Generating Maps from OpenStreetMap Data for a Serious Game

Florian Kempter

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2015

© Copyright 2015 Florian Kempter

This work is published under the conditions of the *Creative Commons License Attribution–NonCommercial–NoDerivatives* (CC BY-NC-ND)—see http://creativecommons.org/licenses/by-nc-nd/3.0/.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 28, 2015

Florian Kempter

Contents

Abstract

This work discusses the integration of real-world map data into serious games to explore issues surrounding global ecology. Several projects from the fields of urban planning, serious games and map generation are presented, including an analysis of their respective drawbacks in an educational setting. The desired tool should be capable of harvesting relevant information about any city and generating a map that is both recognisable and playable. Data that is already collected in urban computing is presented and compared to its availability in a map service database. The exemplary serious game *Urban Space Upgrader* is evaluated to reveal limiting factors such as the absence of data in some locations. The thesis provides multiple approaches to handle this problem, including an algorithm that fills empty spaces automatically.

Kurzfassung

Diese Arbeit behandelt die Einbindung von realen Kartendaten in ein Lernspiel um die Erforschung von globalen Umweltproblemen zu ermöglichen. Mehrere Projekte mit dem Thema ‘urban planning’, ‘serious games’ und ‘map generation’ werden vorgestellt und auf ihre jeweiligen Nachteile in einer Lernumgebung analysiert. Das gewünschte Werkzeug muss imstande sein, relevante Informationen über eine beliebige Stadt zu sammeln und daraus eine wiedererkennbare und spielebare Karte zu generieren. Daten, die bereits im Bereich ‘urban computing’ gesammelt werden, werden auf ihre Verfügbarkeit in einer Landkarten Datenbank geprüft. Das exemplarische Lernspiel *Urban Space Upgrader* wird ausgewertet um mögliche Störfaktoren, wie beispielsweise das Fehlen von Daten in einigen Regionen, aufzudecken. Die Arbeit liefert mehrere Ansätze zur Lösung dieses Problems, wie etwa ein Algorithmus der leere Flächen automatisch befüllt.

Chapter 1

Introduction

Industrialization and its resulting damage for the global ecosystem is a major topic and especially first world countries have to be aware of their responsibilities. However many people are either not interested or do not know about their ‘carbon footprint’ even though there is already a big library of documentaries concerning the issue. Therefore the main goal is to educate people about the influence of their behavior on the surrounding ecosystem. Fortunately the information age provides the tools to help accomplish this. The source for learning material are websites such as *Google Maps* or *OpenStreetMap*, map services that in addition to being used for navigation provide a wide range of geographical information. By processing and filtering these data, it is possible to create tools that are relevant enough to interest people about urban planning, a sub-category of ecology. Hereafter, the motivation of creating such a tool is shown as well as its scientific background and relevance to the field of urban planning. Additionally, a short overview about the structure of this thesis is given.

1.1 Motivation

The success of sandbox games such as *Minecraft* show that procedural worlds have proven to provide a long term motivation for the players because every new game the world looks different and contains new challenges and goals to reach. Since educational games would benefit from this feature to keep the users motivated and therefore create a more successful learning environment, it is only reasonable to combine them. The *Sim City* series for example offers generated game maps where players can build their cities and incidentally learn about urban planning. However the main goal of the game is still entertainment which is why changes are needed to shift the focus to an educational purpose. The primarily necessary addition is the usage of real map data instead of fictional data. By using real map data, the users can specifically learn about the urban problems in their native environments and therefore have more motivation for improving the situation. An exemplary topic is for a user to learn about the lack of recreational areas next to a residential part of the city, which may result in a significantly lower satisfaction of the people living there. By gaining this knowledge, the resulting consequence may be petitions in order to create new parks and thus generally impart critical ecological problems. The source of the tool is *OpenStreetMap* because it is freely available and provides the necessary information about urban areas. It is the foundation of a map generator that is able to create an abstraction of every desired location. An exemplary educational game assumes the role of the medium between user and learning material.

1.2 Scientific Relevance

The research question and therefore scientific foundation of this thesis can be stated as follows:

> What is a sufficient way of harvesting *OpenStreetMap* data and using it for serious games about urban planning?

'Sufficient' means that the maps that are created from the database should resemble the origin and therefore be recognized. For example if Linz, Austria is recreated, the user is able to identify certain areas and connect them to the real city. Furthermore, it means that the game map is structured to enable the implementation of various serious games. The final goal of the tool is to collect information from real maps that are relevant to urban planning and transform them to a game environment that is both realistic and educational. Seen from an economical perspective, the market for urban planning and educational games is big and therefore demand for relevant software is high. Still only a few tools are developed for the topics, especially when combining both of them.

1.3 Structure

This thesis is segmented into seven chapters. The current chapter explains the motivational background and scientific relevance of the topic as well as the basic requirements of a tool to fulfill the determined research question. Chapter 2 explains the three main terms that the thesis is concerned with and how to combine them. The mentioned terms are 'urban planning', 'procedural generation' and 'serious games'. Chapter 3 continues to present examples of already existing tools that connect the previously defined topics and point out their application areas. By evaluating this state of the art, Chapter 4 presents the conceptual progress of the tool, particularly its requirements and the resulting choice of used technologies. Chapter 5 proceeds to show the actual implementation of the tool, the problems that occurred during development and how they were resolved. Since there were multiple solutions to the obstacles, Chapter 6 evaluates the different approaches based on a list of criteria. Lastly, Chapter 7 presents some special cases of map generations, ways to improve the tool further and an assumed general development of the topic in the future.

Chapter 2

Definition of Terms

The generation of game maps by utilizing *OpenStreetMap* data for a serious game consists of three important parts: urban planning, procedural generation and serious games. None of them appeared recently and therefore each one has already a respective field of research. Procedural generation and serious games firstly occurred around 1970 and urban planning has its history even earlier. Nevertheless, they are still developing due to new technologies. This chapter explains these terms and supports them by examples.

2.1 Urban Planning

'Urban' has its origin in the Latin word 'Urbanus' which translates to "of or belonging to a city" [53]. Simply put, by adding the word 'planning', the topic is about outlining a city or town. The Romans created the term and had a huge influence on the process by documenting it greatly. Although, according to LeGates [5], they were not the first to put emphasis on the field. Ancient civilizations in Egypt or Asia built functioning cities between 3000 and 2500 BC. The creators must have applied urban planning to a certain degree to fulfill the demands of such a city. The requirements certainly changed during time but some still persist, for example the need for drinkable water. Ancient cities solved this problem by building residential areas next to rivers and freshwater lakes. The Romans started to develop a water distribution system by building aqueducts, the predecessor of modern water pipes. Another important factor is health. Not only sewerages to prevent the contamination of groundwater but also powerhouses that increase the exhaust gas pollution need to be placed intelligently.

Of course modern urban planning is much more complex than thinking about how natural resources and other goods are transported around the city. LeGates [5] has determined that different demographic groups have different needs and there is an increasing conflict between them. The upper social class for example has a request for luxurious establishments such as golf courses while the lower class wants social facilities such as public kindergartens. Since the financial means of urban planning are restricted by the government, it has to maintain a balance of fulfilling the demands of those classes. With growing complexity of a city more versatile design tools are needed for supporting it. The most common tool that has been used since the beginning of urban planning is a plain piece of paper. It still supports the process today. A modern two-dimensional plan can be seen in Figure 2.1. Every design process starts with simple sketches of the desired result to simplify the workload later on. But since a paper can only contain so much information to remain clear, other tools are needed. A common way of planning areas of a city and presenting them to others are three-dimensional models as seen in Figure 2.2. They have the advantage of showing the height of buildings and surfaces which is very important for evaluating the physical possibility of a plan. They also impart proportions of structures and outline ideas more precise. Of course the information age brought a huge improvement to the working field. Modern computers can simulate cities quicker and calculate the possible outcome of design approaches in the forefront. This process is called ‘Urban Computing’. Yu Zheng [19], one of the lead researchers for *Microsoft*, defines the term as collecting data from urban areas via various sensors.

A large scaled data collection is important, because modern technologies also show other problems that need to be considered. The industrialisation for example has had left behind a big carbon footprint which is harmful for nature and in conclusion harmful for every human being. Statistics [28] show that air pollution in the United States has been drastically reduced since 1990 due to the collection of information about the environment and adjustments to urban planning. But not only air pollution can be detected, processed and visualized automatically to react to changes. Zheng’s work features a list of seven categories with their respectively own dataset that can be gathered by sensors:

- **Urban planning**: The category contains traffic flow, human mobility and points of interest. These data can be summarized as areas where many people need to travel from one point to another and therefore create congestions. Taken measures are for example expansions of streets that are heavily used or the launch of new public transportations.
- **Transportation**: In contrast to the previous bullet, ‘transportation’ concentrates on a single road user and how to get it to its goal. The collectable data are estimations of the fastest and most energy efficient routes. This is already integrated in every GPS (Global Positioning System).
- **Environment**: This point was already mentioned in the previous section and contains air quality and noise pollution. These two information are very important in larger cities where the recommended average

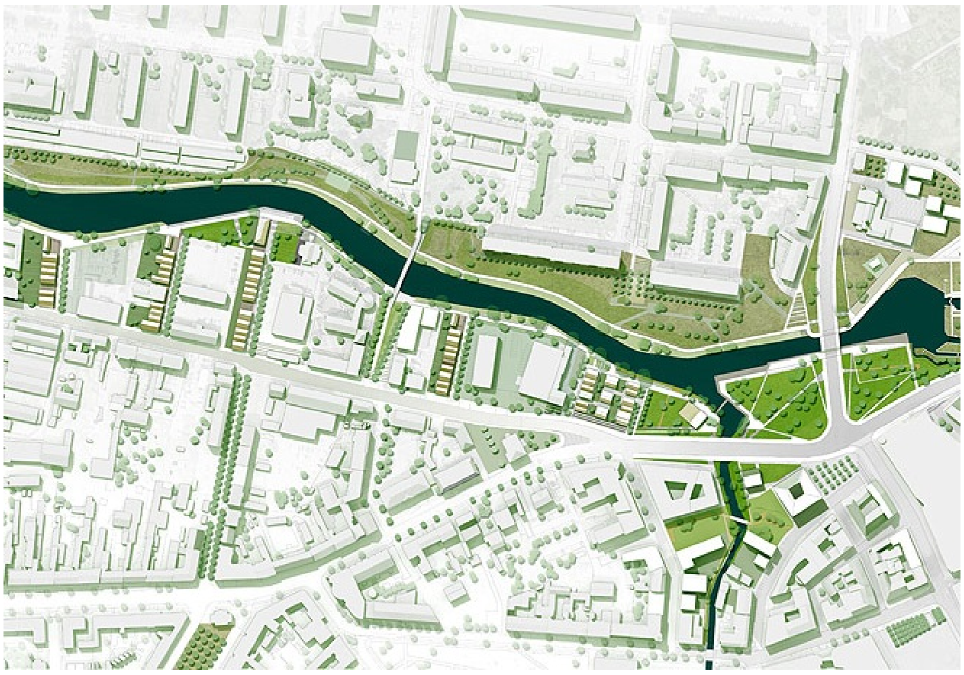


Figure 2.1: A two-dimensional plan of the canal promenade in Eberswalde, Germany [52].

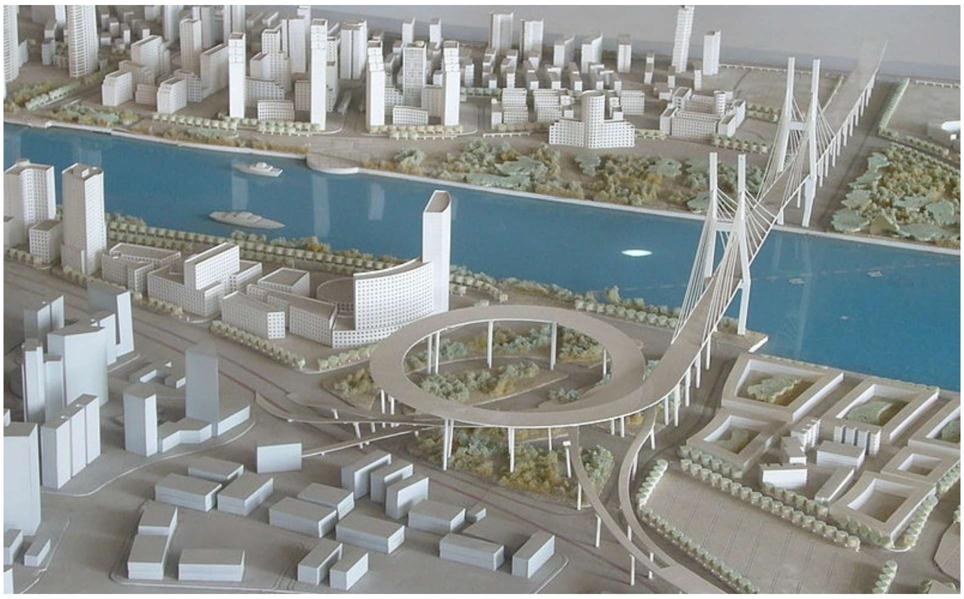


Figure 2.2: A three-dimensional model of a Shanghai bridge, at the Shanghai Urban Planning Exhibition Center [39].

value is most likely excessive and noxious.

- **Energy**: The data for this category are gas and electricity consumption. They require long-term planning, because urban planners try to maintain a regulated and moderate use of available resources. Solar energy, electric cars and other inventions with renewable energies are already improving this area greatly.
- **Social**: Information in this group is as relevant to citizens as to visitors of a city. The dataset contains resident profiles, local experts and location recommendation. They are, except the profiles, already searchable in review websites such as *Yelp* and improve communication among businesses and individuals by visualizing preferences and demands in services.
- **Economy**: In opposition to the previous type of structure, this one is mainly for service contractors and not consumers. It shows the overall interest in businesses and therefore prevents oversupply of certain goods. This also helps the market to stay competitive by identifying monopolists.
- **Safety and security**: The last category is hard to measure because it needs to foresee data at an early stage. Its purpose is to prevent issues such as traffic anomalies or natural disasters such as floods. Depending on the geographic location of a city, events like these are less relevant. On a global scale though, every urban planner has to bear them in mind.

Most of this information is already collected and used for planning and can be openly examined in location-based services such as *Google Maps* or *OpenStreetMap*. Naturally, the data can be combined to draw conclusions. For example areas with insufficient public transportation suffer from heavily used streets and therefore traffic jams and increased air pollution. Companies such as *Facebook* or *Google* use the social and economic aspect of urban computing to create detailed profiles of members based on their residence. A geographical location improves the relevance of advertisement by targeting it specifically at people living near the business. Furthermore, by presenting local urban data to citizens, an educative value can be achieved [9, 11].

2.2 Procedural Map Generation

Procedural processes originate in the field of computer graphics. One of the big players in this domain is James D. Foley who wrote several books about the topic. He recorded [2] that the history of how pictures are displayed on a computer is closely related to the technologies that are available at the time. Although occurring in the 1950s, William Fetter was the first who used the term ‘procedural’ in this context in 1960 at his job for *Boeing Aircraft Co.*

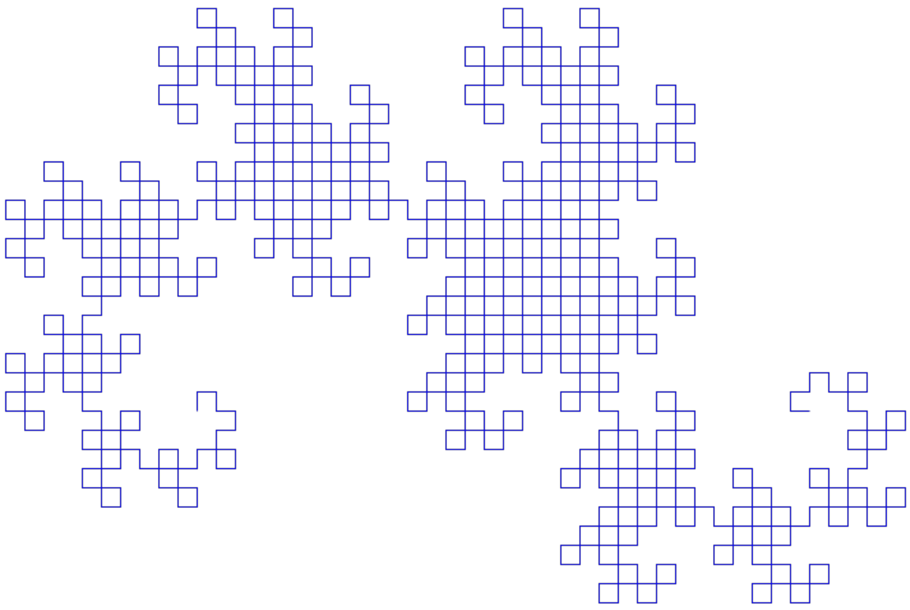


Figure 2.3: The structure is generated by a L-system and is called dragon curve [34].

[35]. The pictures at this time had to be put together manually by either having each pixel on the screen be colored individually or defining a geometrical figure by determining its vertexes in a coordinate system. The technologies improved and it was soon possible to build the first three-dimensional figures. Further development increased the complexity and level of detail of the models and likewise the amount of time needed to create them. Since a computer's first intention was to solve mathematical problems, as soon as digital images started to emerge, people thought about automating the process. A precursor of modern procedural generation is the L-system [13]. An example of it is shown in Figure 2.3.

With each iteration the L-system gets more detailed until an end state predetermined by the user is reached. This process of creating something new from a given dataset every time the input changes is called procedural as the result is not crafted manually but via calculations. Rules help assume the outcome but have to be flexible enough to work on a randomized data set in a given range. Basically put, the L-system's random variable is whether to turn left or right and the constant input is the degree of the turn. The result can be used as an street map for a generated city.

An example for software that uses procedural generation is the video game *Minecraft* [24]. The map of *Minecraft* is made of blocks that resemble materials in the real world. There are for instance blocks of water, dirt or diamonds. Every time the player starts a new game the blocks are placed

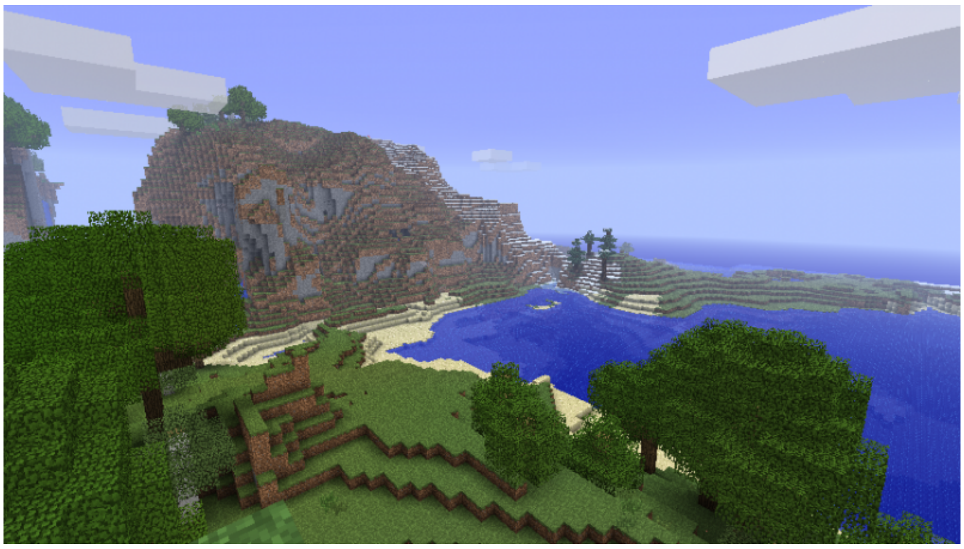


Figure 2.4: The image shows a randomly generated map in *Minecraft* constricted by a rule set to resemble a landscape that would be possible in the real world [29].

randomly but restricted to a rule set. Exemplary rules are that water is always on sand and can never float freely in the air or that diamonds are under the surface below a certain number of other blocks. Those are only some examples, because there are many more needed to create the illusion of a real landscape. A map's size is only restricted by the hardware of the computer but to prevent errors beforehand they are limited to a maximum number of about 900 quadrillion blocks [47]. Figure 2.4 shows an example of such a map. The purpose of generation in this case is to create varying experiences every time a new game is started by the player. If the maps were crafted by hand, there would be much fewer of them due to the enormous amount of time it would take. Since the game gained its popularity due to the fact that it is statistically very unlikely to see the same game map twice, this method has many supporters in the game development industry.

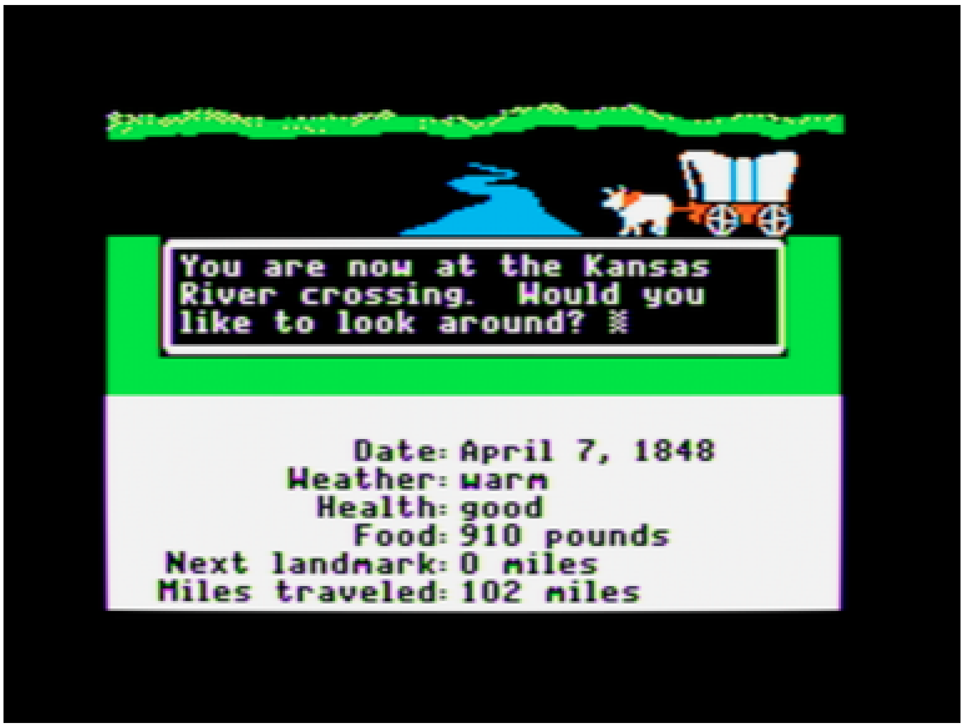
Creating varying experiences is not the only purpose procedural map generation has. To illustrate the previous section about urban planning, procedural map generation is useful for handling large amounts of data and visualizing them on maps. Based on the findings of the University of Oulu [9] this can increase the involvement of citizens by generating an individual map of their surroundings and therefore give specific information to possibly relevant problems. Archon Fung's analysis on how to improve urban planning [3] aids this claim by stating that residents need more information about existing problems and possible solutions to participate in improving their home city.

2.3 Serious Games

At about the same time that early computers managed to show animated pictures, programers at the MIT (Massachusetts Institute of Technology) [35] were able to develop the first video game. Since the only purpose of computers at the time was to solve mathematical problems, a milestone for home computers as they are known today was laid. The first popular game was *Spacewar!* which is about two space ships that have to shoot each other to get points and thus win. Although this concept sounds simple, people were fascinated by it, because no one ever experienced something like this. The reason why *Spacewar!* was intriguing is mainly due to the used technology, but also because it was a fun experience. Therefore it can be concluded that the goal of the game was to entertain its users. The reason why this is emphasized is that serious games, in contrast to regular games, do not focus on this goal. Michael and Chen, programers with broad empirical knowledge in the video game industry, state that a serious game's goal is primarily education and not fun. This does not mean they have no enjoyable value which is why they define them as follows [7]:

> Games may be played seriously or casually. We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement. This does not mean that serious games are not, or should not be, entertaining.

According to the IFETS (International Forum of Education Technology and Society) [17] the enjoyable part in serious games can actually be used to motivate students and therefore foster a better understanding of certain topics. The organization tested this claim with an educational game called *VR-ENGAGE*. It is set in a fantasy world where the users need to answer geographical questions about Greece to proceed and ultimately defeat dragons to collect a powerful artefact. By having inserted learning material inside an adventure, the IFETS researched if a test group of students who used the software had more knowledge about the topic than a group that studied with classical learning material. The result was, with every student having approximately the same previous knowledge, that those who studied with the educational game had better overall scores in a final test. A more recent Greek study by the Department of Physical Education and Sport Science in Athens [10] affirms this result but limits the applications. They say that serious games or digital game-based learning, as they call it, can only grant a long-term motivation to students if a constant variety of learning environments is provided. Despite these results, *VR-ENGAGE* was not established in the rest of the world, probably because of its subject and that many people still see video games as a recreational activity. A more famous example



```
You are now at the Kansas
River crossing.  Would you
like to look around?

         Date: April 7, 1848
      Weather: warm
       Health: good
         Food: 910 pounds
Next landmark: 0 miles
Miles traveled: 102 miles
```

Figure 2.5: This image of *The Oregon Trail* shows a decision the player needs to make to proceed [42].

for an educational game is *The Oregon Trail* which teaches about the migration of pilgrims in the early United States of America [23]. The goal of the game is to manage different resources of a pilgrim group and to make decisions for them which can be seen in Figure 2.5. Probably the main reason for its popularity is that many people remember it for its difficulty and the phrase “You have died of dysentery” [40]. A term often used for games of the genre, such as *VR-ENGAGE* and *The Oregon Trail*, is edutainment which is a subcategory of serious games. The sole difference is that edutainment games mainly target students and younger audience and serious games are meant for every age group.

A category that needs to be mentioned in this section as well are video games that have the goal to entertain but still educate their players. The Wisconsin Center for Education Research [18] analyzed several video games and found out that players acquire knowledge of real life topics by playing them. They base their claim on a comparison to “learning by doing”. Most current video games include tutorials to teach players about game mechanics they later need to apply. Then, by putting small obstacles in their way, the game tests their knowledge. Usually the tutorial offers guidance as there are no penalties for failure. For example if a game requires the character to learn how to jump, at first a button will be shown on the screen that needs to be pressed. Then an easy hurdle, like a small fence, will block the way to the goal. If the fence would be replaced by a large pitfall that requires the player to do multiple jumps at once, frustration may come up, because the game did not yet teach about that. The Wisconsin Center for Education Research uses the first-person shooter *Full Spectrum Warrior* as an example for such a learning process. The inspiration for the game is *America's Army* which is produced and used by the United States Army to simulate possible harmful situations without actually being dangerous [20, 26]. Similar to other first-person shooters the player receives different mission goals but with the feature that fellow soldiers need to be kept alive. The missions get more difficult later on and by adapting to various settings the player learns how to protect the team. This is achieved by giving them orders to move or shoot as seen in Figure 2.6. The only difference to its role model is that *America's Army* is a multiplayer game where the teammates are controlled by actual players as seen in Figure 2.7. Soldiers also need to learn this skill during their military service so even if they have deeper knowledge, gamers will still achieve something that is normally only learn-able by joining the army. However, Simon Egenfeldt-Nielsen [1] states in a paper for the IT-University of Copenhagen that the gained knowledge is motivated extrinsic and not intrinsic. This means that the player does not want to learn directly but rather learns in passing. The result is that a basic knowledge may have been achieved but usually can not be applied in a slightly different context. Regarding the previous example this means that if some one successfully finished *Full Spectrum Warrior* or *America's Army* this person is not able to lead a team of soldiers in a real life situation.

Most video games do not claim that the player achieves an applicable knowledge but still are able to inform about the topic. The player does not need to qualify as a soldier or whatever the role is the video game sets after successfully finishing it. An educational game must not be seen as a stand-alone learning material but rather an addition to other types of teaching tools. Still, having only an educational game without other context grants enough information to motivate people to get familiar with the matter on their own.

2.4 Summary

Procedural generation and serious games have a similar background and therefore it is easy to see a combination of them. Urban planning on the other hand seems to have little in common with the other topics at first due to its non-technical background. The advancements of the information age though make it possible for departments to be multidisciplinary. Urban computing already uses digital databases and automated analyzes. By having

Figure 2.6: In *Full Spectrum Warrior* the player can position his soldiers strategically to keep them alive [51].

Figure 2.7: In *Amercia's Army* the player can communicate with other players to reach goals as a team [30].

these data openly attainable, the possibility of other implementations like an educational game are given. The next chapter names some tools already available that cover urban planning.

Chapter 3

State of the Art: Urban Planning Software

This chapter lists some examples that have included the technologies of the previous chapter, both the type of map creation and the goal of education. An obligatory requirement to be recognized in the list is to have the topic urban planning. Since the available information of map services is predetermined to geographical, demographical and similar data, covering not related topics is unrealistic.

3.1 Procedural Urban Planning

The first example is a procedural generator of pseudo-infinite cities created by the RMIT University in Melbourne [4] and based on a L-system. The tool uses random numbers in a certain range to create skyscrapers with diverse appearances and places them on a city grid. Every time the virtual city is generated, the shape and placement of buildings varies. It is called pseudo-infinite because the number of buildings is set by the user beforehand and is only limited by the hardware. Considering the explanation of procedural map generation in Section 2.2, the fixed values or rules are as follows:

- Minimum and maximum number of corners and angles of the shapes serve as floor plan.
- Number and height of floors are merged to form a three-dimensional object.
- Variety of textures cover the finished object.
- Grid of streets on the ground provide a space to place the finished building.

By choosing a value in the predefined range randomly, the virtual city simulates a pseudo arbitrariness of real cities. The term ‘pseudo arbitrariness’ is used as most laypeople do not see the complexity of a growing system

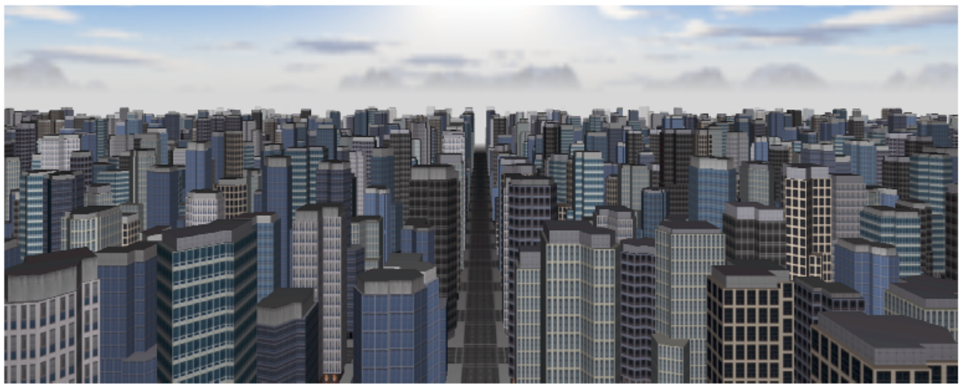


Figure 3.1: Result of a three-dimensional procedural generated city [37].

like that unlike architects who periodically add new structures to it. To simulate the process a procedural generator is required. It is only realistic to manually craft a certain amount of visually different buildings before the amount of work would increase exponentially. The result can be seen in Figure 3.1. The creators of this tool say that its purpose is to show possible problems that may occur. By creating nearly infinite cities, faults in the layout of larger cities can be found quickly and fixed early instead of trying to anticipate the growth and repair afterwards. To achieve this they state that other types of buildings must be added because the prototype currently includes only commercial skyscrapers. A possible usage for this tool is in the urban planning department to foresee traffic anomalies and possible quality of living of citizens.

Another procedural generation of an urban area is offered by 8bitcity.com, a website that transforms several large cities around the world to two-dimensional maps to look like a role play video game from 1980. An example of such a pixelated map is seen in Figure 3.2. According to Brett Camper, the author of the tool, the roads and parks are based on *OpenStreetMap*. Therefore the fixed information in this case is a given database with geographical information. Though it is not mentioned how the maps were generated, it can be assumed that they were only partially created procedurally. Indicators for this are the limited amount of cities and that only the finished abstraction but not the process of the generation is available. Nevertheless, the resulting image matches the requirements of a procedural generated map because the input (top-view of a city) differs from the output (abstracted map). This is achieved by applying a rule set that is flexible enough to work on multiple objects. Unlike the procedural generator of pseudo-infinite cities, an applicability in urban planning is not given. It rather provokes people to “take some time to think about [their] surroundings a little differently”, “set out on a quest” and “be an adventurer” [32].

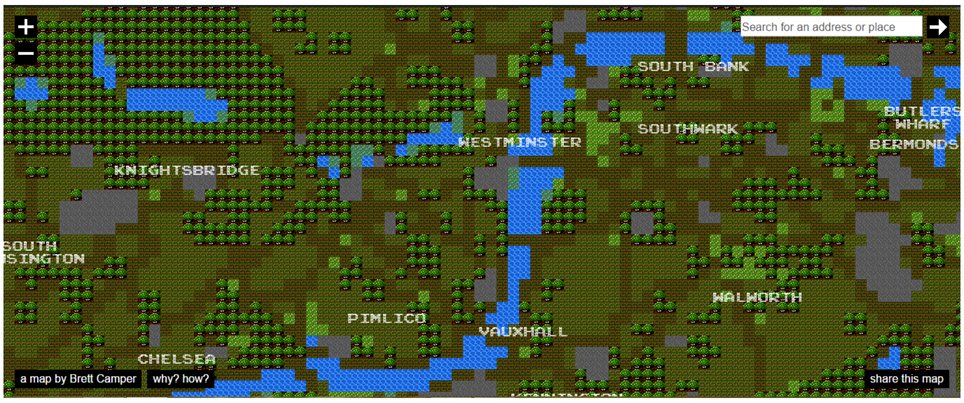


Figure 3.2: Pixelated map of London, Great Britain on 8bitcity.com [33].

3.2 Construction and Management Simulation

The next example is not one particular project but a genre of video games about urban planning. The genre that was defined by Ernest Adams, founder of the IGDA (International Game Developers Association), and Andrew Rollings [14] is called construction and management simulation, which is a sub-genre of simulations. Simulations, as the name implies, simulate certain areas of the real world and make it possible for gamers to experience situations that would otherwise be unavailable. A famous example for a simulation is the *Microsoft Flight Simulator* series where the player takes control of an aircraft and is required to get it from one destination to another [27]. The possibility of choosing from a variety of planes and being put in difficult situations such as landing on a runway with bad weather conditions without ever having to leave the home computer or being in danger makes the game and therefore the genre appealing. A construction and management simulation equally puts the player in the role of an urban planner that has the task to plan the layout of a city by carefully balancing its effectiveness and available resources.

The *Sim City* series made the genre popular and the second part, which is one of the most popular, can be seen in Figure 3.3 [22]. The image shows an already developed city. However, when the player starts a new game only an empty map with plains, hills, lakes and rivers is created procedurally. The game offers an interface as seen on the left part of the image with several tools to construct areas on the game map. The following list explains some of them briefly:

- The green button on the left with a house is used for planning residential areas.
- Right next to the green button is a blue button with a skyscraper on it that is used for constructing commercial areas.

- The red button on the right with a factory on it is for building industrial areas.
- Above the colored buttons are three icons with different transportation methods that are used to connect the areas via streets, railway tracks or ferries.
- The other buttons on the top half are for raising buildings that are important for infrastructure such as power plants, schools or fire departments.
- The buttons on the lower half are for navigation, game options and statistics about the city.

By drawing shapes on the empty map, the player only plans their approximate positions but doesn't construct anything. The game then proceeds to build on them in real-time. The result depends on the surroundings of the construction site. In the case of a residential area there will be small suburban houses or huge apartment complexes. There is a complex rule set that determines the factors of what buildings are going to appear but simply put it tries to simulate the real life. This means that residences near commercial or industrial zones are usually bigger than remote ones. The connection to electricity and fresh water influences is beneficial to the optic of the residence. As a result, areas with access to these residences are generally more beautiful. Additionally, the size of buildings influences the resources that are produced. Every shape or building the player places on the map costs resources which in the case of *Sim City 2000* is money. Based on real urban planning, commercial and industrial areas logically produce more money but to work properly they need a happy population. Otherwise people emigrate which results in a worker shortage. This balance between the needs of citizens and a steady income is the only mission as there is no final goal. By visualizing the growth of the player's city on the map, the motivation of the game derives from growing and expanding. These thereby occurring challenges can be compared to the job of an urban planner.

A recent game of the genre is *Cities: Skylines* by *Paradox Interactive* [25]. The available tools are similar to those of *Sim City* and are used the same way. The player drags shapes as seen in Figure 3.4 and, depending on the surrounding environment and the status of the whole city, the buildings are constructed automatically. Since *Cities: Skylines* was released in 2015, modern technologies such as off-shore wind farms are implemented as well. Additionally, the game displays urban problems not only via statistics but also on the game map. If for example too many people want to use a single-lane motorway exit slip road, a congestion will form and people switch to public transportation. If those are not sufficient enough, crowds will form at bus stops which results in a late arrival at work and therefore in a decrease in productivity.

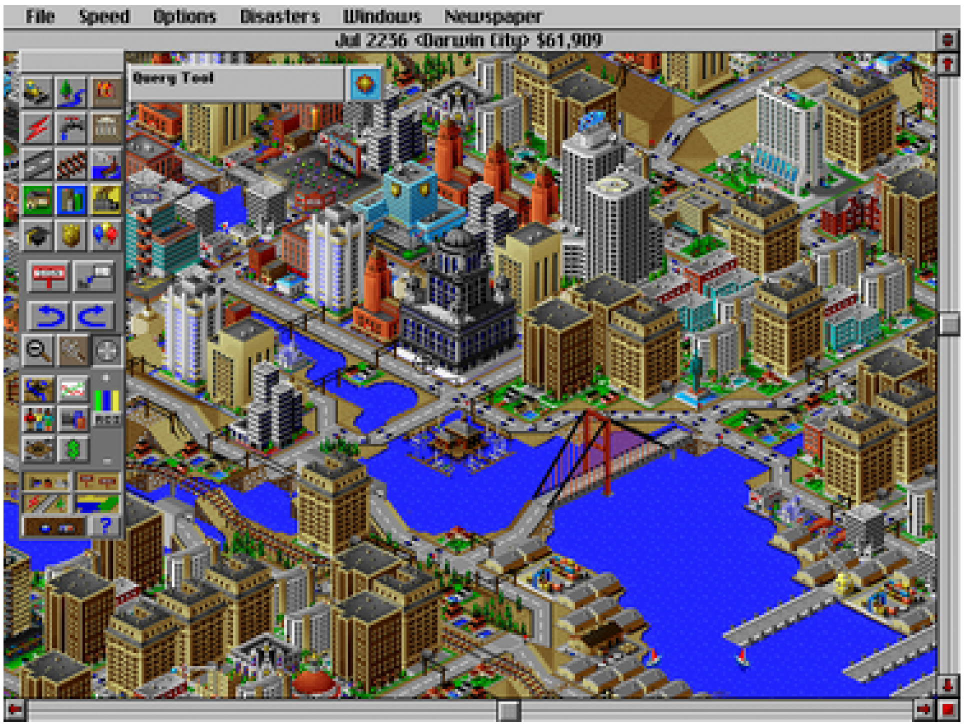


Figure 3.3: A screen shot *Sim City 2000* shows the interface and the already built structures of a city [50].

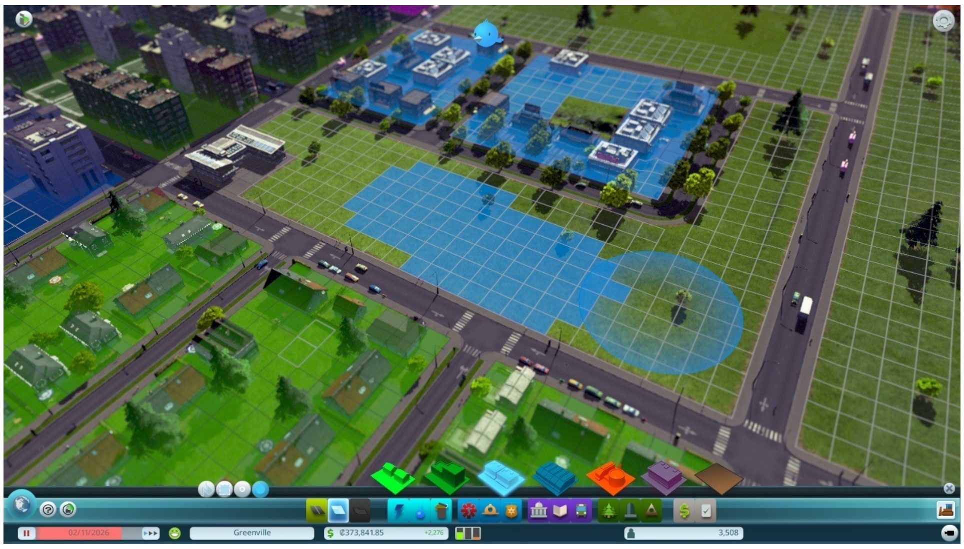


Figure 3.4: The tools given in *Cities: Skylines* let the player draw shapes of residential, commercial or industrial areas [38].

Another game that is worth mentioning in the construction and management simulation genre is *Civilization V* by *Firaxis Games* [21]. According to the Wisconsin Center for Education Research the game series, even though having an entertaining purpose, still lets the player "understand the operation of complex historical modelling" [18]. The difference to *Sim City* is that the player assumes control of the leader of a whole civilisation rather than only one city and therefore has to make decisions on a larger scale. But still, the cities must be managed. Each one needs and produces goods for the whole civilization. The production is done by placing workers on different adjacent tiles. The number of available workers depends on the size of the city. Each tile produces different resources which are either tools that accelerate the construction time of buildings, food to increase the city's population or money to pay the upkeep of the country's army. The interface for distributing these workers can be seen in Figure 3.5 as the hexagons represent the different terrains and the symbols on them their possible output. The buildings in the city modify these factors further but instead of drawing areas there is only one representative instance that grants for example the ability to store food. These city upgrades can also increase the scientific or cultural progress and improve the whole civilization. The gameplay of *Civilization V* has similarities to the two previous examples since an overall balance of food and other resources such as culture must be kept. The happiness of the people influences their productivity. If it falls below a certain value, the workers will go on strike. Referencing a problem in the real world, huge metropolises generate unhappy people automatically due to a lack of space. A contrary point to the other games is that *Civilization V* is round-based so its challenge is not to quickly respond to problems but to observe and process a larger amount of information. Also the game has a final goal which is either to destroy every other player or be the scientifically or culturally most advanced one.

3.3 Serious Games about Urban Planning

Alenka Poplin analyzed several serious games and how they can be utilized to educate about urban planning. Her motivation derives from the viability of the industry and the still lacking producers [11]. Citing Van Eck [16], a professor at the University of North Dakota, the branch was worth \$20 million in 2006 and is steadily growing. Furthermore the sector for education and training was estimated at \$2 trillion [15] in the same year and according to Poplin serious games will become increasingly important in it as well. A project Poplin worked on for the HCU (HafenCity Universität Hamburg) is *Bürger Beteiligung Billstedt (B3)*. The town Billstedt, a quarter of Hamburg, wanted to integrate its citizens in the decision of what to do with their central marketplace. The tool shows a three-dimensional model of the area

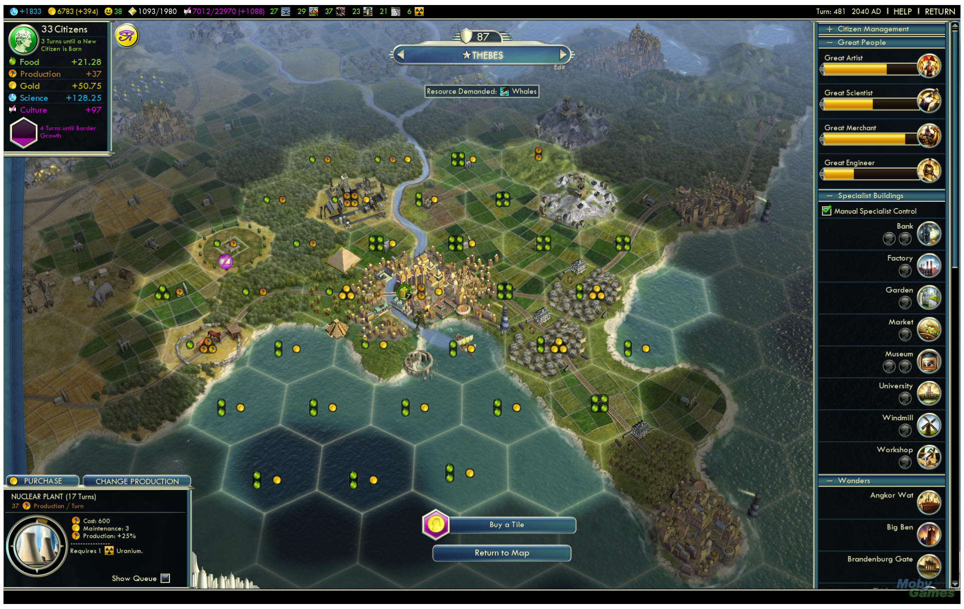


Figure 3.5: The detail view of the city Theben in *Sid Meier's Civilization V* with options to construct buildings and manage resources [43].

where the users can place objects. This is for example a playground as it can be seen in Figure 3.6. The finished versions can be sent to and rated by professional urban planners. The three best designs are published on the town's public website and are considered in the following planning process. Through a chat system the users can directly talk about their ideas with other players or consult an expert about possibilities and further plans of the town. The project's goal is to motivate citizens to participate in the planning of their living environment. Additionally the participants are able to achieve knowledge by having the opportunity to talk to experts and thus gaining detailed information on the subject urban planning.

A project quite similar to *B3* that was analyzed by Alenka Poplin is *NextCampus* [12]. The game's principle is identical but the goal is to integrate citizens of Hamburg and students in the planning of the HCU. The game provides tools to rate existing buildings and suggest the position and function of new ones. An existing building that can be rated by different groups is shown in Figure 3.7. The innovation and insight in urban planning is the available budget for the changes which is seen in the top left corner of the image. By visualizing the constraints of the working field, people are educated about the difficulties and decisions that are to be made.

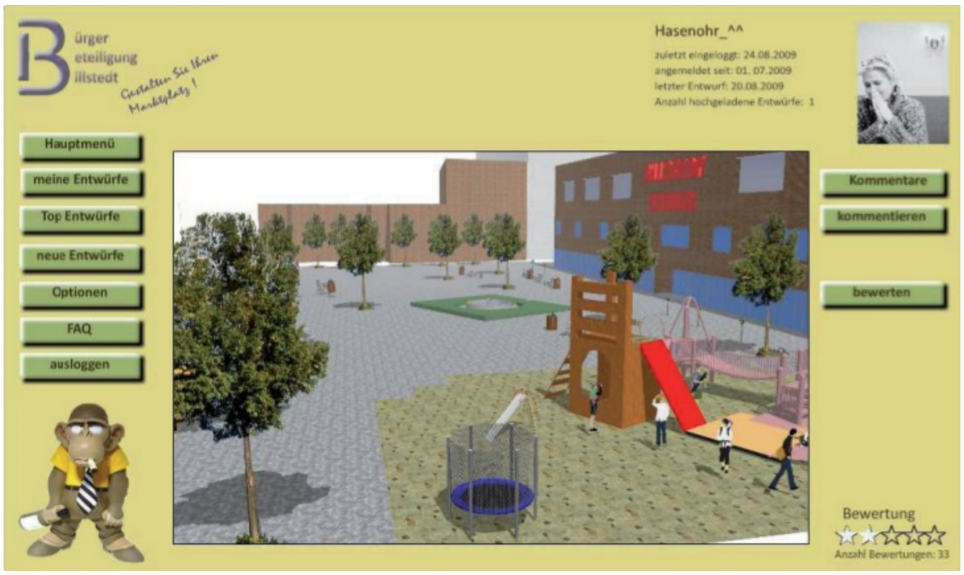
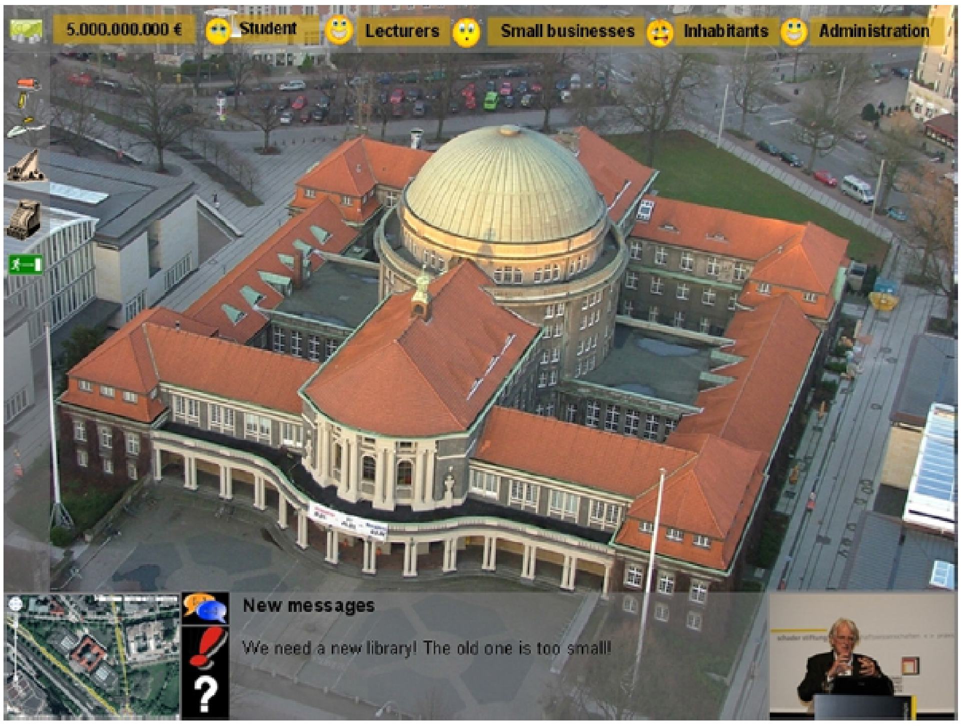


Figure 3.6: The interface of the serious game *B3* in which players can design a playground on a virtual central marketplace [48].



We need a new library! The old one is too small!

Figure 3.7: The user interface of *NextCampus* shows the possibility of rating the library of the HafenCity Universität Hamburg [49].

3.4 Summary

Comparing those projects, the field already offers many tools that simplify and teach urban planning. The departments already use procedural tools to simulate cities with educational or entertaining purpose and in the case of construction and management simulations they are accepted well by non-expert users. As mentioned by Egenfeldt-Nielsen [1] most people only play those games for fun and do not achieve real knowledge about the field. Although, by creating a relevant environment as in the projects of Alenka Poplin [11, 12], the motivation and the resulting educational value can increase significantly. The foundation of a tool that combines this relevance with game mechanics of urban planning simulations is presented in the next chapter.

Chapter 4

Conceptual Design of a Prototype

A tool that offers relevant information about urban planning to users has several requirements to work properly. The foundation is the learning material. The data is provided ideally by a web-based map service that has enough tags and information to ensure a diversity and recognizability of several different cities. This means that the user is able to generate a map of every desired area and to identify it based on the layout and unique structures. Furthermore the result must contain enough urban planning related data to qualify as a map for serious games. Assuming that the service provides enough data, the tool must fulfill certain criteria, content-wise as well as structural, in order to achieve a permanent learning success. This is sustained by an easily usable navigation to retain the users motivation. All priorly noted obstacles are examined in this chapter and result in a concept for an implementable prototype.

4.1 Technologies

The process of creating an openly available procedural tool requires several design decisions like possible technologies to harvest and visualize information. To be freely attainable by any person, the ideal medium for this software is the internet. The versatility of the world wide web demands an immediate accessibility from every major web browser. This excludes the necessity of installing additional plug-ins or other software.

The primary reason for designing a web-application rather than a native application is the need for an internet connection. Though software written in Java is able to collect information on the web while running on the desktop, the implementation is simpler and faster in a browser. A feature the tool provides is for example a search function to enable the user to chose any city for the serious game. To avoid the need of a large local database to gather the needed information it is reasonable to directly implement the search and result online. The latest and therefore ideal technology for web applications are the languages HTML5 and CSS3. Since over 50 percent of web-browsers used [44] are compatible to HTML5 and its drawing functions, it is suitable for visualizing the collected information. To communicate with a web service and thus gather the needed data set the most commonly used language is JavaScript. Additionally there are many open source libraries available for instance *leaflet* that is used for implementing modifiable maps.

Which technology is used for the tool also depends on the map service's interface in terms of exporting data in a readable and usable form. *OpenStreetMap* offers a complete set of their database in different formats, for example XML or PBF. Both of those languages are computer-readable and integrated in other web-applications. Due to a file size of about 30 to 40 gigabytes, assuming the average download rate is about 25 Mbps [45], requesting this amount of data for each user is unrealistic. To filter out only relevant information a map harvesting tool such as *Overpass Turbo* is required. It accepts statements in a built-in shell that result in complex search requests. The following code example shows a query written with the *Overpass* API:

```
[out:json];
{{geocodeArea:Vienna}}->.searchArea;
(node["tourism"="museum"](area.searchArea););
out;
```

The first line defines the format of the output. In this case the tool creates a JSON object that can be accessed by calling a procedurally generated link. The line is optional and the preset-output is a XML object. After that, the variable 'searchArea' is set to the predefined area of Vienna. This variable can later be used to only search in this zone. The third line determines the actual search query. *OpenStreetMap* has three different types of objects: `nodes`, `ways` and `relations` [46]. The `nodes` are the main elements and contain the latitude and longitude values and the object that are at that point. The `ways` are collections of nodes that describe larger geographical areas or lines of points. Finally, `relations` are elements that describe how multiple points may connect logically. In this case, the query shows all points that contain the tag `museum` in the category `tourism`. This line is extended by the previously set restrictions and therefore the only searched area is Vienna. The command in the last line creates an output of the request in a customizable layout. The result of this example is a JSON object with all museums tagged for touristic purpose in Vienna.

To fulfill the requirements of an openly available tool, the project is a web-application based on HTML5, CSS3 and JavaScript. To help the player search for an relevant area on the world map of *OpenStreetMap*, the tool *leaflet* is integrated as well. The desired location is then harvested and filtered by *Overpass* for information that are shown in the following section.

4.2 Content

The choice of map service primarily decides the availability and usefulness of urban information. *OpenStreetMap* is open source and thus freely modifiable by its users. This is done by simply tagging areas or places. Although, it is possible that there are currently no or wrong data for an area as anyone can add or edit them. As a consequence, certain problems with the map data can occur.

An open library that is structured like *Wikipedia* has its positive and negative aspects. A positive aspect is that the library has many contributors who receive no money and work much faster than a small team of content creators if they are organized. There are several automated processes in geographical analyzes such as finding structures via satellites and assigning them logically to places, but most meta information is still missing. It is very difficult to deter a building's purpose by only knowing its shape and location. Local citizens are able to identify them easily and therefore create a more detailed database. The consequential negative effect is the lack of source of information and without content moderation, opposing purposes of buildings can occur. There are for example several groups of trees in an area near houses and a citizen tags them as a recreational area because of the strollers that pass by regularly. Another citizen knows that a forester works in this area and the trees are used in a neighboring sawmill. Therefore he tags the trees as industrially used forest. Though both citizens are correct, a consistent database is not possible. *OpenStreetMap* handles the problem by providing a guideline for users that want to tag areas on the map. Similar tags are categorized into groups and the following list names some of them:

- `building`: Users are encouraged to use these tags for larger complexes such as an apartment block or cathedrals. Their purpose is to identify a building itself and not the complete institution. Hospitals for example consist of several buildings and other areas such as car parks or recreational facilities. The building in which the actual hospital is stationed is therefore tagged as `building:hospital`.
- `amenity`: This category contains institutions that are interesting to citizens and tourists alike. The tags vary from important infrastructures such as schools or hospitals, to small structures such as benches or public restrooms. All together, their purpose is public commercial and recreational areas. Contrary to `building`, it is used to define a complex of buildings with the same purpose.
- `power`: Similar to the amenities, this group contains areas of infrastructure. Its purpose though is not commercial but industrial. Example tags are power plants, cables or generators.
- `landuse`: These tags cover larger areas. Such as amenities contain separate buildings, a land-usage can have multiple amenities. For example

if several apartment blocks form an area, they can be tagged as residential. Every area that is used by humans but not necessarily developed such as agricultural fields falls in this category.

- `natural`: Just like `landuse`, these tags cover larger areas. The difference is that they are not used by humans. However, some of them may include parts that are used for recreational amenities such as beaches or trails.
- `highway, public_transport, railway`: Those three categories contain the means of traffic such as roads and railways. Everything that is important for transportation such as bus stops, railway stations or motorway accesses is listed here.
- `waterway`: In addition to travelling on roads or railways, there are also ways for vehicles on water. Rivers and channels are listed here. The reason this bullet is separated is that the objects do not need to qualify for transportation such as natural waterways that are not suitable for ships.

There are more categories but they are not as commonly used. The tags are not mandatory but rather a recommendation to maintain a well structured database. The guideline changed due to updates in the past and therefore the map is still not completely consistent but improving vastly. These categories are usable by *Overpass* as filters.

Another problem, databases of this size have, is missing information. Having data about every square meter of the surface of the earth is unrealistic and thus there are many areas in *OpenStreetMap* that have not been tagged by users. Since a relevant learning environment generated on the foundation of an incomplete database is problematic, a way to fill the gaps has yet to be found. There are three possible approaches:

- A place-holder fills the gap until a user defines the area. To sustain a consistent map undefined sections are replaced with other tags that are provided by *OpenStreetMap*. This completes the map until the missing information is added to the database.
- The missing information is generated but restricted by a rule set. This means that an algorithm decides what the most logical category of a certain tile is. For example if there is a tile framed by a lake, the tool assumes that it is lake as well.
- Other sources such as satellite pictures or the database of other map services are consulted such as *Google Maps*. They are then compared to the existing database and missing data is filled in.

Regarding the content of established games about urban planning in Section 3.2, the information that is useful for a serious game can be limited to four categories. To grant a recall value of locations, the `natural` and `waterway` tags are useful for representing geographical areas and `landuse` for representing urban structures. The games *Sim City* and *Cities: Skylines* use similar labels. Nevertheless, these approaches for missing data need to be tested in the implementation to determine their individual effectiveness.

4.3 Visualization

Another requirement for a sufficient tool is its interface and design. To allow a user to easily access the available learning material the software must provide a logical navigation and comprehensible visualization of the data. Following the definition of serious games in Section 2.3, a balance must be found between keeping the user motivated and teaching about urban planning. Since a user's motivation mostly derives from the entertaining value of video games, serious games often base their layout on them. The games closest to urban planning in this case are construction and management simulations that were presented in Section 3.2. All three games enjoy great popularity due to their game design and ease of use and all include a visual representation of the game world, a control panel to modify it and a display of available resources.

Although, being the oldest game on the list, *Sim City 2000* is widely regarded as the best of its series. Thus, mainly due to the technologies of the time, the graphical elements are two-dimensional images. The interface is an assembly of several flat boxes with shadows in order to appear bulky. The elements on the game map are fixated in a dimetric grid that is rotated and tilted by about 45 degrees. This enables the player to view the objects from the top and one of the corners and creates the illusion that the two-dimensional objects are three-dimensional. Logically the point of view can not be altered. The player can place objects only within the squares of the grid but adjacent tiles of the same type connect to larger shapes. The placement is restricted to only one tile per square except for tunnels or other underground structures. As shown in Figure 4.1, the game map uses most of the screen. The control panels to manipulate the game world are located at the left and the top side of the window. The top panel contains options for the program itself such as graphic settings and game speed as well as additional statistics and information about the current city. When clicked, they add new windows to the game map. Under that the resource, name of the city and in-game date are displayed. The date indicates the time proportional to an estimated time that would have passed in the real world. The left panel shows building tools and as every other additional window it can be dragged to every location on the game map which allows a customizable layout.

Cities: Skylines is the most recent construction and management simulation but does not change much in terms of game logic compared to *Sim City 2000*. Although, it offers improved graphics and a simplified layout. The

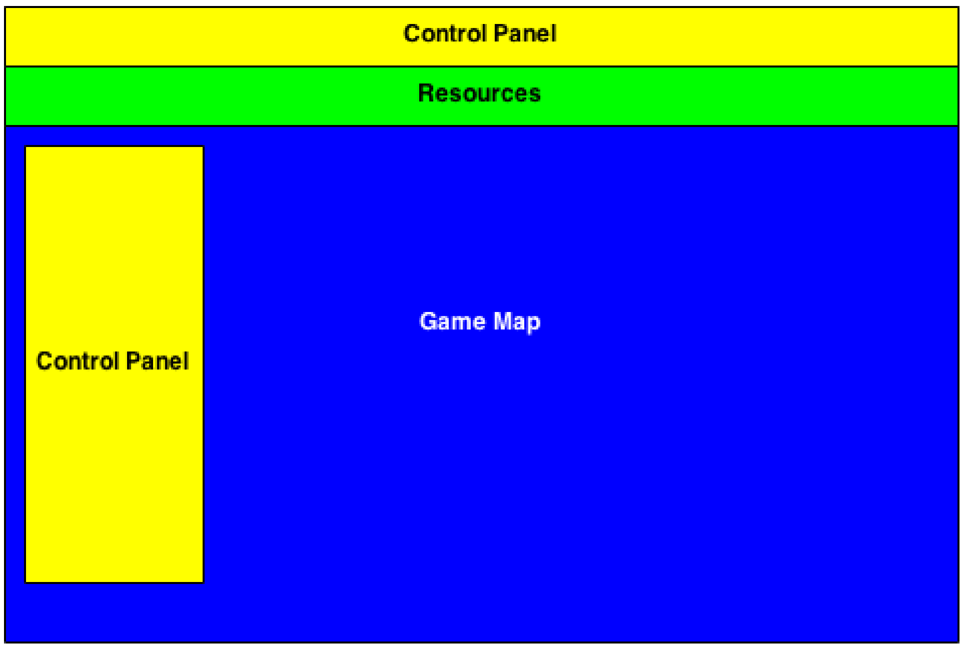


Figure 4.1: Basic layout of *Sim City 2000*.

game map and all objects are three-dimensional and therefore the player can zoom, rotate and tilt the point of view. Similar to its role model, areas can only be combinations of square tiles but instead of a fixed grid they can be attached to freely drawable streets. The map also allows multiple objects to be on top of each other. Figure 4.2 shows the basic interface of the game. The game map fills the screen almost completely and the interface is attached to its bottom. By clicking a button, the control panel extends to show further options or opens new drag-able windows. The bar below that shows, similar to Figure 4.1, the resources, the city's name and the game time.

The last example is *Civilization V*. Although not completely representative for the topic urban planning, it has the most elaborated interface due to its amount of available information. The game map and its objects are three-dimensional but still fixed to a grid. Unlike *Sim City 2000*, the single tiles are not square but hexagonal. The main reason for this is the military part of the game so the player can strategically place armies on the map. Nevertheless the city's interface is influenced by that as well. An abstract model of the standard view can be seen in Figure 4.3. Similar to the other games, the game map takes in the majority of the window and the interface and resources are pushed to the side. Since *Civilization V* has a lot of additional military, political and social information, it has a drop-down menu in the top right corner. Its bullets open additional windows with more details. The panel for creating objects is located at the bottom left corner and its

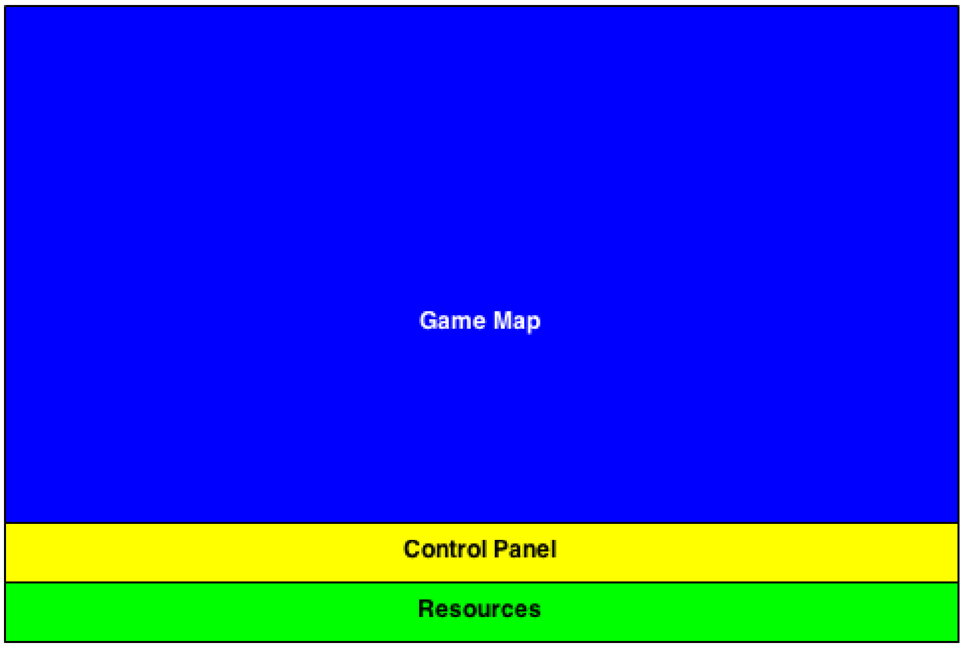


Figure 4.2: Basic layout of *Cities: Skylines*.

information depend on the tile that is currently selected. If a military unit is selected, the possible orders, and in the case of a city, the available buildings are shown. The overall resources are in the top left corner.

Due to the limitation of HTML5, the desired learning tool is two-dimensional and bases its appearance on *Sim City 2000* and *8BIT CITIES* mentioned in Section 3.1. To save space on the browser window, the game map is a non-dimetric grid. Comparing the different layouts with the requirements of the tool, a modification of the *Civilization V* interface fits best. There are different tiles on the map and by clicking on them the player should recognize and interact with them individually. A box in the corner of the screen that changes its information due to the tile that is clicked suits this purpose. Since the only resource to construct objects is money as in *Sim City* and *Cities: Skylines*, this interface item can be reduced. The resulting layout can be seen in Figure 4.4.

After determining the layout the next logical step is to implement the tool in the predefined environment. In summary, this is a web-application that generates an abstraction of a desired map excerpt. Its data derives from *OpenStreetMap* and the main technologies are *JavaScript* and *Overpass Turbo*. The following chapter contains an explanation of how this application works, what obstacles have emerged and how they have been solved.

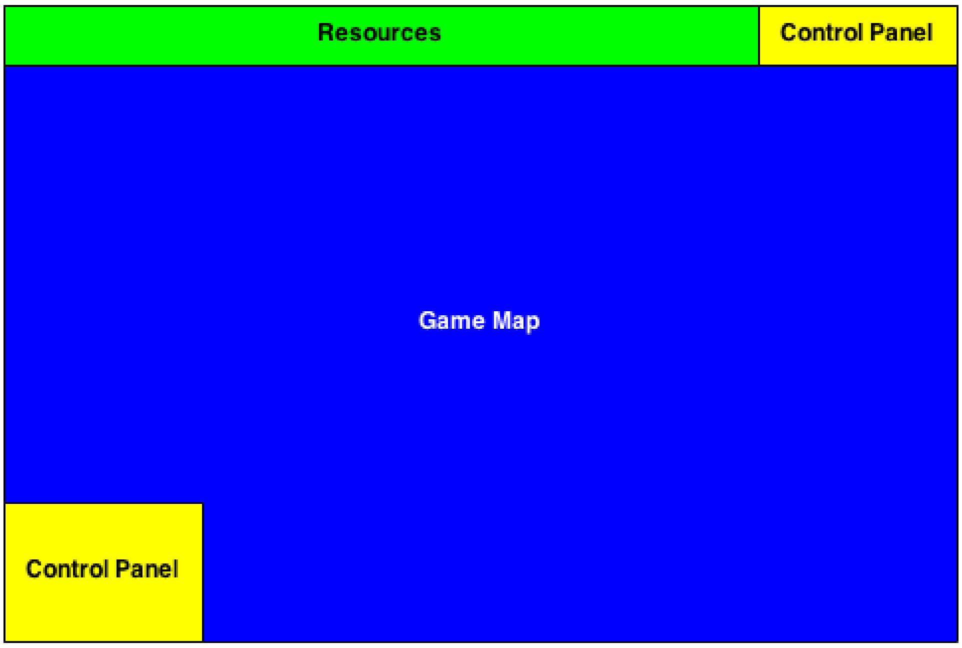


Figure 4.3: Basic layout of *Civilization V*.

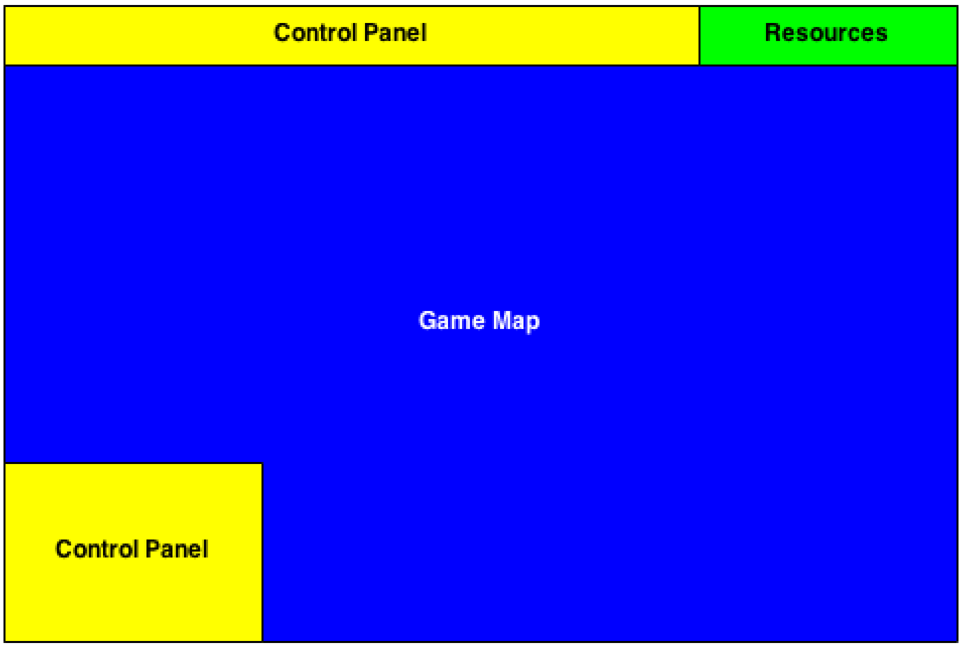


Figure 4.4: Resulting layout for a serious game about urban planning.

Chapter 5

Harvesting OpenStreetMap Data

This chapter examines the implementation of the concept discussed in Chapter 4. The necessary methods for harvesting data from *OpenStreetMap* and implementing an exemplary serious game are shown and described shortly. The given code examples are reduced to core functions to improve their comprehensibility. Problems like missing map data are listed and the applied solutions are presented.

5.1 Map Data Extraction

As previously mentioned, the foundation of a map generator is its given data set which is *OpenStreetMap* in this project. The tool for extracting these data as determined in Section 4.2 is *Overpass Turbo*. It provides the languages *Overpass XML* and *Overpass QL* to build search queries. These queries are transformed to a request link and subsequently deliver the data to a local file. JavaScript interprets the requests and saves the incoming information to an object and uses them for the map generation. As every procedural generation, an input is needed to retrieve the resulting map. The information of *OpenStreetMap* and therefore the queries of *Overpass Turbo* are based on geographical points with latitude and longitude values. That is why the user needs to provide such a point in advance. This is done by searching on an implemented *leaflet* map.

To extract *OpenStreetMap* data, a search query needs to be created which filters and transforms the information to a readable form. The following code is written in *Overpass QL* and tested on overpass-turbo.eu. The chosen language has no influence on the resulting data. Furthermore it only represents the request for one location of the map.

The actual search query is seen in Program 5.1 from line 3 to 10 between the parentheses. Since the objects that are to be harvested are geometrical

Program 5.1: A query in *Overpass* that requests several tags.

```
 1 [out:json][timeout:25];
 2 (
 3 way(around: 100, 48.306940, 14.285830)['name']['landuse'];
 4 way(around: 100, 48.306940, 14.285830)['landuse'];
 5 way(around: 100, 48.306940, 14.285830)['name']['natural'];
 6 way(around: 100, 48.306940, 14.285830)['natural'];
 7 way(around: 100, 48.306940, 14.285830)['name']['waterway'];
 8 way(around: 100, 48.306940, 14.285830)['waterway'];
 9 way(around: 100, 48.306940, 14.285830)['name']['leisure'];
10 way(around: 100, 48.306940, 14.285830)['leisure'];
11 way(around: 50, 48.306940, 14.285830)['name']['highway'];
12 );
13 out tags qt;>;
```

shapes and not only single points on a map, the `way` tag needs to be searched for. The area has a geographical limitation of being inside a 100 meter radius around a point with a latitude of 48.306940 and a longitude of 14.285830. This is approximately the center of Linz, Austria. Furthermore, the radius defines the granularity or level of detail and size of the resulting map. By reducing this value, the map excerpt becomes more detailed but smaller and likewise by increasing it, the excerpt logically gets larger but less detailed.

The content-related limitation is in the square brackets and defines the tag that the query searches for. In this case, *Overpass* harvests areas that are tagged with `landuse`, `natural`, `waterway`, `leisure` or `highway`. The `name` term in front of half of the tags adds tags that are additionally described with a name. The first and the last line of the code are defining the output. Line 1 says that the object's format is JSON and the request may not take longer than 25 seconds or otherwise is aborted. Line 13 filters the result for all tags because this is the only information that is needed for the map generation. The `qt` orders the results ascending after their quadtile id. This translates roughly to their distance to the searched point. This is added because there can be multiple relevant areas near the location and only the closest should be added to the generated map to fulfill the requirement of being identifiable by the user. Although the importance is altered by the generator by prioritizing rivers and streets as those are commonly the most distinctive features of a city. A possible response of the *OpenStreetMap* server can be seen in the following example:

```
"elements": [{
  "type": "way",
  "id": 143291881,
  "tags": {
    "landuse": "commercial"
}]
```

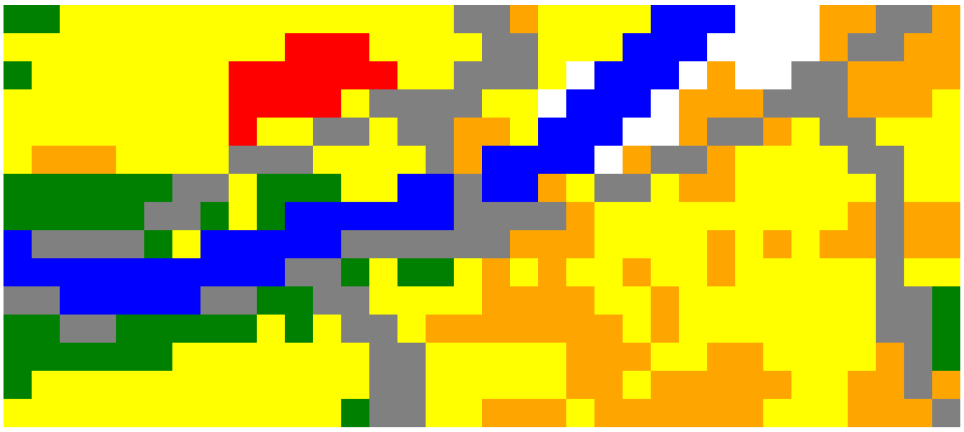


Figure 5.1: A generated preview map of Linz, Austria with each square representing 100 square meter.

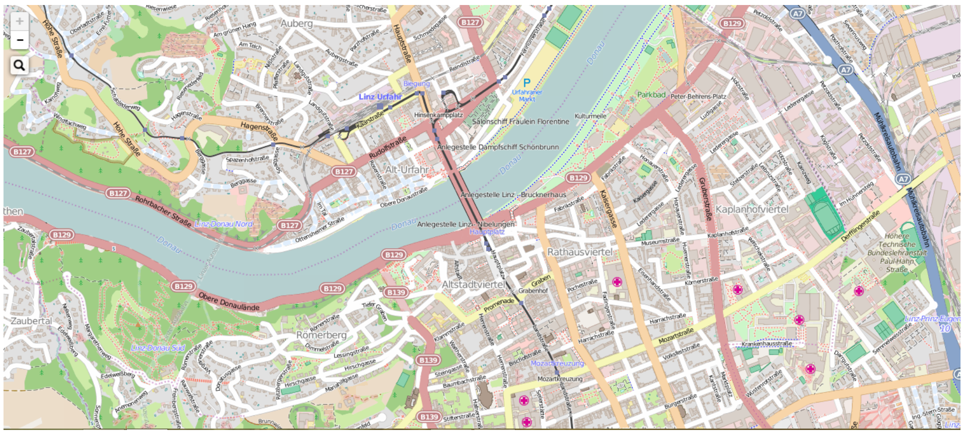


Figure 5.2: Linz, Austria on *OpenStreetMap* and viewed with *leaflet*.

Since the latitude and longitude in the query are fixed values, the transformed request in the local JavaScript has to be modified to create larger, more diverse maps. A loop changes the values for every request so that a two-dimensional array is filled with the respective information of *OpenStreetMap*. To visualize the array and preview the map without any further game logic each tag has a different color assigned. The colors are based on *Sim City* and an example preview of Linz, Austria is seen in Figure 5.1.

By comparing the generated excerpts to the map of *OpenStreetMap* in Figure 5.2, structures are already recognizable such as the blue tiles serving as placeholder for the river Danube in the center. The other tiles are yellow residential areas, orange commercial areas, red industrial areas, green forests and grey streets. White tiles determine that the query has not found any information around the location matching the given criteria. Furthermore, it

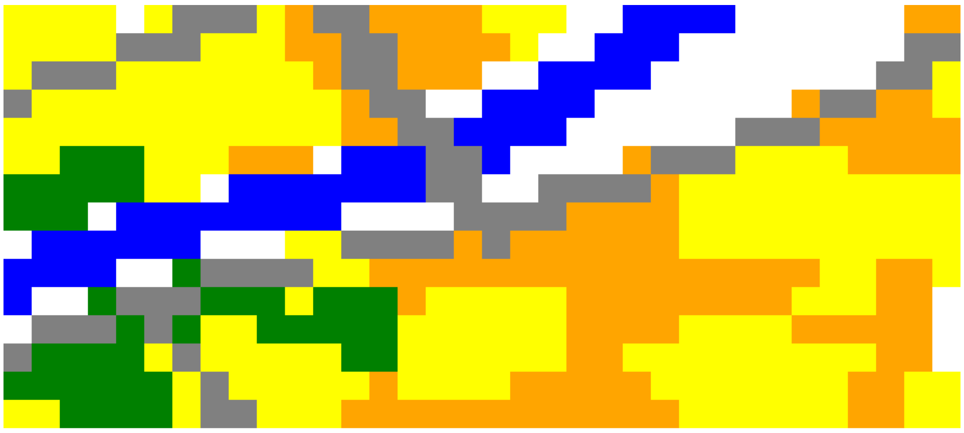


Figure 5.3: Linz, Austria with a high granularity with each square representing 50 square meter contains more details but the complete map is rather small in comparison to Figure 5.1. Furthermore areas where there are no tags like the excerpt in upper right corner below the river are brought out.

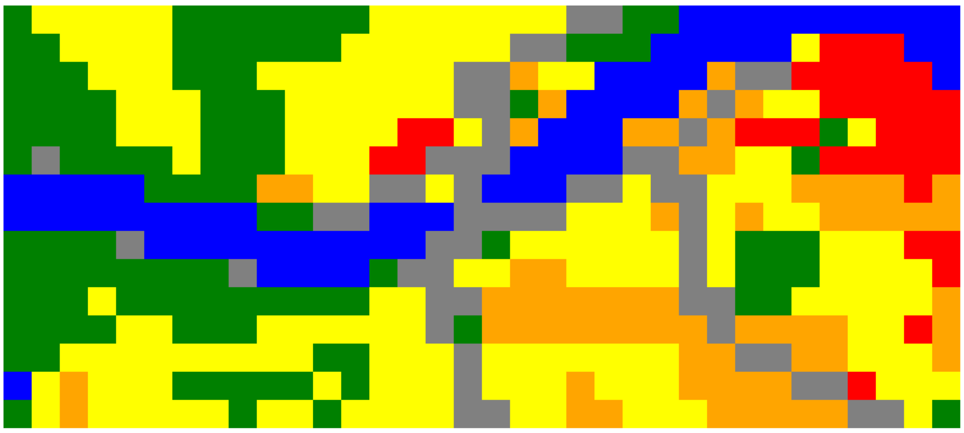


Figure 5.4: Linz, Austria with a low granularity with each square representing 200 square meter covers a larger area but the information are less precise in comparison to Figure 5.1. An example for this are the missing streets on the left side of the map.

was mentioned that one tile represents 100 square meter of the actual map. This decision is based on a comparison of higher and lower granularities and the resulting maps are seen in Figures 5.3–5.4.

5.2 Serious Game Based on Map Data

To test the map generator for its usefulness as foundation for serious games, an example has to be implemented. The game's requirements are based on

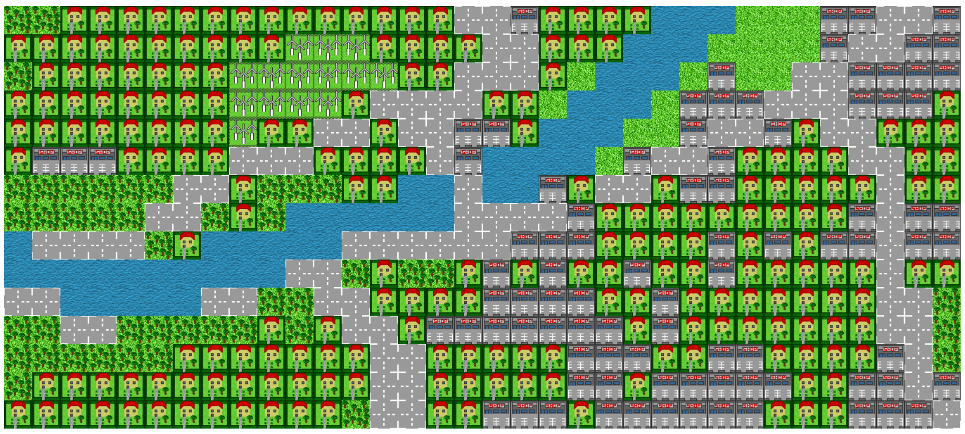


Figure 5.5: The generated game map of Linz, Austria overlayed with the textures of *Urban Space Upgrader*.

the properties predefined in Section 2.3. This includes a learning benefit for the user, a motivating game mechanic and a well structured interface that combines the entertaining and educational purpose.

The game that serves as an example for the features the map generator provides is called *Urban Space Upgrader*. It is two-dimensional due to the two-dimensional map array of the generator. The colored tiles are replaced by icons representing the respective area as seen in Figure 5.5. The goal of the game is to stabilize the three resources (money, health and happiness) that are influenced by the production values of tiles of the generated map (taxes, energy and pollution). Each tile has a negative and a positive output as they are listed as follows:

- Residential areas pay taxes and therefore increase the overall money directly. On the negative side, houses need some energy and produce pollution.
- Commercial buildings pay taxes as well, but to differ from residential houses, they produce more pollution and need more energy. To retain a balance to houses, they increase the money generation effect of their surroundings.
- Industrial tiles produce energy but also pollution. The maintenance of them costs money.
- Forests cost money but they reduce the overall pollution. Logically, they do not influence the energy.
- Water, streets and empty tiles do not produce or cost any resources.

By clicking on any of these tiles on the map, a detail view of them is opened. This window lists their individual production values, their type and other information. To improve the recognizability of the city the name of

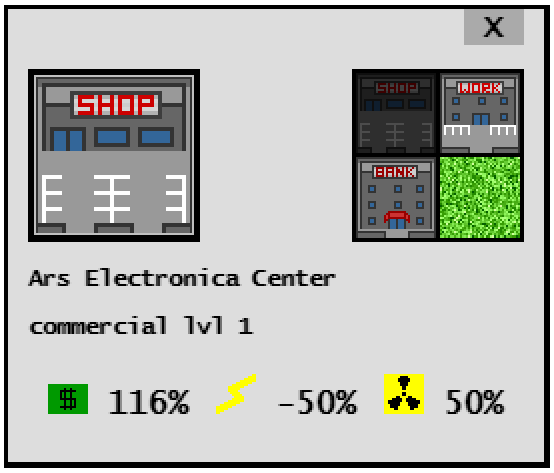


Figure 5.6: A detail view of a tile in *Urban Space Upgrader*.

the area is listed here as well. An example of this detail view can be seen in Figure 5.6.

The overall resources the player has to balance are influenced by these outputs. Money is always directly influenced by taxes so if something produces money, it adds and if something costs, it subtracts from it. Health is a maximum value of 100 minus the total pollution the tiles generate. This results in the objective of balancing forests with commercial and industrial areas. The happiness of the citizens is calculated by the overall health plus the available energy. Therefore a city with many industrial tiles has much available energy and therefore the happiness is increased. Although at the same time the pollution decreases the value again. After a predetermined timespan the resources are updated and the player can see the impact of modifications on the city.

As the name of the game implies, the modifications the player can perform are upgrades to tiles. Every tile has three levels that influence their visualization and production values. The different levels for each tile are shown in Figure 5.7. The rows represent the level whereas the first contains every level one tile and the last one every level three tile. Each column represents the different types of tiles with residential in the first, commercial in the second, industrial in the third and forest in the fourth one. The general rule is that the higher the level is, the higher are the outputs of a tile. For example a level one residential area is represented as a single house with a garden and generates and consumes a regular amount of money and energy. When upgraded, the house will transform to a row house and doubles its costs and productions. Similar, the last level is an apartment block and has three times the output. Every upgrade costs money and by clicking on an upgrade on the detail view of the tile as seen on the right side of Figure 5.6

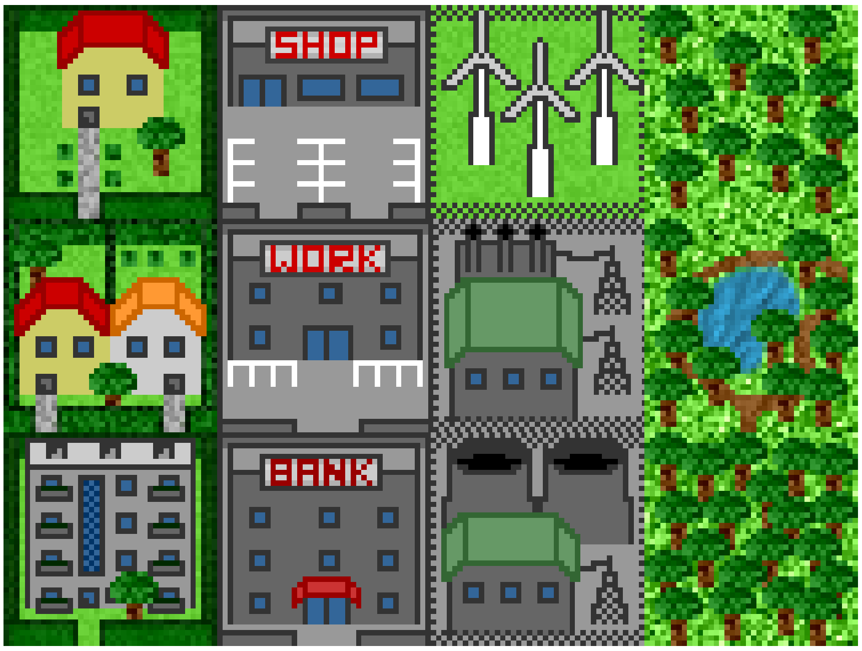


Figure 5.7: The different levels of tiles in *Urban Space Upgrader*.

it is replaced by a construction site without any productions or costs until the next resource update.

The tile upgrades do not only influence their own production but those of surrounding tiles as well. Commercial, industrial and forest areas each have positive and negative effects on their surroundings. Industrial areas for example increase the pollution but decrease the energy consumption of each tile that is in a certain radius around them. The strength of the effect is based on the level of the influencing tile and the distance of the influenced one to it.

5.3 Development Challenges

Working with an external API (application programming interface) often causes unexpected problems that derive from incorrect information in the database. As mentioned in Section 4.2, the data of *OpenStreetMap* derives from a large number of various contributors and therefore results in a non-preventable inconsistency. Obstacles are common and need a working solution to fulfill the predefined requirements of the project.

5.3.1 Loading Time

The generation of the map is procedural and therefore takes time to finish. Since the information is dependent on the response speed of *OpenStreetMap*, the tool does not entirely account for an appropriate loading time. Appropriate in this case refers to the loading time of other web applications, especially browser games regarding their structural similarities to this project.

The map size depends on the window size of the browser used and the size of one tile is defined as a square with a length of 40 pixels. Each of those tiles is requested by the client and it takes *OverpassTurbo* several seconds to respond. Bigger resolutions with over 500 tiles therefore result in loading times up to 15 minutes. This is disproportionately high for a browser game. The reason for this is the `round` statement of the search query in *Overpass*, which is known to be slow because it has to search every point inside the radius. Possible solution approaches are as follows:

- The `round` statement is changed to `is_in` and returns the information of an exact point at the given latitude and longitude coordinates. Since the query only searches at one location rather than each one inside a radius around it, the time to process the request decreases significantly.
- Generalizing the query in terms of applying fewer filters decreases the response time as well. The less information *Overpass* has to return, the faster the response is sent. Instead, the filters are implemented in the JavaScript code. Outsourcing the calculations to the project's server has the advantage of being faster due to less traffic.
- The last approach does not focus on the actual loading time of the request but accelerates the map visualization. By enabling the user to save the created maps as JSON files and load them later on, the game can be started faster. Several maps are preloaded so the player has the possibility to either use them or wait for his own to be created.

The only approach not applicable on this list is the first bullet. `is_in` is faster than `round` though the resulting output is not satisfying in terms of recognizability. Since the query only searches one location, most of the tiles are empty. Latitude and longitude are very precise specifications of a geographical location. The distance of a location to the next is set to 100 meters and therefore the radius of the `around` query is accordingly 100 meters. By replacing it with `is_in` only one small space every 100 meters is analyzed. This may result for example in a commercial area 5 meters next to the point searched to not being considered in the search query. The response would consequentially contain no relevant information and tag the tile as empty. To counteract this problem of many empty tiles, the distance between the steps has to be reduced. Even tough this would improve the accuracy of the tool finding the correct tile, the time the *OpenStreetMap* server needs to respond is disproportionately higher than before due to the

Table 5.1: Comparison of loading times in milliseconds with different queries.

`way(around: 100, lat, lon)`	**Loading Time**
`['landuse'='residential'];`	9759.64
`['landuse'];`	2618.94
`[];`	1807.70

increased amount of request.

The code shown in Program 5.1 is already reduced to a certain degree. By searching for all areas with `landuse`, the time the server needs to process the query is shorter than with `landuse=residential`, `landuse=commercial` and `landuse=industrial` as there are less restrictions and therefore less calculations for *Overpass*. The necessary filtering for relevant information happens on the application side. It is even possible to simplify the query more by disable any filters and therefore request every tag of a location. A comparison of the three different requests can be seen in Table 5.1. Each response time is the result of the average of ten sequential requests to intercept abnormal times due to high traffic on the server side. The test is done in *Google Chrome* version 43.0. The comparison affirms the already expected result that it is faster to apply all filters after requesting the needed information.

By loading maps themselves and saving them locally, the users still have a longer waiting time in the beginning compared to using a pre-loaded one. Although they are then able to start a game with their desired map much faster later on. The generated maps are two-dimensional arrays and therefore can be easily transformed into JSON objects within JavaScript and saved as a local file. With a file uploader the JSON files are loaded into the application where they are parsed to a JavaScript object again and can be used by the game. The file size is depended on the size of the created map but usually does not exceed 100 kilobytes.

5.3.2 Missing Information

As mentioned in Section 4.2, the database of *OpenStreetMap* has vacant spaces due to its user-based content generation. To fulfill the requirements of maps being recognizable and maintain consistency, the missing information needs to be filled in. The tested approaches are a next neighbor algorithm, a predetermined rule set and a game logic approach where the tiles are left empty. They are explained in detail and evaluated in regard of several criteria:

- A heterogeneity must be given. That means, in contrary to homogeneity, the distribution and variety of different types of areas must be diverse enough for the abstraction of the city to be credible. Since there is no strict separation of residential and commercial areas in most cities but rather a conjunction between them, an abstract visualization has to capture and represent this as well.

- As mentioned before, the map should maintain a certain recognizability. For the learning effect to be efficient the chosen map excerpt is ideally as close to the expected outcome as possible. By identifying areas and structures correctly the impact and influence of areas on each other is clearer than with a fictional map. For example a large park at the center of the map should be recognized as such within the abstraction so the user knows its importance for the whole city.
- The realism is closely connected to the recognizability because the player expects that the tool correctly visualizes the area that was chosen. The differentiation of realism is to simulate authenticity of areas that are not known. This means that even if the user has not seen the area before, it must be as believable as possible based on already existing experiences on other cities. A large industrial area for example is usually not surrounded by many recreational parks or residential areas and if the abstraction would present the map in this way, it loses credibility.
- The abstraction also needs to provide an applicability as game map. Since the tool to create an abstract map of an area is only the basis of additional educational software, it has to fulfill some requirements. Firstly, this includes a consistent usage of tags. This means that even missing information are tagged as such because the possibility of using empty tiles must be given to tools. Though, the number of empty tiles is restricted to maintain enough interactivity with the map. Another important point is the difficulty of the game to prevent a loss of motivation if it is too hard in the beginning or too easy later on. These points are tested on the exemplary serious game *Urban Space Upgrader*. It should be mentioned that the result varies depending on the game that uses the map generation. Therefore it may change if another tool is analyzed.

The criteria derive from a grounded theory approach. This means that they are not based on existing evaluations but on a discussion resulting in questions about how to measure the performance of the different approaches. This research methodology is necessary because the field of generating abstractions for a serious game based on an open source map database is not covered explicitly. The three suggested solutions are tested respectively on maps of Salzburg, Munich and London. The cities are chosen due to their geographical distance, amount of missing information and differences in available information. Cities with few empty tiles such as Linz in Figure 5.1 are avoided due to insufficient distinction between the approaches.

The first approach is a ‘next neighbor’ algorithm. The method for deciding the appropriate type of area for an empty tile is based on the importance and distance of already available information. The distance is calculated by regarding every non-empty for every empty tile on the grid. Since every tile in the grid has a `x` and `y` coordinate, the distance between them can be calculated as following:

```
var distance = Math.abs(origin_x - x) + Math.abs(origin_y - y);
```

`origin_x` and `origin_y` are the coordinates of the empty tile and `x` and `y` are accordingly the coordinates of the non-empty tiles to which the distance is calculated. To prevent negative distances `Math.abs()` is used to only return absolute values. The importance of a non-empty tile is determined by the number of same tiles it is surrounded by. For example a residential area with eight neighboring forests has the importance 1. If the area has four other residential areas and four forests around it, the importance of the tile is 1 plus each same tile, which adds up to 5. Logically, a residential area surrounded by eight other residential areas has an importance of 9. The combination of distance and importance results in the following factor for each non-empty tile:

```
var factor = importance / distance;
```

Each factor is compared and the empty tile is then filled with the highest one. The algorithm is loosely based on how a statistical tool of the geographic information system *ArcGIS* calculates the average nearest neighbors of areas in grids [36]. The results of the algorithm used on the exemplary cities are seen in Figures 5.9, 5.12 and 5.15. Considering all three examples and evaluating them after the predefined criteria results in following performances:

- Heterogeneity: The already existing areas are extended to bigger clots and may connect to other areas. This is more homogeneous than heterogeneous, since the map consists of a few large areas with the same type and not many smaller ones. In other words, a variety of many different types of areas is not given for most maps. At least in some cases the clunks are not too big.
- Recognizability: By filling tiles with neighboring areas, the recognizability of cities is often given. Though this mostly derives from rivers or streets, the abstractions of Munich and Salzburg are quite close to the real structures. Nevertheless, the algorithm performs poorly with London, where the bottom right corner is entirely forest due to the large parks around it.
- Realism: The credibility for the next-neighbor method depends, as with the recognizability, on the city. People that have not seen any of the exemplary cities are probably believing that Munich and Salzburg are correctly represented by the abstraction and do not believe that

London has this many forests at one place. Although it is possible for residential areas to be this large, it is rather unusual for European cities.
- Applicability: The exemplary game can transform every tile on the map to a already costing and resource generating element, which is maybe good for other applications but rather bad for this one, since the initial difficulty is very high. There are many tiles of the same type. To maintain a proper balance, the player has to make many changes to the map immediately to prevent a quick decrease of the main resources.

The second approach uses a determined rule set to decide the type an empty tile is filled with. Instead of using the distance and importance of non-empty tiles this algorithm only considers the direct neighbors of empty ones. Unlike the importance of the next-neighbor algorithm, multiple types can have influence on it. Although the surrounding only determines two possible types and their probability. An additional random factor finally decides the filling of the empty tile. The following list contains the requirements of the vicinity, the resulting group of available types and their probability:

- The surrounding contains one or more residential and one or more commercial areas. This results in 40% commercial or 60% residential area for the empty tile.
- The surrounding contains one or more residential areas and one or more forests. This results in 50% residential area or 50% forest for the empty tile.
- The surrounding contains two or more industrial areas. This results in 50% commercial or 50% industrial area for the empty tile.
- The surrounding contains one or more commercial and one or more industrial areas. This results in 40% commercial or 60% residential area for the empty tile.
- The surrounding contains one or more commercial and one or more water areas. This results in 50% residential or 50% commercial area for the empty tile.
- The surrounding contains two or more residential areas. This results in 40% residential area or 60% forest for the empty tile.
- The surrounding contains two or more water areas. This results in 30% residential area or 70% forest for the empty tile.

Because of the more restrictive algorithm, the rules have to be applied several times until there are no more changes to the map. Otherwise only the neighbors of non-empty tiles would be filled in. In some cases multiple iteration do not result in a entirely filled map which is why an additional algorithm has to be executed afterwards. Similar to the first instance, the tile is influenced by already existing types and a random variable to enable a probability distribution within a group. However, the group is not influenced by their surroundings but the total number of types on the map. Additionally the group contains three to four possible types instead of two:

- If residential areas have the highest total number, an empty tile results in 20% forest, 30% commercial or 50% residential area.
- If commercial areas have the highest total number, an empty tile results in 10% forest, 10% industrial, 40% residential or 40% commercial area.
- If industrial areas have the highest total number, an empty tile results in 20% industrial, 40% commercial or 40% residential area.
- If forests have the highest total number, an empty tile results in 10% commercial, 40% residential area or 50% forest.
- If streets have the highest total number, an empty tile results in 10% industrial, 40% residential or 50% commercial area.
- If water tiles have the highest total number, an empty tile results in 10% commercial, 40% residential area or 50% forest.
- If empty tiles have the highest total number, an empty tile results in 25% forest, 25% residential, 25% commercial or 25% industrial area.

The results of these algorithms used on the exemplary cities are seen in Figures 5.10, 5.13 and 5.16. The rules are based on observations and comparisons of different rural and urban areas around Europe. Other continents are less productive due to their lower density of information in *OpenStreetMap*. Interviews with urban planners have also shown that there is no general rule set cities are following so the algorithm tries to simulate the complexity of urban structures by adding chance to the decision process. By evaluating it, based on the predefined criteria, its effectiveness can be measured:

- Heterogeneity: Due to the probability distribution of type decision, the maps have a high heterogeneity. The larger clunks remain, but the empty tiles are filled with various smaller areas up to single tiles.
- Recognizability: In contrary to the first approach, the algorithm often chooses wrong tiles for cities. Though it performs better with London, where the bottom right corner is filled with at least some residential and commercial tiles, it works worse on others. Munich, as well as Salzburg gets many forest tiles around its center which is not entirely true.
- Realism: Since people who do not know the cities often compare them to ones they know structure-wise, this approach performs better than the next-neighbor algorithm. It is believable that every city has some forest or parks among large residential areas and that London may have a lot of parks but still some other urban areas among them.
- Applicability: Comparing the finished map to the previous approach, every tile is immediately changed to produce and cost resources. Al-

though the start of the game is easier than one with the next-neighbor approach since the overall distribution of different type is more balanced in all examples. This improves the applicability for *Urban Space Upgrader* since it is still very challenging when there are many critical areas to improve right off the start.

The last idea to handle the empty tiles is to integrate them into the game. The player can decide what tiles are suited for the currently missing information by upgrading empty fields to a residential, commercial, industrial or forest area. The player is also able to remove tiles that contain information to rebuild parts to improve the city's efficiency. To include the mechanic fluently into the game the removal of resources generates and the upgrading of empty tiles costs money. In comparison to the approaches of filling the gaps, the maps without further modification is seen in Figures 5.8, 5.11 and 5.14. Though no algorithm was used to fill the map, the approach is measured on the given criteria:

- Heterogeneity: Since the heterogeneity is based on the given map in this case, large empty clunks remain, where no information is given. Though it depends on the city, this approach has the worst heterogeneity in comparison.
- Recognizability: This criteria depends on the chosen city as well. Rather than filling the map with wrong information, this approach does not do anything, so the recognizability does neither increase or decrease. Though the user is likely to be less judgemental on missing information, since the seen map is not wrong at least. Although this improvement is rather subjective.
- Realism: This criteria logically only achieves the worst performance since the exemplary cities do not have empty areas just because the city has not been tagged yet entirely.
- Applicability: Regarding the usability of the map for serious games, the approach of leaving the tiles empty works quite well in the case of the exemplary *Urban Space Upgrader*. Since the player can upgrade every empty tile to the desired type it increases the immersion into the game. The player is able to rebuild a city or make changes to it as desired. The level of difficulty starts low and increases as empty tiles do not cost anything and the player can focus on critical areas faster.

Regarding the resulting marks of the abstractions of Salzburg, Munich and London, an ideal solution cannot be identified. Nevertheless, each approach has their advantages and disadvantages. The next chapter confronts each approach based on its performance thus deciding on the most useful algorithm. Additionally, possible improvements are presented and combinations of them are considered to lessen the problem of missing information in *OpenStreetMap*.

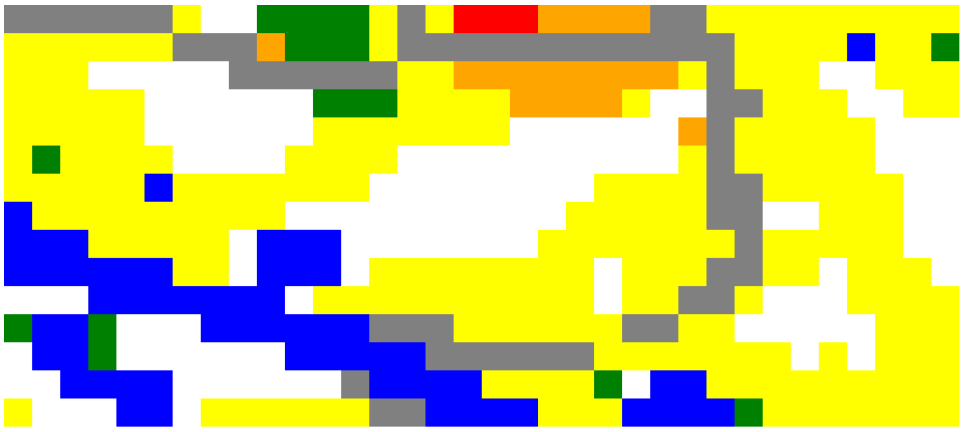


Figure 5.8: A generated preview map of Salzburg, Austria.

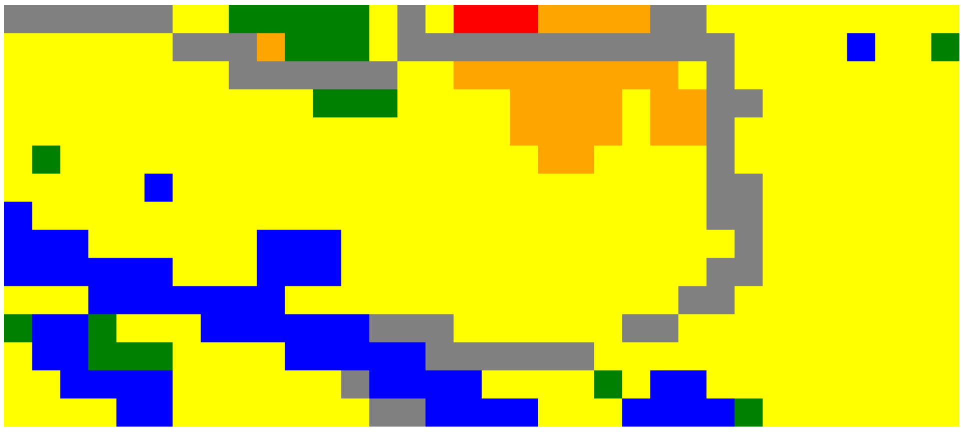


Figure 5.9: A generated preview map of Salzburg, Austria filled with a next neighbor algorithm.

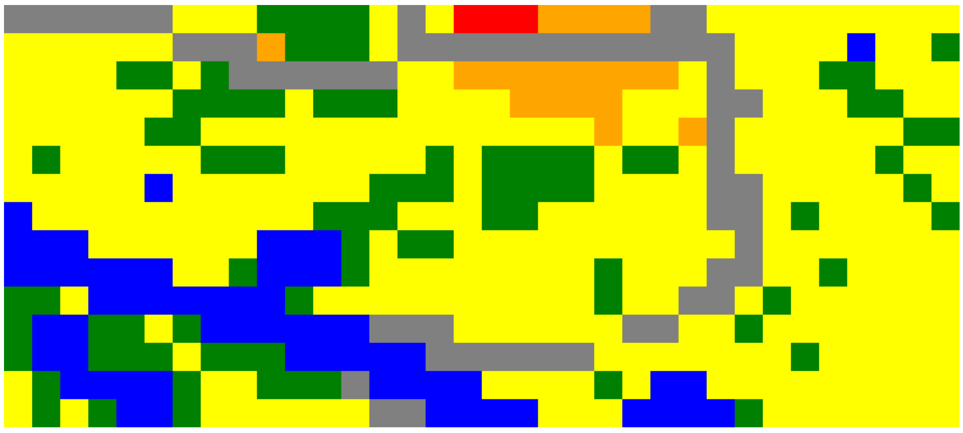


Figure 5.10: A generated preview map of Salzburg, Austria filled with a rule set algorithm.

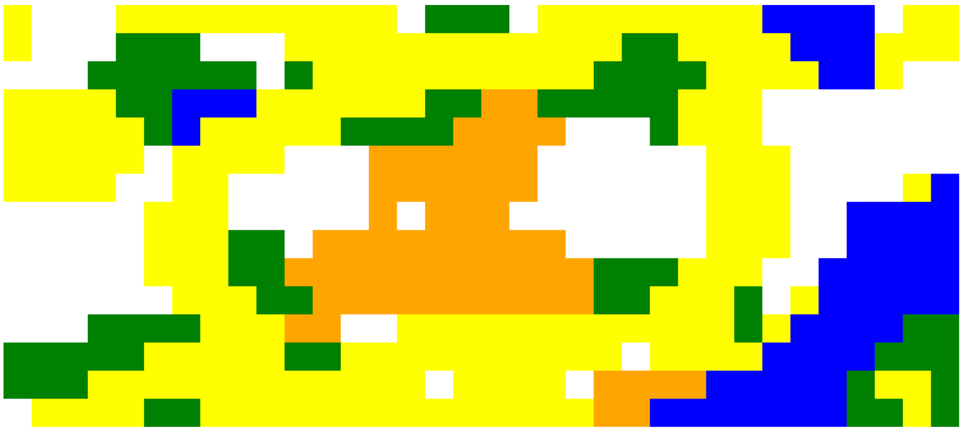


Figure 5.11: A generated preview map of Munich, Germany.

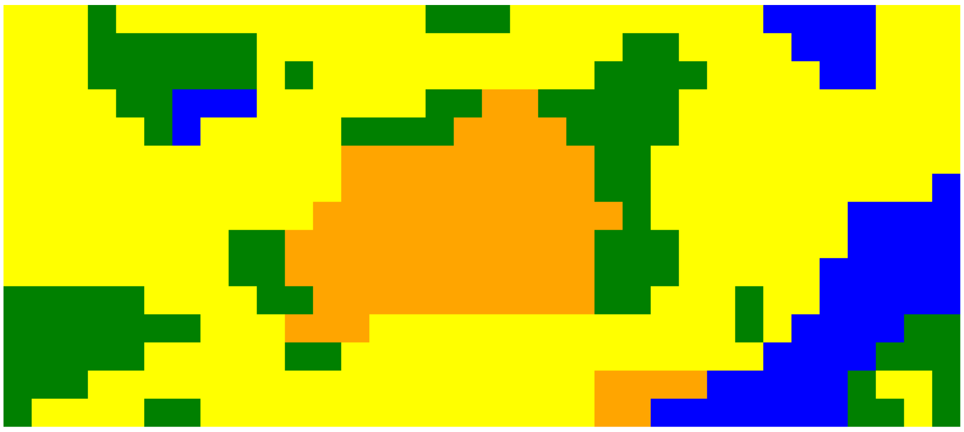


Figure 5.12: A generated preview map of Munich, Germany filled with a next neighbor algorithm.

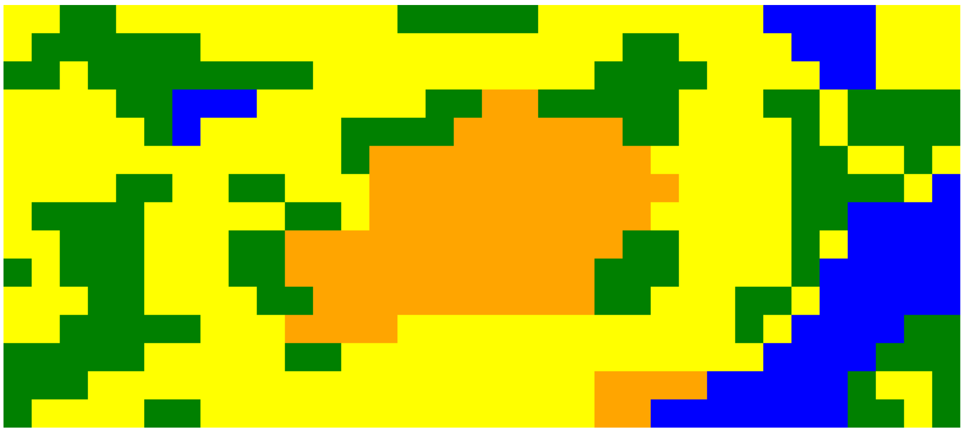


Figure 5.13: A generated preview map of Munich, Germany filled with a rule set algorithm.

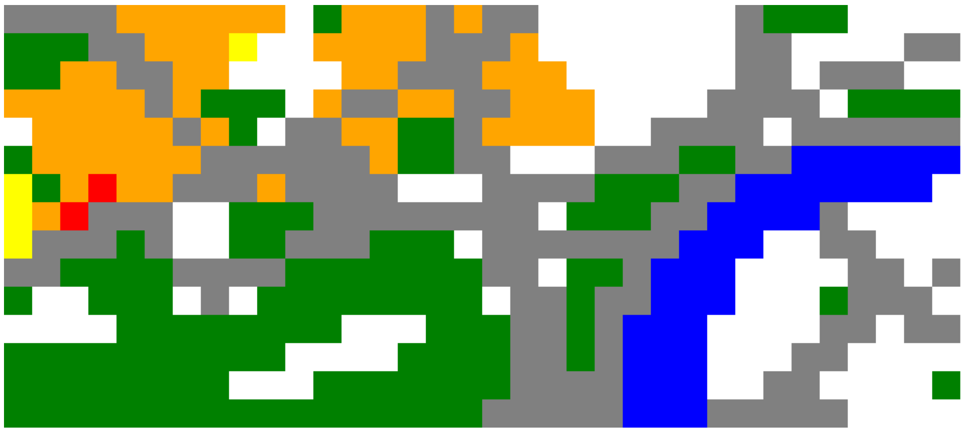


Figure 5.14: A generated preview map of London, Great Britain.

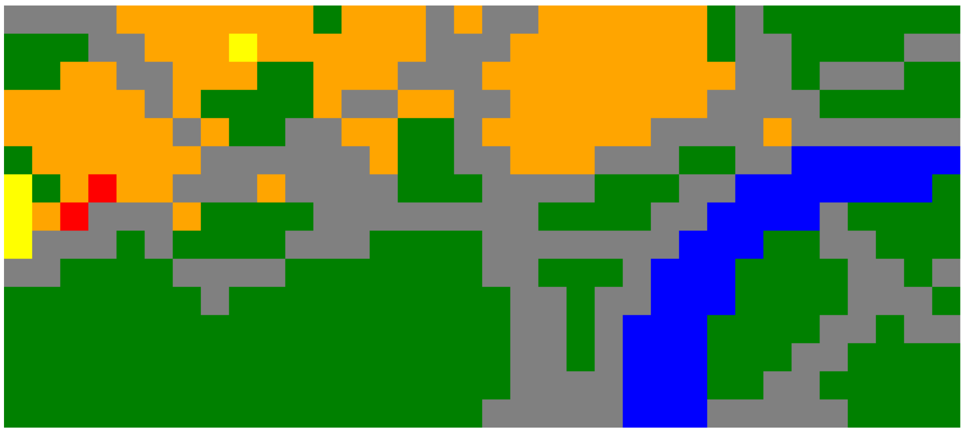


Figure 5.15: A generated preview map of London, Great Britain filled with a next neighbor algorithm.

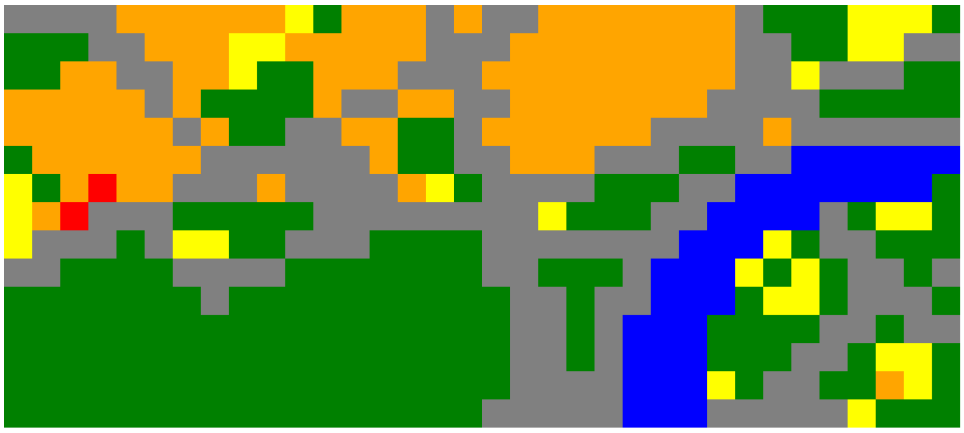


Figure 5.16: A generated preview map of London, Great Britain filled with a rule set algorithm.

Chapter 6

Comparison and Evaluation of Proposed Solutions

The approaches of Section 5.3.2 are not providing an ideal solution to the problem of missing information in *OpenStreetMap*. Therefore a comparison between them is needed to detect their individual performances. Ideally, this results in possible improvements and an enhanced algorithm. The criteria are shortly described and their resulting marks, based on the Austrian grading system as explained in Table 6.1, are presented in a table. Within, advantages and disadvantages are outlined and sequentially discussed afterwards.

In regard to the evaluation results in Table 6.2, the three approaches differ greatly from the predetermined criteria. Beginning with 'Heterogeneity', the 'Rule Set' performs significantly better than the other two. Logically this is due to the type of algorithm or lack of such in the case of the third approach. The 'Next Neighbor' algorithm usually creates homogeneous results because the ranking is based on the distance and size of other groups of tiles with similar types. This means for example that if there are two big

Table 6.1: Explanation of the assigned marks based on the Austrian grading system [31].

Grade	*Meaning*
1	Very Good: This indicates the best achievable performance.
2	Good: This indicates a close but not as good as a very good performance.
3	Satisfactory: This indicates an average performance.
4	Adequate: This indicates a bad but still passing performance.
5	Unsatisfactory: This indicates a very bad and therefore failing performance.

Table 6.2: Comparison of three approaches to handle missing information based on predetermined criteria.

	Next Neighbor	Rule Set	Leave Empty
Heterogeneity	4	1	5
Recognizability	2	3	4
Realism	2	1	5
Applicability	4	3	1

cluster of tiles of the same type with empty space in between, the groups will most likely conjoin. This can be seen by comparing Figures 5.11–5.12 where the green areas right of center have connected. Since the 'Rule Set' does not care about the size of other tiles and only about its directly adjacent neighbors, the result differs greatly. By applying some rules that have a high chance of inserting something new if two neighboring tiles share a type, a more complex pattern is created. If there is just one different type of tile next to a large cluster, the probability of changing other tiles around it increases. Though it depends on the types that are inserted, a difference can be clearly seen when comparing Figures 5.15–5.16 where the 'Rule Set' creates a much more diverse pattern in the bottom right corner. Because there is no algorithm used in the 'Leave Empty' approach, there is logically a large cluster of empty tiles that does not improve the heterogeneity of the map. When opposing the three approaches, the 'Rule Set' is the most appropriate because it is the only algorithm that creates a diverse map that is comparable to the complex structures of a real city.

The next criteria is 'Recognizability'. In contrast to the previous criteria, the resulting grades of the approaches do not differ greatly. Nevertheless, the 'Next Neighbor' algorithm scores the best grade when being applied to the exemplary cities. The created homogeneity of the approach, though impractical for the previous point, creates more recognizable areas. Since there are no new tiles added like in the 'Rule Set', the user is more likely to believe the correctness of the map. This is observable on Figures 5.9–5.10. The 'Rule Set' adds additional green forest tiles to the right part of the map even if there are no such areas. Although the algorithm correctly assumes the larger green area at the center of the map coincidentally. 'Leave Empty' scores lowest logically because the empty areas are not recognizable. However, it does not fill in tiles incorrectly or otherwise it would have scored worse. 'Next Neighbor' does not base its pattern on chance to assume areas correctly wherefore it is superior due to its higher reliability. It has to be mentioned that the criteria is not ideally fulfilled because no approach was able to completely fill in every exemplary map correctly. A possible improvement to this is to compare the given map to other sources such as *Google Map Maker* which has a similar map tagging functionality as *OpenStreetMap*. Their main difference is that *Google's* map service is a closed system which implies that information that are contributed by users belong to *Google* and are not open source. That is why a majority of the contributor-community prefers *OpenStreetMap* [41]. Another approach is to include satellite images to the map generation. Though they do not contain any meta information about human-build areas as there is no difference between residential and commercial areas from this perspective, certain terrains such as forests, water, streets and buildings are recognizable by their divergent colors. Regarding the project, this works with an additional request of information concerning geographical points where no tags have been found. If for example the satellite image on this coordinate is blue, it can be assumed that this area is water. Though buildings and streets can share the same color, the height of the point is an additional indicator for distinguishing them. Furthermore the book 'Remote Sensing and Image Interpretation' [6] contains methods of processing satellite imagery with more complex interpreters that are able to identify different structures based on their gathered experience. For example baseball fields can be recognized by their shape and by including the visible shadows in order to determine the height and shape of objects.

Though closely related to 'Recognizability', the criteria 'Realism' has a different outcome. Instead of 'Next Neighbor', the 'Rule Set' is preferable in the case of credibility. By basing the algorithm on observation on other European cities, the user is more likely to believe the validity of the generated map. This can be comprehended by comparing Figures 5.15–5.16. In the case of the 'Next Neighbor' approach, the bottom right corner is completely filled with green forest areas due to the large clusters of forests on the other side of the water. This is not completely wrong, because central London consists of many recreational parks. Nevertheless, 'Realism' is concerning a believable pattern more than the actual truth. For people that do not know central London but other European metropolises it seems unlikely for it to consist of such an amount of green areas. A pattern more common is for example that streams have recreational parks with trees at the riverside and residential or commercial areas behind them. This is due to many ecological dependences of rivers and trees. Trees are for example helping with the pollution reduction of rivers and serve as protection of water resources as stated in a research for the United Kingdom Forestry Commission [8]. The application of this rule can be seen by observing Figures 5.8–5.10. Though this depends on the origin of the user and therefore the personal knowledge about urban patterns, maps filled with the 'Rule Set' seem slightly more authentic than the others. If the approach leaves the map empty, any kind of realism is disturbed because it is improbable to have blank areas in the city center and therefore it logically performs worst.

The last criteria is concerning the applicability of the map for serious games. This is the only category where the 'Leave Empty' approach is preferable to the others as it behaves best for the exemplary game *Urban Space Upgrader*. The game adds production and cost value to each tile on the map. The goal for the player is to maintain balance of these values to keep the main resources, which are money, health and happiness, as high as possible. This is done by removing, creating or upgrading tiles to influence their positive and negative output. The overlay of the game works for each approach equally because the tested cities all created a playable game map. Though, a great discrepancy is shown by comparing the starting difficulty of the game. By filling empty spaces with tiles that already have influence on the overall resource-balance, the provided challenge is disproportionately high[1]. 'Rule Set' performs slightly better than 'Next Neighbor' as the generated types are more diverse. Regarding Figures 5.9–5.15 it is noticeable that in both examples one type is highly dominant. In London these are forests and in Salzburg residential areas. Including the overview of production and cost in Section 5.2 these maps will have an excess of one and a lack of another resource. 'Leave Empty' however starts with more empty tiles that are neutral in terms of production and cost. Therefore the player is able to fill in the gaps as desired and incrementally creates a balanced city.

Recapitulating the comparison of all four criteria, it can be said that the 'Rule Set' algorithm works best for generating cities that are most likely to be recognized and if not, at least be accepted as such, as the error margin of wrong tiles is not very high [2]. In the case of the exemplary serious game *Urban Space Upgrader* the approach of leaving the tiles empty has the highest applicability of the map. Though, as mentioned previously the criteria 'applicability' depends on the serious game used. Therefore other possible usages of the map generation are shown hereafter. Additionally these algorithms are only an example of possible solutions to fill in missing map information. To keep the reasonable bounds of this paper other approaches than 'Leave Empty', 'Next Neighbor' and 'Rule Set' were not tested. Since its goal is to provide the foundation for serious games, the selected approaches are sufficient for a real world implementation.

[1]In the case of a window size of 1366×768 the map consists of 510 tiles.

[2]A more precise percentage of falsely filled in tiles can not be given at the moment due to the preceded problem of *OpenStreetMap* not having sufficient or wrongly tagged information about areas.

Chapter 7

Further Development

The cities and algorithms that have been chosen in the previous chapter are only exemplary and do not represent the complete array of possible solutions to the problem of generating an abstract map that is suitable for serious games. Although the mentioned problems are solved sufficiently enough to satisfy the predetermined criteria, there is still much room improvement. These improvements basically aim for an extended variability in applicability that involves the possibility of using more cities and integrable tools.

As mentioned, only a small percentage of cities were chosen to test the map generator. A constraint that has not been covered are special characteristics in the structure of a city. Amsterdam, Netherlands is exemplary for an abnormal layout and the resulting map is seen in Figure 7.1. Apparent from the real map in Figure 7.2, Amsterdam has a large number of rivers in-between buildings. Since the generator is prioritizing rivers and streets before observing other types, the result of a large accumulation of small waterways is a large cluster of blue tiles. Based on the criteria in Chapter 6 it would perform significantly worse because instead of a large cluster many smaller water tiles are expected. A similar result can be expected if a city has many highways with the tag ‘primary’ because those are used for identifying the streets in the requested extract. A possible solution for this would be either to reduce the distance between the tiles and therefore increase the granularity to get a more detailed map or to remove the prioritizing of tiles for maps that contain a majority of them. However increasing the level of detail would equally increase the loading time drastically. Another inconsistency that prevents the generator to work globally is the incompleteness of the *OpenStreetMap* database. Though information about streets are available in almost every location, other tags like residential, commercial or industrial areas are mostly common in Central Europe. This hinders the creation of meaningful maps because the algorithms for filling do not work with insufficient input. A possible solution in this case that is beneficial to *OpenStreetMap* and the user is to provide a direct link to the location openstreetmap.org that is lacking data. Thereby the user may be motivated to add missing tags to be able to play the desired area. Incidentally the database of *OpenStreetMap* is growing. Even if the data base is updated in this way, a complete consistency of map information is improbable due to the enormous amount of landmass that has to be covered wherefore the function to fill in the gaps is still needed and improvable. A realizable approach is to add categories to the rule based algorithm. Additional information can be collected about the geographical location, size and population density of inhabited areas as far as they are available. Since the structures of cities differentiate greatly based on their location, this factor is considerable for determining the prominent type of land use. The scale of this can be from a continental one down to provinces. For example western Germany has more industry due to the available coal deposits in the ground whereas souther Germany has shown to have increased commercial areas in comparison. Logically, the size of a city is also significant for those assumptions as smaller towns as well as cities with a high population density are likely to have primarily residential areas.

The map generator is not meant to only operate *Urban Space Upgrader* but is rather supposed to work as a framework for different serious games. Though both programs share the same JavaScript file, they are strictly divided to easily replace one single part. The process of searching a map with *leaflet* and generating its abstraction with the possibility to fill the map allows other tools to use real maps as game element. Especially serious games about urban planning are able to profit from this. For example a game where natural disasters have to be prevented by placing fire departments or dams is able to teach about affected areas in cities. Or if the highway prioritizing is improved, a game where the player has to deliver packages from one point of the map to another by using the available streets can inform about the traffic situation in cities and why congestions form. The only restriction within the tool is the two-dimensional grid layout but because other games about urban planning like *Sim City 2000* or *Cities: Skylines* use a similar structure and are able to include most of the topics of urban planning that were mentioned in Section 2.1 it is effective enough to provide the foundation for many other educational games about the field.

Generally speaking the functions of map services have increased drastically over the last years as there are many competitors like *Google Maps*, *Apple Maps* and *OpenStreetMap*. While navigation is one of the main uses for those maps, the newer functions are aiming at an urge to discover and explore ones surroundings. The next step of exploring something is to understand how it works and as already mentioned there are several games like *Sim City* or *Cities: Skylines* that already fulfill some of these desires by simulating the complex coherences of urban planning. Though the education part in these simulations is given it is not dominant enough to communicate an adequate amount of information to count as a learning material. This

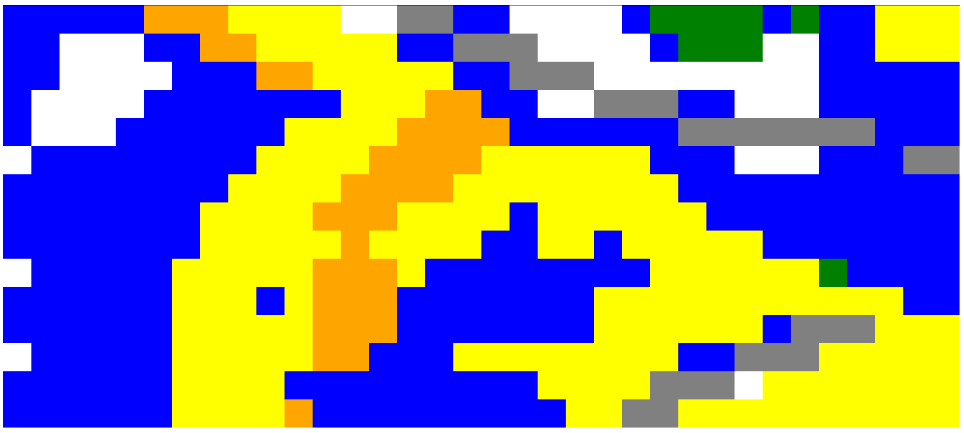


Figure 7.1: A generated preview map of Amsterdam, Netherlands.

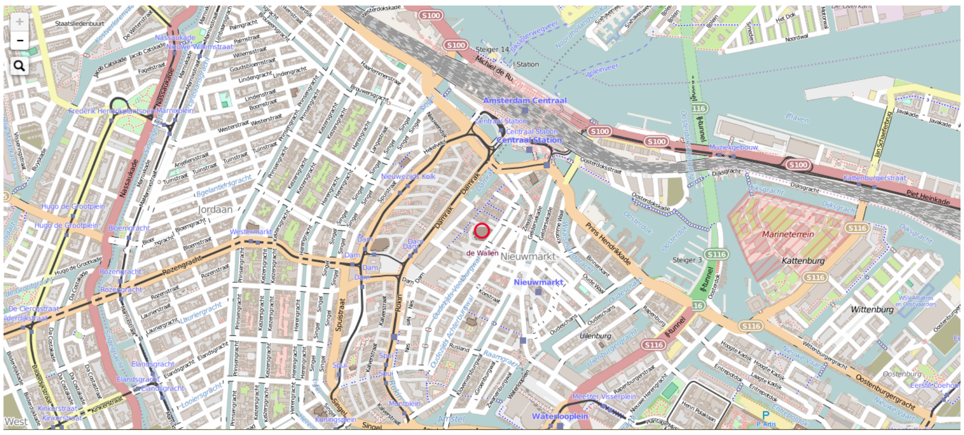


Figure 7.2: Amsterdam, Netherlands on *OpenStreetMap* viewed with *leaflet*.

is where the map services have the ability to collaborate by providing the necessary information in form of an openly usable learning platform. Regarding the amount of resources that is put into mapping the world with the *Street View* cars of *Google* and the great number of voluntary contributors of *OpenStreetMap*, the goal of providing a tool that is adaptable to every location on earth is definitely achievable.

References

Literature

[1] Simon Egenfeldt-Nielsen. “Overview of research on the educational use of video games”. In: *Digital kompetanse* 1.3 (2006), pp. 184–213 (cit. on pp. 12, 22).

[2] James David Foley. *Computer Graphics: Principles and Practice.* Addison-Wesley Professional, 1996 (cit. on p. 7).

[3] Archon Fung. *Empowered participation: reinventing urban democracy.* Princeton University Press, 2009 (cit. on p. 9).

[4] Stefan Greuter et al. “Real-time procedural generation of ’pseudo infinite’ cities”. In: *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia.* 2003, pp. 87–94 (cit. on p. 14).

[5] R.T. LeGates and T. Adams. *Early Urban Planning.* Early urban planning Bd. 9. Routledge, 2004 (cit. on p. 4).

[6] Thomas Lillesand, Ralph W Kiefer, and Jonathan Chipman. *Remote sensing and image interpretation.* John Wiley & Sons, 2014 (cit. on p. 50).

[7] David R Michael and Sandra L Chen. *Serious games: Games that educate, train, and inform.* Muska & Lipman/Premier-Trade, 2005 (cit. on p. 10).

[8] Tom Nisbet et al. “Woodland for Water: Woodland measures for meeting Water Framework Directive objectives”. In: *Forest research monograph* 4 (2011), p. 156 (cit. on p. 50).

[9] Johanna Nuojua et al. “Exploring web-based participation methods for urban planning”. In: *Proceedings of the Tenth Anniversary Conference on Participatory Design.* 2008, pp. 274–277 (cit. on pp. 7, 9).

[10] Marina Papastergiou. “Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation”. In: *Computers & Education* 52.1 (2009), pp. 1–12 (cit. on p. 10).

[11] Alenka Poplin. “Games and serious games in urban planning: study cases”. In: *Computational Science and Its Applications-ICCSA 2011*. Springer, 2011, pp. 1–14 (cit. on pp. 7, 19, 22).

[12] Alenka Poplin. “Playful public participation in urban planning: A case study for online serious games”. In: *Computers, Environment and Urban Systems* 36.3 (2012), pp. 195–206 (cit. on pp. 20, 22).

[13] P. Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag New York, 1990 (cit. on p. 8).

[14] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on game design*. New Riders, 2003 (cit. on p. 16).

[15] Tarja Susi, Mikael Johannesson, and Per Backlund. *Serious Games : An Overview*. Tech. rep. HS- IKI -TR-07-001. University of Skövde, School of Humanities and Informatics, 2007. URL: http://www.diva-portal.org/smash/get/diva2:2416/FULLTEXT01.pdf (cit. on p. 19).

[16] Richard Van Eck. “Digital game-based learning: It’s not just the digital natives who are restless”. In: *EDUCAUSE review* 41.2 (2006) (cit. on p. 19).

[17] Maria Virvou, George Katsionis, and Konstantinos Manos. “Combining Software Games with Education: Evaluation of its Educational Effectiveness.” In: *Educational Technology & Society* 8.2 (2005), pp. 54–65 (cit. on p. 10).

[18] David Williamson et al. “Video games and the future of learning”. In: *Phi Delta Kappan* 87.2 (2005), pp. 104–111 (cit. on pp. 11, 19).

[19] Yu Zheng et al. “Urban computing: concepts, methodologies, and applications”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.3 (2014), 38:1–38:55 (cit. on p. 5).

Films and audio-visual media

[20] United States Army. *America’s Army*. Microsoft Windows. 2002 (cit. on p. 12).

[21] Firaxis Games. *Civilization V*. Microsoft Windows. 2010 (cit. on p. 19).

[22] Maxis. *SimCity 2000*. DOS. 1994 (cit. on p. 16).

[23] MECC. *The Oregon Trail*. Apple II. 1985 (cit. on p. 11).

[24] Mojang. *Minecraft*. Microsoft Windows. 2011 (cit. on p. 8).

[25] Colossal Order. *Cities: Skylines*. Microsoft Windows. 2015 (cit. on p. 17).

[26] Pandemic Studios. *Full Spectrum Warrior*. Microsoft Windows. 2004 (cit. on p. 12).

[27] subLOGIC. *Microsoft Flight Simulator*. DOS. 1982 (cit. on p. 16).

Online sources

[28] United States Environmental Protection Agency. *Air Pollution*. last checked: 13.04.2015. 2010. URL: http://www.epa.gov/airtrends/2011/report/airpollution.pdf (cit. on p. 5).

[29] Mojäng Aktiebolag. *Minecraft Screenshot*. URL: http://minecraft.gamepedia.com/File:Beta.png (cit. on p. 9).

[30] United States Army. *Screenshot of America's Army: True Soldiers*. URL: http://www.gamespot.com/americas-army-true-soldiers/images/ (cit. on p. 13).

[31] BGBl. *Gesamte Rechtsvorschrift für Leistungsbeurteilungsverordnung*. last checked: 24.08.2015. 2015. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10009375 (cit. on p. 48).

[32] Brett Camper. *8-Bit City*. last checked: 14.07.2015. URL: http://8bitcity.com/map (cit. on p. 15).

[33] Brett Camper. *8-Bit City » London*. URL: http://8bitcity.com/map?London (cit. on p. 16).

[34] António Miguel de Campos. *Dragon curve, made using an L-system*. URL: http://commons.wikimedia.org/wiki/File:Dragon_curve_L-system.svg (cit. on p. 8).

[35] Wayne Carlson. *A Critical History of Computer Graphics and Animation*. last checked: 14.04.2015. 2003. URL: http://design.osu.edu/carlson/history/lessons.html (cit. on pp. 8, 10).

[36] Esri. *How Average Nearest Neighbor works*. last checked: 08.07.2015. 2014. URL: http://resources.arcgis.com/en/help/main/10.2/index.html#//005p0000000p000000 (cit. on p. 41).

[37] Stefan Greuter et al. *Figure 1: Real-time procedural virtual city* (cit. on p. 15).

[38] Paradox Interactive. *City Skylines: zoning mechanic*. URL: http://news.softpedia.com/news/Cities-Skylines-Sold-250k-Units-During-First-24-Hours-on-Sale-475712.shtml (cit. on p. 18).

[39] Jordiferrer. *Model of a Shanghai bridge, at the Shanghai Urban Planning Exhibition Center*. URL: http://commons.wikimedia.org/wiki/File:Urban_planning_museum_-_Model_of_a_Shanghai_bridge.JPG (cit. on p. 6).

[40] Jessica Lussenhop. *Oregon Trail: How three Minnesotans forged its path.* last checked: 15.04.2015. 2011. URL: http://www.citypages.com/2011-01-19/news/oregon-trail-how-three-minnesotans-forged-its-path/ (cit. on p. 11).

[41] Meghan McDonough. *Google Map Maker vs. OpenStreetMap: Which mapping service rules them all?* last checked: 25.08.2015. 2013. URL: http://www.digitaltrends.com/computing/google-map-maker-vs-openstreetmap-id-editor/ (cit. on p. 50).

[42] Minnesota Educational Computing Consortium (MECC). *The Oregon Trail Screenshot.* URL: http://en.wikipedia.org/wiki/File:OregonTrailScreenshot.png (cit. on p. 11).

[43] MicroProse. *Screenshot: The city screen.* URL: http://www.mobygames.com/game/windows/sid-meiers-civilization-v/screenshots/gameShotId,496025/ (cit. on p. 20).

[44] NetMarketshare. *Desktop Browser Version Market Share.* last checked: 28.04.2015. 2015. URL: http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0 (cit. on p. 24).

[45] Ookla. *HOUSEHOLD DOWNLOAD INDEX.* last checked: 28.04.2015. 2015. URL: http://www.netindex.com/download/ (cit. on p. 24).

[46] OpenStreetMap. *Elements.* last checked: 28.04.2015. 2014. URL: http://wiki.openstreetmap.org/wiki/Elements (cit. on p. 24).

[47] Markus Persson. *Terrain generation, Part 1.* last checked: 14.04.2015. 2011. URL: http://notch.tumblr.com/post/3746989361/terrain-generation-part-1 (cit. on p. 9).

[48] Alenka Poplin. *B3 Game: Design your Marketplace.* URL: http://de.slideshare.net/beniamino/games-and-serious-games-in-urban-planning-study-cases (cit. on p. 21).

[49] Alenka Poplin. *Fig. 5. Design of the user interface of the NextCampus game.* (Cit. on p. 21).

[50] Maxis Software. *Screenshot of an average city made in SimCity 2000.* URL: http://en.wikipedia.org/wiki/File:Sc2kscr.png (cit. on p. 18).

[51] Pandemic Studios. *Screenshot of Full Spectrum Warrior.* URL: http://www.gamestar.de/spiele/full-spectrum-warrior/32225.html (cit. on p. 13).

[52] SMAQ - architecture | urbanism | research. *Urban Plan.* URL: http://www.smaq.net/2010/01/stadtpromenade-am-finokanal-eberswalde-deutschland/?lang=en (cit. on p. 6).

[53] Wiktionary.org. *urbanus.* last checked: 13.04.2015. Jan. 2015. URL: http://en.wiktionary.org/wiki/urbanus%5C#Latin (cit. on p. 4).