman Lee, Mingyu Lim, and Chansu Yu. Scalable data management using user-based caching and prefetching in distributed virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '01, pages 121–126, New York, NY, USA, 2001. ACM.

[105] W. Pasman and F.W. Jansen. Comparing simplification and image-based techniques for 3d client-server rendering systems. *IEEE Transactions on Visualization and Computer Graphics*, 9:226–240, 2003.

[106] Jingliang Peng, Chang-Su Kim, and C. C. Jay Kuo. Technologies for 3d mesh compression: A survey. *J. Vis. Comun. Image Represent.*, 16(6):688–733, December 2005.

[107] Jingliang Peng and C.-C. Jay Kuo. Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 609–616, New York, NY, USA, 2005. ACM.

[108] Jan Petzold, Faruk Bagci, and et al. Global and local state context prediction. In *Artificial Intelligence in Mobile Systems 2003*, AIMS 2003, 2003.

[109] Jan Petzold, Faruk Bagci, Wolfgang Trumler, and Theo Ungerer. Comparison of different methods for next location prediction. In *Proceedings of the 12th international conference on Parallel Processing*, Euro-Par'06, pages 909–918, Berlin, Heidelberg, 2006. Springer-Verlag.

[110] G. Popescu. On scheduling 3d model transmission in network virtual environments. In *Proceedings of the Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, DS-RT '02, pages 127–, Washington, DC, USA, 2002. IEEE Computer Society.

[111] George V. Popescu and Christopher F. Codella. An architecture for qos data replication in network virtual environments. In *Proceedings of the IEEE Virtual Reality Conference 2002*, VR '02, pages 41–, Washington, DC, USA, 2002. IEEE Computer Society.

[112] Joachim Pouderoux and Jean-Eudes Marvie. Adaptive streaming and rendering of large terrains using strip masks. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 299–306, New York, NY, USA, 2005. ACM.

[113] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, New York, NY, USA, 1992.

[114] Jean-Charles Quillet, Gwenola Thomas, Xavier Granier, Pascal Guitton, and Jean-Eudes Marvie. Using expressive rendering for remote visualization of large city models. In *Proceedings of the eleventh international conference on 3D web technology*, Web3D '06, pages 27–35, New York, NY, USA, 2006. ACM.

[115] Lawrence R. Rabiner. Readings in speech recognition. chapter A tutorial on hidden Markov models and selected applications in speech recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[116] 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, March 1999.

[117] Robert Osfield et al. Open scene graph - open source real-time graphics middle-ware. `http://www.web3d.org`.

[118] Jérôme Royan, Patrick Gioia, Romain Cavagna, and Christian Bouville. Network-based visualization of 3d landscapes and city models. *IEEE Comput. Graph. Appl.*, 27(6):70–79, November 2007.

[119] Jérôme Royan, Christian Bouville, and Patrick Gioia. Pbtree - a new progressive and hierarchical representation for network-based navigation in densely built urban environments. *Annales des Télécommunications*, 60(11-12):1394–1421, 2005.

[120] Gernot Schaufler. Dynamically generated impostors. In *GI Workshop on Modeling - Virtual Worlds - Distributed Graphics*, pages 129–135. Infix Verlag, 1995.

[121] Gernot Schaufler and Wolfgang Stürzlinger. A three dimensional image cache for virtual reality. *Comput. Graph. Forum*, 15(3):227–236, 1996.

[122] Dieter Schmalstieg and Michael Gervautz. Towards a virtual environment for interactive world building. Technical Report TR-186-2-95-08, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, June 1995. human contact: technical-report@cg.tuwien.ac.at.

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

[124] Jens Schneider and Rüdiger Westermann. GPU-friendly high-quality terrain rendering. In *The 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006 (WSCG 2006)*, volume 14, Bory, Czech Republic, 2006.

[125] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 231–242, New York, NY, USA, 1998. ACM.

[126] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *Trans. Sig. Proc.*, 41(12):3445–3462, December 1993.

[127] François Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum*, 16:C207–C218, 1997.

[128] Sandeep Singhal and Michael Zyda. *Networked virtual environments: design and implementation.* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

[129] SkyServer. Hierarchical Triangular Mesh. `http://www.skyserver.org/htm/Old_default.aspx`.

[130] Richard Southern, Simon Perkins, Barry Steyn, Alan Muller, Patrick Marais, and Edwin Blake. A stateless client for progressive view-dependent transmission. In *Proceedings of the sixth international conference on 3D Web technology*, Web3D '01, pages 43–50, New York, NY, USA, 2001. ACM.

[131] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for computer graphics: theory and applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[132] José P. Suárez, Agustín Trujillo, Manuel de la Calle, Diego D. Gómez-Deck, and Jose M. Santana. An open source virtual globe framework for ios, android and webgl compliant browser. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, COM.Geo '12, pages 22:1–22:10, New York, NY, USA, 2012. ACM.

[133] T-mobile Czech Republic. P3 Communications and BUT: Press release - 3G and 3G networks performance testing results. `http://t-press.cz/tiskove_zpravy/2011/1224/`.

[134] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: a virtual mipmap. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 151–158, New York, NY, USA, 1998. ACM.

[135] Hai Tao and Robert J. Moorhead. Progressive transmission of scientific data using biorthogonal wavelet transform. In *Proceedings of the conference on Visualization '94*, VIS '94, pages 93–99, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[136] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 611–622, New York, NY, USA, 2004. ACM.

[137] Natalya Tatarchuk. Practical dynamic parallax occlusion mapping. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[138] Gabriel Taubin, André Guéziec, William Horn, and Francis Lazarus. Progressive forest split compression. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 123–132, New York, NY, USA, 1998. ACM.

[139] Eyal Teler and Dani Lischinski. Streaming of Complex 3D Scenes for Remote Walkthroughs. *Computer Graphics Forum*, 20(3):17–25, September 2001.

[140] Danny To, Rynson W. H. Lau, and Mark Green. An adaptive multiresolution method for progressive model transmission. *Presence: Teleoper. Virtual Environ.*, 10(1):62–74, February 2001.

[141] Danny S. P. To, Rynson W. H. Lau, and Mark Green. A method for progressive and selective transmission of multi-resolution models. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '99, pages 88–95, New York, NY, USA, 1999. ACM.

[142] T. Ulrich. Rendering massive terrains using chunked level of detail control, 2002.

[143] Visualization Sciences Group (VSG). Open inventor - 3d graphics toolkit for industrial-strength application developments. `http://www.vsg3d.com/open-inventor/sdk`.

[144] Visualization Sciences Group (VSG). Open inventor - 3d graphics toolkit for industrial-strength application developments. `http://www.vsg3d.com/open-inventor/sdk`.

[145] Simonas Šaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. *SIGMOD Rec.*, 29(2):331–342, May 2000.

[146] Wei Wang and Jinyuan Jia. An incremental smlaoi algorithm for progressive downloading large scale webvr scenes. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 55–60, New York, NY, USA, 2009. ACM.

[147] Web3D Consortium. Web3d - open standards for real-time 3d communication. `http://www.web3d.org`.

[148] Lance Williams. Pyramidal parametrics. *SIGGRAPH Comput. Graph.*, 17(3):1–11, July 1983.

[149] Andrew Wilson and Dinesh Manocha. Simplifying complex environments using incremental textured depth meshes. *ACM Trans. Graph.*, 22(3):678–688, July 2003.

[150] 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, UK, 2000. Springer-Verlag.

[151] Sung-Eui Yoon, Brian Salomon, Russell Gayle, and Dinesh Manocha. Quick-vdr: Out-of-core view-dependent rendering of gigantic models. *IEEE Transactions on Visualization and Computer Graphics*, 11:369–382, 2005.

[152] Ma Zhaoting, Li Chengming, and Pan Mao. An incremental LOD method based on grid and its application in distributed terrain visualization. *Geo-spatial Information Science*, 8:128–132, 2005.

[153] Yu Zheng and Xiaofang Zhou, editors. *Computing with Spatial Trajectories*. Springer, 2011.

[154] Zhi Zheng, Tony K. Y. Chan, and Edmond C. Prakash. Rendering of large 3d models for online entertainment. In *Proceedings of the 2006 international conference on Game research and development*, CyberGames '06, pages 163–170, Murdoch University, Australia, Australia, 2006. Murdoch University.

[155] Zhi Zheng, Prakash Edmond, and Tony Chan. Interactive view-dependent rendering over networks. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):576–589, May 2008.

[156] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering personal gazetteers: an interactive clustering approach. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, GIS '04, pages 266–273, New York, NY, USA, 2004. ACM.

[157] Minhui Zhu, Sebastien Mondet, Géraldine Morin, Wei Tsang Ooi, and Wei Cheng. Towards peer-assisted rendering in networked virtual environments. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 183–192, New York, NY, USA, 2011. ACM.

Appendix A

Scene streaming and rendering

The following video shows the streaming and rendering of the terrain:
http://www.youtube.com/watch?v=s8Ge-Vro96I&feature=youtu.be

The following video shows the rendering of the terrain from top view:
http://www.youtube.com/watch?v=ZUzSy49gsPU&feature=youtu.be

Appendix B

Scene rendering screenshots

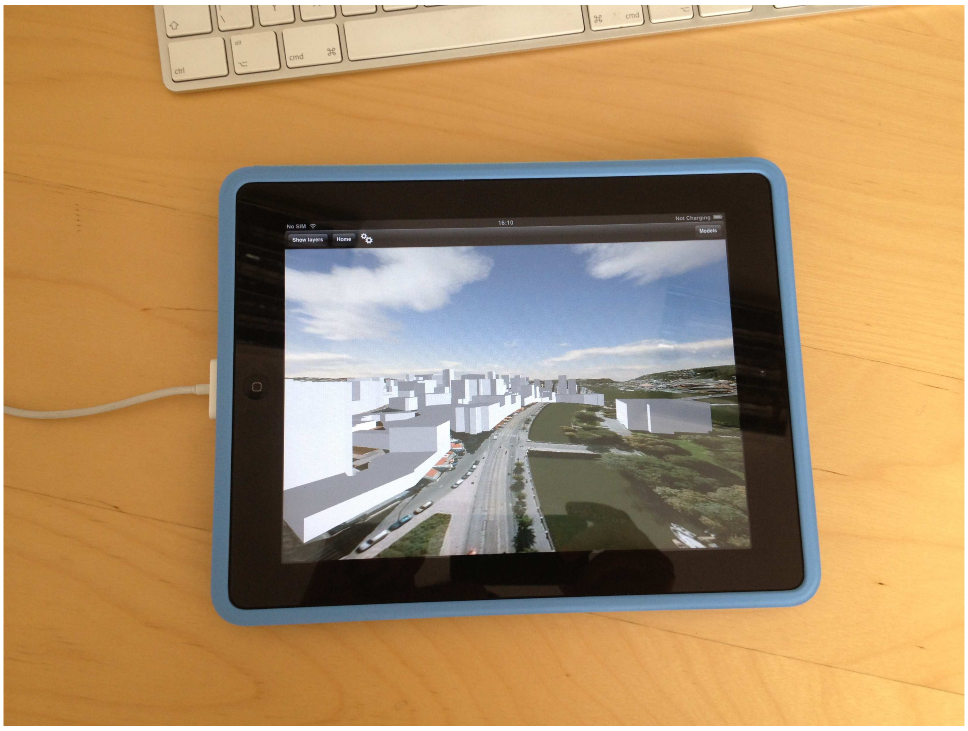


Figure B.1: Rendered terrain with buildings from user view on iPad 2 device. The data are streamed from remote server using the scheduling and prefetching mechanism proposed in this thesis.

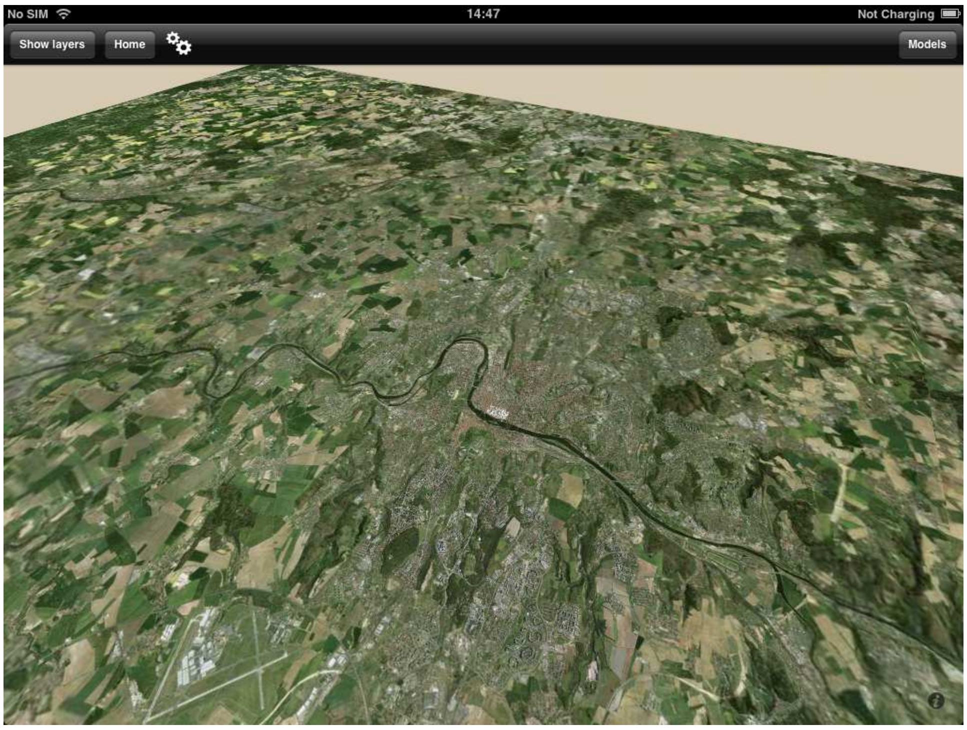


Figure B.2: Overall view of the rendered scene on iPad 2 device.

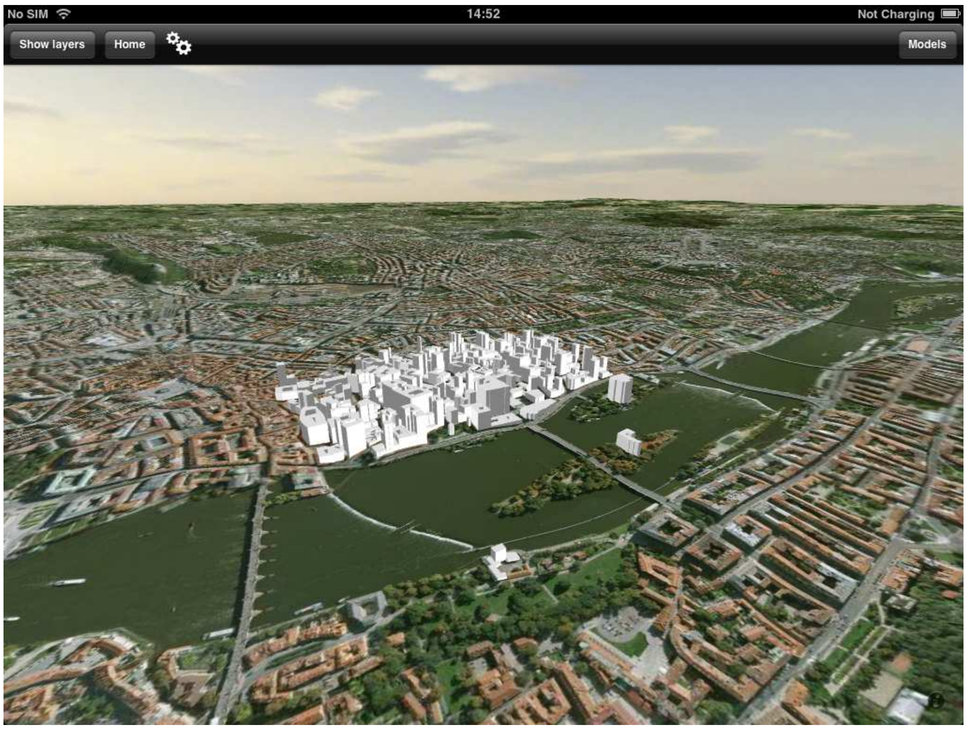


Figure B.3: Bird view of the rendered scene.

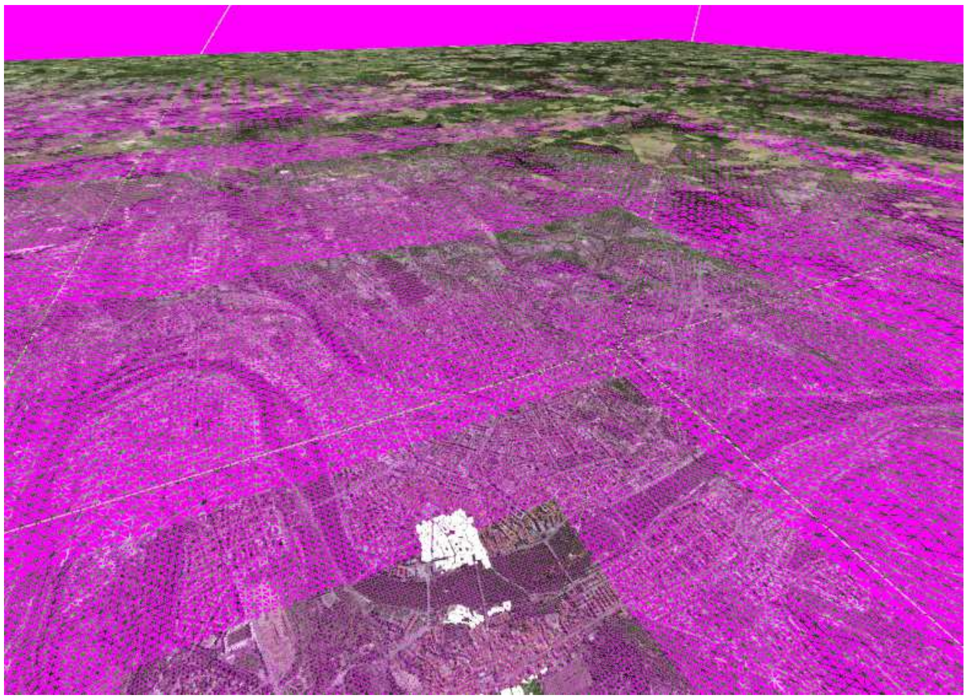
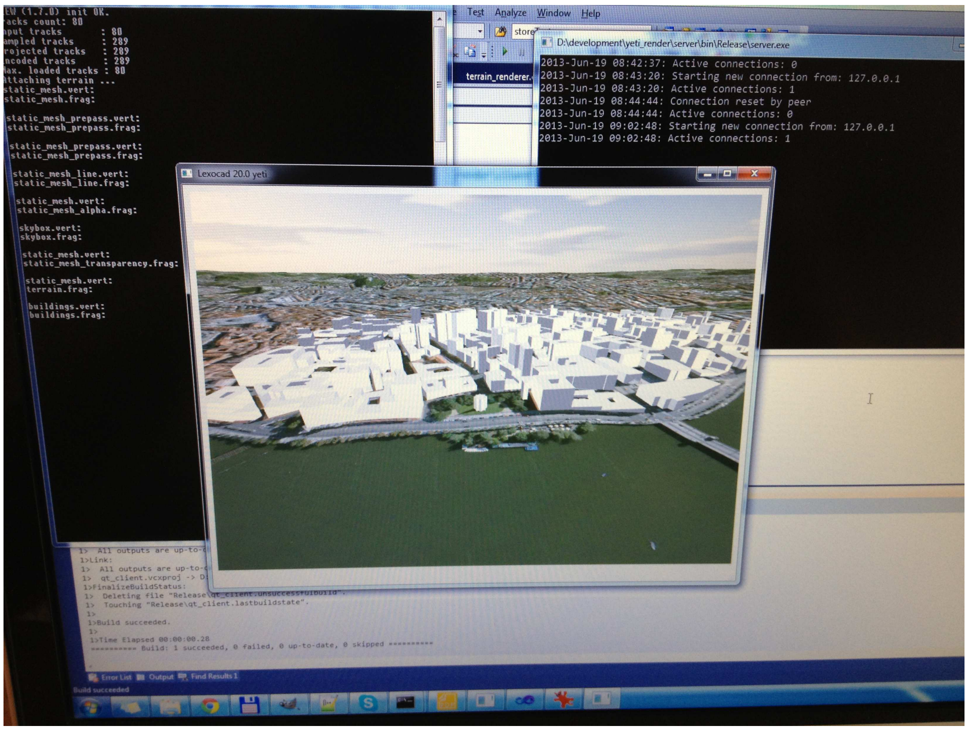


Figure B.4: Wire-frame rendering of the terrain.



```
EW (1.7.0) init OK.
racks count: 80
nput tracks         : 80
ampled tracks       : 289
rojected tracks     : 289
ncoded tracks       : 289
lax. loaded tracks : 80
Attaching terrain ...
static_mesh.vert:
static_mesh.frag:

static_mesh_prepass.vert:
static_mesh_prepass.frag:

static_mesh_prepass.vert:
static_mesh_prepass.frag:

static_mesh_line.vert:
static_mesh_line.frag:

static_mesh.vert:
static_mesh_alpha.frag:

skybox.vert:
skybox.frag:

static_mesh.vert:
static_mesh_transparency.frag:

static_mesh.vert:
terrain.frag:

buildings.vert:
buildings.frag:
```

```
D:\development\yeti_render\server\bin\Release\server.exe
2013-Jun-19 08:42:37: Active connections: 0
2013-Jun-19 08:43:20: Starting new connection from: 127.0.0.1
2013-Jun-19 08:43:20: Active connections: 1
2013-Jun-19 08:44:44: Connection reset by peer
2013-Jun-19 08:44:44: Active connections: 0
2013-Jun-19 09:02:48: Starting new connection from: 127.0.0.1
2013-Jun-19 09:02:48: Active connections: 1
```

```
1>  All outputs are up-to-
1>Link:
1>  All outputs are up-to-
1>  qt_client.vcxproj -> D:
1>FinalizeBuildStatus:
1>  Deleting file "Release\qt_client.unsuccessfulbuild".
1>  Touching "Release\qt_client.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:00.28
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure B.5: Rendering of the terrain on windows.