Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Miles of Tiles“ Tessellation in Geoinformatics

vorgelegt von

Katja Neubarth, BSc
U1389, UNIGIS MSc Jahrgang 2008

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Innsbruck, 04.10.2010

Statement of Originality

I hereby certify that the content of this thesis is the result of my own work. This thesis has not been submitted for any degree or other purposes, neither in whole nor in part. To the best of my knowledge and belief, it contains no ideas, techniques, quotations or any other material from the work of other people unless acknowledged in accordance with standard referencing practices.

Salzburg, 1st October, 2010

Katja Neubarth

Contents

List of Figures

List of Tables

Acronyms

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

BLOB Binary Large Object

BMP Bitmap

CSV Comma-separated Values

DBMS Database Management System

DGG Discrete Global Grid

DGGS Discrete Global Grid System

DTM Digital Terrain Model

DVD Digital Versatile Disc

ECW Enhanced Compression Wavelet

EPSG European Petroleum Survey Group

EXIF Exchangeable Image File Format

FAT File Allocation Table

GCP Ground Control Point

GDAL Geospatial Data Abstraction Library

GIF Graphics Interchange Format

GIS Geographic Information System

GiST Generalised Search Tree

GML	Geography Markup Language
GUI	Graphical User Interface
HFA	Hierarchical File Architecture
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JFIF	JPEG File Interchange Format
JIF	JPEG Interchange Format
JPEG	Joint Photographic Experts Group
KML	Keyhole Markup Language
KVP	Key Value Pair
LBS	Location-based Service
MBR	Minimum Bounding Rectangle
MIME	Multi Purpose Internet Mail Extension
MrSID	Multi resolution Seamless Image Database
NTFS	New Technology File System
OGC	Open Geospatial Consortium
OGP	International Association of Oil and Gas Producers
OSGeo	Open Source Geospatial Foundation
PNG	Portable Network Graphics
PRJ	Projection
QTM	Quaternary Triangular Mesh
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SPOT5	Système Probatoire d'Observation de la Terre 5

SQL	Structured Query Language
SRS	Spatial Reference System
SVG	Scalable Vector Graphics
TIFF	Tagged Image File Format
TIN	Triangulated Irregular Network
TMS	Tiled Map Service
TOAST	The Oversized-Attribute Storage Technique
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USGS	United States Geological Survey
UTM	Universal Transversal Mercator
WFS	Web Feature Service
WGS84	World Geodetic System 1984
WKB	Well-known Binary
WKT	Well-known Text
WMS	Web Map Service
WMTS	Web Map Tile Service
XML	Extensible Markup Language
ZOT	Zenithial OrthoTriangular

Abstract

Tessellation in all its aspects plays an important yet largely hidden part in geoinformatics. This thesis aims to shed light on the basics, methods and applications of tessellation. Tiling, as tessellation is sometimes called, provides the base for primary data structures such as raster data, tiled raster maps and digital elevation models. Concepts based on tessellation play an important part in secondary data structures which are used in spatial databases as well as in raster datasets.

The thesis gives an overview of the basic concepts of tessellation. This involves different scientific disciplines: mathematics, computer science and information theory as well as geographic information science. The fundamental root of tessellation lies in mathematics, with geometry providing basic terms and definitions. Computer science comes into play with image processing methods and algorithms to produce, store and access tiles. Above all geographic information science provides the common denominator for all these disciplines and adds some ingredients of its own, namely spatial access methods, spatial reference systems and georeferencing.

These concepts find many practical applications in the field of geoinformatics. Some of the best known and fastest map services on the Internet – such as Google Maps and Bing Maps – are based on tiled raster maps. They provide an easy possibility to publish all kinds of maps on the Internet. As tiled raster maps have become quite popular over the last few years, efforts have been made by international organizations such as the Open Geospatial Consortium to standardize tile services. Standards such as the OGC's Web Map Tiling Service Implementation Standard address the need for open standards for tiled map services. As tiled web map services constitute one of the most popular uses of tessellation, they are given special consideration as an applications of tiles. The steps involved in generating tiles from a raster image, the actual fragmentation of the image and questions concerning storage of the finished tiles are discussed. The reassembly of tiles into a seamless map has its place as well as tile naming schemes employed by open standards and proprietary services.

Kurzfassung

Tessellation spielt in der Geoinformatik eine wichtige, wenn auch versteckte Rolle. Diese Arbeit will die Grundlagen, Methoden und Anwendungen der Tessellation erläutern. Kachelung, ein Synonym für Tessellation, birgt die Grundlagen für Primärdatenstrukturen wie Rasterdaten und digitale Höhenmodelle. Konzepte die auf Tessellation beruhen spielen auch in Sekundärdatenstrukturen eine wichtige Rolle. Diese sekundären Datenstrukturen werden sowohl in räumlichen Datenbanken als auch bei Rasterdaten eingesetzt.

Die Arbeit gibt eine Übersicht über die grundlegenden Konzepte der Tessellation. Diese kommen aus unterschiedlichen wissenschaftlichen Disziplinen: Mathematik, Informatik und Informationstheorie, sowie Geoinformatik. Die Wurzeln der Tessellation sind in der Mathematik zu finden, genauer in der Geometrie, die die grundlegenden Begriffe und Definitionen liefert. Die Informatik steuert Bildbearbeitungsmethoden und Algorithmen bei, die der Erstellung, Speicherung und dem Zugriff auf Kacheln dienen. Die Geoinformatik ist der gemeinsame Nenner für diese unterschiedlichen Fachrichtungen, und trägt eigene Techniken bei: Georeferenzierung, räumliche Bezugssysteme und Zugriffsmethoden.

Diese Konzepte haben einige praktische Anwendungen in der Geoinformatik. Einige der bekanntesten und schnellsten im Internet zugänglichen Kartendienste – wie Google Maps und Bing Maps – basieren auf gekachelten Rasterdaten. Kacheln bieten eine einfache Möglichkeit, um Karten im Internet zu veröffentlichen. Gekachelte Karten sind in den letzten Jahren immer bekannter und häufiger verwendet worden, daher haben internationale Organisationen wie z.B das Open Geospatial Consortium Anstrengungen unternommen gekachelte Kartendienste zu standardisieren. Das Ergebnis dieser Bemühungen ist der Web Map Tiling Service Implementation Standard, der die Notwendigkeit für offene Standards in diesem Bereich abdeckt.

Nachdem Kartendienste, die auf Kacheln basieren eine der bekanntesten Anwendungen der Tessellation darstellen, werden sie im Rahmen dieser Arbeit ausführlich behandelt. Die Schritte, die vonnöten sind um eine Karte in Kacheln zu zerlegen, der Prozess des Zerschneidens und Fragen der Speicherung der fertigen Kacheln werden erörtert. Das Zusammensetzen der Kacheln in eine nahtlose Karte finden ebenso ihren Platz wie Namensschemas für Kacheln.

Chapter 1

Introduction

1.1 Motivation

All the best known Internet map services use tiles as the basic building block for their services. Google Maps, Bing Maps, Yahoo Maps as well as the community project OpenStreetMap use tiled maps. The tiles are prerendered from either vector or raster data. Other community efforts as well as businesses use tiled maps as a means of efficient and easy presentation of spatial data on the Internet.

Once a tiled map is assembled, it only takes simple integer arithmetic for basic operations such as geopositioning and conversion between real world and pixel coordinates, thus making it fast and efficient. This efficiency makes tiled mapping a good choice for resource sensitive applications such as web mapping and maps for mobile devices.

There are numerous ways to turn spatial data into a tiled map. Some of the most commonly used GIS software products offer the possibility to create, store and access tiles. These include proprietary as well as Open Source software.

The geospatial community has already recognized the importance of tiled maps, especially for web map services. In February 2009, the Open Geospatial Consortium (OGC) released a candidate standard for tiled mapping – the Web Map Tiling Service Implementation Standard, WMTS in short. In April 2010 the candidate became an official standard. The specification draws on the Tile Map Service Specification by the Open Source Geospatial Foundation (OSGeo) [35] and the Tiled WMS by NASA OnEarth [21], thus representing a true community effort. This open tiling specification is just emerging, and it remains to be seen if it can gain momentum. It is also not the only way to serve tiles of importance: the methods and techniques used by the big players in the field, like Google and Microsoft, are also worth looking into.

Tiled mapping involves a lot of different disciplines. Yet there is no comprehensive paper on the principles, basics and applications of tiles and tiled maps. There is plenty of material on the individual parts, but it is widely scattered across different sources. It is intriguing to bring all this material together in one place and give an in-depth overview of the different parts that make tiled maps work.

1.2 Objectives

The main objective of this thesis is to explore the background and basics as well as the uses and applications of tile based mapping, with a special emphasis on tiled web maps.

This starts with the mathematical background and covers the geoinformatics and computer science methods involved. It also includes the tile making process as well as tile storage and access methods.

Parts of this process are covered by the OGC's WMTS candidate standard. It is another objective to explore this emerging open standard as well as proprietary tile serving techniques. Another goal is to give examples of applications that allow the creation of map tiles or use tiled mapping.

1.3 Approach and Methodology

This thesis draws its material from diverse scientific disciplines. Consequently, the first step is to identify the relevant topics of each of the following disciplines: mathematics, computer science and geoinformatics. The next step is to find literature that incorporates these topics, at least in part. This applies to all chapters of this thesis.

The second chapter uses literature research as an approach to describe tiling basics. The topics that apply will be discussed in detail. The mathematical background draws its content largely from the work of Kaplan [19] and Wolfram MathWorld [1]. Computer science and geographic information systems overlap in parts, therefore these parts are largely based on Worboys [53] and Longley [24]. There are numerous online resources to be taken into account as well.

The third chapter focuses on tiled mapping as it is known from online mapping services. It identifies the steps involved in the tile making process, and gives descriptions of freely available tile making applications. The relevant functionality of these applications is identified by examining the chosen programs and tiling sample raster data.

The final chapter is based on Internet as well as literature research to identify common fields of applications for tiled mapping.

[1]http://mathworld.wolfram.com/

The sample data used throughout this thesis is a Multi resolution Seamless Image Database (MrSID) image of Palo Alto, available from the University of California website, Berkeley[2].

1.4 Excluded Topics

This thesis focuses on tiles and tiled mapping as it is used by online mapping services today. This includes a mention or even a closer look at some of the technologies involved, e.g. Asynchronous JavaScript and XML (AJAX) and JavaScript. An in depth explanation of theses technologies would be beyond the scope of this thesis. Google, Bing and other commercial web map services provide Application Programming Interfaces (APIs) as a means to integrate maps into web sites and overlay spatial data. There are also a number of Open Source projects, most notably OpenLayers[3], that aim to provide the same functionality. It is not an objective of this thesis to explain or demonstrate how these APIs work.

Another topic that is important for applications that use tile based mapping but would go beyond the scope of this thesis are caching techniques on tile servers. All in all, the computer science methods involved in tiled mapping can only be touched upon briefly.

1.5 Structure of the Thesis

This thesis is divided into the following chapters:

Tiling Basics This chapter introduces the basic concepts of tiling. Starting with the mathematical background of tessellation and tiling, it moves on to the relevant computer science methods. These range from image processing techniques to data structures that use tiling as a means to partition space efficiently. It concludes with an overview of the GIS concepts and methods involved, such as georeferencing images, resolution and spatial reference systems. It also includes examples for the application of tiles in geoinformatics, apart from online map services.

The Tile Making Process discusses the steps involved to generate tiles from a raster image. It includes considerations to be made before a map is tiled. These concern the system(s) involved, base maps and single tiles. The actual fragmentation of the raster map into tiles, applications that offer this functionality and questions concerning storage of the finished tiles are discussed as well. The chapter then moves on to access methods for tiles stored in file systems and databases and introduces the OGC's WMTS as a complete reference for accessing tiles

[2]http://ucblibrary4.berkeley.edu:8088/xdlib//maps/brk00010.00000077.xml
[3]http://openlayers.org/

and delivering the finished map to a client as well as highlighting some proprietary methods.

Applications Focuses on the most important fields of application for tiled maps: web mapping and mobile map applications as well as giving examples of applications for applications that create tiles and tile servers.

Summary Summarizes the conclusions of the results of this thesis and tries to give an outlook into the future of tiled mapping and its applications.

Chapter 2

Tiling Basics

Tiling has its fundamental roots in mathematics – chiefly in geometry. It finds a number of applications besides its uses for in geoinformatics.

It is used in computer graphics for texture generation, sampling theory, remeshing and pattern generation [19]. Computer science as a whole uses tiling as a secondary data structure, for instance to index large (spatial) databases. Its uses are not limited to modern, technical applications, though. It has been used in ornamental and decorative art for centuries.

2.1 Mathematical Background

The terms tiling and tessellation are often used synonymously. Tiling, according to Weisstein is

> "‘A plane-filling arrangement of plane figures or its generalization to higher dimensions."’ [50]

Tessellation is a tiling by regular polygons only [49]. A regular polygon is equilateral and equiangular, which means that all sides are of the same length and all interior angles are the same. Worboys and Duckham, on the other hand define tessellation in a geoinformatics context as

> "’...a partition of the plane, or a portion of the plane, as the union of a set of disjoint areal objects."’ [53]

They go on to define regular tessellation as partitioning of the plane by congruent regular polygons and irregular tessellation as tiling by other, non regular, polygons.

In this thesis, we follow Worboys’ and Duckhams broader definition of tessellation – be it regular or irregular. This allows the inclusion of Triangulated Irregular Networks (TINs) and Thiessen polygons, which are widely used in geoinformatics. The most important regular polygons for the purpose of this thesis are squares and equilateral triangles; regular hexagons are less important in a Geographic Information Systems (GISs) context.

The concept of tessellation can be extrapolated to higher dimensions – in three dimensional space a tessellation constitutes a tiling of regular polyhedra, and in n dimensions a tiling of regular polytopes [49].

Tilings can be classified according to a host of criteria: regular versus irregular, symmetric versus asymmetric, periodic versus aperiodic amongst others. The distinction between regular and irregular tessellation applies best in geoinformatics. For the purpose of this thesis, only those tessellations relevant to GISs are taken into account and discussed in detail.

2.1.1 Tiling

Weisstein's previously mentioned definition of tiling is meant to give a general idea of the concept behind tiling. But which properties does a plane figure, or one of Worboys' and Duckhams areal objects have to fulfil to be a tile? What are the conditions that have to be met so that an arrangement of such plane figures can be called a tiling?

These properties and conditions are stated by Kaplan, who gives a more comprehensive definition of tiling and the properties of tiles [18]:

1. Every tile is a closed topological disc.
2. Every point in the plane is covered by at least one tile.
3. The intersection of two tiles is a closed curve, a point or simply empty.
4. The tiles are uniformly bounded. There exist $u, U > 0$ such that every tile contains a closed ball of radius u and is contained in a closed ball of radius U.

For the purpose of this thesis, the second and third definitions are most important. The second point conveys that a plane – a map – is completely covered by tiles, thus ensuring that every piece of information the map contains is covered. The third property covers the fact that the coverage is seamless. This is important, because it means that a map can be assembled from tiles without overlaps. It also ensures that a tile does not have a disconnected boundary.

There are certain terms associated with tiles and tilings, illustrated in figure 2.1.

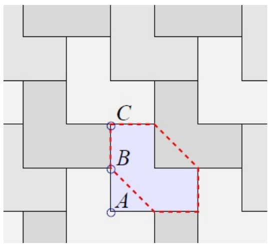


Figure 2.1: Important terms and concepts of tiling [19]

Point A in figure 2.1 is a *shape vertex*, denoting a vertex of an individual tile. In contrast, B denotes a *tiling vertex*, which is a vertex where three or more tiles meet. Point C is both a shape and a tiling vertex, although a tiling vertex is not necessarily a shape vertex and vice versa. A tiling vertex is the point of origin as well as the endpoint of a *tiling edge*. The polygon formed by joining all tiling vertices of a single tile is called the *tiling polygon*, outlined in red. If all the shape vertices are also tiling vertices, the tiling is called an *edge-to-edge* tiling.

If all tiles in a tiling have the same shape, the tiling is referred to as *monohedral*. In mathematical terms this means that all tiles are congruent to a shape T, the so-called *prototile* [47].

If a monohedral tiling is periodic, it is a regular tessellation. A tiling is periodic if it repeats regularly in every direction. For the two dimensional plane that means the tiling repeats regularly in two directions. To verify that a tiling is periodic lattices can be used. A lattice is a grid made up of evenly spaced parallel lines. The parallelograms formed by this lattice are known as *period parallelograms*. The region that is enclosed within a period parallelogram is known as the *fundamental region*. If the fundamental region can be shifted so that it tiles the plane, then the tiling is periodic.

Figure 2.2 illustrates this: the image on the left shows the tiling, the middle one shows the lattice superimposed on the tiling and the rightmost picture shows the fundamental region that can be repeatedly translated to make up the tiling.

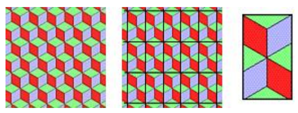


Figure 2.2: Periodic tiling (adapted from [45])

Monohedral periodic tilings in the Euclidean plane – tessellations – are the most important form of tiling for the purpose of this thesis. Tilings are possible in higher dimensions as well, but (web)maps are two dimensional; we want to tile maps, so we stay in the Euclidean plane.

Tilings exist in a number of other forms, some of which are quite well known. Penrose tiles for example are a representative of aperiodic tilings and M.C. Escher's tilings are quite famous.

Tiling is a wide field with a number of applications in various fields besides GISs and mathematics. Many resources can be found on the web, starting points for further study of tiling can be The Geometry Junkyard[1] or Wolfram Mathworld[2].

2.1.2 Tessellation

Tessellation is the basis for two important spatial data representations. Regular tessellation provides the basics for the raster data model, which is important in geoinformatics in particular and in computer science in general. Irregular tessellation is the basis for TINs and Thiessen polygons (Voronoi diagrams).

Regular Tessellations

A regular tessellation is an edge-to-edge tiling of the plane by congruent regular polygons [18]. In the Euclidean plane, only three regular tessellations exist: those by equilateral triangles, squares and hexagons. The tessellation by squares is the most important form for geoinformatics and computer science, as it provides the basis for images in the form of raster data.

[1]http://www.ics.uci.edu/ eppstein/junkyard/topic.html
[2]http://mathworld.wolfram.com/topics/Tiling.html

The three regular planar tessellations are shown in figure 2.3.

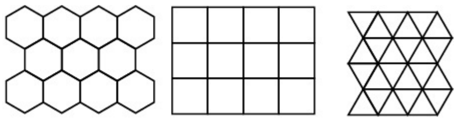


Figure 2.3: The three regular tessellations

The tessellation used by tiled web map services is the regular nested tessellation, where a tessellated figure is further tessellated by the same figure. This applies to tessellation by triangles and squares only, as hexagons do not fit together to form a larger hexagon [53]. The nested square tessellation of the plane has another application: the quadtree data structure. This structure is widely used for holding spatial raster data, as an index as well as a naming convention for individual tile file names. These applications are discussed in sections 2.2.3 and 3.3 respectively.

Tessellation does not stop with regular polygons. There are semiregular tessellations, in which the plane is partitioned not by one regular polygon but by two or more polygons. There exist eight such tilings of the plane, called the Archimedean tessellations.

Irregular Tessellations

Irregular tessellations are tilings by polygons that are not all regular. These tilings can look arbitrary and network-like [27]. Irregular tessellations constitute the basis for two important data structures in geoinformatics: TINs and Voronoi diagrams.

A prominent form of irregular tessellation is tiling by triangles, referred to as *triangulation*. Every surface has a triangulation, although it may takes an infinite amount of triangles [51].

A widely used triangulation method is the Delaunay triangulation. This method creates a triangulation whose constituent triangles are as nearly equilateral as possible for an irregular tessellation [53]. The base for a Delaunay triangulation is a set of discrete points. The defining property of a Delaunay triangulation is that the circumcircle of each triangle must be empty. I this case, empty means that no point of the set is within the circumcircle. Another property of the Delaunay triangulation is, that the outer edges of the triangulation form the convex hull of the given set of points.

Figure 2.4 shows a Delaunay triangulation and its *dual*, the Voronoi diagram. In a dual graph, each face of the original graph is associated with a node in the dual graph, these nodes are connected if the faces of the original graph are adjacent.

To create a Delaunay triangulation, Voronoi diagrams are often used. The base of a Voronoi diagram is a set of discrete points as well. Around

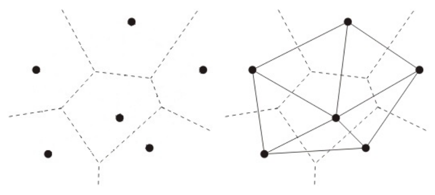

(a) Voronoi diagram (b) Delaunay triangulation

Figure 2.4: A Voronoi diagram and its dual, the Delaunay triangulation

each point P a polygon is created – this polygon has the property that the area contained within is nearer to P than to any other point of the set. These polygons are termed *proximal polygons*. The set of resulting polygons forms the Voronoi diagram.

Voronoi diagrams and TINs are the most important types of irregular tessellations for GIS. Even though they are not important for tiled web mapping, both data models will be briefly introduced in section 4.1.3, as they are the the chief representatives or irregular tessellations in GISs.

2.2 Computing Perspectives

The tiles used in tile based mapping are usually square images. To acquire, store and use these images a number of different computing techniques and concepts are used. Firstly, to turn a seamless map into tiles, concepts originating in digital image processing come into play. Secondly, to store and access tiles efficiently, database technology or file systems and various file formats are used.

2.2.1 Image Processing

Tiled maps are a concept to present multi resolution raster data efficiently. These collections of tiles, or *raster mosaics*, can be of arbitrary spatial extent. Digital image processing provides several concepts and techniques that help to organize these mosaics and make access and storage more efficient.

Image Mosaics

Image mosaics are comprised of several input images, which may overlap in some areas. These input rasters are stitched together to form a new output raster. The process of stitching the input rasters together can include complex operations, especially in overlapping areas where the new raster cell has to be formed from two or more input cells. These operations can either take the the minimum or maximum cell value, or the mean cell value amongst other possibilities [40].

The difference between tiles and conventional image mosaics is that tiles never overlap. This is stated by the third point of Kaplan's definition of tiling in 2.1.1. The property tiles and mosaics share is the following: they are both stitched together using several input images in the correct sequence in order to form a complete map. Thus, tiles can be thought of as a subset of image mosaics – the class of non overlapping, seamless mosaics.

Image Pyramids

The *image* or *resolution pyramid* is a concept that has its roots in image processing. It is used to efficiently display an image at multiple resolutions. In raster maps it is used to present information at different scales as well as a means of generalization and information reduction.

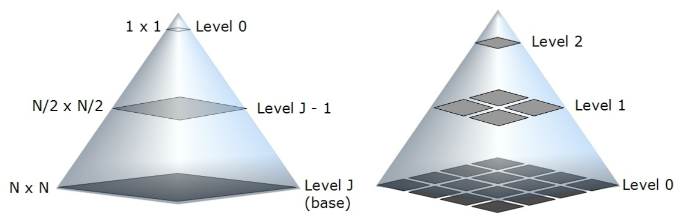
At the base of the resolution pyramid lies a high resolution image. With every consecutive level of the pyramid, a group of adjacent pixels is mapped onto one composite pixel, thus reducing the number of pixels in the image and with it the resolution. This process is then repeated on the newly created image, further reducing the image size and resolution. In theory, the process can be repeated until all that is left is an image the size of one pixel. Depicted graphically, the series of images looks like a pyramid – hence the name.

The pyramid type described above is the so-called *redundant* pyramid. There are several methods to determine the value of the one pixel the group is mapped on. Resampling is one of them, as is the averaging of the input pixels' values – the rules of these methods depend largely on the application. Another type of pyramid is the *additive* pyramid. In contrast to the redundant pyramid, this type start with a low resolution image. The pyramid itself consists of *difference* images. These images are acquired by computing the difference between each pixel. These are added to the original coarse image in order to deliver the image at the desired resolution [27].

Tiled maps share the multi resolution approach with image pyramids: the image pyramid becomes a tile pyramid. In a tile pyramid, each level consists of multiple tiles. The tiles have a constant size in pixels. The tiles themselves are not represented through image pyramids. Even though the basic concept of presenting raster data at multiple resolutions is shared between the two, there are some major differences.

For one, a tile pyramid consists of multiple images at all levels but the highest whereas an image pyramid contains only one image at all levels. This means that in order to show the whole map covered by a tile pyramid it may be necessary to access a lot of tile images. This hardly ever applies though, because tiled mapping is usually used in conjunction with a viewport – a window of a fixed size displaying the desired part of the map. In this way only a small number of tiles has to be accessed to display the part of the map covered by the viewport.

In contrast to the image pyramid, where the image size shrinks as the pyramid rises, the tiles do not vary in size. The major difference between image and tile pyramids is that the single image in an image resolution pyramid covers the same spatial extent at all pyramid levels, whereas a single tile image in a tile pyramid covers different spatial extents at different pyramid levels.



(a) Image pyramid (b) Tile pyramid

Figure 2.5: An image pyramid and a tile pyramid

Figure 2.5 shows a comparison of an image and a tile pyramid. At the base level J the image pyramid is at its greatest resolution, the image being at its maximal size of NxN. In this example, four pixels are mapped onto one pixel in the next resolution level. At level $J - 1$, the number of pixels has been reduced to half the original number. This goes on until level 0 is reached, where the pyramid consists of one pixel only. In reality, the image pyramid would be truncated at an earlier level as images of one pixel do not have many practical applications.

At the base of the tile pyramid, the number of tiles is at its maximum as well – sixteen tiles in the example image. At the next level, the number of tiles needed to cover the map's complete extent has been reduced to four, one tile covering the same spatial extent as four tiles at the lower level.

2.2.2 File Formats

The most common way to store a digital image is a file. To do this a wide variety of file formats exist, providing different properties. Image file formats can be separated in two branches: raster and vector image formats. In the context of tiles and tiled mapping it is not relevant if a tile is stored as a raster or vector image, although the raster format is more common by far.

A file format is a set of rules on how to encode the raw image data so that a program can access and display the image correctly. This set of rules provides information on important properties of the image: bit depth, colour system, image size and resolution as well as (de)compression rules amongst other metadata. Without these rules, the raw image data would be almost impossible to access and render. An image file contains the metadata according to the set of rules as well as the raw image data itself.

One way to identify the file format at a glance is the file extension, a suffix to the file name. Another way to identify a file is by reading its file header. The header consists of a number of bytes at the beginning of the file, which contain the metadata prescribed by the rules of the specific file format.

File formats are often standardized, thus enabling a variety of programs to access and manipulate the raw image data contained in the file. Some of the most universally implemented standards are Joint Photographic Experts Group (JPEG), Portable Network Graphics (PNG) or Tagged Image File Format (TIFF)

One of the most important rules in a file format describes the compression method used. Raw image data can be very large, depending on the number of pixels and colour depth. For example, an image 1024×1024 pixels at a colour depth of 256 colours per pixel would need one byte to store each pixel, approximately one megabyte for the whole image file. At a colour depth of 16 million colours per pixel, every pixel would need three bytes of storage space, increasing the file size to three megabytes.

In geoinformatics, a lot of images are sometimes needed to cover the desired spatial extent at different resolutions, thus increasing the needed storage space and processing time and power further. To keep the file size and thus the need for storage space in check compression methods are used.

Compression Methods

One commonly used way to classify compression methods is the division into lossless and lossy compression.

Lossless compression is fully reversible. The raw data is encoded in such a way that every byte of the original image can be reconstructed from the compressed data. To achieve this, redundancies or patterns in the image data are used. These patterns are identified and encoded according to a compression technique.

Contrary to lossless compression, lossy compression is irreversible. The raw data is compressed in such a way that the original data cannot be fully restored. To compress the data, another kind of redundancy is utilized – perceptual redundancy. This method provides better compression ratios [27]. The compression exploits the fact that the human eye has limitations. It dismisses all data that is not essential for the image's quality. For instance, the human eye cannot perceive very dark or bright colour tones – these would be smoothed over in a lossy compression [46].

Compression Techniques

There are a number of techniques utilized in compression. Often a combination of these is used to achieve optimal compression. Although there are many more compression techniques, the ones mentioned are often important in image file formats.

Run-Length encoding: reduces repeating value sequences by replacing the sequence with only two characters: the value to be repeated, and the number of times it is repeated. Thus the source sequence AAABBC-CCC, which uncompressed would need nine bytes, becomes A3B2C4, reducing the number of bytes needed to store the data to six bytes. This technique is well suited for images which contain large regions with identical values.

Differential Encoding: is another technique that exploits the fact that images often contain similar or identical regions. This technique does not store the actual pixel values but computes the difference to neighbouring pixels and stores this value. These difference values are usually smaller than the actual values, thus saving space. Differential encoding is also used as a preprocessing method for subsequent, different compression techniques.

Huffman Coding: is a statistical compression method that achieves lossless compression. In essence, it assigns the shortest available code to the symbol that is the most likely to occur. The available codewords are stored in a code table, which is created for every single image.

File Format Standards

File formats are a set of metadata, or information about the image data itself. A multitude of image file formats exist, some of them are open standards, some proprietary formats[3]. In tiled mapping, especially on the web, it is important for the files to be as small as possible – thus compressed image formats are employed almost exclusively. Of these, JPEG and PNG are the most common.

The term JPEG is often used do refer to image files, although in reality JPEG is a standardized compression method. The official ISO/IEC name is *Digital Compression and Coding of continuous-tone still images*[4]. It contains standards for both lossless and lossy compression methods and specifies the techniques involved. In reality not many implementations of the lossless compression method exist, although the lossy compression is widely used for all types of images.

The lossy JPEG standard utilizes a number of compression techniques, Run-Length Encoding and Huffman Coding among them [14]. The actual file format that implements – and is defined in – the JPEG standard is JPEG Interchange Format (JIF), although it is rarely used in its entirety.

[3]http://www.fileinfo.com/filetypes/image
[4]http://www.itu.int/rec/T-REC-T.81/en

Most commonly the JPEG File Interchange Format (JFIF) is used, which represents a narrower implementation of JIF inasmuch as it does not include the lossless compression method. Another actual file format that implements the JPEG standard is Exchangeable Image File Format (EXIF).

Thus the file extensions that denote JPEG compliant files are many: .jpg, .jpeg, .jpe, .jfif, and .jif are the most common. It is also possible for other file formats to include JPEG compressed data, e.g. for thumbnail images.

PNG was developed as an alternative to the Graphics Interchange Format (GIF). This was desirable because of license issues and the fact that GIF only supports 256 colours. PNG uses lossless compression which is achieved in a two stage process. The first step applies a filter to the image data. This filter uses a differential encoding technique thus reducing the original pixel values to difference values. The second step applies the compression technique DEFLATE[5], which partly uses Huffman Coding.

There are three types of PNG files: 8 bit, 24 bit and 32 bit. The number of bits refers to the amount of storage for colours. 8 bit PNG files can store up to 256 colours, whereas the 24 bit type can store up to 16 million colours. Both these types support simple background transparency, but only the 32 bit type supports variable transparency for each pixel.

Compared to JPEG, PNG files take up more disk space, as they use a lossless compression method. PNG is the image format of choice for graphics containing objects with sharp transitions, such as raster maps. JPEG on the other hand is better suited to handle photographic images like satellite and aerial imagery. If these are used as background images, or if file size is important, then compressed formats are well suited.

Compression influences the geometric precision of an image. The greater the compression ratio, the more distorted the images becomes. This may make it difficult to interpret the image – visually or automatically. Image file formats that use lossy compression are not suitable for all applications, e.g image analysis. Therefore many GISs support other file formats. Some of these formats are proprietary, like Enhanced Compression Wavelet (ECW) or MrSID, some are open like TIFF.

[5]http://tools.ietf.org/html/rfc1951

2.2.3 Tiles as Secondary Data Structures

Up until now, this thesis has been concerned with tiles as a primary data structure and tessellation as a means to partition a specified region. But tessellation has other applications as well: it is used to index spatial databases; to index, store and access raster data, as a spatial data structure, as a basis for naming schemes for file based tiles, it has even been considered as a compression method by Samet in [42]. All these applications are known as secondary data structures. Some of the concepts and ideas stemming from secondary data structures can be applied to tiles as primary data structures as well.

Concrete use of tessellations as secondary data structures include indices for spatial databases as well as for large raster datasets. Some of these indices can be represented as hierarchical tree structures – the quadtree being a prominent example. It also provides the basis for the quadkey naming scheme employed throughout the Bing Maps service or the Google Maps satellite imagery service.

Indices

Generally speaking, an index is a concept that facilitates and accelerates information retrieval. Indices are used in spatial databases as well as for large datasets. There are three main types of indices used in a geoinformatics: grid indices, quadtrees and R-trees [24]. Of these, grids and quadtrees are based on tessellation.

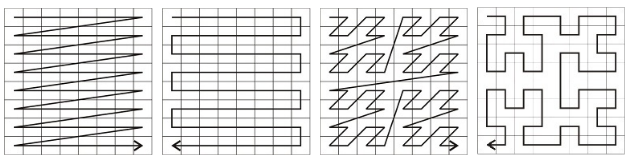
Grid Indices are one of the oldest methods to facilitate finding geographic information. A regular grid, also known as a mesh, represents a tessellation of the plane by rectangles or squares. Every cell of the plane can be addressed by an index of *(x,y)*. This concept can be applied to three dimensions as well, a z coordinate being added to the index. Grids form coordinate systems in two and three dimensions.

Latitude and longitude lines are a grid covering the earth from pole to pole. They are frequently included on paper maps and atlases for reference purposes. There are various other grid systems for both paper and digital maps, e.g. Universal Transversal Mercator (UTM) which is used globally. Still, many countries have their own national grid systems, tailored to their special requirements.

Grids are not only used on paper and digital maps, but as data structures and database indices as well. These grid structures are usually rectangular or square in shape, partitioning a planar region containing geographic data into equal sized cells. They can be applied to raster as well as vector data, are easy to implement and robust. One factor that influences the index' performance is the tile addressing system, or tile ordering in the plane.

The underlying problem of efficient tile ordering is that two dimensional spatial data structures have to be mapped onto a one dimensional storage system. In other words, the challenge is to find a path through the tesselated region that preserves the nearness of adjacent tiles in the tile ordering. The *tile index*, the algorithm that describes that path, ensures that the tiles neighbouring spatial objects are located on are in each other's vicinity in the secondary data structure as well [53]. In reality, no tile index is able to preserve the nearness of all indexed geographic features.

Some commonly used tile indices are depicted in figure 2.6.



(a) Row (b) Row Prime (c) Morton (d) Peano-Hilbert

Figure 2.6: Common tile indices [53]

The first one, row ordering, is the most intuitive tile ordering system, one row of grid cells follows after another. Row ordering is simple, but it only preserves nearness between adjacent cells in the same row, not across rows.

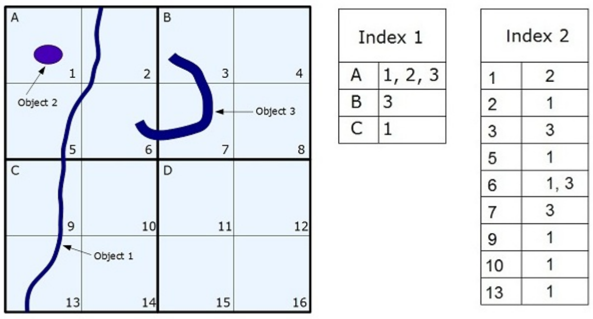
Row prime ordering is an enhancement of simple row ordering, preserving the nearness of cells at the rows' end, if nowhere else.

Morton and Peano-Hilbert ordering are rather more sophisticated tile indices. Morton ordering, also known as *Z-order curve*, preserves nearness well. To create a Morton ordering, first the so called *z-values* – or Morton numbers – have to be computed for all grid cell coordinates. The z-values are the result of interleaving the bits of the binary values of a coordinate pair and converting the resulting binary value back to the decimal system. Thus, the coordinate $(6, 7)$ becomes the binary z-value 111101, which is 61 in the decimal system. The z-curve follows the resulting numbering in the grid.

The Peano-Hilbert method is closely related to Morton ordering, the main difference being that the transitions in a Peano-Hilbert tile index can only occur between cells sharing an axis. It is even better at preserving nearness, although it is more complicated to construct.

A frequently used grid index is the multilevel fixed cell grid [24]. Figure 2.7 shows a two-level grid index applied to a region containing three geographic objects.

The first level index splits the region into four cells labelled A, B, C and D. The index is then built according to which object lies completely or partially within which cell. Cell A includes all objects whole or partially. Cell B includes a part of object 3, cell C parts of object 1. Only cell D is empty. The second level index again partitions the region into square cells – only this time the cell size is smaller, allowing for an index with finer granularity. The second index has empty cells as well: cells 4, 8, 11, 12, 14, 15, and 16 do not contain any objects or parts thereof.



Index 1	
A	1, 2, 3
B	3
C	1

Index 2	
1	2
2	1
3	3
5	1
6	1, 3
7	3
9	1
10	1
13	1

Figure 2.7: Two-level grid index (adapted from [24])

The empty cells at both index levels point to a major drawback of the fixed grid index: The number of geographic objects per cell varies strongly, some cells contain many objects, some do not contain any objects at all. This fact becomes more pronounced if the distribution of the objects is not uniform. If there are clusters of objects, some cells will be overfull while others remain empty. The relationship between object distribution and grid cell size clearly influences the index' performance. If the distribution deviates too far from uniform, it would be desirable to have a grid structure that adapts to the objects' distribution [53]. The *grid file* is such a data structure. It allows the grid cells to adapt to the data distribution pattern, thus constituting an irregular tessellation.

Quadtree indices can be thought of as a grid data structure that adapts to the underlying data. The grid is not just draped over the region, but constructed dynamically. This is done by recursively subdividing the plane into four quadrants until every subdivision is completely filled by one object [24]. Quadtrees are used as indices for raster data as well as vector data. The name quadtree comes from the underlying data structure, which, depicted graphically, resembles an upside-down tree. The most common geometric shapes used for the partitioning are squares and rectangles. Nevertheless, other shapes such as triangles and hexagons are employed as well.

A tree in general is a hierarchical data structure starting at the *root* node. The root node splits into further nodes. There are two types of nodes: *leaf* nodes are nodes that do not have any more *child* nodes, *internal* or *non-leaf* nodes do have child nodes. A non-leaf node is known as the *parent* of its child nodes, child nodes at the same level are called *siblings*.

A quadtree's non-leaf nodes have exactly four children [53]. Quadtrees can be used as indices for spatial databases and raster datasets as well as a compression method for spatial databases. This data structure is not limited to two dimensions – it can be adapted to three dimensions, which is known as the *octree*. The octree's non-leaf nodes have eight instead of four children.

The quadtree used for raster data as well as areal and line data is known as the *region quadtree*. To build the index, a square region containing heterogeneous data is divided into four quadrants – this process is repeated until every quadrant contains only homogeneous data. These repetitions are known as *quad levels*, or *levels of tiling*. This leads directly to a significant advantage of the region quadtree: it permits the use of different resolution levels. The tree grows in areas with more detail whereas the subdivision stops in homogeneous areas.

To illustrate the division process figure 2.8 shows successive partitioning of a simple example raster and the resulting quad levels.

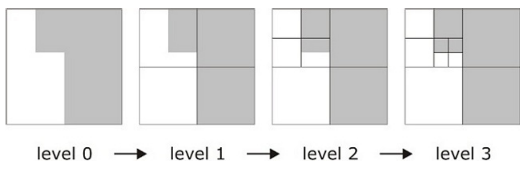


Figure 2.8: Quad levels of a raster (adapted from [53])

To build the corresponding tree structure at first a rule has to be established concerning the ordering of the child nodes. A commonly used rule is the division of each quadrant at each tree level into a northwest, northeast, southwest and a southeast quadrant.

Figure 2.9 shows the quadtree structure for the example raster used in 2.8. The tree starts at the root node and splits into four nodes at quad level 1. Black nodes represent none-leaf nodes, whereas white and grey nodes represent leaf nodes.

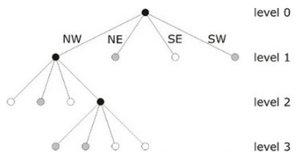


Figure 2.9: Quadtree for the example raster used in figure 2.8

Quadtrees can be very efficient for storing and accessing rasters. They are best used with rasters containing large homogeneous areas or rasters that rarely change. If a raster is too heterogeneous, there are no areas to combine. This fact is best illustrated by a chequerboard pattern, which leads to a tree structure that is in fact a Morton ordered tile index [53]. If a raster is frequently updated or changed in another way, such as by translation or rotation, the corresponding tree data structure has to be rebuild from scratch every time. This may lead to a completely different tree structure.

Quadtrees are not only used for areal data, but for point data as well. The data structure employed for point data is the *point quadtree*. The tessellation method used by the point quadtree is an irregular one, as the region containing the data is divided dynamically according to the data distribution. The division is always centred on a data point, thus dividing the region into rectangles.

A variant of the point quadtree is the *PR* quadtree, *PR* standing for Point Region. It uses the same subdivision method as the region quadtree, dividing the region into squares until every square contains one point only.

A defining criterion for a point quadtree is the order in which the points are inserted into the data structure. The same set of points can lead to wildly different trees, depending on the sequence of the points. The sequence can be determined by a (numerical) attribute or some other identifier, but more commonly the set of points is ordered by location before the index is built. This ensures that the tree does not become one sided and thus inefficient.

The root node of a point quadtree is the region itself. The first point inserted into the index divides the whole region into four irregular quadrants. The next point inserted only subdivides the quadrant it is situated in. This process is repeated until all the points in the set are indexed. The tree structure is built like a region quadtree, with the quadrants ordered northwest to southeast.

As mentioned above, there are quadtrees for raster data as well as for vector data. Points and raster data are covered by point and region quadtrees respectively. A quadtree index used for linear vector data such as polylines, linear networks or polygon boundaries is the *PM quadtree*. The *PM* stands for Polygonal Map. To be precise, the term PM quadtree describes a family of quadtrees designed to index and store point and line data [42]. This family includes the PM_1, PM_2 and PM_3 quadtree. The difference between these quadtree types lies in the rules they follow to achieve the tessellation and the associated tree data structure.

The PM_1 quadtree starts to divide a square region of the plane and the spatial objects contained within just like the region quadtree does. Every quadrant is split into four sub-quadrants, until all the rules for the PM_1 quadtree are satisfied. These rules can be summarized as follows:

- Every leaf node bounds a region that contains only one vertex.
- Every leaf node that contains a vertex may only contain a q-edge that is incident with this vertex. The term *q-edge* describes a part of a line cut by a quadrant.
- If a leaf node does not contain a vertex it may only contain one q-edge.

The PM_2 quadtree in comparison only follows the first two of these rules.

The grid and quadtree indices are based on rectangles or squares, and by far the most common in the world of geoinformatics. However, there are other, more exotic uses for grid and quadtree indices. Some of these combine ideas taken from grids and quadtrees to create new forms of indices. They employ other shapes such as triangles and hexagons to divide a plane or a sphere.

Location Codes

Location codes are described by various sources, in various contexts using various names. Lee and Samet describe them as a way to use the quadtree data structure without building the associated tree. The codes are used to label the cells of a Discrete Global Grid (DGG) tessellated by triangular faces. The data structure containing the locational codes is called a *linear quadtree* and the labelling scheme uses *triangular codes* [22].

Dutton uses location codes to identify the triangular faces of a Quaternary Triangular Mesh (QTM) based on an inscribed octahedron [5], labelling them QTM identifiers. QTMs are an example of the more exotic applications of tessellation in geoinformatics – they are described in greater detail in section 4.1.1.

Li refers to location codes as *quadcodes*, and describes them as a hierarchical labelling system for a region partitioned by a square quadtree [23]. The origin of these quadcodes lies in image processing.

All these different approaches have in common that they view the location code system as an alternative way to represent quadtrees. It consists of an underlying data structure, the linear quadtree, and a labelling scheme. The linear quadtree is a data structure based on quadtree indices, although it does not use a tree structure per se to store the tree nodes. Instead, the location codes – or quadcodes – of each tree node are stored in a simple array. The order in which the quadcodes are stored may correspond to a particular traversal order of the underlying tree structure. A quadcode can consist of numbers or letters. To construct the linear quadtree it does not matter if the prototiles used to tessellate are squares, triangles, or another regular polygon.

Quadcodes based on square quadtree decomposition have several properties that make them interesting for spatial applications: the codeword itself contains information about geometric properties such as area, distance from other quadrants, location in the tessellated region and adjacency. It facilitates accurate description of quadrant size and dimension. The code can be converted to and from geographic coordinates, if the region is accurately georeferenced. A property the quadcode inherits from its close relation, the quadtree, is the use of multiple resolution levels in the whole region or in subregions thereof. Equally important, these geometric properties can be easily calculated by using simple arithmetics [23].

To understand why location codes have these properties, one must take a closer look at the assembly and structure of the quadcode. Mathematically, the quadcode is a base-4 code, which means that it uses four code symbols to construct the individual codewords [23]. Each quadrant of a tessellated region is assigned one of the code symbols. If valid code symbols are 0, 1, 2 and 3, a valid codeword can only consist of one or more of those four symbols. At each subdivision step one of these numbers is appended to the parent code. The parent code is the quadcode of the square the newly created quadrant lies in. Figure 2.10 illustrates the construction process of a codeword.

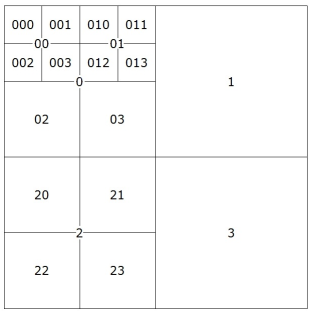


Figure 2.10: Quadcode construction

At the start of the decomposition process, a square region is subdivided into four quadrants. The upper left quadrant is assigned the quadcode 0, upper right is 1, lower left and lower right are assigned 2 and 3 respectively. Subsequent subdivisions of quadrant 0 are a combination of the parent code – in this example 0 – and the codes assigned to the four subdividing quadrants. The upper left quadrant of the second level of the tessellation is thus assigned the quadcode 00. Staying within the upper left quadrant, the third level is prefixed with the parent quadrant's location code 00, and gets its own code of 0 – the complete quadcode for the third level of the upper left quadrant is therefore 000. The illustration shows the subdivision and naming process for three levels of quadrant 0 and two levels of quadrant 2.

The length of the codeword implies the number of subdivisions that have been performed. This way, the resolution level of a quadrant can be directly deducted from the quadcode length. As in a quadtree structure, a region can be split into different resolution levels, according to the needs arising from the data contained within.

The geometric properties that can be read directly or calculated from the quadcode mostly relate to subregions. These subregions are known as *elementary squares*. The quadcode of an elementary square provides information about a number of geometric properties, e.g. the location of the square in a Cartesian coordinate system. In order to do this, a characteristic point of the square has to be agreed on. The point that is commonly used is the upper left corner of the elementary square. Other properties are the distance between two elementary squares and adjacency between two squares.

An important point in spatial applications is the formation of (sub)regions. These regions often extend over multiple quadrants, thus they cannot be represented by one elementary square alone. To represent a region, a set of quadcodes is used. The elements contained within the set are the elementary regions. Figure 2.11 shows a subregion of a tessellated region in light grey. The set of quadcodes representing this subregion is: 3, 10, 11, 13.

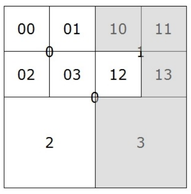


Figure 2.11: Quadcode subregions

Other properties that can be derived from the codewords themselves are area and length of elementary squares. The quadcode system for square spatial decomposition is compact and easy to implement. Quadcodes render it unnecessary to physically build a tree structure, which has to be traversed to find regions, compute lengths and other properties. Instead, all these operations can be performed using an array structure, which provides a more direct access to the region and the data contained within.

Location codes convert two dimensional coordinates, be they geographic latitude/longitude coordinates or Cartesian coordinates into a one dimensional string, thus facilitating comparison with other converted coordinates and preserving an important property – namely that of spatial proximity. Two tiles that have similar quadcodes are usually close together in reality.

2.3 GIS Perspectives

The tiles used in tiled web mapping are essentially small equal-sized image files, created by regular square tessellation. What does it take to turn these image files into small maps themselves and to assemble them into a greater, seamless map at different resolution levels? This section aims to explain fundamental concepts from the world of geoinformatics and geography in general that lie behind this transformation. Important applications of irregular tessellation in geoinformatics, namely TINs and Voronoi diagrams, are explained in section 4.1.3.

2.3.1 Raster Data in Geoinformatics

Raster and vector data are the prevalent data models in GISs. The use of tessellation as a secondary data structure provides some important techniques and methods for handling vector data, but raster data is the primary data basis for tiled web map services, hence the special consideration at this point.

The raster data model provides a simple structure to store spatial data. It can be used to perform spatial analysis and overlay operations. Its major drawback is the sheer volume of raster datasets. Depending on the required resolution, datasets demand large amounts of disk space as well as processing power and time.

Raster data itself is based on the regular, usually square tessellation of an area of interest. The prototiles are called raster cells or pixels. The regular tessellation leads directly to the organization of raster data as a matrix consisting of rows and columns. Geometrically, a raster can be seen as a Cartesian coordinate system, with its origin in the upper left corner and two perpendicular axes.

The pixels themselves have values, stored in a matrix. If there is only one value matrix, the image is referred to as a *single band* raster. These can be binary images which can only have one of two values, one or zero. They are usually displayed as black and white images. Greyscale images can contain values from zero to another value, like 255 or 65535, and are displayed using shades of grey. Aerial photographs are often displayed as greyscale images. To display colour with a single band raster, colour lookup tables are used. Every entry in the lookup table corresponds to a combination of red, green and blue colour values. The pixels themselves store a cross reference to the entry in the lookup table.

Rasters containing multiple values matrices are called *multi band* raster. Satellite images are frequently multi band images, as each sensor acquires another part of the electromagnetic spectrum. The number of bands defines the spectral resolution of an image.

These values are mapped to information concerning one or more real world phenomenon. Additional information for these values can be stored in a raster attribute table. Values can be of different data types, e.g. integer or floating point numbers as well as a special *noData* value signalling the absence of data. Raster data can be used to portray either discrete or continuous phenomena. Discrete data consists of objects that have a clear definable boundary, e.g buildings, roads and lakes [24]. Discrete data is modelled using *area raster*, where the cell value applies to the whole pixel area.

Continuous data – also called surface or field data – portrays phenomena that change as they move away from a fixed point across the surface. This point can be the source of a phenomenon, e.g. an oil spill. Continuous data include temperature, precipitation levels as well as elevation data. Continuous data is often modelled with *point rasters* – the attribute value is applicable solely to the centre point of the pixel. These values are often taken from measurements.

Point and area rasters have a dual relationship. Prototiles are described by their vertices as well as by the tessellation rules. This becomes especially clear when looking at irregular tessellations like TINs, which can be represented using an irregular tessellation method like the Delaunay triangulation and the set of points the tessellation is based upon.

Raster data can be divided into four classes as follows [27]:

1. *Raster Images* are greyscale or colour images usually acquired through remote sensing methods. They contain a lot of different colours, and therefore information. Through the application of spatial metadata they can be used as background images or basemaps. Satellite imagery and aerial photos fall into this category.

2. *Raster Maps* can be either scanned paper maps or the product of a digital map creation process, often based on satellite or aerial imagery. They usually contain less information than raster images as they tend to focus on one or more themes representing discrete phenomena such as land use.

3. *Thematic Rasters* represent mostly continuous real world phenomena such as temperature or precipitation levels.

4. *Regular Digital Terrain Models* are a subset of digital terrain models, as they utilize a regular elevation matrix. This simplifies the general handling of the model.

Tiled web map services can use all four of these raster classes as long as they are available in a supported file format or database. In practice only raster images and raster maps are used. The most important properties of raster data are cell size and as a direct consequence, resolution.

2.3.2 Resolution

There are three types of resolution: spectral, temporal and spatial [24]. Spectral resolution, or the number of bands is described above. Temporal resolution is the frequency with which an image collection of the same spatial extent is updated. The last, and for many applications most important type of resolution is spatial resolution.

Spatial resolution for one thing depends on the manner in which the image is acquired. A common source for raster data in GISs is remote sensing. Remote sensing methods include satellite sensors and aerial photography, but also more exotic methods like balloons and masts. In the case images are captured through remote sensing, the limitations and specifications of the equipment directly influence the spatial resolution. The cameras and sensors used have inherent limitations concerning level of detail that can be captured. The Système Probatoire d'Observation de la Terre 5 (SPOT5) satellite, for instance, has sensors that capture images where one pixel covers 1000×1000 meters down to 2.5×2.5 m per pixel. The cameras used in aerial photography provide even finer level of detail with approximately 0.1 meters to 5 meters [24].

Another common raster data source are scanned paper maps. As in remote sensing, the equipment greatly influences the spatial resolution.

Some rasters are created using spatial analysis, like Digital Terrain Models (DTMs). In this case, the spatial resolution is determined by the density of the base data set. For example, in a digital elevation model the density of the altitude measurements determines the pixel size. The more points the base set provides and the closer together they are, the greater the level of detail of the resulting raster

Spatial resolution in raster data is usually expressed in pixel size. If the spatial resolution is 10 meters, any object smaller than 10 meters will not be recognizable for what it is. A pixel covers a certain spatial extent, determined by its width and height.

Figure 2.12 illustrates the difference various pixel sizes make.

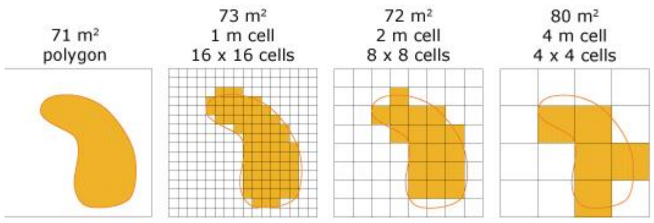


Figure 2.12: The influence of cell size in spatial resolution (adapted from [9])

Images with a pixel size covering a small spatial extent are high resolution images as they show a high level of detail. On the other hand, images with a pixel size covering a large spatial extent are low resolution because the level of detail in the image is small.

One thing that has to be considered when choosing the adequate spatial resolution for an application is that cell size directly influences the amount of disk space and processing power needed. The smaller the cell size, the greater the level of detail as well as the storage space and processing time needed.

The difference between map scale and spatial resolution often leads to confusion. Spatial resolution describes the size of a single pixel. For example, 10×10 meters means that the image has a resolution of 10 meters. Map scale describes the ratio between a distance unit on the ground and a distance unit on the map. For example, a scale of 1:25.000 centimetres means that one centimetre on the map covers 25.000 centimetres (250 meters) on the ground.

The higher the resolution, the more detail can be seen on the image. On the other hand, the smaller the scale the less detail can be made out. The spatial resolution does not change, regardless of the scale the image is displayed at.

2.3.3 Georeferencing Images

The term georeferencing in general describes methods and techniques for assigning a location reference to a piece of information, or simply put, to specify the location of an object or phenomenon [24]. There are various ways to reference an object's location: placenames, postal addresses or postal codes and cadastral systems are some that are used universally.

To be truly useful, a georeference should have these two properties: it should be globally unique and persistent in time. Uniqueness is desired so as not to confuse one location with another. Changing a georeference would result in additional confusion, as well as rendering it impossible to work with a georeference in a GIS context over a longer period of time.

Placenames, for instance, may refer to more than one location. To make them truly unique, additional references have to be given. An example for this are city or town names in North America. Many populated places share the same name. To distinguish between these locations, the state name is appended to the place name.

Another class of georeferencing methods are the so-called *metric* systems. These systems are based on measurements of the earth rather than names or codes. Included in this class are the systems that are most important in geoinformatics: coordinates systems. The well known and globally used latitude/longitude system, as well as other coordinate systems are among them. Universally used systems are UTM as well as World Geodetic System 1984 (WGS84), other systems are limited to certain areas of the globe, like the State Plane Coordinate System used in the USA or the National Grid of Great Britain.

Metric georeferencing systems have various advantages for the use in geoinformatics: first of all, it is possible to do calculations with numeric coordinates. This means that relations between two or more locations can be calculated, e.g. distances. Another advantage is that the spatial resolution of a location can be as accurate as needed: the only limiting factors are the measuring equipment and the number of available decimal places.

The term georeferencing, when applied to raster data, describes a two step process. First, spatial reference information is added to a raster image or data set, second, its pixel coordinates are transformed to the desired map coordinate system. Sometimes these two are seen as distinct concepts, referred to as georeferencing and geocoding respectively [27].

The spatial reference information added to the image comes in the form of a Spatial Reference System (SRS) that provides the basis for the map coordinate system, as well as positional information, consisting of Ground Control Points (GCPs) and a transformation matrix.

The SRS consists of two parts: the first is a geodetic reference system including a reference ellipsoid, geodetic datum, measurement unit and map projection. The second is the coordinate system, which can be either of the following: geographic, geocentric, projected or local.

There is a tremendous amount of SRS in use throughout the world today. To avoid confusion, they need to be specified in a unique, extensible and preferably standardized way.

One such system is the OGC's Well-known Text (WKT) system, specified in the Simple Feature Access specification[6]. It uses a markup language to specify the integral parts of a SRS. The WKT string specifies the datum, ellipsoid, coordinate system and map projection. WKT is widely used to store and transfer spatial reference information, for example ESRI uses a variant of the OGC's WKT to specify the SRS for its shapefiles. WKT as

[6]http://www.opengeospatial.org/standards/sfa

well as its binary counterpart Well-known Binary (WKB) are supported by numerous applications and database systems.

The following string gives an example of WKT strings. It is a representation of the reference system WGS84, omitting the map projection.

```
GEOGCS["WGS84",
DATUM["WGS1984",
SPHEROID["WGS84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],
AUTHORITY["EPSG","6326"]],
PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4326"],
AXIS["Longitude",EAST],
AXIS["Latitude",NORTH]]]
```

There are other systems in use to specify SRS, such as the system employed by the OGC's Geography Markup Language (GML) or the Proj4[7] string which would represent WGS84 in the following form:

```
+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
```

SRSs are usually maintained by national mapping authorities, who ensure that all the parameters are correct and up-to-date. Another authority that has gained importance as a reliable source for SRSs over the last few years is the International Association of Oil and Gas Producers (OGP), formerly known as the European Petroleum Survey Group (EPSG).

This organization has compiled a comprehensive database, the Geodetic Parameter Set. It contains reference ellipsoids, geodetic datums, measurement units as well as projected and geographic coordinate systems. Every piece of data is identified by an EPSG code. The database is both available online for queries and download.

Another important point of georeferencing in regard to raster data is the manner in which the reference is stored. There are two ways to accomplish this: the first one is internal storage, which means that all the spatial information is stored within the image file itself, usually in the file header. The second method is external storage: information about the SRS is saved in an external file or database table.

As the georeferencing process involves two distinct steps, some file formats use a combination of the two storage methods. Sometimes a separate file is used to store information about the SRS, or it is stored in the file header, while the transformation data is stored in a file or vice versa.

A popular external storage option for transformation data is the use of a *world file*. The world file was introduced by ESRI as a means to store transformation data in an external, easily readable file format. The file itself is a simple, plain-text file. It includes the information needed to transform the

[7]http://trac.osgeo.org/proj/

pixel coordinate system into a map coordinate system using a six parameter affine transformation, but it does not contain any information about the raster's SRS. The parameters are stored as six lines with the following content [11]:

Line 1 (A): dimension of a single pixel in map units in x direction

Line 2 (D): rotation terms

Line 3 (B): rotation terms

Line 4 (E): negative dimension of a single pixel in map units in y direction

Line 5 (C): x coordinate (in map coordinates) of the upper left pixel's centre

Line 6 (F): y coordinate (in map coordinates) of the upper left pixel's centre

Parameter E is negative, because the coordinate origins of raster images and maps are different. Images have their origin in the upper left corner while a map has its in the lower left corner.

The naming of the world file follows either of these two conventions: the world file bears the same name as the image file, but the letter *w* is added to the extension. A file raster.jpg thus becomes raster.jpgw. The other way is to take the first and third letter of the original extension and add a *w* to it – raster.jpg becoming raster.jgw.

This is the information needed to georeference (and geocode, according to [27]) a raster image. The actual image-to-world association is done by using GCPs.

A raster image is georeferenced using already referenced spatial data. The SRS of this data is known. The referencing is done by establishing a link between a point on the already referenced data and a point on the raster data – this point is referred to as a GCP. These GCPs are points that are easy to identify on both datasets, like a crossroads or a confluence. The points are then utilized to warp the raster image from its own coordinate system to the system used in the map. In this process, every pixel is assigned a real-world coordinate from the appropriate system. The georeferencing process yields better results if the GCPs are not clustered in on area of the image, generally one is placed in every corner and some are positioned over the remaining area.

Raster data does not have any explicit location information, it is implicitly contained in the grid structure. This grid structure defines location within the raster through coordinate origin, orientation, cell size and matrix dimension.

The transformation process changes the origin of the raster to that of the map coordinate system, which is generally in the lower left corner. It transforms the discrete pixels into continuous map units and rotates and translates the image if necessary.

As is the case with most transformations and conversions, this process is error prone. The main error source comes from the fact that it has the be known if a map coordinate represents the pixel's centre point or one of its corners. If this is not taken into account, a potential error of magnitude $\sqrt{2} \times \delta x/2$ may be introduced.

There are several transformation methods available: the affine (polynomial), spline or adjust transformations.

The first-order polynomial transformation is widely used in GIS applications [10]. To apply an affine transformation, three GCPs are sufficient, but it is recommended to use more than that. The affine transformation ensures overall accuracy of the transformed image, although it does so at the cost of localized inaccuracies. Second and third order polynomial transformations are used as well, although they are more complex and thus require more processing time as well as GCPs.

The affine transformation is carried out using six parameters. These parameters are stored in the aforementioned world file (see description in 2.3.3). They are used to construct the transformation matrix to compute the map coordinates:

$$\begin{pmatrix} x\prime \\ y\prime \end{pmatrix} = \begin{pmatrix} A & B \\ D & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} C \\ F \end{pmatrix}$$

This matrix is applied to every pixel in the image, transforming the original x and y to $x\prime$ and $y\prime$, the computed map coordinates. A and E are the scale factors in x and y direction respectively; B and D the rotation factors (which are usually zero) and C and F the translation terms.

The spline transformation is not optimized for overall accuracy, but does ensure local accuracy. It needs at least ten GCPs, but using more points makes the transformation more precise.

The adjust transformation is optimized for overall as well as local accuracy. It uses both a polynomial transformation and interpolation methods borrowed from TINs.

Chapter 3

Tiled Mapping

Tiled Mapping, although it has been known and in use for some time, has gained momentum over the past few years. Faster and cheaper Internet connections, as well as sophisticated software technology for dynamic web pages paved the way for web map services such as Google Maps and Bing Maps. These services are largely responsible for the increased popularity and use of Internet map services.

This chapter aims to explain the different aspects of tiled mapping: cutting a map into tiles, storing and retrieving tiles and reassembling them into a seamless map.

It starts with the elementary building block: the tile itself.

3.1 Preliminary Considerations

Tiled maps are not suitable for every kind of map and application, but for web map services they are an ideal choice. In this area, tiled maps can be seen as the industry standard. Their success is not based on tiles alone, but on the use of tiled maps in conjunction with Internet technologies such as AJAX, which provide the APIs that make these services readily accessible and easy to use for software developers and users alike.

Tiles provide the base for these services. Tiled maps can be rendered at high speed - the tiles may be small maps, but to client applications – i.e. web browsers – they are just a set of images to be displayed. This way users do not have to wait for the map to be drawn, and then displayed.

Another advantage of tiles is that the information content and quality of the map image can be preset. In some map services, every piece of information has to be redrawn when the user pans or zooms the map. A tiled map service simply loads the relevant prerendered tile images. The prerendered image may be a terrain map, a street map or any other type of map - this has little impact on loading times. An image file with a high information content will be slightly larger than one containing little information, but the difference in loading time will be almost imperceptible. Of course, this can be seen a drawback as well - map tiles are fairly static. Hence they are unsuited to present realtime data by themselves. They can be used as basemaps, with the realtime – or other – data added as an overlay, using an API. This type of map has become known as a map mashup. Map mashups are explained in greater detail in section 4.2.

3.1.1 Map Considerations

Tiles are cut from maps, so the first thought before creating tiles has to be about the original map, as it provides the base for the tessellation. An important property of the map in this context is information content and closely linked to it, scale.

A tiled mapping service consists of the same map at different scales. The map has to be legible and easy to understand at all scales. In order to achieve this goal, information content and map design have to be considered carefully for each scale.

To illustrate the concept of a set of maps at different scales - a multiscale map - a comparison to the image pyramid can be made (see 2.2.1). An image pyramid is a useful tool to show an image at different resolution levels. It usually starts with a high resolution image at the base and shows the same image at lower resolution levels as the pyramid rises. A multiscale map implements the same concept: at the base of this "map pyramid" lies the map at its smallest scale, containing the highest amount of information and detail. As the pyramid rises, the spatial extent stays the same, but the level of detail decreases. For example, the individual buildings in a city are replaced by a polygon, or the city that is represented as a polygon becomes a point as the scale changes.

The process of adapting the information content, or level of detail, contained in a map is known as generalisation [24]. Often, scale serves as the rule that specifies the level of detail a map contains. There are several geometric methods to adapt spatial objects to the level of detail in order to create a useful map: *simplification* aims to reduce the number of vertices e.g in a polyline or polygon boundary and *collapse* replaces an areal object by a point object, to name but two.

A completely different class of generalization concerns attributes of spatial objects. There are two ways to generalize on an attribute level: *classification* reduces the number of classes by reclassifying attributes, thus merging and reducing object classes, whereas *symbolisation* replaces map symbols with other, more appropriate symbols. An exhaustive survey of generalization methods for digital cartography is given by McMaster and Shea in [25].

Scale is important to determine the level of detail contained in a map, but not the only rule that controls generalization. The purpose and audience of the map have to be kept in mind as well. Along with the information content, symbology and labelling have to be clear and adapt to the different scales.

3.1.2 Tile Considerations

In a mathematical sense, a tile is a plane figure, and a tiling is a set of disjoint figures that completely covers the plane. In geoinformatics, the definition of the term *tile* is more specific. In the Web Map Tile Service (WMTS) Implementation Standard, the OGC defines a tile as

> "‘ a rectangular single piece of a uniform fragmentation of a pictorial representation of geographic data that can be accessed by a pair of row, column indices and scale value."’ [31]

This single piece of a pictorial representation has a number of important properties that have to be considered before a map is tiled.

Tile size directly influences performance, storage and compatibility with other tile based map services. A smaller tile size puts applications that use tiles at an advantage: if tiles are requested, less data has to be transmitted, thus decreasing data traffic and bandwidth needs. On the other hand, a small tile size means that more individual files have to be generated, which can increase storage requirements. If the tiles are stored in a database, smaller tiles lead to an increased number of table records, thus slowing down queries. This is especially important when considering that the number of tiles quadruples with every increase in resolution level.

A larger tile size may optimize disk usage, but it can also increase data traffic. This is especially important if the tiles are created for use by an internet service, where data traffic and bandwidth considerations can be significant factors.

Most tiled map services use 256 pixels as the default tile size [44] [8]. Other tile sizes may lead to compatibility problems if they are to be used as overlays or basemaps in conjunction with other tile services. If the tile size is smaller than 256 pixels, then the tiles have to be stitched together before they can be used as overlays, tiles using a larger size might need to be cut up. The de-facto default tile size of 256 is not arbitrary: any tile size that is a power of two makes calculations – e.g. transforming a geographic into a pixel coordinate – easier and is more storage effective than other tile sizes.

Scale Tiled maps do not use scale as it is known from paper maps or GISs. Every tile comes at a fixed scale - the *zoom level*. Tiled mapping systems usually provide a limited amount of zoom levels, starting at level 0 or 1. At this starting level, they show the whole extent of the map.

The highest level of detail is usually dictated by the information content and purpose of the map, rarely ever exceeding level twenty. This is because the number of tiles rises with every zoom level.

Which zoom level represents which scale has to be determined before the tiles are cut. Consecutive zoom levels usually present the map at half the scale of the preceding level.

The number of available zoom levels influences the whole tile creation process as well as the number of tiles, and thus the storage requirements. The more zoom levels there are, the more tiles are generated - and the number of tiles quadruples with each level. To illustrate this, the following example taken from Microsoft's Bing Maps service is used [44]:

At zoom level one, corresponding to a scale of 1:295.829.355, the whole earth is covered by four tiles. The highest available level is twenty-three, at a scale of 1:70. To cover the whole earth at this level, approximately 7×10^{13} tiles are needed.

Spatial Resolution The distance on the ground that is covered by a single pixel changes with each zoom level. To calculate the spatial resolution, or *ground resolution*, as it is sometimes called, the latitude has to be taken into account as well.

As most tiled mapping systems use a Mercator projection, spatial resolution is fairly easy to calculate. The following formula, adapted from [44], can be applied to most tiled map services. The earth radius is is assumed to be 6378137 meters and the map width is in pixels.

$$res = \cos(lat \times (\pi/180)) \times earthradius/mapwidth \qquad (3.1)$$

Spatial Reference The SRS that georeferences the tiles is the same as the one that the map as a whole uses. If the original reference system cannot be used for some reason, the whole map has to be reprojected before it can be cut into tiles.

The vast majority of tiled map services use the WGS84 Web Mercator (Auxiliary Sphere) projected coordinate system, which has become the de-facto standard for tiled maps.

Shape Square tiles are the de-facto standard in the world of tile based mapping. All the big service operators, like Google and Microsoft, use square tiles. This shape provides the easiest means of creating seamless maps from the individual tiles. It also provides the easiest base for the calculation of distances and other geometric properties.

Image Format The image format used to store the tiles influences the overall quality of the map that is presented. If has an impact on storage needs as well as data traffic. It may also limit the number of uses the tiles can be put to: image formats that support transparency allow tiles to be draped over a basemap.

The most common image formats for tiles are PNG and JPEG (for a more detailed description see section 2.2.2). JPEG files are generally smaller in size, but as it uses lossy compression, the image quality may suffer. PNG uses a lossless compression method, hence the files are larger. One advantage of PNG is that it supports background transparency.

These tile properties are a part of the design choices to be made before cutting a map into tiles. They are in part determined by the conditions dictated by the map itself and the system: information content, scale dependencies as well as available storage space and bandwidth.

3.1.3 System Considerations

Tiling a map, storing and updating as well as maintaining the tiles can make high demands of a computer system. To meet these demands, the system has to be well designed, with the properties of the map and the proposed map service in mind. These requirements can be divided into two parts: tile creation and tile storage.

Tile Creation: The biggest performance penalty of tiled mapping occurs at creation time. It can be time consuming to create large tile sets for multiple resolution levels. The time needed to tile a map depends on the available processing resources, the number of zoom levels, the map size and the map's information content. The parts of the map that contain more detail include more colours and changes of colours, thus increasing the file size of these tiles. Areas that are more homogeneous yield smaller files. It can be equally time consuming to copy or move the tile sets from one machine to the other, as it can occur when switching from a test to a production environment.

Tile Storage: When considering system requirements for a tiled map, another important point is whether the tiles are held in a database of simply stored in the file system. These two storage possibilities are discussed in greater detail in section 3.3. Most tiled maps use prerendered tiles, stored in a file system, but tiles can also be rendered on the fly, when a client requests them. For this, either vector or raster data can be the basis. This method poses different requirements than the simpler prerendering method.

3.2 Tile Cutting

Tile cutting is a simple and straightforward operation, and it has been used in image processing applications for some time. Its main use in image processing is similar to its use in tiled mapping: to increase performance when dealing with large images at multiple resolution levels by creating either seamless or overlapping image mosaics.

It has been implemented in many graphics editing and processing applications under different names. Photoshop variously calls it image slicing or mosaicking, ImageMagick refers to the operation as cropping.

3.2.1 Prerendering Tiles

Most tiles are prerendered, not served on demand. This is crucial to the success of tiled mapping on the Internet, as it enables the fast display and seamless zooming and panning as seen in Google Maps and other services. Prerendering means that tiles are not cut when a client application requests them, but before they are actually needed. They are then stored for future use.

Tile cutting algorithms range from sophisticated ones for irregular tessellations to simple ones for regular tessellations. Algorithms for irregular tessellations include triangulation methods like the Delaunay triangulation and construction algorithms for Thiessen polygons (see 2.1.2). A regular square tessellation can be accomplished using e.g. a quadtree data structure. These algorithms can be applied to both raster and vector data.

The algorithm for regular square tessellations of a raster map is conceptually simple. In a nutshell, every resolution level of a chosen image is subdivided into squares, starting and stopping at a predefined level.

If a raster map provides the input data, the algorithm may perform other operations in the course of the tiling as well, like resampling or compressing.

Problems arise if the original raster map size is not a power of two. There are two ways to deal with this problem, one uses a preprocessing approach, the other a method that corrects the size while creating the tiles.

Resizing: In this case, the original map is resized before the tiles are created. Two of the resizing operations available are stretching and shrinking the image. Another resizing method that is rarely used, as it represents a loss of information is the simple omission of edge pixels.

Image resizing belongs to the group of interpolation methods. In image processing, interpolation represents the usage of known pixel values to estimate the values of unknown pixels [14]. There is a multitude of image resizing techniques in use today. Commonly used methods use a nearest neighbour approach, differing mainly in the number of neighbouring pixels used in the interpolation.

The simplest method is the nearest neighbour method itself, which uses one neighbouring pixel only to calculate the new value. This approach causes serious artefacts especially along straight lines, and is therefore rarely used.

Two prominent and widely implemented methods are bilinear and bicubic interpolation. The bilinear method uses four neighbouring pixels to calculate the unknown value, whereas bicubic interpolation uses sixteen pixels. Bicubic interpolation is the default method in image processing applications such as Adobe Photoshop [14]. It is more complex and thus computationally more expensive than the bilinear variant.

There are many more interpolation methods of various complexity which are suitable for different applications, like polynomial and spline interpolation.

The process of resizing an image always introduces distortion. On the one hand, the amount of distortion depends on the original image size: the more pixels have to be interpolated to reach a power of two, the higher the distortion. On the other hand, the manner of distortion depends on the interpolation method used, e.g. bicubic interpolation is better at preserving details than the bilinear method.

Padding: This method pads the edge tiles that do not have the predefined size due to the map's inadequate dimension with spacer pixels until these tiles reach the chosen size. This method does not distort the map image itself, but it adds artefacts to some of the tiles. The size of these artefacts depends solely on the number of pixels needed to pad the tiles in question. If a file format that supports transparency, e.g PNG, is used, the disruption caused by these artefacts can be kept at a minimum.

The padding method is computationally less expensive than resizing through interpolation, although it may make the edge tiles visually less appealing. Both of these approaches are used by tile cutting applications - e.g. Microsoft's Map Cruncher uses the padding method.

3.2.2 Tile Cutting Applications

As mentioned above, tile cutting is a part of many image processing applications. There are a lot of standalone image tile cutters as well, e.g. Zoomify[1]. Most of these applications only cut an image into tiles, without providing the functionality needed to create tiled maps: georeferencing and reprojecting.

Due to the increasing popularity of tile based mapping services, applications have been developed specifically to cut raster maps into seamless square tiles. These applications generally refer to the operation as tile cutting, or simply tiling. GIS software like ArcMap and ArcGIS Server provide tools to create tiles from both vector and raster data as well as store and manage the tiles. There are also third party extensions like Arc2Earth[2], that offer additional features, such as creating metadata for Google Earth.

The choice of the tile cutting application determines important properties of the output tiles, e.g. image format and tile addressing system. These

[1]http://www.zoomify.com/
[2]http://www.arc2earth.com/

properties are important for client applications that use the tiles as well as general system considerations, such as storage requirements. Tile addressing systems are discussed in greater detail in section 3.4.

Among the standalone applications that offer the full functionality necessary to create tiled maps from raster data are MapTiler, Microsoft Map Cruncher and Gmap Image Cutter. An example for an application that creates tiles from vector data is GMapCreator.

Table 3.1 gives an overview of the functionality and features of all four applications.

	Map Cruncher	MapTiler	GMap ImageCutter	GMapCreator
Georeferencing	positioning of GCPs	world file or manual coordinate input	none	takes vector data with PRJ file as input
Reprojecting	yes	yes	no	yes
Input Formats	JPEG, PNG, TIFF, PDF, BMP, EMF, WMF	JPEG, PNG, TIFF, MrSID, HFA	JPEG, PNG, TIFF, GIF, BMP	Shapefiles, CSV
Spatial Metadata	proprietary XML file	Tiled Map Service (TMS) tile map resource	none	none
Adjustable Properties	zoom level	zoom level, output format, sample viewers	zoom level	zoom level
Sample Viewers	Bing Maps	Google Maps, Google Earth and OpenLayers	Google Maps	Google Maps
Output Formats	PNG	PNG, JPEG	PNG, JPEG	PNG
Tile Addressing System	Bing Maps Quadkey	Google Maps, TMS	Google Maps Quadkey	Google Maps Quadkey

Table 3.1: Comparison of tile cutting applications

GMapCreator

This application[3] - developed by the Centre for Advanced Spatial Analysis at the University of London - supports vector data as input; shapefiles and Comma-separated Values (CSV) files to be precise. The shapefile may contain polygons, lines or points. To transform the data to the Mercator projection used by Google Maps, a Projection (PRJ) file needs to be present. GMapCreator uses the GeoTools[4] library to do the actual transformation.

The application was developed to facilitate thematic mapping using Google Maps as a basemap [2]. A thematic map displays the spatial pattern of a non-spatial phenomenon on a map. These phenomena are represented by attributes. Different attribute values, or groups of values are typically represented by different colours. GMapCreator allows the user to choose the attribute, set the value groups and adjust the colours. The application also takes into consideration that the number of tiles quadruples with every additional zoom level by allowing the user to set the starting and ending level. To further limit the number of tiles, the user can set the spatial extent tiles should be generated for. This is a useful feature if the input shapefile covers a large area, but only a subregion thereof should be turned into tiles.

Figure 3.1 shows a thematic map using Google Maps as a basemap, the overlay generated by GMapCreator depicts the number of households per county in the state of Florida. The sample viewer this image is taken from is created by the application.

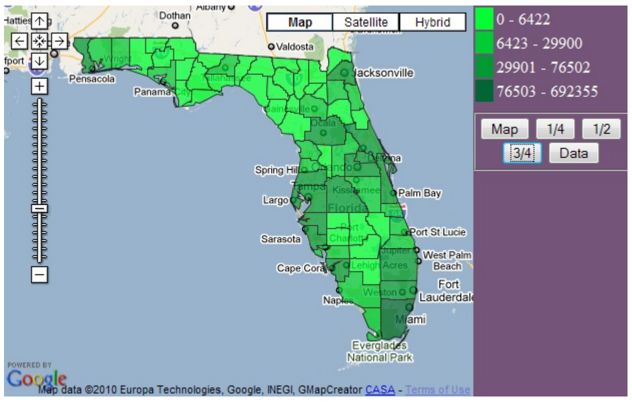


Figure 3.1: Overlay tiles created by GMapCreator used with Google Maps

[3]http://www.casa.ucl.ac.uk/software/gmapcreator.asp
[4]http://www.geotools.org/

GMap Image Cutter

Like the GMapCreator, the GMap Image Cutter was developed by the Centre for Advanced Spatial Research[5]. This application is in fact a simple image cutter that does not enable georeferencing or reprojecting. It does not take any spatial information provided with the map image, e.g. world files into account. The reason why this program is nevertheless interesting for tiled mapping is that it uses the Google Maps satellite imagery tile addressing system to label the generated files, hence making it possible to use the tiles in conjunction with Google Maps. It also creates a sample web page for the generated tiles which utilizes Google Maps to display the image.

This way, the tiles can be used for tiled mapping with little extra effort.

MapTiler

MapTiler[6] is based on utilities provided by the Open Source software package Geospatial Data Abstraction Library (GDAL). Specifically, it uses *Gdaltranslate* to georeference the map image, *Gdalwarp* to reproject it and *Gdal2tiles* to do the actual tiling. MapTiler combines all these utilities and provides them with a Graphical User Interface (GUI).

MapTiler implements the OSGeo's TMS specification. This standard will be explained in greater detail in section 3.4.2. MapTiler not only creates the tiles themselves, but the necessary directory structure specified by the TMS as well. It does not create spatial metadata in the form of a WKT file, or a world file, but uses the format specified by the TMS specification instead [12].

To georeference the image, either a world file or the map's bounding box coordinates can be used. The *Gdalwarp* utility is used to reproject the image. It provides a variety of options to specify the spatial reference system: WKT files, EPSG codes as well as the PRJ files can be used.

MapTiler provides support for a host of input raster formats: PNG, JPEG, GIF, TIFF and Bitmap (BMP) are among the commonly used file formats. Besides these the application supports file formats specific to GIS software, e.g MrSID and Erdas Imagine's Hierarchical File Architecture (HFA) format. Another input possibility is provided in the form of access to Web Map Service (WMS) servers.

MapTiler was developed with the publishing of raster maps on the Internet in mind. Therefore it provides a number of options to generate simple Hypertext Markup Language (HTML) pages to view the generated tiles as overlays in web map services like Google Maps and web map toolkits OpenLayers. It also provides an option to generate a Keyhole Markup Language (KML) overlay for use in Google Earth.

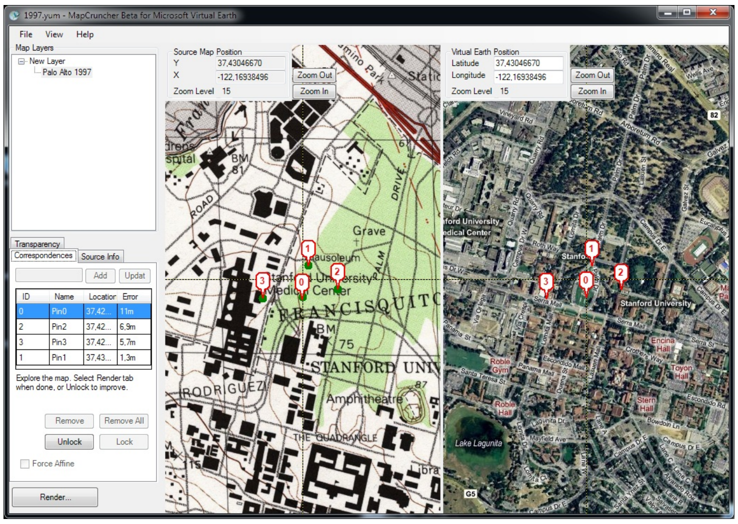
[5]http://www.casa.ucl.ac.uk/software/googlemapimagecutter.asp
[6]http://www.maptiler.org/

Map Cruncher

Map Cruncher is a tool developed by Microsoft Research[7]. Like MapTiler, its purpose is to tile raster maps in order to publish them on the Internet - in this case in conjunction with the Bing Maps service. It provides a GUI and it handles georeferencing and reprojecting as well as tiling a map.

The map is georeferenced by positioning GCPs. The already referenced data necessary to georeference the map is the same map data and aerial imagery that the Bing Maps service utilizes. This data uses the de-facto standard projection and datum for tiled web map services: a Mercator projection and the WGS84 datum.

Figure 3.2 shows Map Cruncher's user interface: the left map contains the user map that has to be tiled, the map on the right hand side is the Bing Maps reference data.


1997.yum - MapCruncher Beta for Microsoft Virtual Earth

Source Map Position
Y 37,43046670
X -122,16938496
Zoom Level 15

Virtual Earth Position
Latitude 37,43046670
Longitude -122,16938496
Zoom Level 15

ID	Name	Location	Error
0	Pin0	37,42...	11m
2	Pin2	37,42...	6,9m
3	Pin3	37,42...	5,7m
1	Pin1	37,43...	1,3m

Explore the map. Select Render tab when done, or Unlock to improve.

Figure 3.2: The Map Cruncher user interface

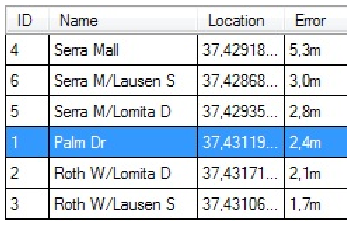
The figure also shows the GCPs, which are referred to as correspondences in Map Cruncher. In contrast to MapTiler, the input map's projection does not have to be known, as Map Cruncher uses an approximate reprojection to warp and position the source map [7]. To accomplish this, two transformation methods are available: the affine and the bivariate polynomial reprojection. In case of the affine transformation, both the standard method using six parameters (see 2.3.3) and a subclass called rigid reprojection are used. The rigid reprojection method requires only two GCPs to be positioned by the user, the third point is determined by forming a right isosceles triangle including the two positioned points on both maps. The rigid reprojection

[7]http://research.microsoft.com/en-us/um/redmond/projects/mapcruncher/

is especially useful if the source map uses a conformal projection, like the Mercator projection.

The polynomial transformation used is a quadratic reprojection, which is better at preserving curves than the affine method [7]. The disadvantage of the quadratic transformation is that it requires the user to position six GCPs.

To give the user the opportunity to further adjust the accuracy, Map Cruncher displays the GCPs as well as the estimated deviation in meters between the point the user positioned and the calculated point on the reference map. Figure 3.3 shows the table containing the points, sorted by deviation.



ID	Name	Location	Error
4	Serra Mall	37,42918...	5,3m
6	Serra M/Lausen S	37,42868...	3,0m
5	Serra M/Lomita D	37,42935...	2,8m
1	Palm Dr	37,43119...	2,4m
2	Roth W/Lomita D	37,43171...	2,1m
3	Roth W/Lausen S	37,43106...	1,7m

Figure 3.3: Map Cruncher ground control points and estimated error.

Map Cruncher stores spatial metadata in a separate file in Extensible Markup Language (XML) format. This file contains information about the transformation as well as information about the map's spatial extent. It does not contain information about the tiles' spatial reference system.

3.3 Storage

Tile storage goes hand in hand with tile cutting - as soon as the original map is divided, the tiles need to be stored. This can either be done by storing the tiles as files in a file system, or as records in a database.

3.3.1 File Systems

From a user's point of view, a file is the smallest unit of data that can be stored on a storage medium. From a conceptual point of view, a file is a collection of related data. A file system provides the utilities and organisational tools that enable a user to create, save, name and manipulate files [3].

Most modern file systems are hierarchically organized, which means that they have a starting point, the so-called root directory which contains all other directories or files. The directories in the root may contain other directories, known as subdirectories. This directory structure is often referred to as the directory tree. The tree can reach an arbitrary level of depth, or it may be limited by constraints dictated by the operating system.

Directory structures play an important part in tile addressing systems, and are often specified in the pertaining standards, e.g. TMS.

File systems are an integral part of the operating system, hence there are a myriad of file systems in existence. Most operating systems provide support for more than one file system, sometimes even for file systems of other operating systems. Linux based systems largely use a member of the ext* family of file systems, while Windows based systems use the New Technology File System (NTFS), or File Allocation Table (FAT) in older versions.

Files are generally stored in secondary memory, also known as secondary storage. This form of storage has the distinct advantage of being persistent - in contrast to primary memory, which is faster, but volatile. Magnetic or optical devices as well as flash memory are usually used for secondary memory. Commonly used magnetic storage media are hard discs and magnetic tapes, optical devices include compact discs and Digital Versatile Discs (DVDs). Flash memory represents a relatively young storage technology, with memory sticks being typical representatives.

Magnetic storage, in the form of hard discs, is a convenient and cheaply available method of storing files quickly, and therefore widely used. The smallest data unit on a hard disc is called a block. The block size depends on the file system, but compared to the size of a file it is rather small - so files often span across more than one block. To access a file, the disc's read/write head moves to the physical address of the block the file is located on and scans the block until the right location in the block is reached. This mechanical movement - known as *seek time* - influences the overall performance of a system greatly. Performance, in this context, describes the response time to a file query [53]. In order to keep seek time low, a file that spans over several blocks ideally occupies physically adjacent disc blocks. The physical organization of files on a disc is referred to as *file organization*. Other hardware related factors that influence performance are the time it takes the disc to rotate to the correct position, known as *latency* and the time it takes to transfer the block into the processor, known as *transfer time*. These are some of the factors that have to be considered when storing tiles on a hard disc. The overall performance of a tiled mapping service is influenced by other factors as well, e.g available bandwidth.

File access and manipulation is in large part a task the application using the file has to manage. GIS applications provide support for a varying number of file formats - both vector and raster.

The main advantages of using a file system and files for tiled mapping are the simplicity and flexibility this concept provides. It does not take any additional software to host the tiles, and they can easily be moved or copied from one computer (and file system) to another. Traditional GIS applications often need to perform a task by searching the data contained within the file. Therefore, specialised data and access structures have been developed for different spatial file formats.

Tile based mapping services do not need to search the data within a file, what they do need to search is files within directories - hence the often elaborate directory structures employed by tile addressing systems (see section 3.4). The search for individual tile files is handled by the utilities the file system and the operating system provide.

Another problem that arises for GIS applications when using files is multi-user access. File systems - in contrast to database systems - often do not implement mechanisms that ensure data integrity and security. In this case, the applications themselves have to provide these features themselves [27]. Tiled mapping services do not need to open a file over a long period of time, as they are usually viewed rather than edited. In this case too, the on board mechanisms the file system provides are enough to handle multiple access.

3.3.2 Databases

A database is, according to [24],

> "an integrated set of data on a particular subject."

A database can either be stored in a file - like ESRI's Personal Geodatabase - or in a Database Management System (DBMS).

Databases and DBMS have a number of advantages compared to files:

- All the data is stored in a single, centralized location. This prevents data redundancy and reduces maintenance costs and efforts.

- Data becomes detached from applications - the same data can be used by multiple applications. It can be accessed over standardized interfaces and via a standardized query language, both provided by the DBMS. This query language provides tools to define data structures, manipulate data and control access.
- Multiple users can access the same data at the same time, as the DBMS provides techniques to ensure data integrity and security.
- A database is better suited to handle large amounts of data than a single file, or a number of files.
- Databases can be indexed - this increases search speed considerably. Every DBMS provides a number of different index methods, like the b-tree or quadtree indexes.
- Databases provide transaction management. This means that any changes made to the data - update, deletion or insertion - are monitored and can be reversed. Databases implement the ACID principle - atomicity, consistency, isolation, durability. Atomicity means that if one part of a transaction goes wrong, the whole transaction goes wrong. This ensures that the data is in a consistent state, before and after every transaction. Data that is used in an ongoing transaction is isolated, which means that other transactions cannot access the data as long as it is not in a consistent state. If the transaction is completed successfully, and the database is in a consistent state, then the changed data has to be stored durably. If any kind of failure occurs, the changes the transaction made can be fully reversed. This functionality is provided by the DBMS, in a file based system it has to be implemented by the applications using the data.

The downside of databases is that with a DBMS another, rather sophisticated, piece of software is added to the stack. The DBMS has to be configured and maintained, and the data administered. Professional DBMS are expensive, both in acquisition and maintenance. The decision whether to acquire a database or to use files has to be considered carefully. For a small number of users file based solutions may be more efficient and economical.

There are three main types of DBMS in use today. Relational DBMS consist of a number of tables, which in turn are made up of rows and columns. This is by far the most widely used database type. Object DBMS are able to store objects - complete with state and behaviour - in the database, as well as providing object query tools. The third type is a hybrid of the other two types: the object-relational DBMS. It combines the relational model with the capability to store objects. This type is the most important for GIS, and most databases that offer support for spatial data are object-relational DBMS [24].

This support usually comes in the form of a spatial extension to the actual database. These extensions offer spatial data types to store geographic data as well as functions and query tools to manage and retrieve the data.

The OGC's Simple Feature Access specification specifies such data types and functions. The data types are point, line and polygon, and they are defined as objects. The base object is *geometry*, and it has an associated spatial reference system, consisting of a coordinate system and projection. All other objects, like line linear rings and polygons are derived from this base type. The defined functions are intended to test spatial relationships between two spatial objects, and include methods like touch, within and disjoint.

Most spatial database extensions implement the Simple Features Access specifications at least in part - like SQL Server[8] and Oracle Spatial[9].

The data types defined in the Simple Features specification are intended for vector data, and the specified functions are intended to be performed on vector data. Tiled mapping can make use of vector data, but its primary data source is raster data. How spatial databases store and manage raster data will be explained in the following sections with the help of two examples: Oracle Spatial and PostgreSQL/PostGIS.

Oracle Spatial

To store raster data, Oracle Spatial provides the SDO_GEORASTER data type. Oracle Spatial is an object-relational DBMS, and this data type is an example for the functionality based on the object model.

To add raster data to the database, a column of type SDO_GEORASTER is added to a table. To insert the image data, Oracle Spatial provides an import procedure. This function takes various parameters to describe the image. For one, the file format has to be specified. Supported file formats include PNG, GIF and TIFF. The import function allows the specification of the block size to be used in the raster data table, both are explained later in greater detail. Other parameters allow the import of an associated world file, as well as other properties describing the image.

[8]http://www.microsoft.com/sqlserver/2008/en/us/spatial-data.aspx

[9]http://www.oracle.com/technology/products/spatial/htdocs/data_sheet_9i/9iR2_spatial_ds.html

The SDO_GEORASTER class has five attributes, described in table 3.2 [20] [33]:

Name	Type	Description
rasterType	number	The raster type consists of five digits, specified in the form [d][b][t][gt]. [d] represents the number of spatial dimensions, currently only two dimensional data is supported. [b] describes the number of bands in the image: 0 specifies a single band, 1 a multi band raster. [t] will specify if a temporal dimension exists in future versions of Oracle Spatial. [gt] is the georaster type - currently only 01 is valid.
spatialExtent	SDO_GEOMETRY	This attribute represents the image's Minimum Bounding Rectangle (MBR), using the SDO_GEOMETRY data type. The SRS is stored in the SRID attribute. If the image is georeferenced, the coordinate system is the same one the raster image uses. If it does not have a spatial reference system associated, the MBR is set to null.
rasterdataTable	varchar	Specifies the name of the associated raster data table, which stores cell information of the image. This attribute will be discussed in greater detail later on.
rasterId	number	Together with the RASTERDATATABLE attribute, this serves as the unique identifier for the raster object.
metadata	xmlType	This attribute can store additional information, e.g compression type and quality as well as transformation coefficients for georeferencing.

Table 3.2: The SDO_GEORASTER class structure

The attribute rasterDataTable plays an important role in the structure of the raster image storage model. It has the data type SDO_RASTER, which is used to store raster data in general. The raster data table stores the image's cell data - the actual image data.

Images can be very large, therefore they are split into *blocks*. Every piece of data is stored as a raster block, uniquely identified by the row and column block numbers in the SDO_RASTER object. If the image has more than on band, the band block number is added to the unique identifier. The raster block is of type Binary Large Object (BLOB). Besides images, this data type is used in databases to store audio and video data, even executable programs can be stored. Oracle BLOBs can hold up to four gigabytes of data.

Figure 3.4 illustrates the blocking concept employed by Oracle Spatial for a single band image.

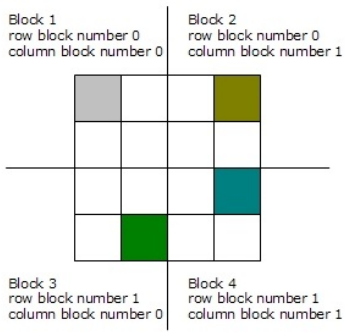


Figure 3.4: Blocking in Oracle Spatial

For multi band images, this concept has to be adapted to accommodate the additional information. Oracle Spatial uses *interleaving* to store the cell values for each band in the image. There are several methods in existence to do this:

- Band Sequential: This method simply stores the cell values for one band after the other.
- Band Interleaved by Line: This method stores the cell values of the first line of the first band, followed by the first line of the second band until every first line of every band is stored, and then continues with the second line of the first band.
- Band Interleaved by Pixel: Every cell of every band is stored after the other.

The best method depends on the application the image is intended for. The interleaving method can be specified when the image is imported.

Table 3.3 show the structure of the SDO_RASTER data type.

Name	Type	Description
rasterId	number	This identifier must be the same as the id in the corresponding SDO_GEORASTER object.
pyramidLevel	number	This attribute specifies the resolution level in the image pyramid this block belongs to.
bandBlockNumber	number	The number of the block's band
rowBlockNumber	number	The number of the blocks' row.
columnBlockNumber	number	The number of the block's column.
blockMBR	SDO_GEOMETRY	The bounding box of this block. The coordinate system uses the image's own pixel space, so the SRID of the SDO_GEOMETRY is set to null.
rasterBlock	BLOB	The actual image data contained in the specified block.

Table 3.3: The SDO_RASTER class structure

Oracle Spatial provides several techniques that are important for tiled maps: georeferencing, indexing and image pyramids as well as functionality to divide a raster image into tiles.

Georeferencing: There are two ways to georeference an image in Oracle Spatial: when the image is first imported into the database, and afterwards through a special function.

At the time the image is imported, a world file and the SRS can be specified. This information can be added and changed after the image is imported as well. The function SDO_GEOR.GEOREFERENCE allows manual georeferencing. The input parameters for this function are the image itself, the spatial reference system id and six coefficients. These coefficients are stored in the metadata field of the SDO_GEORASTER object. They are the same ones that are specified in a world file. To warp the image, Oracle uses a polynomial transformation [33].

Indexing: Oracle provides two index methods for spatial data. Vector data – that is, data of type SDO_GEOMETRY – can be indexed using either a R-tree or a quadtree index. This provides a backdoor to index raster data as well – by indexing the spatial extent attribute in the SDO_GEORASTER object, which is of the type SDO_GEOMETRY.

The quadtree index uses a regular rectangular tessellation, which is applied to an individual geometry. The tessellation stops when a user defined termination criterion is reached. This can either be the tile size, the maximum number of tiles, or the maximum resolution level.

Morton numbers (see 2.2.3) can be calculated for the tiles within a resolution level and stored as strings or numbers. This way standard Structured Query Language (SQL) operators can be used, further increasing performance [34].

Image Pyramids: The SDO_GEOR.GENERATEPYRAMID method can be used to generate an image pyramid. As input parameters, it needs the number of pyramid levels that should be generated and the image itself. The pyramid type is specified in the metadata attribute of the SDO_GEORASTER object. Currently, only redundant pyramids are supported. This type is referred to as *decrease*. The other valid value for the pyramid type is *none*, which means that no pyramid will be created. Future releases of Oracle Spatial will support additive pyramids –pyramid type *increase* – and *bidirectional* pyramids [33].

The pyramid images are stored in the same raster data table as the original image. The pyramid level a record belongs to is specified in the pyramid level attribute.

Tiling: Oracle Spatial refers to this procedure as subsetting [20]. To cut an image into tiles, the function SDO_GEOR.GETRASTERSUBSET can be used. This function returns the specified region as a BLOB. The function takes the following parameters as input:

- GeoRaster: the raster image to take the subset from.
- Pyramid Level: the pyramid level on which to perform the subsetting.
- Window: This parameter can take on either of two types. If it is of type SDO_GEOMETRY, then the specified values represent a MBR in the image's SRS. If it is of type SDO_NUMBER_ARRAY, then it specifies the upper left and lower right corners of the image in pixel space. This parameter also determines the next one.
- Band/Layer Numbers: If the window parameter has the data type SDO_GEOMETRY, then this parameter represents the layer numbers to perform the operation on. If it is SDO_NUMBER_ARRAY, it represents the band numbers. If the parameter is null, the operation is performed on all layers or bands respectively.
- Raster BLOB: This object holds the result of the subsetting operation – the actual image data.

Another way to retrieve tiles is the the function SDO_GEOR.ExportTo. It exports the image either into a file or into a BLOB. Its primary purpose is to export the whole image, but it provides a parameter, *cropArea*, that allows the specification of a region to be cropped. It supports the PNG and TIFF file formats and exports georeference information in the form of a world file.

Oracle Spatial loads and stores raster data, thus allowing the storage of already existing tiles. It also provides georeferencing methods. But by providing a method to create image pyramids and the subsetting operation it also provides two key ingredients to create tiles at different resolution levels, thus making it suitable for tile based mapping.

PostgreSQL/PostGIS

PostgreSQL is an Open Source object-relational DBMS. Like Oracle, it has a spatial extension: PostGIS. This extension implements the OGC's Simple Features Specification. Accordingly, it provides a standard geometry data type for vector data. It also provides a geography data type for vector data – the difference being that the geography data type is based on a sphere and uses angular coordinates rather than being based on a plane like the geometry data type. PostGIS includes functions for spatial analysis as well as manipulating and processing spatial data [37].

Neither PostgreSQL nor PostGIS offer a specific data type for spatial raster data like Oracle's GeoRaster type. PostgreSQL provides support for large objects, like images and audio data. This can take on the form of the so-called *large object* facility, which stores all data from such an object in a single table called *pg_largeobject*. It returns the unique identifier of the object generated at creation time for further handling. Using this method, a single large object can have up to two gigabytes. The second possibility PostGIS provides is The Oversized-Attribute Storage Technique (TOAST). This technique uses compression to store large amounts of data. It provides support for up to one gigabyte of data [17].

Although neither PostgreSQL nor PostGIS formally support spatial raster data, the combination of the two allows the import and storage of spatial raster data in a roundabout way. To provide a georeference for the raster data, a MBR can be used. There are no special functions to process or manipulate raster data, like the generation of subsets or image pyramids.

Plans to change this are well underway, though. The WKT Raster project started in 2009 and a beta version is currently available. It is an extension of PostGIS that has to be installed separately, although it is planned to merge it into the standard PostGIS module. WKT Raster's goal is to add raster support to PostGIS. Specifically, WKT Raster wants to

> "...implement the RASTER type as much as possible like the GEOMETRY type is implemented in PostGIS and to offer a single set of overlay SQL functions operating seamlessly on vector and raster coverages. " [39]

WKT Raster introduces a new data type to PostGIS, namely the raster type. To store an image, a column of this type is be added to a table. Each row that has a field of type raster represents a so-called tile. This field holds only the raw image data, metadata is stored in the table *raster_columns*. The raster data type is a composite PostgreSQL type, containing attributes needed for georeferencing, information about the bands and the actual image data. All these attributes are accessible by functions.

Table 3.4 shows the structure of the raster data type [39].

Name	Type	Description
version	integer	The version of the WKB format.
number of bands	integer	Number of raster bands.
width	integer	The image width.
height	integer	The image height.
pixel size x	number	The pixel width in the same unit as the SRS.
pixel size y	double	The pixel height in the same unit as the SRS.
upper left x	double	The x coordinate of the upper left pixel's centre.
upper left y	double	The y coordinate of the upper left pixel's centre.
rotation x	double	Rotation along the x-axis.
rotation y	double	Rotation along the y-axis
SRID y	integer	The SRS.
is offline	bit array	This flag specifies whether the band is stored in- or outside of the database.
has nodata value	bit array	Specifies if a band has a nodata values.
pixel type	bit array	The pixel type for each band. It is specified as a string, e.g. 32BF denoting a 32 bit floating point value.
nodata value	array	The nodata value for each band. The actual data type of each entry in the array depends on the pixel type.
values	array	The data values for each band. This column only contains data if the the is offline flag is set to false.
band number	integer	The number of the band stored outside of the database.
path	string	The path to the image file.

Table 3.4: The raster data type structure

The georeferencing attributes pixel size, upper left and rotation correspond to the six parameters stored in a world file.

The information contained in the metadata table raster_columns is depicted in table 3.5:

Name	Type	Description
r_table_catalog	varchar	This column is mostly left blank. It owes its existence to the fact that the it exists in the geometry metadata table as well.
r_table_schema	varchar	This column contains the schema name - the PostgreSQL default value is *public*
r_table_name	varchar	This column contains the name of the table that contains the raster column.
r_column	varchar	The name of the raster column. This, together with the first three columns fully qualifies the raster column within the DBMS.
srid	number	The identifier of the SRS. The column is itself a foreign key, referencing the table SPATIAL_REF_SYS. PostGIS uses EPSG codes.
pixelTypes	varchar array	Together with the nodata values, this column indirectly specifies the number of bands. It is an array containing the pixel type used in each band.
out_db	boolean array	WKT Raster provides the possibility to store data within the database, or as a file in the file system. In the latter case, only a path reference to the file in the file system is stored. This will be explained in greater detail later on. This information is stored separately for every band.
regular_blocking	boolean	In a regularly blocked raster, every cell has the same size, and every row and column has the same number of pixels.
nodata_values	double array	This column contains the values for the pixels containing no data for every band.
pixelsize_x	double	The pixel width in the same unit as the SRS.
pixelsize_y	double	The pixel height in the same unit as the SRS.
blocksize_x	double	The raster width in pixels.
blocksize_y	double	The raster height in pixels.
extent	geometry	The MBR, stored as a polygon geometry.

Table 3.5: The raster_columns table structure

To import raster data, an external utility is currently needed. This utility is *Gdal2WKTRaster*, part of the GDAL package. In future releases, this utility will be fully integrated into WKT Raster, but for now it has to be run from the command line [29]. This importer supports a lot of input file formats: JPEG, PNG, TIFF as well as GeoTIFF and many more[10]. It supports the import of single files as well as of multiple files into a single table. The image data is stored as WKB in the database and no information about the original file format is retained. It is also able to divide a raster file into regions at the time it is imported. This is the same functionality like Oracle Spatial's subsetting operation, the difference being that it is executed when the raster is imported into the database. Another option that can be specified at import time is the creation of an index on the raster's MBR. Some of the metadata needed to fill the raster_columns table has to be specified when importing the raster, e.g the spatial reference system or the width and height if the image is to be divided at import time. Some parts of the metadata are filled in by Gdal2WKTRaster, like the pixel type. Lastly, Gdal2WKTRaster can import just the metadata for a raster image. This means that only the path to the image is stored in the database while the actual data remains in file format on the storage medium.

The option to store an image as a reference only must also be supported by the DBMS. In the case of WKT Raster, this is made possible by setting the out_db flag in the metadata table raster_columns to true before importing the image. This has a number of advantages: the image can be stored in a common file format, and is therefore readily accessible to a lot of GIS and other applications. Another advantage is the simplicity of storing data as a file - no import or export is needed to access the data.

On the other hand, if the file is moved or renamed, then the reference stored in the database is rendered moot. Another point that has to be considered when using references is that the DBMS has to have access privileges to the directory. Databases offer transaction management, which would benefit multi user editing of the image. Finally, as WKT Raster plans to offer raster processing functions in future releases, it would slow down these operations considerably if the image had to be loaded from a file before it could be used.

In future releases, alternative ways to import data are planned as well: the first of these is a function that allows the creation of a raster image from text, called RT_RasterFromText. As input - besides the raw data values for each band - this method takes all the values that have to be filled in the metadata table raster_columns. Additionally, it needs the six parameters specified in a world file to georeference the image. The other method to import data is from a WKB, called RT_RasterFromWKB, which takes the binary data and an identifier for the reference system as input [38].

PostgreSQL provides two indices that are suitable for spatial data: an R-

[10]http://www.gdal.org/formats_list.html

tree index and a Generalised Search Tree (GiST) index. This index divides data into three types: objects that overlap, objects that are inside and objects to one side [37]. Like Oracle Spatial, PostGIS indexes the raster's MBR. The R-tree poses a number of problems in PostgreSQL: it does not support spatial objects that are larger than eight kilobytes - these objects cause the index to fail while it is being built. PostGIS itself largely uses GiST indices.

As described above, the SRS can be specified when the image is imported into the database. The format can be either an EPSG code or a Proj4 string. The transformation information is either provided in a world file, or embedded in the file itself. If this information is missing at import time, then Gdal2WKTRaster assumes default values for the pixel size.

To change these values, WKT Raster provides a number of functions. ST_SetGeoReference sets the transformation parameters in a single statement, other functions allow the setting of single parameters, e.g. the pixel size. It is also possible to change the SRS. To reproject an image, currently external utilities from the GDAL package have to be used.

To export raster images, or subsections thereof, as of now, utilities from the GDAL package have to be used as well. GdalTranslate converts from one raster format to another. It is able to read WKT Raster from a PostgreSQL database, and save the data in a raster file format. Alternatively, GdalWarp can be used. This tool converts between raster formats as well, and is additionally able to reproject images. Future releases of WKT Raster will include special database functions to export raster data directly into widely used file formats. Some of the functions that are planned are ST_AsJPEG, ST_AsPNG and ST_AsTIFF. All these functions allow the export of one or more bands of data. These functions not only take raster as an input, but geometries as well, thus allowing the rasterisation of vector data. Other planned export functions provide support for formats such as KML and Scalable Vector Graphics (SVG) [39].

In contrast to Oracle Spatial, WKT Raster does not provide - and will not in future releases - functionality to generate and maintain image pyramids [39]. Reduced resolution versions of an image have to be created with external tools and stored as separate images in the database. Another function that is - as of now - not provided is a function to divide the image into regions, like the subsetting function in Oracle. If an image needs to be tiled, it has to be done at import time, with the functionality provided by Gdal2WKTRaster.

As mentioned at the beginning of this section, the aim of WKT Raster is to provide functionality that allows to perform operations on both raster and vector data. Some of these functions have already been implemented in the currently available beta version, like converting a raster to polygons based on pixel values. Further raster processing and manipulation functions will be available in the upcoming version, e.g. creating the union of two rasters, or a raster and a geometry.

Oracle Spatial vs. WKT Raster

Oracle Spatial and WKT Raster are only comparable up to a point, as they model spatial raster data differently and follow different concepts. Oracle distinguishes between the image as a spatial object, represented by the SDO_GEORASTER class, and the associated image data, stored in a SDO_RASTER object. The first object holds spatial metadata as well as data concerning the image a whole, whereas the latter holds the data itself, and attributes that concern an image block. The metadata attribute of SDO_GEORASTER can hold both spatial and image processing information for any needs that may arise.

WKT Raster views the image as one object that should be able to stand on its own, and therefore stores it in a single column of type raster. This composite data type provides all the necessary information to perform basic GIS raster operations. With the metadata table raster_columns there exists the possibility to register an image column and assorted metadata, but this is not mandatory. To store other attributes, WKT Raster relies on the intrinsic functionality of an object-relational DBMS.

Another difference is that WKT Raster aims to provide functionality that allows the seamless editing and processing of vector and raster data, whereas Oracle provides functions that edit either raster or vector data. WKT Raster provides methods to convert from raster to vector data and vice versa, Oracle does not.

What complicates this comparison further is the fact that WKT Raster is only in its beginning stages, with lots of functionality planned but not implemented yet [39]. It remains to be seen if WKT Raster can gain acceptance.

3.4 Serving Tiles

Once the tiles are cut and stored persistently, they can serve their purpose: to be reassembled into a complete map by a client application and displayed. To do this, the client has to know the tiles' properties. To know which tiles are adjacent, the addressing system has to be known. The addressing system manifests itself in the file name, or if the tiles are stored in a database, in an attribute stored with the image data. But it is not just the tile name that can be important: the directory structure, zoom level, spatial extent, pixel size and image format are all properties that are vital to know for a client application. The values these properties take on, and how they are published for clients to access, depend on the tiling scheme.

There are different tile addressing systems in use - some are proprietary, and the tiles are only accessible through APIs provided by the companies, e.g Google Maps and Bing Maps. This saves the clients accessing the tiles from finding out about the aforementioned properties themselves, as the tiles are accessed and assembled via API functions.

These services and their APIs can also be used to serve one's own tiles as an overlay. But to bypass these proprietary services and the terms of use attached to them, other systems have to be used. There are two open standards that describe tiled mapping services: the OSGeo's TMS specification, and based on it the OGC's WMTS Implementation Standard.

Both these specifications describe a tiling scheme. They cover every aspect of serving tiles via standardized interfaces and describe the tiles themselves, their spatial extent and other necessary properties.

3.4.1 Web Map Tiling Specification

WMTS specifies methods and techniques for services that aim to provide maps based on prerendered tiles stored as image files. It is the most comprehensive open standard dedicated to tiled map services. The standard is divided into three parts: the Service Metadata document, which is mandatory in all OGC standards, the Tile and associated requests and operations, and the Feature Information, which is optional. The base for these data structures and documents, requests and responses is the OGC Web Services Common Standard. WMTS adds to the already defined data structures where necessary.

WMTS is designed to satisfy the requirements of both the resource oriented and the procedure oriented software architectural styles [31]. To this end, valid request and response methods are defined for three representatives of the procedure oriented architecture: Key Value Pair (KVP), simple XML and XML contained in Simple Object Access Protocol (SOAP) envelopes. This software design approach focuses on exposing procedures to be used by client applications. A client requests a procedure to be executed, and gets an object containing the requested data in return. In this case, procedures are e.g. the *GetCapabilities* request, which returns a document containing metadata in XML format.

The resource oriented approach is represented by the Representational State Transfer (REST) architecture. It follows the classic client-server architecture: a client sends a request to a server, which processes the request and returns a response. The resources are usually documents, identified by a Uniform Resource Locator (URL). The document defines valid requests and Hypertext Transfer Protocol (HTTP) based URL patterns that allow clients to access metadata and tiles as resources identified by URLs.

Service Metadata

The GetCapabilities method used in procedure oriented architecture returns the Service Metadata document, encoded in XML format. In the REST approach, this document is made available via a URL. For a client application planning to use the service, this information is vital. The capabilities described in the document can be grouped into five categories:

Service Provider: This section contains metadata about the organisation administrating the service, e.g. name, address and other contact information.

Service Identification: General information about the service, e.g. the service version, title and abstract.

Operations Metadata: Information about the operations provided by the service, e.g. method names and their parameters. If the service provides only a REST implementation, this section is left blank.

Contents: This section describes the actual content the service offers. It is explained later in greater detail.

Themes: This part is optional; it contains descriptions of the themes provided by the layer(s). The themes are independent of the layers section, thus allowing multiple thematic descriptions.

The metadata category that gives information about the tiles themselves and their properties is the Contents section. It introduces the basic terms and concepts that create the hierarchy used in the WMTS specification to order and address tiles and their properties: layers, tile matrix sets and tile matrices.

A layer, in this context, is the top level data structure. It links to one or more tile matrix sets and gives important information about these tile sets: the available image formats, the MBR that contains all tile matrix sets in this layer, as well as a link to the tile matrix sets themselves. The link refers to the identifier given to the tile matrix set in the corresponding part of the contents section. The tile matrix set link is mandatory.

If the service provides a REST interface, the resource URL has to be specified in the layer section. This element provides information about the image format, the resource type – which is currently restricted to tile – and a template for the URL the tiles are available from. This template is divided into the base address, followed by variables representing the style, tile matrix set and tile matrix identifiers and tile row and column indices. Below is an example for a resource URL pattern:

```
http://www.wmts.com/{Style}/{TileMatrixSet}/{TileMatrix}/{Row}/{Column}.png
```

There are two ways to define the layer's bounding box: the first is by specifying a bounding box where in addition to the bounds the SRS has to be stated. The second is by specifying the WGS84 bounding box, which takes WGS84 as the given coordinate system.

There can be one or more layer(s) in the contents section, each layer being uniquely identified by a code. A layer can be described by using one or more of the descriptive fields: title, abstract and keywords as well as a field for additional metadata.

An integral part of the Contents section is the Tile Matrix Set. This data structure functions as a container for one or more tile matrices that share certain properties, e.g. the spatial extent they cover and the SRS. In this container, the common properties for the tile matrices are specified.

Table 3.6 gives an overview of the fields contained in the tile matrix set data structure.

Name	Type	Description
Identifier	codeType	The data type of this field is defined in the OGC Common Web Service Standard, which in turn is based on an ISO standard [30]. It contains a unique identifier for the tile matrix set.
Title	text	This field contains a short description of the tile matrix set's contents. There are additional fields available to provide more detailed descriptions of the contents, namely abstract and keywords.
Supported CRS	Uniform Resource Identifier (URI) type	This field is mandatory, and must contain a reference to a valid coordinate system.
Well-known scale sets	URI type	This field can contain a reference to a scale set that is used in other tiled map services, thus ensuring that this tile matrix set is an adequate overlay. The concept of well known scale sets will be described in greater detail later on.
Bounding box	Bounding Box type	This field specifies the MBR in the defined coordinate system.
Tile Matrix	Tile Matrix type	This describes the tile matrix. Refer to table 3.7 for more details.

Table 3.6: The Tile Matrix Set data structure

The property that allows a service to be used in conjunction with other services - be they commercial like Google Maps or WMTS compliant providers - is the well-known scale sets property. The scale sets allow the service to offer tile sets that are compatible to the most popular tile based mapping services in existence today.

In fact, a well-known scale set consists of two components: a coordinate reference system and the scale set itself. The scale set consists of a number of predefined scales. A tile matrix set can support exactly one well-known scale set. The scale set's parameters can be found at the URI the property specifies. In order to fully support such a scale set, the tile matrix set must use the same coordinate system and offer tile matrices ranging from the largest scale in the set to a lower scale. It is not necessary for the tile matrix set to offer tile matrices at all the scales defined in the set.

WMTS defines some well-known scale sets specifically to be compatible with Google Maps and Bing Maps. The *GoogleCRS84Quad* well-known scale set is compatible with the satellite imagery supplied by Google Maps, and the *GoogleMapsCompatible* scale set can be used for tile matrices that are compatible with both Google and Bing Maps [31].

The tile matrix data structure describes the tile sets and the actual tiles that are available through the service. Table 3.7 explains the components.

Name	Type	Description
Identifier	codeType	This field contains a unique identifier for the tile matrix.
Title	text	This field is used to describe the contents of the tile matrix. Other descriptive fields are abstract and keywords.
Scale denominator	number	This number represents the scale of all the tiles in the matrix.
Top left corner	GML point	This specifies the top left corner of the tile set using the coordinates and coordinate ordering defined in the SRS specified in the tile matrix set. It represents the minimum x and the maximum y coordinate of the bounding box of the tile matrix.
Tile width	number	The width of a tile in pixels for this tile matrix.
Tile height	number	The height of a tile in pixels for this tile matrix.
Matrix width	number	The number of tiles that make up a matrix row.
Matrix height	number	The number of tiles that make up a matrix column.

Table 3.7: The Tile Matrix data structure

Each tile matrix in a set is optimized for a particular zoom level. The identifier is unique for each tile matrix in each set, and the specification recommends to use a name that indicates the scale or resolution for this tile matrix. The tile matrix space is completely defined by the properties stated in the data structure. It also allows the addressing of single tiles, which is vital for a client. Every tile can be addressed by its row and column index. Properties such as the tile matrix' bounding box, conversion between pixel coordinates and coordinates in the system used can be done with the given parameters.

Figure 3.5 illustrates the tile geometry as defined in the WMTS specification.

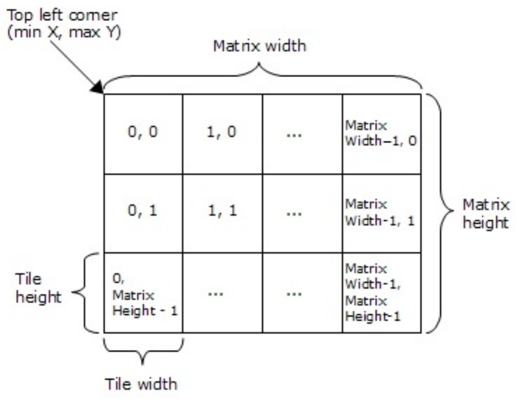


Figure 3.5: The WMTS tile space (adapted from [31])

A tile matrix set section of a service metadata document for a service offering tiles covering Europe at two zoom levels could therefore look like this:

```
<TileMatrixSet>
  <Identifier>Europe_WGS84</Identifier>
  <SupportedCRS>urn:ogc:def:crs:OGC:1.3:CRS84</SupportedCRS>
  <WellKnownScaleSet>urn:ogc:def:wkss:OGC:1.0:GoogleCRS84Quad
  </WellKnownScaleSet>
  <TileMatrix>
    <Identifier>EU_Zoom_1</Identifier>
    <ScaleDenominator>795139219.9519541</ScaleDenominator>
    <TopLeftCorner>70 -30</TopLeftCorner>
    <TileWidth>256</TileWidth>
    <TileHeight>256</TileHeight>
    <MatrixWidth>4</MatrixWidth>
    <MatrixHeight>4</MatrixHeight>
  </TileMatrix>
  <TileMatrix>
    <Identifier>EU_Zoom_2</Identifier>
    <ScaleDenominator>397569609.9759771</ScaleDenominator>
    <TopLeftCorner>70 -30</TopLeftCorner>
    <TileWidth>256</TileWidth>
    <TileHeight>256</TileHeight>
    <MatrixWidth>16</MatrixWidth>
    <MatrixHeight>16</MatrixHeight>
  </TileMatrix>
</TileMatrixSet>
```

The service metadata document provides a client application with all the information necessary to request tiles. The request is done with a *GetTile* operation, or, in the case of a REST approach with a URL pointing to the tile in question.

Tile Request

The WMTS specification views a tile as a resource that comes in the form of an image file containing cartographic content. The response to a tile request is always one complete tile image. The tile size and available image format(s) are specified in the service metadata document.

Below is a valid tile request using XML encoding:

```
<GetTile service="WMTS" version="1.0.0"
         xmlns="http://www.opengis.net/wmts/1.0">
 <Layer>Europe</Layer>
 <Style>default</Style>
 <Format>image/png</Format>
 <TileMatrixSet>Europe_WGS84</TileMatrixSet>
 <TileMatrix>EU_Zoom_1</TileMatrix>
 <TileRow>1</TileRow>
 <TileCol>1</TileCol>
</GetTile>
```

The GetTile request used in KVP and simple XML as well as SOAP interfaces needs to specify certain parameters in order to receive a tile. For one, it has to specify the complete hierarchy employed by the specification, which consists of the layer, the tile matrix set and tile matrix identifiers.Another mandatory parameter is the style. To finally reference the desired tile, the tile column and row index are used. Further mandatory parameters include output image format.

In the case of KVP encoding, the response comes in the form of a binary image in the specified format streamed via HTTP. In the case of XML encoding, the raw image data is encoded in an XML element, and has to be assembled into an image by the client application.

A service that uses REST, which represents the resource oriented approach, returns no binary data to the client. Instead, the client can construct a valid URL that points to the desired tile. The base for the response URL is the template attribute of the resource URL element in the layers section. This template defines the URL pattern: starting with the domain name and possibly subdomain names, the URL is built of the style, tile matrix set and tile matrix identifiers as well as tile row and tile column. The tile column represents the actual file name. The specification defines no particular order for the variables, but the one described here is recommended. To construct the URL, the client has to replace the variables with values taken from the service metadata document.

To get the tile described in the XML encoding example from a hypothetical service located at www.wmts.com, the following URL has to be constructed by the client application:

```
http://www.wmts.com/default/Europe/Europe_WGS84/EU_Zoom_1/1/1.png
```

The image located at this URL represents a part of a tiled map, with the size, bounding box and scale specified in the service metadata document.

Feature Info

The Feature information section of the WMTS specification describes the object or objects located at a specific pixel on a specific tile. To request this information, a *GetFeatureInfo* request is made in procedure oriented architecture, whereas the RESTful approach provides a so-called feature resource.

The feature information document may have any format, but the specification recommends the use of the GML Simple Features Profile. Other formats that can be used are simple XML or HTML. The presence of additional information for a certain layer is made known in the layer section of the service metadata document. If the infoFormat element is defined for a layer, then it can be considered *queryable*, i.e. the GetFeatureInfo request returns data.

To execute a feature info request, the original tile request is embedded into the feature info request. The feature info request itself consists of two parameters to mark the pixel position, J for the row and I for the column as well as the format of the response. The feature information response contains the additional information for the specified pixel: this can be any information considered useful, e.g. elevation, temperature measurements or other information that can be found in attributes for spatial data.

3.4.2 Tile Map Service Specification

Unlike WMTS, the TMS specification is not an official standard. According to the OSGeo web site it is intended as a guideline for the implementation of client-server mapping software that uses multi-resolution image pyramids. It aims to standardise the way tiles are requested by clients as well as the way a server describes the tiles it is offering [35].

In many ways it resembles a downsized and simpler version of the OGC's effort. For example, it does not provide SOAP or KVP interfaces, but only a REST interface and there is no equivalent of the Feature Information. So, only the first two parts of the WMTS specification are matched: the service metadata is composed of the root and tile map service resources, and the tile request is known as the tile map resource.

Adhering to the REST approach, all these resources are accessed via URLs. The base URL is given in the root resource and subsequent requests deliver more and more specific metadata about the tiled map service, until – at the last step – a tile image is returned. Figure 3.6 illustrates the steps that are necessary in order to acquire a tile. The URLs used in the image refer to the ones used in the following examples.

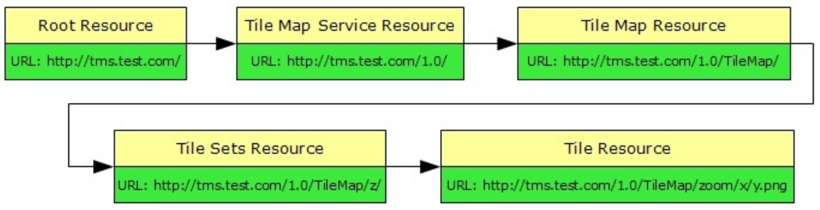

Root Resource
URL: http://tms.test.com/
Tile Map Service Resource
URL: http://tms.test.com/1.0/
Tile Map Resource
URL: http://tms.test.com/1.0/TileMap/
Tile Sets Resource
URL: http://tms.test.com/1.0/TileMap/z/
Tile Resource
URL: http://tms.test.com/1.0/TileMap/zoom/x/y.png

Figure 3.6: TMS request chain

Root Resource

This resource, located at the very root of the site offering the service, simply describes the different versions a service is available in and specifies their exact URLs. It has to be available as an XML document directly at the service's root. The document takes on the following form:

```
 <Services>
  <TileMapService title="TMS_v1" version="1.0" href="http://tms.test.com/1.0/" />
  <TileMapService title="TMS_v2" version="2.0" href="http://tms.test.com/2.0/" />
</Services>
```

The different versions of the tile map service listed in the root resource are described in greater detail in the next resource in the chain, the tile map service resource.

Tile Map Service Resource

At the address given in the *href* attribute of the root resource, the metadata describing the service itself – the tile service resource document – is located. This resource is divided into three sections: mandatory and optional descriptive information as well as the *tile maps* section.

The descriptive information consists of a mandatory service title and abstract, and an optional part that contains contact information about the service provider and keywords describing the service.

The tile maps section contains one or more *tile map* elements. These contain information about the actual tiled maps the service offers. The WMTS equivalent to this is the tile matrix set, which in turn contains the tile matrices. The information contained in a tile map element is basic: a title for the map, the SRS, the tile map URL and a *profile*. The SRS attribute uses EPSG codes almost exclusively. If a reference system is not available in the EPSG database, then it is possible to define it within the resource document using the WKT format.

A valid tile map service element may look like this:

```
<TileMapService version="1.0" services="http://tms.test.com/1.0/">
   <Title>TMS_v1</Title>
   <Abstract>Long description of the service</Abstract>
   <TileMaps>
     <TileMap
       title="North America"
       srs="EPSG:4326"
       profile="global-geodetic"
       href="http://tms.test.com/1.0/NA" />
     <TileMap
       title="Europe"
       srs="EPSG:4326"
       profile="global-geodetic"
       href="http://tms.test.com/1.0/EU" />
   </TileMaps>
 </TileMapService>
```

The *profile* attribute is the equivalent of the well-known scale set, trying to provide a way to facilitate interoperability between tiled map services by predefining a fixed set of scales. The profile is an attribute of the tile map service element, and it reoccurs as an attribute of the tile sets element. This provides the possibility to redefine the profiles for a single tile set of a tile map. Profiles will be explained there in greater detail.

Tile Map Resource

A *tile map* is, for all intents and purposes, a complete representation of a spatial extent, i.e. the whole area is covered by tiles and no blank areas or no data values remain. Like the tile map service resource, the tile map resource defines mandatory and optional descriptive elements. This resource once more defines the SRS, which must not deviate from the one given in the tile map service resource.

The spatial extent of the tile map resource is defined in the *bounding box* element using the units the reference system specifies. The *origin* element describes the real world coordinates of the lower left corner of the tile with the tile coordinates (0,0). The tile geometry differs from the one used in the WMTS specification inasmuch as tiles with negative coordinates are allowed. Figure 3.7 shows the tile geometry as defined in the TMS specification.

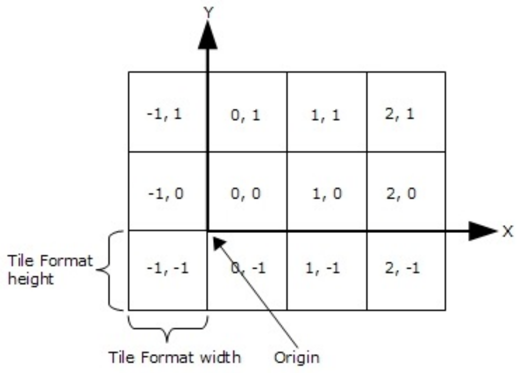


Figure 3.7: The TMS tile geometry

The *tile format width* and *tile format height* shown in figure 3.7 are defined as attributes of the *tile format* element. This element has two other important attributes: the mime type and the extension. The mime type, also known as the content type refers to the Multi Purpose Internet Mail Extension (MIME) type, which commonly identifies file formats in web browsers. It consists of two parts: the first part refers to the general purpose of the content, e.g. audio, video or text, whereas the second part specifies the exact file format, e.g. PNG or JPEG. In the case of the mime type attribute used here, the first part has to be *image*, and the second part specifies the image file format. On the face of it it may seem redundant to use two attributes, but when considering the JPEG file format and its multitude of possible extensions, this two-part definition appears sensible.

The last element contained in the tile map resource is the *tile sets* element. The *tile set* element(s) contains the base URL of the actual tiles, defined in the *href* attribute. The tile set element contains two more attributes: *units-per-pixel*, which describes the relation of map unit to pixel for this particular tile set, and *order*, which specifies the pyramid level of this tile set.

A valid tile map resource document looks like this:

```
<TileMap version="1.0" tilemapservice="http://http://tms.test.com/1.0">
 <Title>North America</Title>
 <Abstract>Longer decription of the Tile Map</Abstract>
 <SRS>EPSG:4326</SRS>
 <BoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
 <Origin x="-180" y="-90" />
 <TileFormat width="256" height="256"
             mime-type="image/jpeg"
             extension="jpg" />
 <TileSets profile=global-geodetic">
  <TileSet href="http://tms.test.com/1.0/NA/0"
           units-per-pixel="0.9"
           order="0" />
  <TileSet href="http://tms.test.com/1.0/NA/1"
           units-per-pixel="0.6"
           order="1" />
 </TileSets>
</TileMap>
```

To access a single tile, the URL provided by the href attribute in the tile set element is suffixed with the X coordinate of the desired tile, thus providing the complete directory name. The image file name is assembled using the Y coordinate as the file name and adding the extension specified in the tile format element. A valid tile image URL, using the example above could look like this: `http://tms.test.com1.0/NA/0/0.jpg`

The tile sets element itself has only one attribute, namely *profile*. There are four possible profiles:

Global-geodetic: Tile sets which support this profile can only have EPSG:4326 as an SRS. The units-per-pixel attribute must be calculated using the following formula:

$$unitsPerPixel = \frac{0.703125}{2^z} \tag{3.2}$$

z denotes the zoom level, which must be zero or greater. The reason for this is that at the initial zoom level there are two tiles of 256×256 pixel covering the earth. The origin of a tile set implementing this profile is always at 90°S 180°W. The reason for this becomes apparent when looking at figure 3.7.

Global-mercator: The mandatory SRS of this profile consists of a simple Mercator projection and a WGS84 datum. This is not defined in the EPSG database, so the OSGeo defined it on its own authority, giving it the code *OSGEO:41001* [35]. It is similar to the one used in the global-geodetic profile, but uses e.g meters instead of decimal degrees as map units. The units-per-pixel attribute uses a different formula from the global-geodetic profile:

$$unitsPerPixel = \frac{78271.516}{2^z} \tag{3.3}$$

This produces a map that, at the initial zoom level, has four tiles at a tile size of 256×256 pixels. The origin is at (-20037508.34, -20037508.34).

Local: Any valid SRS can be used in this profile. The formula for the calculation of the units per pixel is simple in this profile: 2^z. Again, z denotes the zoom level, which must equal or be greater than zero. Another requirement for this profile is that the directories under which the tiles for a specific zoom level are located must be named after that zoom level, i.e at the zoom level 9, the tile set's href attribute must be `http://tms.test.org/1.0/NA/9`.

None: If a tile set does not implement a profile, the value of the attribute is set to *none*, rather than omitting the attribute altogether.

3.4.3 Proprietary Systems

Most commercial tile based map services provide APIs to access their tiles. This eliminates the need to request information about layers and tiles, request single tiles and assemble them into seamless maps, as it can be done with functions provided by the API. Consequently it is not strictly necessary to know the tile addressing system in use to create a map. Some providers offer a description of the tile addressing scheme as part of the official API documentation. Microsoft offers a detailed explanation of the tile system employed by Bing Maps [44]. Their tile naming system is explained in section 2.2.3. Google have a short description in the API documentation [15].

The APIs can be used to overlay custom made tiles over the basemap provided by the service. Mostly, this is a simple process, but it entails accepting the technical restrictions and legal conditions dictated by the service provider. The services do not offer additional spatial information about their tiles, like the feature information part of WMTS. Neither do they offer spatial metadata as it is contained in the service metadata document.

Nevertheless is it possible to access the tiles via URLs, although this is often explicitly forbidden by the terms of use of these services, e.g. Google requires that the API is used to use its base data [16].

Chapter 4

Applications

Tessellation and tiling have many applications in geoinformatics. Irregular tessellation is a regular feature in GIS software, primarily for surface modelling. TINs and Voronoi diagrams are the best known of these tessellations.

A multitude of regular tessellation methods are involved in DGGs, which utilize both a quadtree data structure and grids, as well as employing one of the many incarnations of location codes to address tiles. Location codes themselves bridge the gap between traditional GIS software and web map services using tiled maps.

Tiled web map services are the most widespread and well known application of tiled mapping. Tiles also provide the base for mobile mapping applications most recently used in Location-based Services (LBSs).

4.1 Tessellation in Geoinformatics

4.1.1 Discrete Global (Geodesic) Grids

The DGG is a data structure that uses both grids and quadtrees.

> "A Discrete Global Grid consists of a set of regions that form a partition of the earth's surface, where each region has a single point contained in the region associated with it. Each region/-point combination is a cell." [41]

DGGs are used to reference high resolution datasets that commonly used grids - such as the latitude/longitude system - cannot cover accurately enough.

The cells can have irregular shapes, such as the division of the earth's surface into land masses and water bodies, or regular shapes with evenly distributed points. The regularly shaped regions are created using regular tessellations of a sphere. A DGG is the basic building block of a Discrete Global Grid System (DGGS). A DGGS consists of a sequence of DGGs draped over the sphere at increasingly finer grid width, and thus, resolution.

There are several ways to construct a DGG. One is to use a grid based on the latitude/longitude system. This has several advantages: the system is well known and widely used, both by datasets and GIS software; most users are familiar with it and it is easy to implement and display. It also has some serious limitations. For one, it does not produce cells of equal area. The cells become more distorted the farther north or south one moves on the grid. At the poles, the grid cells even turn from rectangles into triangles. These singularities complicate things for an application using a DGGS, forcing the use of specially adapted grid systems for the polar regions.

Another construction method for a DGGs is tiling. In this case, the term tiling describes the application of a square grid to a sphere that is already projected on a plane. The grids based on this method suffers from some of the same problems map projections do: there is always some kind of distortion imminent in the process of projecting a spherical surface onto a plane. Accordingly, the grids suffer from the same distortion the map projection does.

The final method to construct a DGG is by embedding a polyhedron in a sphere and hierarchically partitioning the faces of the polyhedron using a decomposition method such as the quadtree. The resulting grid systems are known as Geodesic DGGs, because they are inspired by the work of the architect R. Buckminster Fuller and the geodesic dome [41].

For the inscribed polyhedron, only five possibilities exist. These are the so-called *platonic solids*. The bodies - also known as regular solids - are regular polyhedra whose faces consist of congruent regular polygons [48]. Figure 4.1 shows the five platonic solids, from left to right: the tetrahedron, cube, octahedron, dodecahedron and icosahedron.

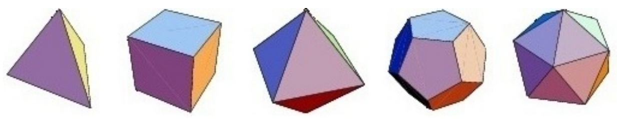


Figure 4.1: The five platonic solids (adapted from [48])

According to [41], it is highly unlikely that one Geodesic DGG is ideal for all applications. Therefore, when constructing a DGG using this method, a number of design choices have to be made with the purpose of the grid in mind. These are:

- The platonic solid to be embedded in the sphere
- The orientation of the solid in relation to the sphere
- The spatial decomposition method and prototile used to tessellate the faces of the solid
- The conversion method between the spherical face and the prototile

- A technique to allocate points to each grid cell

A desirable property of the inscribed solid is that it has small faces, as they reduce distortion. This property is best satisfied by the icosahedron, which therefore has been widely used when constructing a Geodesic DGG. The cube along with the tetrahedron has the largest faces of the solids and is therefore relatively unsuited as base polyhedron. Nevertheless, the cube has been used as a base solid because its square faces are easy to partition using a standard quadtree data structure.

The orientation depends strongly on the use and purpose of the grid. The most commonly used solids are the icosahedron, dodecahedron and octahedron [41]. In case of the icosahedron, a possible orientation is one that places a vertex at each pole and aligns an edge emitted from the north pole with the prime meridian.

The four possible methods of spatial decomposition are tiling by squares, triangles, diamonds and hexagons. Squares are an obvious choice, because the standard region quadtree uses squares, and a host of algorithms and implementations thereof already exist and are readily available. A special form of a square is the diamond, which can be formed by joining two adjacent triangles.

Many of these algorithms are easily transferable to quadtrees using a triangle to tessellate. Triangles are an obvious choice when using an icosahedron or octahedron as base solid, as these polyhedra have themselves triangular faces.

The last tessellation figure that can be used in Geodesic DGGSs is the hexagon. There are a number of problems attached to tessellating a sphere with hexagons. Firstly, it is impossible to completely tile a sphere with hexagons [41]. There will always be artefacts occurring around all vertices of the base polyhedron. The shape of the artefact depends on the base solid. If the inscribed polyhedron is an octahedron, the artefact is a square; if it is an icosahedron, the artefact is a pentagon. The second problem arises from the fact that it is not possible to divide a hexagon into smaller hexagons, or create a hexagon by combining smaller hexagons. The figure that results from aggregating hexagons can best be described as an almost-hexagon, a figure which is made up by seven hexagons. This entails that it is easy to use hexagons for one resolution grids, but not for DGGSs. Despite this, hexagons, or structures based on tessellation by hexagons, are used in DGGSs. The almost-hexagon consisting of seven real hexagons - a structure also known as the Generalized Balanced Ternary.

Conversion methods can be split into two categories: direct spherical subdivision creates the subdivision on the part of the sphere that maps to the corresponding part of the planar face of the base solid. The map projection method uses an inverse map projection to map the planar face partition to the sphere.

The point allocation method that is best suited depends on the conversion method, but as a rule of thumb the centre points of the planar cell can be used.

A fairly well known example for a Geodesic DGGS using triangular spatial decomposition is the QTM. The QTM describes a grid scheme for the earth based on a triangular mesh, thus making it a regular spherical tesselation. The mesh is built up to the desired resolution level using quadtrees. The quadtree used by the QTM uses equilateral triangles to partition space.

To generate the mesh, an octahedron is embedded in the sphere so that two of its vertices are tangent to the poles and the rest touch the equator at 0°, 90°, 180° and 270° longitude. Figure 4.2 shows the initial octahedron embedded in the globe. To refine the mesh further, the midpoint of every edge is raised to the surface, thus creating four new triangles for every face of the octahedron. This process is repeated until the desired level is reached [6].

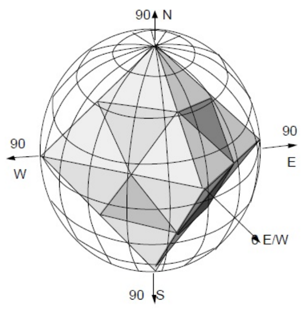


Figure 4.2: An embedded octahedron - the basis for a QTM [6]

To create the corresponding quadtree the northwest to southeast ordering used in the other quadtrees is replaced by a numbering system. Every triangle is further divided into four parts labelled 0, 1, 2 and 3. At the next quad level, the number denoting the higher level is retained. This leads to longer labels for triangles situated on a deeper tree level. These quadcodes are discussed in greater detail in the next section.

4.1.2 Quadcodes

Quadcodes are a practical application of tiles as a secondary data structure. They are used to address tiles which in turn are a primary data structure.

It is possible to convert quadcodes to and from geographic latitude/longitude coordinates. How this is done, and how complex a method has to be employed, depends on the labelling scheme which in turn depends on the underlying spatial decomposition method. To convert between geographic coordinates and quadcodes based on a square spatial decomposition is a straightforward matter. If the spatial partitioning is based on triangles - like most DGGSs - then the naming scheme, and with it the conversion methods change, generally becoming more complex.

QTM Addresses

This addressing system is an example for quadcodes based on triangular decomposition. The naming scheme proposed by Dutton in [5] is explained in greater detail.

Dutton proposes the QTM as a geodesic DGG. The base solid is an octahedron, oriented so that two of its vertices are tangent to the poles, and the remaining four touch the equator at 0°, 90°, 180° and 270°. The octahedron has eight triangular faces, which provide the root for the eight triangular quadtrees decomposing the sphere. The alignment of the embedded solid to the latitude/longitude grid is chosen because it makes it easier to convert to and from geographic coordinates.

The naming, or in this case rather the numbering scheme, assigns an *octant number* to each triangular face of the inscribed octahedron. The numbering starts with octant 1 in the northern hemisphere at the prime meridian, progressing clockwise around the globe and switching to the southern hemisphere at the prime meridian again, ending with octant number 8. This octant number is prefixed to each quadcode, thus providing the unique identifier to keep the eight quadtrees apart.

Each octant is decomposed into four isosceles right triangles. The shape of the triangles is a property of the projection method, the Zenithial OrthoTriangular (ZOT) projection and plays an important part in the calculation of the quadcodes [5]. The ZOT projection unravels the octahedron and stretches its triangular faces so that they fit in a square - thus changing the octahedron's original equilateral triangles to right isosceles triangles. This fact applies to every triangle at every level of the quadtree. The centre of the projection is the north pole, the south pole is located at all four corners of the square - this is a consequence of the fact that all the faces of the southern hemisphere are cut from the pole to the equator and are mirrored across the central axes [4].

The triangles are numbered according to the quadcode scheme described above, the only difference being the prefixed octant number. The codeword is a quaternary code, consisting of one of the symbols in the set $0, 1, 2, 3$. The centre triangle is always given the number 0, the other triangles are numbered according to the adjacent vertex of the parent triangle.

Figure 4.3 shows the numbering of three levels of triangles, assuming that the parent triangle has code 0 and lies in octant number 1. In the world of the Octahedral QTM, the node labels are called *QTM identifiers* or *quaternary addresses.*

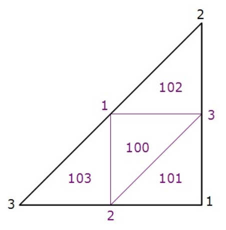


Figure 4.3: Construction of quaternary addresses

Bing Maps Quadcodes

A practical application of quadcodes in the world of tiled web mapping is the Bing Maps Service[1]. In this tiled web map service, the earth is projected using a Mercator projection. The Mercator projection preserves shape, but distorts area and scale. These downsides are compensated by the fact that - almost - the whole earth fits on a perfect square. This ensures that north is always on the upper side and south on the lower side, as well as east being on the right hand side and west on the left hand side. The only areas that are not shown on the map are the poles. This is because the Mercator projection goes to infinity at both poles - thus the latitudes shown on the map range from approximately 85° north to 85° south [44].

This projected map is the starting point for the use of a square tessellation of the whole planet. It uses a standard region quadtree with variable resolution levels. Quadcodes are given yet another name - quadkey, an abbreviation of quadtree key. The maximum depth, and thus quadcode length, available is twenty-three. This does not mean that the whole earth is available in twenty-three resolution levels. Rather it means that the tessellation has multiple resolution levels, according to the level of accuracy needed to display the data contained in certain regions. At a resolution level of twenty-three, the ground resolution would be 0.0187 meters per pixel at the equator [44] - a level of detail that is unnecessary for a web mapping service. The use of variable resolution levels exploits one of the main advantages of the quadtree data structure. The quadcode is constructed from the set containing the numbers $0, 1, 2, 3$ in the order depicted in figure 2.10.

This system makes it fairly easy to convert between quadcodes and geographic coordinates. The datum used is WGS84. Besides quadcodes and

[1]http://www.bing.com/maps/

geographic coordinates, tile and pixel coordinates play an important part in the calculations involved in the conversion methods.

Each tile is not only identified by a quadcode, but by (X,Y) coordinates, called tile coordinates, as well. The origin of the Cartesian coordinate system lies in the upper left corner of the map - analogous to the coordinate origin in raster images. At each level of subdivision, the number of tiles quadruples and the grid that covers the map becomes successively finer.

Pixel coordinates have a number of properties in common with tile coordinates: the number of pixel coordinates changes with every resolution level and the origin of the pixel coordinate system is situated at the upper left corner. At the lowest resolution level, the map image is 256×256 pixels in size. This number rises with every resolution level and can be calculated for a specific resolution level using the following formula: 256×2^z, with z being the resolution level.

There are two ways to calculate the quadcode for a specific resolution level from geographic coordinates. One way is to calculate the tile coordinates first, the other way takes the additional step of calculating the pixel coordinates first, then deriving the tile coordinates from them.

As this map uses a Mercator projection, this is done using the formula for the spherical version of the Mercator projection. The map's sole use is for display on a screen, which makes the fact that this causes additional distortion negligible. To calculate the X and Y tile coordinates from a longitude λ and a latitude ϕ, the following formulas are used [44]:

$$X = \lfloor \frac{(\lambda + 180)}{360} \times 2^z \rfloor \tag{4.1}$$

$$Y = \lfloor \frac{0.5 - \log(\frac{1+\sin\phi}{1-\sin\phi})}{4\pi \times 2^z} \rfloor \tag{4.2}$$

Again, z denotes the resolution level. If another projection was used, these formulas would have to be changed to accommodate this projection. If the pixel coordinate needs to be calculated, equations 4.1 and 4.2 need to be multiplied by 256.

To give an example, at resolution level 3, the geographic coordinate 47°N 11°E translates into the tile coordinates (4,2) and the pixel coordinates (1024, 512).

To derive the quadcode from the tile coordinates, at first the bits of the binary representations of the tile coordinates are interleaved. The result is then interpreted as a quaternary number. For the example latitude/longitude of 47°N 11°E, the quadkey would be calculated as follows: the binary representation of the tile coordinate (4,2) is (100,010). If the digits (bits) of the two binary numbers are interleaved – starting with the Y coordinate and including leading zeros – the result is 011000. Conversion to a quaternary number results in 120 as the quadkey for 47°N 11°E at resolution level 3.

In the Bing maps system, these quadkeys are used as filenames for the individual tiles, which are stored in the PNG file format.

The Bing maps system makes use of many concepts mentioned so far: it uses regular square tessellation, a linear quadtree data structure, bit interleaving to create unique identifiers, which in turn is an application of Morton space ordering and the concept of tile pyramids. Thus it combines many forms of tessellation.

4.1.3 Irregular tessellations

Regular tessellations play a prominent role in geoinformatics. They provide the base for the raster data model and secondary data structures. Irregular tessellations are not as common, nevertheless they are used as data structures for DTMs and as secondary data structures, e.g the region quadtree [27]. According to Worboys and Duckham a tessellation is irregular if the polygons that form the tessellation are not regular and equal [53]. Some irregular tessellations use similar polygons, such as triangles or squares, while some use variable polygons.

Some irregular tessellations are constructed based on a set of spatial objects, mostly points. The construction algorithm adapts the mesh to the data distribution. This way, data density is taken into account and reflected in the grid structure. Each grid cell can be defined based on the same amount of base objects. This results in bigger cells in areas of less data, and smaller cells where the base objects are denser [36]. TINs and Thiessen polygons belong to this group.

Other irregular tessellations are not generated by mathematical formulas or algorithms, but are based on real-world phenomena. These can be natural objects like land use, or artificial like cadastral units or political boundaries.

In chapter 2, tiling was defined as "‘a plane-filling arrangement of plane figures"’ [50]. According to this definition, any area that is completely filled by irregular polygons constitutes an irregular tessellation. Following this definition, political entities such as countries, provinces or other administrative areas are irregular tessellations. For example, if the area of interest is a continent, or just a square sector of a continent, it is completely covered and divided by countries.

On the other hand, a polygon class that does not completely cover an area is not an irregular tessellation. This way, a lake polygon class that only covers part of an area of interest is not an irregular tessellation.

Triangulated Irregular Networks

TINs are a data structure that is widely used for surface modelling as well as isopleth construction [24] [36]. They are suitable for representing terrain variability, such as ridges, ditches, planes and mountains. To model a surface as lifelike as possible, the set of points providing the base for the triangulation has to be carefully chosen. The quality of the resulting surface depends on the amount and quality of the sample points. In areas that contain more points the approximation is better, as is the spatial resolution.

TINs belong to the group of 2.5-D data structures - so called because their dimensional properties lie between 2-D and 3-D. True 3-D structures are able to store multiple z-values for one location, enabling them to model overhangs and tunnels. 2.5-D structures on the other hand only store one z value per location.

TINs are constructed using a set of points, scattered irregularly over an area of interest. The z-value of these points is based on measurements, ensuring that the set contains important points, such as saddle points and peaks as well as other minima and maxima in the area. This way, the true shape can be approximated better.

Each of the points has one z-value. The triangulation method usually employed to construct the mesh is the Delaunay triangulation (see 2.1.2). This method produces triangles that are as equilateral as possible [53]. The reason why equilateral triangles are best suited for a TIN is the following: the points in the set represent sample measurements. This means that the modelled surface is at its most accurate near these sample points. In an equilateral triangle, every point is as near to the sample point as possible.

A TIN consists of nodes, edges and triangles. Neighbouring points are joined by edges, forming the triangles. Each triangle is a part of the surface, thus providing information about this part of the surface, such as slope, aspect and area. The triangulation describes properties of the whole surface, e.g. volume, line of sight between points and it facilitates surface analysis, e.g assessment of landslide risk or drainage studies [24].

There are two possibilities to store a TIN. The first is to store the triangles themselves: an identifier, the nodes and the neighbouring triangles' identifiers are stored for every triangle. The second method is node based: an identifier, the node and the identifiers of the neighbouring nodes are stored. A neighbouring node is one that is connected to the node in question by an edge.

The difference between these two methods is minimal. In principle, triangle storage is better suited to applications that analyse areas, such as slope and drainage, whereas node storage is better for the generation of contour lines.

Thiessen Polygons

Thiessen polygons are also known as Voronoi diagrams or the Dirichlet tessellation. As with TINs, the tessellation is based on a set of points. Each polygon contains only one point of the base set, and every point in each polygon lies closer to this base point than to any other point of the set [52].

In geoinformatics, Thiessen polygons were originally used to estimate rainfall. To do this, the rainfall measured at a sample location is applied to its whole Thiessen polygon. The resulting map shows the same rainfall measurements in each polygon, which change abruptly at each polygon boundary.

The uses of Thiessen polygons goes beyond rainfall estimates. They are used as a base for nearest neighbour searches as well as a method for generalizing vector databases [24]. Their use in as a spatial interpolation method is limited, as the abrupt change in values at polygon boundaries is often improbable.

4.2 Web Mapping

Web mapping is arguably the most popular application of tiles today. Most Internet map services use tiles as the basis on which their APIs are built. These services and the release of APIs and with itthe possibility to create map mashups helped to bring about the rise of *neogeography.*

The term web mapping is applied to a variety of web sites and applications that have spatial content. According to [26]:

> " It can mean a simple web page that shows a satellite image or a Flash-based application with high levels of interaction, animation and even sound effects. But, for the most part web mapping implies a web page that has some sort of interactive map component. "

Expanding that definition, Neumann describes web mapping as a series of actions that includes not only the presentation of maps on the Internet, but the design, implementation and generation of said maps as well [28]. Above all, web mapping concerns itself with the technologies employed in this process. These include the technologies used by the APIs used to access the maps, standard web technologies as well as data storage technologies.

Together with web mapping, the terms web GIS and web cartography have been introduced. Web cartography considers the maps themselves, their usability, their aptitude for digital media and other non technological aspects. Web GIS is sometimes used synonymously with web mapping, although the two terms are distinct. Web GIS focuses on analytical and geodata processing abilities and components. A web map is often used to show the results of analysis and processing performed by a web GIS.

The difference between web GIS and web mapping is becoming increasingly fuzzy as the APIs for web maps evolve, becoming more sophisticated. They often provide services and functions that allow simple geoprocessing or analysis. Among these functions are geocoding, elevation requests for a given location and calculating routes.

Web mapping has a number of advantages over traditional paper as well as digital maps:

Actuality: Web maps can be updated when the need arises. Theoretically, this enables them to display spatial information in real time, thus reacting to events and phenomena as they occur.

Easy dissemination: Interested parties can access the information without having to acquire new maps or wait for them to be printed and sent to them. Web maps turn the acquisition of spatial information from a push to a pull service, shifting the responsibility of getting up-to-date information from the provider to the user.

Additional content: The use of standard web technologies like HTML allows the integration of additional information in various formats, be it text, images or audio and video. This content can either be directly integrated in the site containing the map, or it can be linked to, turning the web map into an information hub.

Multiple data sources: Through the use of standardised services such as WMS multiple data sources can be integrated into one map. The map provider is not responsible for maintenance and technical support of these sources.

Accessibility: Through adhering to standards, a web map can be displayed in almost any browser. This ensures that the intended audience can access the information without technical problems.

Interactivity: Web maps usually provide some interactive features: zooming and panning the map, changing between a map view and aerial or satellite imagery as well as adding and removing other layers.

Easy assembly: Nowadays it is fairly cheap to obtain the hardware necessary to operate a web server, it is also possible to lease web hosting services. Software, like web servers or databases, as well as software to create and host maps is freely available as open source projects continue to spring up and mature.

Collaborative mapping: The application of tried and proven Internet technologies to maps and GIS in general has opened mapping to a greater public. Theoretically, maps could be created by anyone. Projects such as Open StreetMap have been made possible by the advent of the Internet in general and web mapping in particular.

On the other hand, web maps have some disadvantages as well:

Spatial data: Spatial data is still not freely available for the whole planet, and buying it can be expensive. Community efforts such as Open StreetMap are becoming increasingly interesting as a free and open data source, and some satellite imagery is freely available globally.

Reliability of external data sources: The fact that multiple external data sources can be integrated in one map can turn from an advantage into a disadvantage. Data sources can change their technical specifications, or disappear altogether without notice.

Data quality and copyright: When using external data sources, copyright issues have to be carefully checked to make sure that no legal issues arise. The same applies to data quality.

Technical limitations: Technical problems can pose serious issues for web mapping services. Performance issues can arise when too many requests are made simultaneously, slowing the service down considerably. External data sources can be unavailable, limiting the information content. Screen size is limited, so it may require a lot of panning and zooming to view the desired area. These technical limitations especially apply to mobile maps, which, when viewed in a mobile web browser are special form of web maps. Mobile devices have other limitations than desktop computers. Mobile maps are introduced in section 4.2.3.

Development: Even though it becomes increasingly easier to acquire free spatial data and create maps, mapping still needs special skills in the field of geography and computer science. Creating an interactive web map combining different data sources and publishing it on the Internet is still a complex undertaking.

Web mapping is a relatively new application of geoinformatics, covering a broad range of web sites and applications. An early classification of web maps identifies three types of online maps: view-only atlases, interactive atlases and analytical atlases [28].

Today's expanded technical possibilities render this classification almost moot. The aforementioned definitions of the term web mapping facilitate the division into static and dynamic maps, further splitting it into view only and interactive maps. The first classification method refers to the map's content, whereas the second refers to interaction possibilities.

Static Web Maps: These maps are usually just images, lacking all interactive features. The formats used for this web map type can be either raster or vector files, e.g PNG or SVG. They are often scanned paper maps, and are therefore not frequently updated. This type of web map is very simple and, if it uses a raster format, does not require any special technologies or plugins.

Dynamic Web Maps: Dynamic, in this case, refers to the fact that the maps are created dynamically from one or more data sources, be they local databases or Internet services. The output format of the map can be the same as for static maps. This map type still does not provide any interactive features.

View Only Web Maps: Both static and dynamic web maps are essentially view only maps, as they do not offer interactive features like zooming and panning.

Interactive Web Maps: The default features that interactive web maps offer today are zooming and panning the map. These features help to explore the map and allow the user to focus on a region of interest. More advanced features allow the user to add additional layers or search for directions.

An example for an interactive web map is shown in figure 4.4. The map uses the OpenLayers framework and offers zoom as well as pan controls and an additional layer via WMS showing buildings.

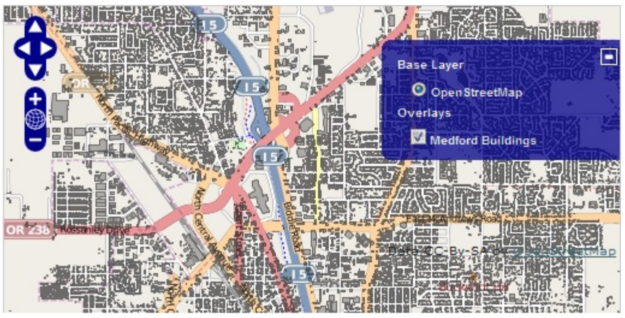


Figure 4.4: Interactive web map [32]

Nowadays, interactive maps are the de-facto standard of web mapping. Interactive maps using tiles are usually known as *slippy maps*, and the concept they represent is the map mashup.

Figure 4.5 illustrates the main concepts, technologies and terms used in tile web mapping.

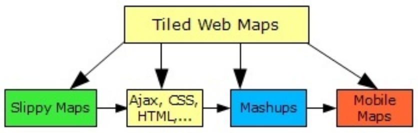


Figure 4.5: Web mapping: terms and concepts

4.2.1 Slippy Maps

The technologies and base data used to create web maps vary, but all interactive maps services use a client-server architecture. There are many server side technologies that can be used to serve web maps. There are solutions that handle vector as well as raster data, some that enable database access, some that only support file access. There is also special server side software that handles tiled maps, such as TileCache[2] or GeoWebCache[3].

The client part is taken on by a web browser. Current browsers are able to display raster data in the form of JPEG, PNG or GIF files. Other file formats can be supported, but some may require separate plugins to be installed. Some browsers support vector data formats natively, e.g Mozilla Firefox supports SVG [43]. Other browsers require additional plugins to be installed, and important GIS vector data formats, such as shapefiles are not supported at all. This constitutes one of the reasons why raster data is prevalent in web mapping services, and why tiles are represented as raster data.

The client side of a web mapping service uses standard Internet technologies, such as JavaScript, HTML and XML. This combination of technologies recently has become popular under the name AJAX. The real gain this technology brings is the speed that the asynchronous data transfer provides. This communication paradigm is embodied by the *XMLHttpRequest* object which is implemented in most web browsers by default. It facilitates the transmitting of data in the background, without locking the user interface. Thus, the user can keep on working while the requested data is being loaded.

The combination of a set of pre-rendered tiles at fixed scales and the interaction possibilities provided by the AJAX technology produces the *slippy map*. The success of slippy maps as the primary vehicle for web maps is possible because of the speed effected by loading only the tiles that can be displayed in the viewport, i.e. the part or the screen reserved for viewing the map. Depending on the zoom level, only a limited spatial extent is shown in the viewport. To enable seamless panning, adjacent but invisible tiles are often preloaded. The term slippy map is sometimes used synonymously with tile based mapping [1].

How slippy maps work and provide controls to interact with the map will be explained in the following section, using the AJAX based Open Source web mapping toolkit OpenLayers as an example. The purpose of this is to further the grasp of slippy maps as the primary application of tiles in web mapping.

OpenLayers

OpenLayers itself is completely written in JavaScript. It was developed by MetaCarta starting in 2005, and the currently available version is 2.10. It supports not only the use of locally stored tiles, but also tiling schemes using open standards, proprietary tiling schemes and geospatial protocols

[2]http://tilecache.org/
[3]http://geowebcache.org

and formats. Among these are GML and Web Feature Service (WFS) and the possibility to use other vector data formats.

OpenLayers implements the possibility to use the tiles of existing web map services such as Google Maps and Bing Maps and overlay, as well as integrating layers from ESRI's ArcIMS Server. It also supports tiles served according to the WMTS and TMS specifications. All these different tile access methods, or *layers*, in OpenLayers terminology can be combined as desired.

OpenLayers is an object-oriented toolkit, providing classes to cover every aspect of web mapping. To create a simple web map with interactive features, a small number of classes is sufficient. These classes are explained briefly in the following section.

Map This is the core class of OpenLayers. It provides the main object to which all layers and interaction controls are attached. The visible manifestation of this object is the map integrated in a web page. On its own, objects of this class do not provide a lot of functionality, neither do they offer any content. It allows some control of the map's behaviour and provides means of administration via diverse functions [43]. To add content, an object of type layer has to be explicitly added to the map.

Layer As the name OpenLayers suggests, this class is the one that provides the map object with actual content. It implements the base functionality and properties that all map layers share. To add an actual layer containing spatial information, a number of subclasses are available. These subclasses implement the actual tiling schemes specified by the tiling methods they represent. These can be roughly divided into four categories.

Figure 4.6 illustrates this classification.

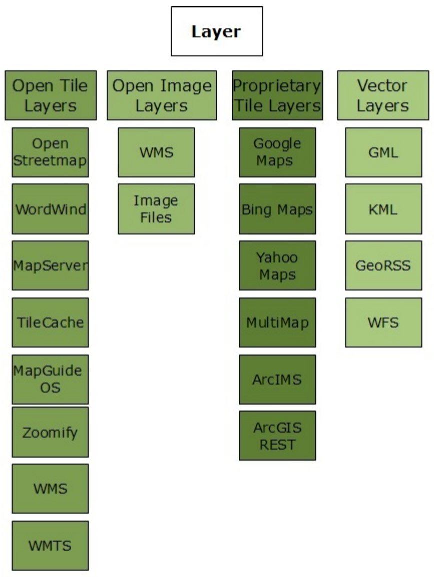


Figure 4.6: OpenLayers Layer class

Open Tile Layers: Into this category fall layers that conform to the WMTS and TMS specifications as well as layers that use open web map services as a source for tiles, e.g. Open StreetMap or WorldWind. Other layer implementations of this group provide support for tiles served with Open Source toolkits such as MapServer.

Open Image Layers: OpenLayers not only provides support for tile based web mapping, but for services that serve up untiled raster maps. The most prominent example for such a service is the OGC's WMS. As long as the spatial extent and URL are known, basically any image can be embedded using OpenLayers.

Proprietary Tile Layers: This group facilitates the inclusion of proprietary web services. It also allows the use of the different base maps provided by these: street maps, satellite imagery and hybrid.

Vector Layers: This category includes GML and KML layer implementations.

A layer can be either a *base layer*, or an *overlay*. Base layers are mutually exclusive, i.e. only one can be active at a given time. Overlays can be stacked on top of one another and turned on or off as needed.

Thus all objects of type layer provide spatial information, but no interactive features. To add panning, zooming and other interactive features to the map, the control class is needed.

Control The de-facto default for interactive web maps is to have pan and zoom controls. Some display a scale bar or an overview map as well, others sport a layer switcher. OpenLayers provides all these default controls, and the possibility to add more.

Pan/Zoom Control: This is a standard control, uniting the pan control which enables panning in the four cardinal directions and a control feature to change the zoom level. A related control element, the Pan/Zoom bar provides the same functionality - the difference being that the zoom control is represented as a sliding scale rather than just with plus and minus signs.

Overview Map: This control displays a small map in a corner of the main map. The overview map is assigned a layer to give an overview of. The scale of the map is such that the surroundings of the layer are clearly visible and put in a spatial context.

Layer Switcher: This control displays all layers that are defined for this map, dividing them into base layers and overlays.

Scale: There are two ways to add a scale: the scale control, which simply states the current scale as a ratio in the form of, e.g. 1:1.000.000, or the scale line control, which displays a scale bar with measurements and units.

Graticule: This adds a latitude/longitude grid to the map.

There are numerous other interactive controls which provide means to control the behaviour of the keyboard, or mouse events, or for manipulating vector data.

There are many more classes in OpenLayers, such as a class for geographic latitude/longitude coordinates, a class to represent a single tile and its properties like size and image format. There are classes that provide support for vector geometry, following the OGC's Simple Features specification, and classes that provide utilities for both vector and raster data.

But the three classes described above are the most important in the OpenLayers toolkit. They provide the basic objects and functionality to create a map, add content and interactive controls - these three classes are enough to create a fairly sophisticated interactive web map.

Figure 4.7 shows the controls and objects introduced above, highlighting the controls and layers.

The above example was assembled using sample data from the United States Geological Survey (USGS), available from the University of Berkeley (see section 1.3). The images were tiled using MapTiler. The tiling scheme used is the one specified in the OSGeo's TMS specification, which can easily be integrated with OpenLayers. When creating the tiles, MapTiler offers the option to create a sample page for OpenLayers, which – with some changes – was used in this example.

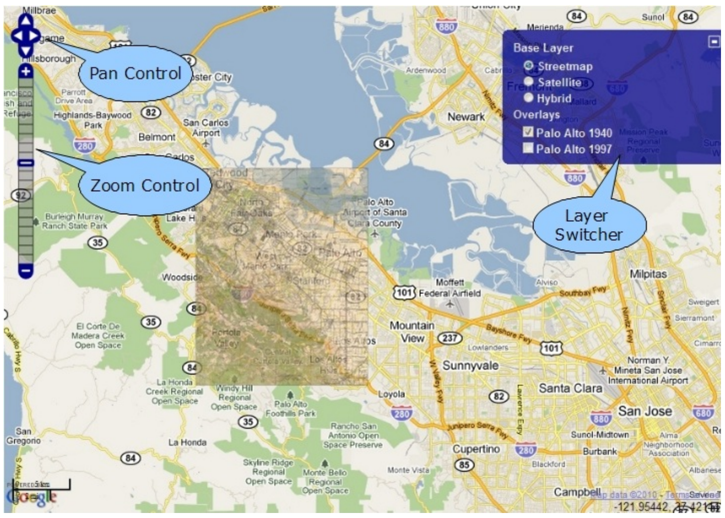


Figure 4.7: Interactive web map

4.2.2 Map Mashups

Map Mashups is a relatively new term, describing the use of a web map service's API in conjunction with other spatially referenced data sources. The result is then presented using the map service's user interface integrated into a web site [13]. Mashups started shortly after the release of the Google Maps service. At first, Google did not allow public access to the API for its map service, so the first mashups where created by reverse engineering the closed API and adding external data sources.

The first mashups included ChicagoCrime, which was later integrated into EveryBlock[4] and HousingMaps[5], which combines Google Maps with real estate data taken from the classified service CraigsList. In response to these first mashups Google released the Google Maps API. Since then, the other major service provider have released their own APIs. Even some national mapping organisations like the Ordnance Survey in Great Britain have released APIs

A multitude of map mashups has sprung up on the web: some in response to events such as natural disasters, others overlaying historic maps or displaying a variety of spatial variables.

Representative of the variety of map mashups, figure 4.8 shows a service presenting historic maps of Manchester, overlaid over an Ordnance Survey map[6].

[4]http://chicago.everyblock.com/
[5]http://www.housingmaps.com/
[6]http://manchester.publicprofiler.org/

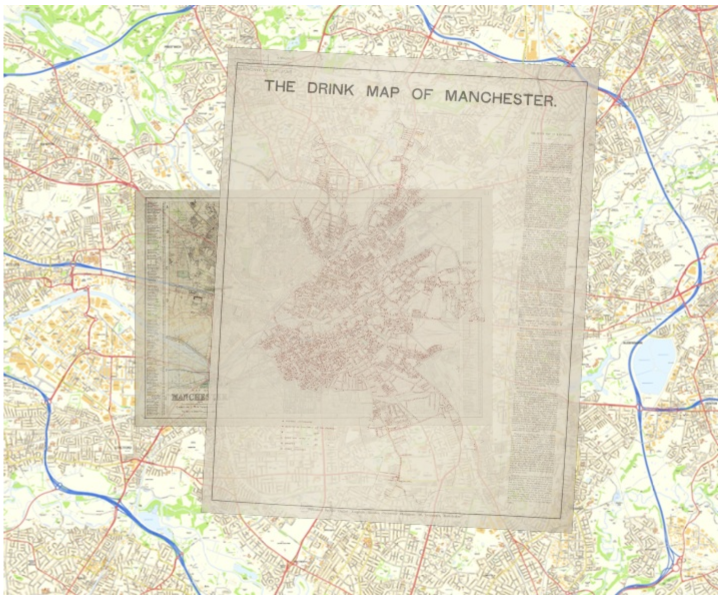


Figure 4.8: Map Mashup using Ordnance Survey base data and two historic maps of Manchester

4.2.3 Mobile Maps

Mobile maps, if viewed in a mobile web browser can be seen as a subset of web map. There are other applications of mobile maps as well, usually involving LBSs. Mobile maps have special requirements, as their purpose is to be displayed on a mobile device, which differs greatly from a desktop computer. Mobile devices have technical limitations that are not important for a stationary computer. These have to be considered when bringing maps to mobile devices.

Display: Mobile device screens are small and they may not be able to display as many colours as a stationary screen. Mobile maps have to be adapted to this, being legible even on small screens and in an unsuited environment.

Interaction: Interaction possibilities are restricted. Modern smartphones often have a touchscreen and a keyboard, but older devices lack these possibilities. This influences the interaction features a mobile map can offer.

Power: A major drawback of mobile devices is that they only have limited battery life. The more functionality a device offers, the higher the power consumption.

Network Connectivity: Wireless network connectivity follows different rules than wired connectivity. Quality of service can change rapidly. This influences the available bandwidth and data transmission speed.

Storage Space: Some services use devices to store data, to enable faster or offline access. Modern smartphones do not have serious problems with this, but again, older devices may only have very limited storage.

Computing Power: Mobile device processors are less powerful than their stationary counterparts.

All these limitations and restrictions have to be taken into account when choosing spatial data for a mobile map service. Some of these issues can be tackled when the map is created, some can be addressed when the service is in use.

- Information Content: the content can be adapted to suit the purpose of the map, omitting all unnecessary information or moving it to overlays that can be switched on when needed.
- Cartographic means: as some mobile devices can only display a limited set of colours, the map can be adapted to use only a limited set of colours as well. Another consideration is that mobile maps are used in an environment where conditions are not ideally suited to reading a screen, thus the use of high contrast and few colours can be beneficial.
- Caching: If enough storage is available, maps or parts thereof can be cached on the device itself.
- Computing: the need to do time and computing power consuming calculations and procedures should be shifted away from the mobile device when possible.

Tiles are well suited to be used in mobile maps. The points mentioned above can all be realised using tiled maps.

Most mobile devices support standard image file formats like JPEG, thus the basis for the use of tiles is a given. Specially adapted tiles can be rendered for mobile applications. Using a transparent file format, these tiles can represent different layers and be used accordingly. Tile image files can be transmitted one after the other, making the whole process of transmitting a map less error prone by dividing it into packets. Single tiles can be stored on the mobile device for faster or offline access. To calculate tile bounds, or geographic coordinates for a single pixel or vice versa, simple operations suffice.

Chapter 5

Summary and Conclusion

5.1 Summary

The goal of this thesis was to give an overview of tessellation and its uses in geoinformatics, with a special focus on tile based web mapping, as this is today's most popular applications of tiles.

It started by explaining the fundamental concepts, terms and methods of tessellation, drawing its sources from mathematics, computer science, information theory and geography. All these fields contribute to tessellation and its product, tiles in their many forms, as they are used in geoinformatics. The tile of choice for tile based mapping is square, the associated form of tessellation is the square regular tessellation, also known as tiling. This simple tiling method forms the base for one of the most successful applications of tiles, online map services.

The tiles used by these services are based on raster maps which are available at different resolution levels, thus forming a tile pyramid. To actually create tiles and a tiled map service to serve these tiles, a lot of different issues have to be considered. This thesis addressed the most important considerations to make before tiling, and to give a glimpse of software that does the actual tiling. Storage in file systems was addressed just as databases and their possibilities and capabilities to store raster data, form pyramids and cut a map into tiles were. This was followed by an excursion into open standards that define protocols for tiled map services.

The final part of this thesis concerned applications of tessellation and tiles. Important tessellations, like TINs and Voronoi diagrams were described just as the rather exotic yet interesting DGGSs. Following that were quadcodes, which lead directly to web mapping. As a representative of modern web mapping frameworks, OpenLayers was described in detail.

Figure 5.1 shows the trinity that forms the fundamental concepts and data basis of tiled mapping: a map – usually a raster map – forms the basis for the service. This simple map is then turned into a resolution pyramid and finally cut into tiles.

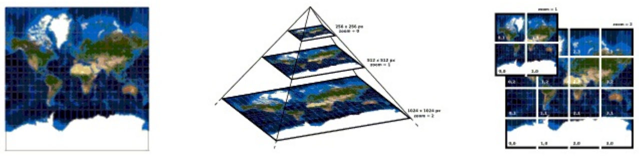


Figure 5.1: Map, pyramid and tiles

5.2 Conclusion and Outlook

Tessellation is an integral part of geoinformatics and GIS. TINs as well as Thiessen polygons are well established data structures, quadtrees are used in spatial databases, and the latitude/longitude system itself can be seen as a spherical tessellation.

Internet map services that use tiles are still evolving, adding functionality and becoming more sophisticated with every new release of their APIs. Apart from the big players in the field and their services, there exist now the possibilities for small organisations and even individuals to publish maps on the Internet. This an be accomplished either with the help of said big players or by using free software and open standards.

Of the two existing open standards, WMTS is the more comprehensive and offers more possibilities to a tile service. Especially the Feature Information, which turns the service from a mere tile service into something that offers additional information at the pixel level, makes this specification interesting and forward looking. Additionally, it is an official standard by the OGC, thus ensuring that it is maintained, updated and universally accepted. If it will be universally implemented as well remains to be seen.

The other representative of open tiling standards is TMS. This specification, although simple and easy to implement, lacks the comprehensiveness its counterpart offers. Its handling of profiles, as it calls a common scale set for tiles, is not easily expandable, and it gives the general impression of being ill-conceived and less mature that WMTS.

Another issue that will be important in the future of tiled mapping is storage. Tiles can take up a lot of storage space, if files are used maintenance and updating can be cumbersome. On the other hand, storing tiles in a database needs a lot of expertise and can become expensive, as spatial databases with raster support are mostly proprietary products. Files are certainly the storage method that is simpler and faster to implement, but databases provide additional functionality, which can be used to enhance the information a service has to offer.

An interesting development in the field of spatial databases is WKT Raster, a module that can be installed on top of the PostgreSQL/PostGIS Open Source database. Its aim is to offer functions that can be used on both raster and vector data, thus enabling seamless handling of all the data contained in a database. This project is currently in its beta phase, so it remains to be seen if it can fulfil its promises. If it does, it certainly opens up new ways to integrate different data sources and offer these as tiles to use in an online service. It also allows the whole process of creating, storing and serving tiles to be handled using open software.

From today's point of view, the future looks bright for online map services, and with them, tessellation and tiles.

Bibliography

[1] Adnan, M., A. Singleton, and P. Longley: *Developing Efficient Web-based GIS Applications.* Techn. rep., Centre for Advanced Spatial Analysis, 2010.

[2] Centre for Advanced Spatial Analysis: *GMapCreator.* URL, http://www.casa.ucl.ac.uk/software/gmapcreator.asp,. Accessed 30.07.2010.

[3] Dhamdere, D.: *Operating Systems: A Concept-based Approach.* Tata McGraw-Hill, 2009.

[4] Dutton, G.: *Zenithial Orthotriangular Projection.* In *Proceedings of the Tenth International Conference on Computer-Assisted Cartography (Auto-Carto 10)*, pp. 77–95. American Congress on Surveying and Mapping, 1991.

[5] Dutton, G.: *Encoding and Handling Geospatial Data with Hierarchical Triangular Meshes.* In *Symposium on Spatial Data Handling*, pp. 505–518, 1996.

[6] Dutton, G.: *Digital Map Generalization using a Hierarchical Coordinate System.* In *In Auto Carto*, pp. 367–376, 1997. http://www.spatial-effects.com/papers/conf/GDutton_AC13.pdf.

[7] Elson, J., J. Howell, and J.R. Douceur: *MapCruncher: Integrating the World's Geographic Information.* SIGOPS Oper. Syst. Rev., 41(2):50–59, 2007, ISSN 0163-5980.

[8] ESRI: *Available Map Cache Properties.* URL, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html/Available_map_cache_properties/005300000098000000/,. Accessed 05.07.2010.

[9] ESRI: *Cell size of Raster Data.* URL, http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Cell_size_of_raster_data,. Accessed 18.06.2010.

[10] ESRI: *Georeferencing a Raster Dataset.* URL, http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Georeferencing_a_raster_dataset,. Accessed 22.06.2010.

[11] ESRI: *World File Format.* URL, http://resources.arcgis.com/content/kbase?fa=articleShow&d=17489,. Accessed 22.06.2010.

[12] GDAL: *Gdal2tiles.* URL, http://www.gdal.org/gdal2tiles.html,. Accessed 26.07.2010.

[13] Gibin, M., R. Milton, P. Mateos, and P. Longley: *Collaborative mapping of London using Google Maps: the LondonProfiler.* Techn. rep., Centre for Advanced Spatial Analysis, 2008.

[14] González, R.C. and R.E. Woods: *Digital Image Processing.* Prentice Hall, 3rd ed., 2008.

[15] Google: *Google Maps JavaScript API V2.* URL, http://code.google.com/apis/maps/documentation/javascript/v2/overlays.html,. Accessed 26.10.2010.

[16] Google: *Google Maps/Earth Terms of Service.* URL, http://www.google.com/intl/en_at/help/terms_maps.html,, 2010. Accessed 25.08.2010.

[17] The PostgreSQL Global Development Group: *PostgreSQL 8.4.4 Documentation*, 2010. Accessed 09.08.2010.

[18] Kaplan, C.S.: *Computer Graphics and Geometric Ornamental Design.* PhD thesis, University of Washington, 2002.

[19] Kaplan, C.S.: *Introductory Tiling Theory for Computer Graphics.* Morgan & Claypool Publishers, 2009.

[20] Kothuri, R., A. Godfrind, and E. Beinat: *Pro Oracle Spatial.* APress, 2004.

[21] Jet Propulsion Laboratory: *OnEarth Tiled Service.* URL, http://onearth.jpl.nasa.gov/tiled.html,. Accessed 02.02.2010.

[22] Lee, M. and H. Samet: *Navigating Through Triangle Meshes Implemented as Linear Quadtrees.* ACM Transactions on Graphics, 19:79–121, 1998.

[23] Li, S.X. and M.H. Loew: *The Quadcode and its Arithmetic.* Commun. ACM, 30(7):621–626, 1987. http://uhost.rmutp.ac.th/sangsan.t/paper/.

[24] Longley, P., M. Goodchild, D. Maguire, and D. Rhind: *Geographic Information Systems and Science.* Wiley, 2nd ed., 2005.

[25] McMaster, R. and S. Shea: *Generalization in Digital Cartography.* Association of American Cartographers, 1992. http://www.survey.ntua.gr/main/courses/geoinfo/admcarto/lecture_notes/generalisation/bibliography/mcmaster_shea_1992.pdf.

[26] Mitchell, T.: *Web Mapping Illustrated.* O'Reilly, 2005.

[27] Nebiker, S.: *Spatial Raster Data Management for Geo-Information Systems - A Database Perspective.* PhD thesis, ETH Zürich, 1997. http://www.igp-data.ethz.ch/berichte/Blaue_Berichte_PDF/63.pdf.

[28] Neumann, A.: *Encyclopedia of GIS*, ch. Web Mapping and Web Cartography, pp. 1261–1269. Springer, 2008.

[29] Obe, R. and L. Hsu: *PostGIS in Action.* Manning Publications Co., 2010.

[30] OGC: *OGC Web Services Common Standard*, Apr. 2010.

[31] OGC: *OpenGIS Web Map Tile Service Implementation Standard*, Apr. 2010. http://www.opengeospatial.org/standards/requests/54.

[32] OpenLayers: *WMTS Example.* URL, http://openlayers.org/dev/examples/,. Accessed 16.09.2010.

[33] Oracle: *Oracle Spatial GeoRaster*, 2005. http://dba-services.berkeley.edu/docs/oracle/manual-10gR2/appdev.102/b14254.pdf, Accessed 04.08.2010.

[34] Oracle: *Oracle Spatial Quadtree Indexing*, 2005. http://dba-services.berkeley.edu/docs/oracle/manual-10gR2/appdev.102/b14254.pdf, Accessed 04.08.2010.

[35] OSGeo: *Tile Map Service Specification.* URL, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification,, 2009. http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification, Accessed 07.01.2010.

[36] Peuquet, D.J.: *Representations of Space and Time.* The Guilford Press, 2002, ISBN 1572307730.

[37] PostGIS: *PostGIS 1.5.1 Manual*, 2010. http://www.postgis.org/download/postgis-1.5.1.pdf.

[38] Racine, P.: *PostGIS WKT Raster Specifications Version 1.0.* 2008. http://www.cef-cfr.ca/uploads/Membres/WKTRasterSpecifications1.0.pdf.

[39] PostGIS WKT Raster: *PostGIS WKT raster.* URL, http://trac.osgeo.org/postgis/wiki/WKTRaster,, 2010. Accessed 09.08.2010.

[40] Reihs, F.: *Organisation und Verwaltung von Rasterdaten digitaler Geländemodelle.* Master's thesis, University of Salzburg, 2007.

[41] Sahr, K., D. White, and A.J. Kimerling: *Geodesic Discrete Global Grid Systems.* Cartography and Geographic Information Science, 30(2):121–134, 2003.

[42] Samet, H.: *Applications of Spatial Data Structures.* Addison–Wesley, 1990.

[43] Schütze, E.: *Current state of technology and potential of Smart Map Browsing in web browsers.* Master's thesis, University of Applied Sciences Bremen, 2007. http://www.smartmapbrowsing.org/diplomarbeit_EmanuelSchuetze.pdf.

[44] Schwartz, J.: *Bing Maps Tile System.* URL, http://msdn.microsoft.com/en-us/library/bb259689.aspx,. Accessed 18.10.2010.

[45] ScienceU: *Periodic Tilings.* URL, http://www.scienceu.com/geometry/articles/tiling/periodic.html,. Accessed 16.03.2010.

[46] Tan, L.: *Image File Formats.* Biomedical Imaging and Intervention Journal, 2006. http://www.biij.org/2006/1/e6/e6.pdf.

[47] Weisstein, E.W.: *"Monohedral Tiling" From MathWorld - a Wolfram Web Resource.* URL, http://mathworld.wolfram.com/MonohedralTiling.html,. Accessed 17.03.2010.

[48] Weisstein, E.W.: *"Platonic Solid" From MathWorld - A Wolfram Web Resource.* URL, http://mathworld.wolfram.com/PlatonicSolid.html,. Accessed 25.05.2010.

[49] Weisstein, E.W.: *"Tessellation" From MathWorld - A Wolfram Web Resource.* URL, http://mathworld.wolfram.com/Tessellation.html,. Accessed 08.02.2010.

[50] Weisstein, E.W.: *"Tiling" From MathWorld - A Wolfram Web Resource.* URL, http://mathworld.wolfram.com/Tiling.html,. Accessed 08.02.2010.

[51] Weisstein, E.W.: *"Triangulation" From MathWorld - a Wolfram Web Resource.* URL, http://mathworld.wolfram.com/Triangulation.html,. Accessed 17.03.2010.

[52] Weisstein, E.W.: *"Voronoi Diagram" From MathWorld - a Wolfram Web Resource.* URL, http://mathworld.wolfram.com/VoronoiDiagram.html,. Accessed 23.06.2010.

[53] Worboys, M. and M. Duckham: *GIS - A Computing Perspective.* CRC Press, 2nd ed., 2004.