A semantic approach to patch-based procedural generation of urban road networks

Edward Teng
A semantic approach to patch-based procedural generation of urban road networks

THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Edward Teng
born in Jakarta, Indonesia

TU Delft

Computer Graphics and Visualization Group
Department of Intelligent Systems
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl
A semantic approach to patch-based procedural generation of urban road networks

Author: Edward Teng
Student id: 4085094
Email: edward.teng@student.tudelft.nl

Abstract

A road network is one of the core elements of urban environments, strongly defining their layout. Procedural modeling has been increasingly used to create such road networks. However, many procedural methods are complex and difficult to master by non-experts, often have a limited and hard to control expressive range, and require a variety of specialized input data to generate a complex road network. To mitigate this, some methods proposed to use stochastic data on road patches extracted from example maps to design a road network following a given urban style. This thesis presents a novel patch-based method that uses the semantics of individual patches to help guiding the procedural generation. Our approach combines the advantages of patch-based generation with the parametric-based methods. Due to the intuitive character of semantic parameters and tags, our approach provides for an easy customization of fictive road network creation, allowing a user to easily define various types of road network styles, containing only the desired features and structures of real-world road networks.

Thesis Committee:

Chair: Prof. Dr. E. Eisemann, Faculty EEMCS, TU Delft
University supervisor: Dr. ir. Rafael Bidarra, Faculty EEMCS, TU Delft
Committee Member: Dr. ir. Willem-Paul Brinkman, Faculty EEMCS, TU Delft
Preface

This thesis project is a part of my Master of Science degree in Computer Science. The topic of the project deals with procedural content generation, which I find very interesting.

Looking back, when I first started on this project, I had a slightly different vision. However, the most important part is that I have done what I originally wanted, which is creating a different approach to procedurally generate road networks. At last, I would like to thank my supervisor, Rafael Bidarra, for his suggestions and weekly discussions. I would also like to thank Klaus Hildebrandt and Huijuan Wang for their suggestions and Milan van der Kuil for organizing a part of the usability study.

Edward Teng
Delft, the Netherlands
May 22, 2017
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Research question</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Outline</td>
<td>3</td>
</tr>
<tr>
<td><strong>2 Related work</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Parametric-based methods</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Example-based methods</td>
<td>6</td>
</tr>
<tr>
<td>2.3 General road network overview</td>
<td>8</td>
</tr>
<tr>
<td><strong>3 Approach</strong></td>
<td>11</td>
</tr>
<tr>
<td>3.1 Cell initialization</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Network propagation</td>
<td>12</td>
</tr>
<tr>
<td><strong>4 Patch-based method</strong></td>
<td>15</td>
</tr>
<tr>
<td>4.1 Semantic identification</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Generation overview</td>
<td>17</td>
</tr>
<tr>
<td>4.3 Cell initialization</td>
<td>19</td>
</tr>
<tr>
<td>4.4 Network propagation</td>
<td>22</td>
</tr>
<tr>
<td>4.5 Discussion</td>
<td>28</td>
</tr>
<tr>
<td><strong>5 Prototype implementation</strong></td>
<td>31</td>
</tr>
<tr>
<td>5.1 Graph structure</td>
<td>31</td>
</tr>
<tr>
<td>5.2 Application pipeline</td>
<td>32</td>
</tr>
<tr>
<td>5.3 Landmark creation</td>
<td>33</td>
</tr>
<tr>
<td>5.4 Parametric-based generation</td>
<td>34</td>
</tr>
<tr>
<td>5.5 Generation constraints</td>
<td>37</td>
</tr>
<tr>
<td>5.6 Patch-based components</td>
<td>38</td>
</tr>
</tbody>
</table>
## CONTENTS

5.7 Post process - Cell repair ........................................ 43
5.8 Graphical user interface ......................................... 45

6 Results and performance ........................................ 47
6.1 Patch-based method results .................................... 47
6.2 Parametric-based method results ............................... 50
6.3 Combination ....................................................... 56

7 Usability testing .................................................... 61

8 Conclusion .......................................................... 65

Bibliography .......................................................... 67

A Glossary ............................................................. 69
A.1 Parameters ......................................................... 69
A.2 Patches ............................................................ 72
A.3 Scelix ............................................................... 77
A.4 Usability test ....................................................... 78
Chapter 1

Introduction

There exists methods to automatically create content using algorithms. In computer graphics, procedural modeling is a term to refer to those methods and it has been used for years, for example, to model terrains, buildings, plants and trees. Applying procedural modeling to create complex and large virtual content, such as cities, reduces the time consumption drastically in comparison to manual construction.

Road network generation is a crucial part of city modeling, because a road network defines a major part of the layout of the city. Several procedural modeling methods have been proposed to generate such content, ranging from simple grid patterns to complex and realistic urban styles. Figures 1 and 2 show a road network visualization and a couple of road network structures of Manhattan and London respectively. Such urban road network structures are called *patches*.

![Figure 1: A road network visualization of Manhattan and two road structures (patches) within the selected area](image)

As can be seen from the figures, Manhattan has a more typical grid street plan, while London has a more organic road plan. There are procedural methods that can generate a similar road pattern as the one being used in Manhattan by defining the generation parame-
ters. However, it is very difficult to generate a distinguishable road style or have comparable patches as the one being used in London by employing similar methods. This is where the method which uses patches extracted from the real world excels at.

Besides this generation incapability, the use of parameters has another drawback: it requires some knowledge to configure the parameters to achieve a typical road network style. This causes content generation to be difficult for users who have no knowledge of modeling city layouts. Hence, it is important for the approach to be intuitive. Despite these drawbacks, it is easier to define parameters rather than finding and using patches to create simple road network styles, such as grid, industrial or suburban. However, it is essential that the approach is capable to create such road network styles, i.e., by means of altering the available generation parameters. Thus, the controllability of the approach is as important as being intuitive.

By using patches as the main input for the road generation, the method can generate any realistic road networks. Unfortunately, the current method that utilizes patches lacks the utilization of patch semantics. The semantics of a patch in this context is defined as any feature which embodies some patch function or meaning, and is very useful for the road generation and its controllability. The semantics can be taken from each individual element of the patch (e.g. street) or the patch as a whole. Additionally, the semantics can also be used when modeling buildings, e.g. knowing that a patch is a roundabout, a roundabout texture can be automatically applied on the circular area where the patch was used. The topic of this thesis is the generation of road networks, thus the modeling/generation of buildings is not within the scope of this thesis. We introduce the term patch-based method as the method that mainly uses patches and its semantics to generate road networks.
1.1 Research question

In this thesis, we dealt with two major challenges: seamlessly integrating patch-based and parametric-based methods in a controllable and intuitive way, and more importantly, representing the semantics of a patch and defining how this semantics can be utilized by a procedural generator. The latter is incorporated in the patch-based method. Using the mentioned methods, we wanted to create ‘fictive’ urban road networks in the sense that they contain features and structures from the real world, but do not share identical styles.

This brings us to the main research question of this thesis:

- How can procedural parametric-based and patch-based methods be integrated into a controllable and intuitive approach to generate fictive urban road networks?

For this question, we developed a semantic approach that combines a patch-based method with the advantages of a parametric-based method for the customized generation of fictive urban road networks. The main focus of this project, however, was in the patch-based method. This method takes the semantics of each individual patch into account and allows patches to be used flexibly (mainly rotatable, but also scalable) during the road network generation. By putting more focus on the semantics, the generated road network is fictive and differs from other procedural road generation methods.

Furthermore, the controllability and intuitiveness of our approach are mostly based on the proposed generation parameters, semantics of the patches and how they are being presented as in high-level settings. To evaluate our approach, we tested the developed application, which incorporates our approach, in a usability test. We also assessed the effectiveness of the controllability of the parametric-based method by analyzing and measuring the features of generated road networks.

Hence, our contributions are the following:

- The use of patch semantics to help guiding a patch-based road network generator
- A controllable parametric-based road generation method with a high expressive power
- A high level settings scheme that allows non-experts to easily create and modify road networks

1.2 Outline

The next chapter provides an overview of a number of related works. Chapter 3 describes the general approach of our patch-based method. Chapter 4 provides an in-depth description of the patch-based method. Chapter 5 covers implementation details of the necessary and supplementary functions for our approach. Chapter 6 shows the results and performance. Chapter 7 describes the findings and conclusion of the usability test. Chapter 8 outlines the conclusion and future work of this thesis.
Chapter 2

Related work

In this chapter, related works that employ parameters and patches to procedurally generate roads are described. Previous researches have been done that utilize a parametric-based method to generate a road network. This is, however, not the case for the patch-based method, but there are multiple works that are based on the example-based approach. The patch-based method is based on one of the example-based approaches. Besides these two methods, there are other but less relevant ones, such as template- (Sun et al. [2002]), tensor- (Esch et al. [2007], Chen et al. [2008]) and agent-based (Lechner et al. [2003], Lechner et al. [2006]) methods.

2.1 Parametric-based methods

The parametric-based methods use specified parameters to generate a road network and depending on the implementation and parameters, a variety of road network styles can be created.

Parish and Müller [2001] proposed a system called CityEngine (CE) to model a city. Road network generation of CityEngine is based on Lindenmayer-system (Prusinkiewicz and Lindenmayer [1991]), which is a rewriting system that uses a set of rules to model the growth process. Nevertheless, the generation is controllable using input maps, real world data and parameters. However, finding the right parameters might be cumbersome due to the iterative nature of procedural modeling. In addition, CityEngine allows user to use image maps, such as elevation, land, water and population density and also statistical data from the real world cities. By using these image maps and data, not only a realistic city can be created, but also a rich environment with hills and rivers can be synthesized. The road network created by CityEngine is made up of highways and streets. Highways connect populated areas while streets grow into the residential areas and provide the residents access to the nearest highways.

The road generation process follows global goals and a goal is controlled by the input image maps. Highways connect the center of the population density according to the population density map while streets follow the dominant street pattern of the area. A dominant street pattern is selected according to the calculation from the input data, for example, an-
2.2 Example-based methods

Unlike the parametric-based method, an example-based method uses real world data to synthesize the road network. There are two variants of example-based methods: parametric and non-parametric. The parametric variant uses extracted parametrized features from the data, while the non-parametric variant utilizes the data directly.

2.2.1 Parametric variant

Aliaga et al. [2008] presented an interactive system to synthesize urban layouts based on a parametric stochastic example-based approach. The generation parameters, such as intersection type, street length and tortuosity, which describes the curvity of street segments, are obtained from the input example.
The user has to provide one or more example urban layout fragments as the input. The
fragments contain aerial-view images and GIS vector data of a city. These fragments are
available in GIS databases. The system extracts the road network attributes and param-
eters from the GIS vector data and parcels from the images. The road network generation
process follows the style indicated by the geometric attributes of the intersection points and
generation parameters. A random walk algorithm is used for the generation. This algorithm
connects the intersection points using a transition probability based on their attributes.

2.2.2 Combination

Nishida et al. [2015] proposed an interactive system to design complex, realistic road net-
works based on examples using a different approach than the one described above. Instead
of only using attributes, this approach mainly uses patches from example maps taken from
OpenStreetMap, which is a free online editable map of the world. The user has to pro-
vide one or more example road networks as input to define the road style for the road
network. The road network consists of arterial roads and local streets and is generated us-
ing two methods: example- and procedural-based. The approach also supports high-level
operations, such as warping, blending and interpolations, to generate road networks that
look different than the ones from the example maps. Warping operation changes the ori-
etination/shape of the example network, blending operation creates a smooth road transition
from one network to another and interpolation operation combines two example networks
into one.

The system automatically detects and extracts patches (for example-based growth) and
calculates statistical features from the example network: the length and curvature of the
edges of the network (for procedural-based growth). During the extraction, dead-end edges
in the example map are also marked as dead-end edges in the patch. Starting from an
initial seed, that is specified by the user or the system, example-based or procedural-based
growth is executed. The default is example-based growth and if this fails, procedural-based
growth will be used. For the example-based growth, patches are used to generate the road
network. A patch is randomly selected according to the usage probability, which compares
the similarity between the connector patch edges and graph edges. Additionally, a patch is
only valid if it passes the legality check: edge segments are above the water level, the terrain
slope is within a certain threshold and there are no intersections with other road segments.
This process is repeated for the vertices created from the growth methods until there are no
more vertices left or if all road segments intersect the boundary of the target area.

After the generation of arterial roads, the local street generation process starts. The
seed placement is the crucial part for this process and depends on how the arterial roads
were created. For the example-based growth, the seeds are placed on the intersection points
between arterial roads and local streets from the example map. In the case of the procedural-
based growth, the statistical features are used to divide those roads into multiple small
fragments and the seeds are put at the boundary between the fragments. Finally, starting
from the created seeds, local streets are generated in the same way as the arterial roads.
2.2.3 Features, pros and cons

This section provides an overview of the features, pros and cons of the example-based approach proposed by Nishida et al. [2015]. Our patch-based method is based on this variant due to the direct use of patches, hence only this approach is being covered in this section. It has four key features:

- Automatically extracts patches from a given example map
- Uses extracted patches directly to generate road networks
- Uses statistical features from the example maps for the procedural-based growth
- Uses the procedural-based growth to assist the generation where the example-based growth fails
- Provides high-level operations, such as warping, blending and interpolations

These features contribute to three main pros:

- Allow users to simply provide example maps instead of explicitly requiring predefined road geometries
- Allow users to design detailed and realistic road networks easily (without specifying any parameter) and quickly
- Allow users to generate road networks that are different than any of the example map by using the provided high-level operations

However, there are two cons:

- The method uses the patch as-is, it does not allow individual patch rotation or scaling, thus lacking flexibility
- Even though the method uses information (semantics) of the patch, such as the type of streets, e.g. arterial or local and dead-end or non dead-end, we find that more features or properties of a patch can be identified and utilized for the generation

2.3 General road network overview

This section provides the general road network overview and information that have been proposed in earlier works, but are relevant for this project.

Road networks without crossovers or viaducts can be represented as a planar geometric graph in which each vertex can be seen as a point in a road network (such as an inflexion or an intersection), and each edge as a street (segment) connecting those vertices. Real-world road network consist of (at least) main and local streets: main streets provide rapid displacement and access by connecting different areas, while local streets grow into (e.g. residential) local areas, providing access to the nearest main streets. In our geometric graph,
main streets define a space partition into main cells; within each main cell, local streets (may) define local cells.

Figure 3 illustrates the generic pipeline and a possible road network generation using a parametric-based method. The pipeline is divided into two main phases, the main and local street generation, and it requires parameter values as the main input. The output is a graph that represents the road network. Main streets are generated before local streets. From the generated main streets, main cells are formed. Local streets are generated in each of those main cells, hence different road styles can be generated in each main cell. It is important to note that constraints are applied during this road generation to ensure that each new road segment adapts to the terrain and the current road network.

Figure 3: The generic pipeline with an illustration of road network generation using the parametric-based method. (1) Main streets are generated, then (2) local streets. Notice that local street generation is executed on one or more main cells.
Chapter 3

Approach

Our approach proposes the definition, representation and usage of patch semantics and the integration of patch-based and parametric-based methods to generate road networks. The semantics plays a crucial role in the patch-based method, because it contributes to the controllability of the method and is also utilized during generation. This chapter describes the general approach of our patch-based method.

The patch-based method is used to generate local streets and extends the pipeline of the road generation given in Section 2.3 (see Figure 3). This extension can be seen in Figure 4. We chose to use parametric-based instead of patch-based generation for main street generation, as it yields quite suitable cells and avoids a strong dependency on the particular set of main road patches used. In addition to parameters and a main cell as the input for local street generation, patches are also required. To propagate local streets within each main cell, a database of patches is first processed, identified and classified for their patch semantics. Our patch-based method iteratively takes each main cell, initializes it during the cell initialization process and grows the local road network within each cell during the network propagation process.

Figure 4: The extended pipeline of the approach
3.1 Cell initialization

This process aims at creating a first set of starting vertices for the local street propagation within a cell. Many possible ways can be devised to achieve this; we have chosen to attach suitable patches to the main streets, and grow the network from there inwards, thus ensuring that the local streets being grown are connected to the main streets. Patch semantics, primarily its structure, determines which patches are suitable for this initialization. Requiring this specific patch structure typically ensures a natural transition from a main to a local street.

In addition to the use of individual patches, it is also possible that some patches are fit to be used repeatedly after one another. To allow this kind of generation, our approach supports the creation of patterns. A pattern, in our case, is a road network structure, which consists of one or multiple patches attached in tandem where they can be repeated multiple times in a predictable way. These patches (or a single patch) that are used to form a pattern is what we defined as the base pattern, see Figure 5. Patterns are used as an alternative to patches for the initialization. Similar to patches, we wanted to make sure that a pattern is connected to the cell, thus we have chosen two additional approaches: use a pattern on one of the main streets of the cell and use a pattern between the main streets.

![Figure 5: An example of two simple patterns and their base patterns.](image)

(1a) A base pattern that only consists of one patch. (2a) A base pattern that consists of two patches. (1a/2b) Patterns created using the base patterns.

3.2 Network propagation

The propagation process focuses on iteratively selecting a patch and appending it to the local streets within a main cell commencing from the starting vertices created during the cell initialization. Naturally, each new patch appended may bring in one or more connectable vertices to the pool. A patch is selected during the patch selection and applied to the graph during the patch insertion.

Patch selection

The patch selection takes into consideration both the network surroundings (e.g. neighboring vertices or edges, available space,...) and each patch’s semantics (e.g. size, connectiv-
Two patch selection methods are attempted when looking up a suitable patch (see Figure 6): one method propagates the network 'sideways' within the cell, using the new patch to bridge two current starting vertices; and the other method typically propagates the network inwards into the cell, simply attaching a fitting patch to one starting vertex.

Figure 6: An example of the two patch selection methods. (1) There are two patches. The blue dots represent the connection vertices. (2a) First method: find patches to append a neighbouring vertex. The red dot represents the starting vertex and the green dot a neighbouring vertex. (2b) One patch is valid for this method as it is the only patch that can be rotated and translated such that it connects the starting and neighbouring vertices. (3a) Second method: find patches without the consideration to connect a neighbouring vertex. (3b) Both patches are valid for this method.

To have an indication of the suitability of a patch, a rating system is introduced. By comparing the ratings of the patches, we can select the most suitable patch.

**Patch insertion**

Given a patch from the patch selection, the patch is applied to the graph in an appropriate manner. However, it is possible that there is no suitable patch available (e.g. no patch fits on the graph), in this case the parametric-based growth is used instead.
Ultimately, an application is developed that incorporates this approach. Using the application, landmarks, such as rivers and hills, can be generated. These landmarks affect the road network generation process. Additionally, patches can be automatically extracted from an example map or manually created and patterns can be manually created using the available patches.
Chapter 4

Patch-based method

In this section we describe in more detail the main steps of our patch-based approach. The method requires a database of patches as input. In our prototype system, patches for this database can either be manually created or extracted from example maps. For the former, we developed an interactive patch editor, that also permits to define patterns. For the latter, we implemented a patch extraction algorithm similar to that described by Nishida et al. [2015].

4.1 Semantic identification

The semantics of a patch is classified into three categories: vertex, edge and patch. The first two categories describe the individual information of the vertices and edges of the patch, while the patch category describes the semantics of the patch as a whole. The patch semantics that we defined in our approach can be represented by tags for each of these categories, as summarized in Table 1.

Vertex and edge

An illustration of the properties of vertex and edge entities can be seen in Figure 7. The propagation process uses snap and clip operations where ever convenient and possible. A snap operation connects an edge to a nearby connectable vertex or on a nearby, non dead-end edge. A clip operation creates an intersection vertex at the intersection of two edges and clips the remaining edge segment. For patches, such operations are only applicable to snappable edges, i.e. edges that are connected to an intersection vertex. The main or local edge property is used to indicate whether and how a patch can be used for initialization, when a patch is applied, its edges are defined as local streets in the road network.

Patch

Figure 8 shows five patches with different tags. Some patch tags indicate geometric or topologic properties, as e.g. roundabout, loop, cul-de-sac, curved and straight. In order to control the patch-based generation, patches with such tags can be filtered out. For example,
4.1 Semantic identification

<table>
<thead>
<tr>
<th>Category</th>
<th>Tag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex</td>
<td>Connectable</td>
<td>A vertex that allows connections</td>
</tr>
<tr>
<td></td>
<td>Intersection</td>
<td>A vertex with at least 3 degree</td>
</tr>
<tr>
<td>Edge</td>
<td>Dead-end</td>
<td>An edge with no connectable vertices</td>
</tr>
<tr>
<td></td>
<td>Snappable</td>
<td>An edge that allows snapping</td>
</tr>
<tr>
<td></td>
<td>Main</td>
<td>An edge representing a main street</td>
</tr>
<tr>
<td></td>
<td>Local</td>
<td>An edge representing a local street</td>
</tr>
<tr>
<td>Patch</td>
<td>Roundabout</td>
<td>A patch with a circular intersection</td>
</tr>
<tr>
<td></td>
<td>Loop</td>
<td>A patch with one or more closed areas</td>
</tr>
<tr>
<td></td>
<td>Cul-de-sac</td>
<td>A patch with one or more dead-end edges</td>
</tr>
<tr>
<td></td>
<td>Curved</td>
<td>A patch with segments forming a curve</td>
</tr>
<tr>
<td></td>
<td>Straight</td>
<td>A patch with only straight segments</td>
</tr>
<tr>
<td></td>
<td>Terminal</td>
<td>A patch with only one connectable vertex</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>A patch suitable for initializing the propagation process</td>
</tr>
</tbody>
</table>

Figure 7: Patch edges and vertices tags.

A terminal patch, by definition, prevents any further propagation as it adds no connectable vertices to the pool. Thus, in general, it is only applied when there is a limited area left. As another example, an initial patch consists of a straight main street edge and an attachment, i.e. a connected set of local street type edges with at least one connectable vertex. In this way, it can be deployed during the initialization phase of a main cell and also during network propagation.

It is possible for a patch to have multiple tags, e.g. cul-de-sac and straight. However, roundabout and loop tags are mutually exclusive due to the fact that a roundabout patch always has a loop. Thus having a loop tag for roundabouts is redundant.
4.2 Generation overview

The generation of the patch-based method is divided into two main processes: cell initialization and network propagation. Figure 9 shows the flowchart of the generation. Given the input, cell initialization creates starting patches or a pattern. During this initialization process, one or more start vertices are added to the graph. These start vertices are the connectable vertices from the patch or pattern and every connectable vertex in the patch will also be a connectable vertex in the graph when that patch is appended. In the case of the parametric-based growth, every created non-intersecting vertex is connectable. These connectable vertices will be used during the propagation process and patches can be attached to these vertices. To avoid confusion, the vertices that still need to be processed by the propagation process are called candidate vertices. Thus every added connectable vertex is a candidate vertex.

After the initialization, the propagation process starts. This process populates the cell with local streets by primarily using patches. It iterates through all candidate vertices. First, available patches are filtered to exclude patches that do not fit in the approximated work space. After filtering, the patch selection process is executed. This process tries to find a suitable patch for the current vertex and two methods are used: (i) breadth-first, that checks whether there is a patch that can connect to a nearby vertex; and (ii) depth-first, that searches for a patch that can be simply appended to the current vertex of the network.

To ensure a certain level of connectivity, the breadth-first method is attempted first. If this method is successful, then the resulting patch is chosen for the current vertex, otherwise the depth-first method is carried out. In a situation where both methods fail to succeed, a
4.2 Generation overview

Patch-based method

Figure 9: The patch-based generation flowchart
parametric-based growth is used instead. Figure 10 illustrates the patch-based generation process.

This generation is mainly controllable through the use specifications of patches. The semantics of a patch, namely the tag property, plays a role in this context, e.g. creating a road network using patches that have curvatures or restricting the use of roundabouts. However, it is also possible that some patches are used too frequently. To avoid this excessive use of patches, the frequency of occurrences of patches is also controllable.

Figure 10: A demonstration of the processes of patch-based generation. (1) The selected main cell. (2) Cell initialization using patches on main streets. (3a) The red dot represents the current vertex. (3b) A patch (in blue rectangle) selected during the breadth-first method is inserted to the graph. (3c) A patch selected during the depth-first method. (4a/4b) The parametric-based growth from the current vertex. (5) The propagation is completed.

4.3 Cell initialization

In this process, initial patches are inserted onto a main cell, in order to produce the first set of starting vertices to be used in the network propagation process. There are various possible methods to achieve this and we implemented the following three, see Figure 11. Additionally, the initialization methods that use patterns have two cases: a case where there
is at least one connectable vertex available from the pattern and another case where there is none available. In the first case, the propagation proceeds from a candidate vertex, however this is not possible for the second case. In this situation, one patch is added to one of the boundaries and the propagation starts from a candidate vertex of the added patch.

Figure 11: Different methods to initialize a cell. (1) The selected main cell. (2) Initialization using patches on main streets. The red dots represent connectable vertices. (3) Initialization using a pattern on a main street, notice that the pattern (in blue rectangle) does not have a connectable vertex. Thus an additional patch (in purple rectangle) is added to supply connectable vertices. (4) Initialization using a pattern between main streets.

**Patches on main streets**

This method places patches on main streets of the cell, see Figure 11.2. To place patches on main streets, *seed points*, i.e. main street positions to be used for patch placement, are calculated so that each patch is properly distributed between all main streets. This distribution ensures that each patch has enough space to grow, thus avoiding two patches being very close to one another.

For this initialization method, the semantics of the patch is used, namely only patches with initial property are eligible. A patch is randomly chosen for every seed point, after which it is translated to the seed point, rotated such that the main streets (of patch and seed point) are aligned. The attachment is then applied on the graph and the line is lost in the main street. The reason for this particular choice of patch utilization on a main street is to enforce a certain pattern (a line and an attachment) to reduce the randomness of initial patches.
**Patch-based method**

### 4.3 Cell initialization

**Pattern on a main street**

This method creates and adds a pattern on a main street, see Figure 11.3. Each base pattern has a scaler and multiplier. The scaler indicates how the base pattern needs to be scaled and the multiplier determines how many times each scaled base pattern is repeated to form a pattern. To determine the two values, first the length ratio is calculated. This ratio is simply calculated by dividing the length of the street by the length of the base pattern from the defined start to end vertices. The straight line from the start to end vertices forms the *base line* of the base pattern. Figure 12 illustrates the scaling and multiplying of a base pattern. The base pattern and main street combination where the scaler lies closest to 1 is chosen to avoid large scaling. To add the pattern to the graph, the base pattern is scaled and repeated according to mentioned values and then rotated and translated in such a way that the base line of the pattern lies on the main street.

![Figure 12: An illustration of base pattern scaling and multiplying.](image)

1. The base pattern, note that a single patch can be used as a base pattern. The blue line represents the base line of the base pattern.
2. The red line represents a main street. This main street is slightly longer than the length of the pattern (and the base pattern repeated twice). To fit the main street, the base pattern has to be upscaled (scaler > 1).
3. The pattern fits the main street, thus there is no scaling needed (scaler == 1).
4. The main street is shorter than the pattern. To fit the main street, the base pattern has to be downscaled (scaler < 1).

If the pattern does not contain any connectable vertex, a main street that lies far from the pattern is initialized with an initial patch, in order to increase the chances of propagation in that cell.

**Pattern between main streets**

Another method is to create a pattern and place it across two main streets, see Figure 11.4. The seed points for the start and end vertices of the pattern are placed in the middle of each main street pair to minimize unnecessary obstacles for the pattern placement. The procedure to find and place a base pattern is fairly similar to the previous method, except that the line connecting the seed points is used instead of a main street.
4.4 Network propagation

The network propagation process populates a main cell with local streets, by iteratively attaching a patch to the vertex currently being used for the propagation, the current vertex. For this, a pool of candidate starting vertices in a queue, which was first populated during cell initialization, is maintained and subsequently grown with the connectable vertices of each new patch appended to the road network. The pseudocode for this process can be found in Algorithm 1. First, the next current vertex is popped from the queue, and a check is done on the feasibility of attaching a patch to it. This check takes into account possible constraints on the current vertex degree as well as on the actual space available, on the computed propagation direction, e.g. the angle to attach the patch to the current vertex. Next, after filtering out patches that would not fit the work area, which is the available space for the current vertex to grow, two patch selection methods are attempted. For both methods holds that only valid patches, i.e. complying with the user-defined constraints, can be used and appended to the graph. If none of the methods finds a suitable patch, parametric-based growth is used instead, as a fail-safe mechanism. Descriptions of the propagation direction, patch filter, the two selection methods and patch insertion are given in the following subsections.

Algorithm 1 Network propagation

1: Input: graph, patches, cell
2: Output: graph
3: while graph.candidateVertices.Count > 0 do
4:   currentVertex ← graph.candidateVertices.pop()
5:   propagationDirection ← calculate the propagation direction for currentVertex
6:   if propagationDirection is valid then
7:     filteredPatches ← filter available patches
8:     if filteredPatches.isEmpty() then
9:       apply parametric-based growth on currentVertex
10:   else
11:     suitablePatch ← apply Breadth-First to find a suitable patch
12:       ▷ Algorithm 2
13:     if suitablePatch == null then
14:       suitablePatch ← apply Depth-First to find a suitable patch
15:       ▷ Algorithm 3
16:     if suitablePatch ! = null then
17:       attach suitablePatch on currentVertex
18:     else
19:       apply parameteric-based growth on currentVertex
4.4 Network propagation

4.4.1 Propagation direction

The propagation direction is an indication of which approximate direction the new patch could take, if attached to the current vertex. The actual direction may fluctuate within a given range, similar to the parametric-based generation. This direction is given by the angle bisector of the largest angle formed by the adjacent edges of the current vertex. As can be seen in 13, the propagation direction depends on the degree of the current vertex.

4.4.2 Patch filter

This process filters available patches in such a way that only patches that fit the work area are eligible to be selected during the patch selection process. During the early stage of the propagation, the work area is more or less substantial, e.g. patches with large size can be utilized, this also depends on the size of the main cell. However, as the propagation progresses, this area becomes smaller and smaller, thus reducing the amount of patches that can be used due to some patches being bigger than the work area.

We used an approximation of the work area instead of the actual work area, because it is computationally less expensive and provides a satisfactory estimation. To find the work area approximation, connectable vertices (that only have 1 degree) from the graph and also the vertices of the main cell are used. The inclusion of main cell vertices provides a more robust indication. The approximated work area is the oriented minimum bounding box of these vertices. Figure 14 shows the process of finding approximated work areas. To check if a patch fits the approximated work area, the size of the bounding box of the patch and the approximated work area are compared.
4.4 Network propagation

Patch-based method

Figure 14: Two examples of the approximated work area. (1) A patch-initialized cell. The current vertex is represented by the red dot, connectable vertices by green dots and main vertices by blue dots. The black dot is not recognized as a connectable vertex because it was created by the same patch as the current vertex. (2) The three main vertices are not usable due to no valid direct connection from the starting vertex. (3) The approximated work area (blue rectangle) for the current vertex. This area is the oriented minimum bounding box of the valid vertices. (4/5) The approximated work area of another current vertex after a number of steps.

4.4.3 Breadth-first method

The main purpose of this method is to find a suitable patch that connects the current vertex with one of the connectable vertices from the graph. The pseudocode of this method can be found in Algorithm 2.

The method has to search for two pairs of vertices that are at (approximately) the same distance: (i) a graph vertex pair, consisting of the current vertex (start) and another connectable vertex (target) of the graph; and (ii) a patch vertex pair, consisting of two connectable vertices (start and target) of the candidate patch. Each patch can have two or more vertex pairs, with the exception of terminal patches, which have zero pairs, thus not eligible for this method, and each pair is evaluated individually. For example, if a patch has three connectable vertices: A, B and C, this means that there are six potential pairs, A-B, A-C, B-A, B-C, C-A and C-B. Pair A-B is different from pair B-A due to the geometric difference when the patch is rotated and translated.

Rotation

Given a patch and its vertex pair, and a graph vertex pair, it is necessary to calculate the rotation of the patch to append those vertices. Assuming that a patch fits perfectly between the current vertex and one of the connectable vertices, the rotation of the patch can be easily
Algorithm 2 Breadth-First

1: **Input:** graph, filteredPatches, currentVertex
2: **Output:** patch or null
3: suitabPatches ← ∅
4: validConnectableVertices ← select connectable vertices
   ▷ graph vertices of degree 1 that are in ‘line-of-sight’ with currentVertex
5: for each vertex in validConnectableVertices do
6:   distanceGraphPair ← distance(currentVertex, vertex)
7:   for each patch in filteredPatches do
8:     pairs ← find connectable vertex pairs in patch
     ▷ at approximately distanceGraphPair of each other
     ▷ (i.e. within the snap radius)
9:     for each pair in pairs do
10:    rotation ← compute patch rotation
     ▷ such that the connection vertices in pair approximately coincide with
     ▷ currentVertex and vertex
11:   orientate patch according to rotation
12: if patch is valid then
13:   rating ← calculate the rating for patch and its pair
14:   add rating, pair and patch to suitablePatches
15: if suitablePatches.Count() > 0 then
16:   return the highest rated patch
17: else
18:   return null

calculated. However, a patch generally does not fit perfectly, thus a patch modification is unavoidable. There are multiple viable methods to calculate a suitable rotation which involve a patch modification. Three of them are:

1. Compute the rotation where the patch is scaled so that it fits perfectly between the two vertices.
2. Compute the rotation where the snappable edge is extended or shortened so that the patch fits perfectly between the two vertices.
3. Compute the rotation where the distance between the target connectable vertex of the patch and the graph is the shortest and then let the snap operation connects those vertices.

Even though the first method sounds reasonable due to retaining the angle and length ratio of the edges of the patch, it has one major problem, namely it can cause an undesired effect: patch too small or too big due to downscaling and upscaling respectively. The second method is better than the third method because it preserves the angles of the edges of the patch. The third method changes the angle and length of the edge due to the utilization of the snap operation to connect to the connectable vertex of the graph. However, the second
method is not applicable for all situations, see Figure 15.2. For this reason, we have chosen to use the second method where possible and the third method if the second one fails.

Figure 15 illustrates two situations where a patch modification and rotation is necessary. The calculation of the edge extension (method 2) can be seen as a triangle problem solving, where the length of two sides and an angle are known, and the length of the third side is in question.

(1) A situation where method 2 is applicable by first extending the edge between the target vertex of the patch (blue dot) and its adjacent vertex (purple dot).

(2) A situation where method 2 is not applicable due to the target of the graph vertex pair is closer than the adjacent vertex (purple dot) to the current vertex, thus method 3 is employed.

Figure 15: Two situations where rotation and modification of the patch (in the orange rectangle) are necessary to connect the graph vertex pair (current vertex (red dot) as start and a connectable vertex from the graph (green dot) as target.

Rating system

All suitable solutions found (patch, patch vertex pair, and graph vertex pair) are collected and rated. The rating system indicates the suitability of the solution for the situation at hand. To preserve the geometry of the patch as much as possible, the rating depends on the difference between the patch vertex pair distance and the graph vertex pair distance, and is given by

\[
1 - \frac{|distancePatchPair - distanceGraphPair|}{distanceGraphPair}
\]  

with \(distancePatchPair\) as the patch vertex pair distance and \(distanceGraphPair\) graph vertex pair distance. Ultimately, the patch and its pair with the highest rating is selected as the most suitable patch for this approach. If there are multiple pairs with identical rating, a random pair is selected.
4.4.4 Depth-first method

The main purpose of this method is to find a suitable patch that can be appended at the current vertex, to grow the network in the propagation direction with some deviation. The pseudocode of this method can be found in Algorithm 3.

Algorithm 3 Depth-First

1: **Input**: graph, filteredPatches, currentVertex, propagationDirection
2: **Output**: patch or null
3: suitablePatches ← ∅
4: for each patch in filteredPatches do
5:   for each connectableVertex in patch.connectableVertices do
6:     attachDirection ← small variation on propagationDirection
7:     orientate patch so that the adjacent edge to connectableVertex is aligned with attachDirection
8:     if patch is valid then
9:       rating ← calculate the rating for patch and its connectableVertex
10:      add rating, connectableVertex and patch to suitablePatches
11:   if suitablePatches.Count() > 0 then
12:     return the highest rated patch
13:   else
14:     return null

Rating system

Unlike the breadth-first method, only one connectable vertex of the patch is used to append it to the current vertex, one rating for each of its connectable vertices is calculated. Per patch, the base rating for each connectable vertex is initialized at 1 and, subsequently, the rating process adjusts it according to a number of situations, as follows:

- for every other connectable vertex of the patch nearby a dead-end edge of the graph decreases the rating by 0.2;
- for every snappable edge of the patch nearby a non-connectable vertex of the graph decreases the rating by 0.2;
- for every dead-end edge of the patch that is not nearby an edge of the graph decreases the rating by 0.2 otherwise increases the rating by 0.1.

The first two specifications reduce the rating of that particular patch attachment, as it could obstruct subsequent propagation. The last specification lowers the rating to avoid the use of cul-de-sac patches in an open area where it can create obstacles for the propagation. However, it increases the rating to promote the use of cul-de-sac patches where there is a limited area, thus having only a little to no effect on the propagation. The start rating,
4.5 Discussion

Our patch-based method utilizes patch semantics and flexibility to generate a plausible urban road network that contains interesting road structures which cannot be created using a parametric-based method. But the approach does utilize a parametric-based growth as a fail-safe mechanism to propagate where no patch is usable.

During the cell initialization process, a pattern can be used to initialize the main cell. A pattern enforces a certain (repeatable) design which can enrich the road network. Besides using a pattern as an initialization, it is also possible to employ patterns during the propagation. Even though the latter is not an option for our current approach, it is one of the interesting things to keep in mind for the future to improve our approach.

Initial patches are necessary to initialize a main cell using patches from the boundaries. However, it is possible that the given collection of patches does not contain such patches, causing the inability to initialize. To avoid this issue, a standard set of initial patches is defined.

The breadth-first method finds a suitable patch that connects the current vertex and one of the connectable vertices from the graph. The keyword here is a suitable patch and not the best possible patch. The current method only seeks one connectable vertex, however a patch can have multiple connectable vertices. Thus a possible improvement is to find and select a patch that connects multiple connectable vertices.

Despite the selection methods and rating systems to find a suitable patch, it is possible that local cells, which are considerably larger than other neighbouring cells and also look unusual, are generated. Figure 16 shows an example of such a cell. One presumption is due to the restrain to not snap to any patch edges to retain the geometry of the patch.

The patch-based generation can be controlled using parameters e.g. for the rating system, by excluding tags and specifying maximum occurrences of a patch or patches with a specific tag. However, it is not possible to place a patch at a certain location or to assure that a patch is used at least once.

decrease and increase values are relative due to every patch is handled similarly. However, it is important to note that the rating is penalized more heavily, e.g. higher decrease value than the increase value, to avoid situations where propagation can be hindered.

Eventually, the patch (if any) with the connectable vertex having the highest rating is selected as the most suitable.

4.4.5 Patch insertion

Given the patch as a result from the previous process, this insertion aligns the patch according to the rotation or direction and appends it to the graph by connecting the selected connectable vertex of the patch with the current vertex of the graph. Even though there are two different approaches, the insertion process is identical for both approaches. This is possible due to the utilization of the snap operation, which is used at least once if the patch is selected using the breadth-first method.
Figure 16: A populated main cell. The green area represents an unusual local cell generated using the patch-based method.
Chapter 5

Prototype implementation

This chapter describes the implementation of the application that incorporates the approach outlined in the previous chapters and also the functions which are necessary or supplementary for the described approach. The application is written in the C# programming language using Windows Presentation Foundation (WPF) for the GUI and QuickGraph, a graph library.

5.1 Graph structure

QuickGraph provides directed or undirected graph data structures and algorithms. For the application an undirected graph was chosen, because street directions are irrelevant for this project. This graph represents the road network and it is made up of vertices and edges where each vertex and edge has a number of properties, which can be seen in Table 2.

Table 2: Edge and vertex property

<table>
<thead>
<tr>
<th>Entity</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>Id</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Connected vertices</td>
<td>The vertices that the edge connects</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>The edge type: main, local, bridge</td>
</tr>
<tr>
<td>Vertex</td>
<td>Id</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Position</td>
<td>The coordinate of the vertex</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>The vertex type: main, local, bridge</td>
</tr>
<tr>
<td></td>
<td>Patch name</td>
<td>The name of the patch if applicable</td>
</tr>
</tbody>
</table>

There are other type forms, such as boundary, promenade and park, however these belong to the main type.
5.2 Application pipeline

5.1.1 Initialization

An area is created using the value specified by the user. Surrounding this area, boundaries are created. These boundaries ensure that the road network generation does not exceed the defined area. Additionally, a quadtree with the same size of the defined area is created and for each added road segment (edge) a quadtree item, which represents the road segment, is added. The quadtree data structure is used to efficiently find nearby edges and vertices.

5.2 Application pipeline

The pipeline of the application is divided into three main processes, the landmark configuration, road network generation and post-process. Figure 17 shows the mentioned pipeline. The input of this pipeline includes parameter values or a preset, patches and patterns if applicable. However, parameters have to be (re)specified to create main or local streets or a different road network style. Patches can be extracted (Section 5.6.2) or manually created (Section 5.6.3) and patterns can also be created (Section 5.6.4). For every step, user interactions are needed. The output of this pipeline is a graph and a heightmap representing the terrain. Using the output data, a 3D environment can be generated using an external software.

This pipeline, however, is not straight forward. River and hill landmarks have to be generated before the road network generation, because these landmarks have an effect on the road network generation. After the landmark generation, streets can be generated according to the approach described in the previous chapters. Any post-process can be used after each main or local street generation. There are multiple post-processes, namely: cell repair, which subdivides an unusual cell into smaller but proper cells, removal of dead-ends, creation and removal of edges and vertices, which allow for a finishing touch on the generated road network. If the road network is fully generated, parks can be created. This creation is only available in this last stage to avoid any unnecessary obstacles for the road generation. More information about landmark generation is given in Section 5.3 and the cell-repair post-process in Section 5.7.
5.3 Landmark creation

Landmarks can be divided into two types, area- and object-typed landmarks. Rivers, hills, parks, squares, forests and glaciers can be seen as area-typed landmarks, while monuments, towers and radio masts as object-typed landmarks. In this project, the focus lies on the first landmark type as it affects the road network generation and can be essential for the terrain creation as well as visual importance. To avoid requiring maps as input such as a water map or height map, user interactions are utilized. We have created tools for the application to allow the user to easily create rivers, hills and parks. The implementation and creation of these landmarks are described below.

Rivers

The river generation method is a fairly similar method to the one described by Smelik et al. [2011]. Given a set of points defined by the user, which forms a polyline, first create a smoother polyline using catmull-rom spline on the polyline. For each point created by catmull-rom spline, draw a circle around the point. The radius of the circle depends on the user input. To create the river, simply find the outline of the drawn circles. This outline can be seen as a promenade along the river which is also defined as promenade type edges in the graph. The actual river is the area surrounded by these edges. Additionally, the user can specify the maximum number of bridges on the created river. To avoid complex terrains, rivers cannot be placed on hills (elevated area) and this also applies the other way around.

Hills

The hill generation is a simplified smooth hill generation described by de Carpentier and Bidarra [2009]. To create hills, the user simply uses the provided brushing tool. The radius and strength of the brush can be specified by the user, in addition a max height option is also available to limit the height of the hills. When a brush instance is applied, the circular area with the specified radius is affected. There is a fall-off ramp towards the outer radius to create a parabolic effect on the circular area. When there are overlapping brush instances, a small value is added to the old height value.

Parks

To create a park, the user has to define a polygon by providing a set of points. Before adding the park to the graph, catmull-rom spline is used to smoothen the curves of the polygon. The following procedures are executed to add the park to the graph:

- Add every edge of the polygon to the graph as park type edges
- Create vertices on every intersection between the graph and park edges
- Remove every vertex and edge inside the polygon
- Snap every vertex that is nearby a park type edge to the closest point on that edge
5.4 Parametric-based generation

The parametric-based generation is based on a graph growing method and populates the area according to the defined parameters. Table 3 shows the important parameters that affect the generation and Figure 18 illustrates most of the parameters, other parameters can be found in Appendix A.1. Depending on the input values, different road network styles, such as grid, rectangular and organic, can be created. Additionally, a high-level setting, namely a preset, which is a set of predetermined parameter values of a certain road style, is provided to simplify the configurations of a road network using parameters. This allows non-experts to quickly generate road networks or to use a preset as a starting point for modifications.

Table 3: Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street length</td>
<td>The length of the road segment</td>
</tr>
<tr>
<td>Street length deviation range</td>
<td>The length deviation range of the road segment</td>
</tr>
<tr>
<td>Minimum street length</td>
<td>The minimum street length allowed</td>
</tr>
<tr>
<td>Vertex Degree range</td>
<td>The vertex degree range has a minimum and maximum. This range is used to define the number of road segments at a given vertex.</td>
</tr>
<tr>
<td>Minimum street angle</td>
<td>The minimum angle between two adjacent edges</td>
</tr>
<tr>
<td>Angle deviation range</td>
<td>The angle deviation range for a road segment in %</td>
</tr>
<tr>
<td>Snap radius</td>
<td>The radius for the snap operation</td>
</tr>
</tbody>
</table>
Prototype implementation 5.4 Parametric-based generation

Figure 18: An illustration of generation parameters. The street length and its deviation range define the length of the road segment being created. The angle of the segment depends on the vertex degree, the angle of the adjacent edges and the angle deviation range. The degree of a vertex is constrained by the vertex degree range specified by the user. However, when there is no growth possible, a vertex may end up having a lower degree than the minimum (leaving e.g. a dead-end). In this illustration, a [2-4] degree range is used. The vertex in green has degree 2, and the vertices in red, degree 3 and 4, which are also connection vertices. The snap radius is used for a snap operation.

The generation procedures are given below:

- Starting from a coordinate, this can be manually defined in the center or randomly chosen if there is a river or hill in the center, a new (candidate) vertex is created at this coordinate.

- From this vertex, propose road segments one by one. The amount of proposed road segments is a random value of the vertex degree range (this value is referred to as the selected degree).

  - The angle of a proposed road segment depends on the selected degree and the angle deviation range percentage. The angle is calculated by dividing 360 by the selected degree (this value is referred to as the main angle) minus a deviation. The deviation is a random value of the deviation range of the main angle. An example: a selected degree of 4 will result in a main angle of 90°, an input angle deviation of 10% will result in a deviation of 9, which means that the angle of the proposed road segment lies between 81 and 99.

    * If the vertex has an original degree of 0, then propose a segment where the angle between the segment and the horizontal axis (as 0°) is the angle calculated above.

    * If the vertex has an original degree of 1, the procedure is similar to the one above except the adjacent edge of the vertex being used instead of the horizontal axis.
If the vertex has an original degree of 2 or more, first find the largest angle between two adjacent edges. If this largest angle is greater than the calculated angle plus the minimum angle, then propose a segment using the calculated angle.

- The length of a proposed road segment is the street length minus a deviation. The deviation is a random value of the street length deviation range, e.g. a street length of 50 and street deviation of 10 will create a road segment with a length between 40 and 60.

- Generation constraints (Section 5.5) are applied for every proposed road segment. If a proposed road segment is valid, then the segment and its vertex are added to the graph. The new vertex is also added as a candidate vertex.

- Repeat the procedures above for every candidate vertex until there is no more candidate vertex left.

Figure 19 illustrates the generation in a high-level manner. After each generation, the minimum cycle basis technique described by [Eberly] is applied on the road network to find main or local cells. The generation of main and local streets differs only in the parameter values and some of the generation constraints, as described in the next section.

Figure 19: Parametric-based generation. (1) Given a starting coordinate and an area, create a vertex (red dot) on the coordinate. (2) Create new road segments and their vertices one by one from the previously created vertex. (3/4) This process is repeated for all new candidate vertices until the road segments reach the boundary.
5.5 Generation constraints

For the generation of plausible road networks, it is important that every parametric-based road segment, as well as every patch selected by the patch-based method, fits within the network being generated. For the sake of plausibility, a variety of customizable parameters and thresholds have been defined, e.g. snap radius, minimum street angle, minimum street length and vertex degree range.

We defined the following six constraints on a segment (or patch) that help enforcing segment validity and increase its overall plausibility: (i) a proposed road segment must connect to a connectable vertex or edge that lies within the snap radius of the end of the road segment; (ii) the angle between two road segments sharing a vertex must be higher than the minimum street angle; (iii) the length of a road segment must be longer than the minimum street length; (iv) the vertex degree may never exceed the maximum degree of the specified range; (v) a bridge must be created to cross a river; and (vi) the steepness of the road segment must be within the threshold. To abide by these constraints, the following operations checks are defined, and applied if applicable:

1. **Connect** the road segment to the nearest vertex (Figure 20.1) or create an intersection to the nearest edge (Figure 20.2) within the snap radius of the target vertex of the proposed road segment. However discard the road segment if any of the following cases is valid:

   - The angle between the proposed and existing edges is lower than the minimum street angle
   - The length of any of the split edges is shorter than the minimum street length
   - The target vertex has the maximum degree allowed

2. **Clip** the road segment when intersecting another edge (Figure 20.3). Similar to the procedure above, if case 1 or 3 is valid, then discard the proposed road segment.

3. **Create** a bridge to reach the other side of the river (Figure 20.4). This only applies if the proposed road segment is a main street (local streets do not cross a river) and if there is a promenade type street within the snap radius. First connect or snap the proposed road segment to the nearest promenade vertex or edge, then starting from the connected or snapped vertex, find the coordinate (and create a vertex on this coordinate) on the other side of the river where the distance is the shortest, and at last, create the bridge to connect these vertices.

4. **Discard** the road segment if the steepness is above the user defined threshold. Each street type has its own threshold, this allows e.g. main streets to have a higher threshold than local streets (Figure 20.4). Due to the downscale of the heightmap, a bilinear interpolation is applied to find the height on a coordinate. To improve the precision of the road segment steepness inspection, the proposed road segment is divided into smaller road segments and for each of these smaller segments, the steepness is examined. The formula of the steepness (in percentage) of a road segment is:

\[
\frac{|height_{Start} - height_{Target}|}{length_{RoadSegment}} \times 100
\]

(5.1)
If the steepness of any divided road segment is higher than the user defined threshold, then the proposed road segment is discarded.

5.6 Patch-based components

This section describes four of application details that are relevant to the patch-based method: special cases, patch extraction and creation, and pattern creation.

5.6.1 Special cases

There are two special cases during the patch selection process, the first case is the use of roundabout patches and the second case is the use of terminal patches. The first case applies for both selection methods, while the second case only applies for the depth-first method. One typical characteristic of a roundabout is that its road segments are usually connected, thus there is no dangling road segment. To retain this characteristic, a roundabout patch can only be used if all of its snappable edges can be snapped or connected to the nearby edges or vertices of the graph. For the second case, a terminal patch is only used if its area is not
comparably smaller than the available work area. This is to avoid leaving substantial empty area.

### 5.6.2 Patch extraction

The application has a patch extraction function, where its purpose is to extract patches from the input OpenStreetMap files (OSM). The idea of this extraction is comparable to the one described by Nishida et al. [2015]. The mapping between the OSM highway type and the graph street type is shown in Table 4. The patch extraction process is shown in Figure 21. A limitation to the road segment length avoids the extraction of inappropriate, such as non-urban-like, patches. In addition, limitations can be imposed on the extractor, e.g. limiting the maximum number of vertices and the maximum number of non-dead-ends, to discard very large or dense patches.

#### Table 4: Road type mapping

<table>
<thead>
<tr>
<th>Road type</th>
<th>OSM highway type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main street</td>
<td>primary, primary_link, secondary, secondary_link, trunk</td>
</tr>
<tr>
<td>Local street</td>
<td>residential, living_street, tertiary, tertiary_link</td>
</tr>
</tbody>
</table>

![Figure 21: A demonstration of the patch extraction.](image)

Figure 21: A demonstration of the patch extraction. (1) Given an area from an example road network graph. (2a) Loops or circular shapes are detected (blue circle) and added as a patch, then (2b) the expansion is applied on the patch. This is simply checking every adjacent edge of the vertices of the patch and if the length is below a certain threshold, then add the vertex and its edge to the patch. (3a) The remaining vertex (red dot) with at least three adjacent edges is selected to create a new patch, then (3b) the expansion is also applied.

When extracting a patch, some of the patch semantics can be taken directly from the OSM file, while others need to be calculated. Table 5 shows the implementation details of some of the patch tags.
**Table 5: Patch tags implementation detail**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roundabout</td>
<td>The patch contains a circle. To find a circle, use minimum cycle basis technique to find cycles and use circular approximation to identify if a cycle is a circle</td>
</tr>
<tr>
<td>Loop</td>
<td>The patch contains a cycle</td>
</tr>
<tr>
<td>Cul-de-sac</td>
<td>The patch contains at least one non-connectable vertex with degree 1.</td>
</tr>
<tr>
<td>Curved</td>
<td>The patch contains at least three consecutive edges where the delta angle of each edge pair is small</td>
</tr>
<tr>
<td>Terminal</td>
<td>The patch contains only one connectable vertex</td>
</tr>
<tr>
<td>Initial</td>
<td>The patch contains straight main streets with connectable vertices at both ends and an attachment along the streets</td>
</tr>
</tbody>
</table>

### 5.6.3 Patch creation

In addition to supplying example maps to extract patches, the user can also create a patch according to his or her preferences using our patch editor. To make the patch creation easier, tools are provided to create specific forms and a simple edge can also be created by connecting two vertices. Currently, we have implemented forms such as circles, rectangles and squares.

After creating the vertices and edges of the patch, the user can specify each edge type, e.g. main or local. Each vertex with 1 degree can also be assigned as a dead-end or a non-dead-end. Vertices with more than 1 degree are automatically dead-ends, because connections are only available through connectable vertices to retain the geometry of the patch. Assigning the dead-end or non-dead-end of an edge is unnecessary due to the fact that the adjacent edge of a non-dead-end vertex is a non-dead-end edge and other edges are considered as dead-ends.

Since the graph is planar, the patch has to be planar as well. To enforce this feature, a vertex will be created on every edge intersection when constructing an edge. After creating a patch, the patch semantics identification process is executed to extract the additional semantics.

### 5.6.4 Pattern creation

Next to the patch editor, a pattern editor can be used to create base patterns, which consist of one or multiple patches. The base patterns have a number of properties that can be seen in Table 6 and 7 and examples in Figure 22 and 23. Using these properties, modifications can be made to the patches, e.g. scaling and flipping (X- or Y-axis) and how the patches are connected, e.g. collinear to the previous patch or collapsing a connection edge.

To create a base pattern, the following procedures must be performed by the user, unless stated otherwise:
Table 6: Base pattern information

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection vertices</td>
<td>The start and end vertices of the pattern</td>
<td>int, int</td>
</tr>
<tr>
<td>Method</td>
<td>The usage of the base pattern: on street or between streets</td>
<td>enum</td>
</tr>
<tr>
<td>Segments</td>
<td>Parts of the base pattern</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Base pattern segment information

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patch</td>
<td>The main component of the segment</td>
<td>patch</td>
<td>Figure 22.1</td>
</tr>
<tr>
<td>Connection vertices</td>
<td>The connection vertices between the previous and new segment</td>
<td>int, int</td>
<td>-</td>
</tr>
<tr>
<td>Scale</td>
<td>The scaling of the segment</td>
<td>double</td>
<td>Figure 22.2a/b</td>
</tr>
<tr>
<td>Repetition vertices</td>
<td>The start and end vertices of the segment</td>
<td>int, int</td>
<td>-</td>
</tr>
<tr>
<td>Repetition</td>
<td>The number of times the segment is repeated</td>
<td>int</td>
<td>Figure 22.3</td>
</tr>
<tr>
<td>Segment flip</td>
<td>The modification of the segment: none, flip X-axis or flip Y-axis</td>
<td>enum</td>
<td>Figure 22.4a/b</td>
</tr>
<tr>
<td>Collinear</td>
<td>The connection collinearity between the previous and new segments’ connection edges</td>
<td>bool</td>
<td>Figure 23.4</td>
</tr>
<tr>
<td>Collapse</td>
<td>The collapsing of the connection edge of the new segment on the previous segment</td>
<td>bool</td>
<td>Figure 23.5</td>
</tr>
</tbody>
</table>

- At least one segment must be added.
  - A patch must be selected. Any patch can be used to create a segment, however it is up to the user to define good segments. Furthermore, there is also an option to use a created base pattern as a segment.
  - (Optional) The segment can be scaled (Figure 22.2a and 22.2b).
  - (Optional) The segment can be repeated using the repeat operation. To use this operation, the repetition vertices and the number of repetition must be defined (Figure 22.3).
  - (Optional) The segment flip property can be used to flip the segment (Figure 22.4a and 22.4b).
  - (Optional) The collinear and collapse properties can be used to modify how the new segment is attached to the previous segment (Figure 23.4, 23.5 and 23.6).
  - The connection vertices must also be defined if more than 1 segment is added. These vertices are used to attach the previous segment with the new segment. However, the new segment can only be connected to the last added segment.
5.6 Patch-based components

Prototype implementation

Figure 22: Examples of the segment adjustment. (1) The selected patch for a new segment. (2a/2b) The patch is upscaled and downscaled respectively. (3) The patch is repeated three times. (4a/4b) The patch is flipped on the X-axis and Y-axis respectively.

- The connection vertices and the method of the base pattern must be specified.

The pattern placement has been described in Section 4.3. However it is also possible for the user to manually specify the placement. An edge of a main cell can be manually chosen for the pattern initialization on a main street. In the case of the pattern initialization between main streets, the user has two options, namely: on a vertex or/and a position on an edge of the main cell.
5.7 Post process - Cell repair

This process aims at repairing unusual local cells by subdividing it into multiple cells. Unusual cells refer to cells that are too large in comparison to the neighbour or too concave. Unusual cells are created due unconnected vertices during parametric-based generation. This process is divided into two phases: the cell detection and cell subdivision.

Figure 23: Examples of the segment connection. (1) An added segment of the base pattern. (2) The selected patch for a new segment. (3) The selected patch is added as a new segment to the base pattern with no additional operation. (4) Adding the segment with the collinear property is true. (5) Adding the segment with the collapse property is true. (6) Adding the segment with collinear and collapse properties are true.
5.7.1 Cell detection

This phase detects unusual cells based on two values: the ratio between the cell area and the convex cell area and the ratio between the cell area and the local cell average area, which is the area of the main cell divided by the amount of local cells in the main cell. However, if a local cell contains a patch edge, this cell is skipped to avoid any modification to the patch. A cell is detected as unusual if at least one value exceeds the threshold.

The aim of the first ratio is to find cells that are too concave while the second is to find cells that are significantly larger than the average local cell size within the main cell. After the detection, the subdivision phase can be applied on the detected cells.

5.7.2 Cell subdivision

This subdivision splits the unusual cell into smaller but proper cells. The procedures of the subdivision are given below and Figure 24 illustrates an example of this subdivision.

- For each vertex, propose a road segment to other vertices that are not directly connected to the vertex

- Check if the angles of the proposed road segment and adjacent edges are not too acute. If this passes then check if the split polygons satisfy the threshold. If so, then add the road segment to the graph, thus subdividing the cell. The first road segment that passes the checks will be used to subdivide the cell

- If the first method does not work, then instead of connecting to vertices, try to connect to the nearest point of the edges of the cell

Figure 24: An example of the cell subdivision. (1) Given an unusual cell (green area) that is detected during the detection phase. (2) Multiple road segments (dotted lines) are proposed to subdivide the cell, but only one segment passes the checks. (3) The valid segment is added to the cell and it is split into two smaller cells.
One of the focuses of this project was to have an intuitive and controllable approach and a part which could be reflected in the graphical user interface (GUI). A GUI is created where the user can easily select what street type to generate, e.g. main and local and what landmarks to create, e.g. rivers, hills and parks. The controls, e.g. parameters, of a street generation or landmark creation can also be seen at a glance.

Furthermore, every executed operation, e.g. landmark creation and street generation, and its generated graph are recorded. Using this information, we provided a revert operation where the user can go back and forth to a previously used operation. A screenshot of the application can be seen in Figure 25. The application is divided into 5 components:

![Screenshot of the application](image)

**Figure 25:** A screenshot of the application. The application has five components: the red rectangle shows the top component, green shows the menu bar, blue shows the work panel, purple shows the work canvas and black shows the revert panel.

The *top component* contains the open and save buttons. This allows the user to save and load the graph. If a graph is saved, a xml file and a heightmap are created. These can be used to model a 3D environment.
The menu bar contains the buttons for each operation category. Clicking on one of the buttons will open the work panel for that category.

The work panel contains parameter fields and buttons that can be used to execute a function.

The work canvas displays the generated graph. This canvas is interactable when a specific function is executed, for example when the "Select region" button is clicked, the application will automatically show cells (areas) on the canvas and each cell can be selected. Additionally, panning is allowed on this canvas.

The revert panel displays all executed operations. Any operation can be selected and a revert to that operation can be done.
Chapter 6

Results and performance

In this section we present a variety of results achieved by our patch-based and parametric-based method, and we analyze the results of our patch-based method and the controllability of the parametric-based generator in order to discuss its expressive range. All results were generated on a computer with an Intel Core i5 processor.

6.1 Patch-based method results

We have a collection of 98 patches, manually created and extracted from real world data, as input for the patch-based generation. A part of the patch collection can be seen in Appendix A.2. Figure 26 illustrates a road network using all available patches and Figure 27 shows road networks with the exclusion of various patch tags.

Table 8 shows results related to the generated road networks in Figures 26 and 27. There are three notable points that can be seen from the table:

1. The differences in the generation time. The generation of a road network using a small patch collection is faster than using a large one. This is due to a higher number of patches would mean more patches need to be evaluated during the patch selection process.

2. The amount of unique patches used is significantly less than the total amount of available patches. This can be partly due to the special case patches, such as terminal and roundabout patches, which can only be used in certain situations. In addition, it can be that some patches, e.g. T and + (simple) structures, are more likely to be chosen than patches with somewhat more complex structures. Therefore, as mentioned in Chapter 4, one way of controlling the output network using the patch-based method is by setting a limit on the frequency of occurrence of a patch. By doing this limitation, e.g. limiting each patch to occur only once, increases the amount of unique patches, although typically decreasing the total amount of patches used.

3. The total amount of patches used is comparable for all networks. The second road network (excluding curved patches) has a considerable lower collection of patches,
yet it uses a comparable total amount of patches to other exclusions. We have seen that in such cases, even though using less unique patches, there are often a few among them that can fit easily in many situations.

The generation time using patch-based generation is dependent on the approximation of the work area to prune large patches, as described in Section 4.4.2. This means that, for the same area covered, it takes longer to generate a road network with a few large main cells than a road network with many smaller cells. As an example, for an area of $4 \text{ km}^2$, 

Figure 26: A road network with an area of $4 \text{ km}^2$ generated using the complete patch database. Main streets (red lines) were generated using the parametric-based method; local streets were generated using both patches (blue lines) and, when needed, the parametric-based method (black lines).
6.1 Patch-based method results

Figure 27: Patch-based generated road networks with an area of 4 km$^2$ generated with specific patch tags excluded.

- (1) Curved patches excluded
- (2) Straight patches excluded
- (3) Loop patches excluded
- (4) Cul-de-sac patches excluded

Generating a patch-based road network with our prototype system takes, on average, 40 seconds if it is divided into 4 main cells, and 23 seconds, if it is divided into 16 main cells.
6.2 Parametric-based method results

The road network styles that can be generated using the parametric-based method are diverse. Figures 28 and 29 show a couple examples of the road network styles. The parameter values that were used to generate the road networks in the figures can be found in Table 9 and 10 respectively.

![Uniform square style](image1)
![Uniform rectangle style](image2)
![Irregular style](image3)

Figure 28: Grid style road networks with an area of $1 \text{ km}^2$.

Table 8: Generation data on the road networks in Figures 26 and 27

<table>
<thead>
<tr>
<th>Tags</th>
<th>Number of patches</th>
<th>Time (s)</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Excluded</td>
<td>Available</td>
<td>Unique</td>
</tr>
<tr>
<td>None</td>
<td>98</td>
<td>58</td>
<td>381</td>
</tr>
<tr>
<td>Curved</td>
<td>39</td>
<td>29</td>
<td>354</td>
</tr>
<tr>
<td>Straight</td>
<td>59</td>
<td>38</td>
<td>335</td>
</tr>
<tr>
<td>Loop</td>
<td>75</td>
<td>44</td>
<td>362</td>
</tr>
<tr>
<td>Cul-de-sac</td>
<td>84</td>
<td>52</td>
<td>345</td>
</tr>
</tbody>
</table>

Table 9: Grid presets

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Square</th>
<th>Rectangle</th>
<th>Irregular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street length</td>
<td>50</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Street length deviation range</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Minimum street length</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Street width</td>
<td>0</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>Street height</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Snap radius</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Vertex degree range</td>
<td>4-4</td>
<td>4-4</td>
<td>4-4</td>
</tr>
</tbody>
</table>
Results and performance

6.2 Parametric-based method results

Figure 29: Organic style road networks with an area of 1 km²

Table 10: Organic presets

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Loose</th>
<th>Dense</th>
<th>Firm</th>
<th>Suburban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street length</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Street length deviation range</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Minimum street length</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Vertex degree range</td>
<td>4-4</td>
<td>5-5</td>
<td>4-4</td>
<td>4-4</td>
</tr>
<tr>
<td>Snap radius</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Minimum angle</td>
<td>55</td>
<td>55</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td>Angle deviation range</td>
<td>30</td>
<td>30</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 11: Parametric-based generation performance

<table>
<thead>
<tr>
<th>Area(m)</th>
<th>#Vertices</th>
<th>#Edges</th>
<th>Total length (km)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000x5000</td>
<td>12000</td>
<td>20000</td>
<td>900</td>
<td>4</td>
</tr>
<tr>
<td>10000x10000</td>
<td>50000</td>
<td>80000</td>
<td>3600</td>
<td>25</td>
</tr>
<tr>
<td>15000x15000</td>
<td>110000</td>
<td>180000</td>
<td>8200</td>
<td>70</td>
</tr>
</tbody>
</table>

The first column shows the area of the road network. The second, third and fourth columns show the approximation of the created total number of edges, total length of streets and total number of vertices respectively. The last column shows the time needed to generate local streets in seconds excluding the time needed to extract the cells.

Each road network was generated in approximately 1 second. Table 11 provides more information of the performance on this generation.

6.2.1 Expressive range

The controllability of our generator is one of the important aspects of this thesis. One way to measure this controllability is to analyze its expressive range. The expressive range indicates the variety of its output content that can be created using the available control mechanisms, e.g. generation parameters. For our approach, we found out that any attempt at measuring and analyzing the expressive range of the patch-based method would be first and foremost dependent on the actual collection of patches, more than on any actual controllable settings. Therefore, we decided to focus the analysis on our parametric-based generator. Similarly to Smith and Whitehead [2010], we analyzed the expressive power of our approach by following four steps: determine the comparison metrics, generate content, visualize the expressive range and analyze impact of parameters.

Comparison metrics

It is important that the chosen metrics measure the features of the road networks. For these metrics, we have chosen the street connectivity and density as these two metrics have a clear meaning and can be precisely defined. For example, a road network with high connectivity means that it is typically 'straightforward' to go from A to B, while a dense road network means that it contains a high amount of streets within an area.

Dill [2004] describes a number of measurements to measure the street connectivity. For our study, we chose the Connected Node Ratio (CNR) as our metric, because it is based on the notion of dead-ends, which contribute to the disconnectivity of a road network, and it has a low computational cost:

\[
CNR = \frac{\#\text{intersections}}{\#\text{intersections} + \#\text{deadends}}
\]  

A high CNR value indicates a relatively low amount of dead-ends compared to the amount of intersections and, thus, a higher level of connectivity. According to some researchers,
Results and performance

6.2 Parametric-based method results

e.g. Benfield [1999] and Schlossberg et al. [2006], a higher connectivity contributes to more walking and biking, which might be useful when designing an urban road network. As for the street density of a road network, which can be visually perceived on a map, is defined as the total street length (in km) per area unit (in km²).

Content generation

Our generator has a wide variety of parameters that can be used to control the output road network, these parameters can be seen in Table 3. For this analysis, we have chosen to generate road networks by varying the minimum street angle and the vertex degree range because these two parameters have a strong impact on the output. We varied the minimum street angle from 50° to 90° with increments of 10° (an angle lower than 50° would be too acute, and higher than 90°, too restrictive). For the vertex degree range, we start at a [2-3] range, a go up to a [4-5] range. For the analysis, all other generator parameters were kept fixed with the following values: street length (50m), street length deviation range (10m), minimum street length (20m), snap radius (20m) and street angle deviation range (10%).

Output visualization

We show the expressive range of the road network generator using hexagonal bin plots, see Figure 30. For each of the plots, we generated 1000 road networks covering an area of 1 km². Each hexagonal bin indicates the amount of networks with fairly similar connectivity and density scores. Plots on the same row share the vertex degree range, but use increasing minimum street angles, while plots on the same column share the minimum street angle, but use different vertex degree ranges. All generated road networks have a connectivity ranging between 0.6 and 1, and a density ranging between 15 and 55.
Figure 30: Expressive range of the parametric-based generator, varying the vertex degree range and the minimum street angle. For each plot, the connectivity (y-axis) is measured by the Connected Node Ratio, see Equation (6.1), and the density (x-axis) is the total street length per $km^2$. 
Impact of generation parameters

Figure 30 shows the expressive range for the generator. From this figure, we can make a number of observations on the impact of the parameters used on the generated road networks:

- Focusing on each column, thus with a fixed minimum street angle, we notice that:
  - The higher the vertex degree range, usually the denser the graph, while the connectivity stays relatively steady.
  - There is a drop of density from vertex degree range $[2-5]$ to $[3-3]$. The reason for this is that the range $[2-5]$ allows up to 5 edge connections for each vertex, and the range $[3-3]$, only up to 3; as a result, in general, more edges can be generated using the former.
  - The last column (for a minimum street angle of $90^\circ$) shows a rather steady density, in contrast to the other four columns. The reason is that this minimum street angle in practice restricts the maximum degree of a vertex to 4, occurring only when the angle between two edges is exactly $90^\circ$ (which does not happen very often). This is not the case for the other columns: with lower minimum street angles, there is more flexibility, and thus a wider margin for street density variations.

- Focusing on each row, thus with a fixed vertex degree range, we notice that:
  - The higher the minimum street angle, the lower the connectivity of the graph and usually also the lower the density.
  - The first row has relatively steady (low) density scores, as the connectivity scores decrease. The reason for this is that, due to the already low vertex degree range $[2-3]$, the road network is already quite sparse, so that a higher minimum street angle only causes less snapping, and hence more dead-ends.

From the information above, we can conclude that the minimum angle and vertex degree range have a noticeable impact on the street connectivity and density. A road network with high density and connectivity can be achieved by specifying a high vertex degree range and a low minimum angle, while a road network with low density and high connectivity can be generated using a low vertex degree range and a low minimum angle. Defining a low vertex degree range and a high minimum angle will create a road network with low density and connectivity. Using in-between values will generate a road network with moderate density and connectivity.

However, as can be seen in the figure, the generator does not cover the whole space defined by the two metrics, namely: the bottom right area. This is an area with high density and low connectivity, which means road networks with a high number of total street length and also a relatively high number of dead-ends. In terms of road networks, having such a configuration seems quite non-plausible, as it would mean having e.g. many unconnected streets twisted very close to each other. We managed to generate such networks by forcing
6.3 Combination Results and performance

an unlikely low snap radius and high minimum street length, but the output is an improbable and non-realistic road network. In other words, the ‘reluctance’ of the generator to create such networks is an intended consequence of the design goal of promoting the generation of plausible road networks only.

In conclusion, based on these two metrics, we can say that the expressive range of our road network generator is quite large, and that the features of its output can be easily controlled by parameters such as minimum street angle and vertex degree range.

6.3 Combination

One of the possibilities of the approach is to create a road network with different road network styles and also using different generation methods. Figure 31 shows such a road network. After the generation of a road network, the graph can be saved, this will generate a xml file and a heatmap. An external software, Sceelix (SCX), a procedural engine with a visual node-based editor, can be used to model the created files into a 3D environment. In addition to the built-in nodes, a number of nodes have been implemented in Sceelix for this 3D modeling process. The node graph created in Sceelix can be seen in Appendix A.3. Figures 32 and 33 illustrate the 3D environment of the road network shown in Figure 31 from different point of views.
Figure 31: A road network with different styles on a terrain with a river and hills. The local streets on the left of the river are generated using the patch-based method, while on the right of the river the parametric-based method has been used. (1) This cell was initialized using a pattern on a main street. (2) Initialized using a pattern between main streets. The other non-numbered cells are each initialized using patches on main streets. (3) Generated using parameter values from Table 12. (4) Generated using the uniform square preset from Table 9. This cell also contains a park. (5) Generated using the irregular preset values from Table 9. (6) Generated using the uniform rectangle preset from Table 9. (7) Generated using the organic suburban preset from Table 10. (8) Generated using the organic firm preset from Table 10. (9) Generated using the organic dense preset from Table 10.
Table 12: Parameters of a sparse road network style.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street length</td>
<td>50</td>
</tr>
<tr>
<td>Street length deviation range</td>
<td>10</td>
</tr>
<tr>
<td>Minimum street length</td>
<td>20</td>
</tr>
<tr>
<td>Vertex degree range</td>
<td>2-3</td>
</tr>
<tr>
<td>Snap radius</td>
<td>20</td>
</tr>
<tr>
<td>Minimum angle</td>
<td>50</td>
</tr>
<tr>
<td>Angle deviation range</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 32: A 3D environment of Figure 31 with the road network.
Figure 33: A 3D environment of Figure 31 with generic buildings.
Chapter 7

Usability testing

We employed a usability test on the application which incorporates our approach to evaluate the intuitiveness of our approach. Using this test, we wanted to have an indication if the proposed high-level settings, e.g. presets, and tools, e.g. landmark generations, are easy to use and helpful for generating road networks.

Participants

We have tested our application with 14 participants, eight of them are students with a psychology background and the other six with a background in computer science (cs). Computer science students have a grasp of the concept of a geometric graph, while psychology students do not. However, all of them have more or less no experience with a road network generation application or a road network generator prior to this test.

Tasks

The usability test consisted out of a set of tutorials and two assignments (Appendix A.4.1) that had to be done using the application. We have chosen to let the participants follow a set of tutorials to get familiar with the application and the generation process. The focuses of the assignments are to create a terrain using the provided landmarks tools and generate different road network styles. Each participant follows the same procedures.

Measures

To collect the data, we used a screen and voice recorder to capture the mouse movement and comments of the participants. Additionally, we also observed the participants during the test. After finishing the assignments, each participant was asked to fill in a questionnaire. The questionnaire can be found in Appendix A.4.2. To asses the usability of the application, we use the System Usability Scale (SUS) questionnaire. The SUS questionnaire is a type of Likert scale (1-5) that is designed to measure an application’s usability. However, we removed the first question (“I think that I would like to use this application frequently”) of the SUS questionnaire because it was irrelevant in our case. In addition, we have added several questions to have an indication of how easy or difficult it was to create landmarks
Usability testing

and generate and modify road networks. Besides this data, we also recorded the completion time for each assignment.

Results and analysis

We present a number of results that we observed during the test and recorder as follows:

- Most participants had difficulties using the revert operation for the first time. What the operation does is reverting the road network to the selected operation, thus selecting and reverting to the last operation will not change anything. Most participants thought that selecting the last operation will undo the operation, thus reverting to the operation before the selected operation.

- Some participants unintentionally generated local streets instead of main streets due to the usage of the wrong tab (see Figure 25 to see what is meant by a tab). This was mainly caused because the work panels of both tabs (local and main street generation) are not distinctive enough.

- Creating edges and removing vertices and edges were proven to be challenging to some participants due to the difficulty to select a vertex or an edge. Selecting a vertex or an edge needs to be precise, meaning the participants have to zoom in on the work canvas to click on a point or line, which most of them did not realize.

- Computer science students had their way around the application better than psychology students.

- While working with patches, some participants tried to create a roundabout, e.g. by selecting only the roundabout tag, as it is the most familiar patch tag term out of the five terms. However, as mentioned in Chapter (and section) 4.5, this is not possible.

- During the assignment, the participants were asked to create hills where streets cannot cross. Some participants tried to specify the steepness threshold as mentioned in Chapter 5.5 for a cluster of hills and using a different threshold for another cluster of hills. However, this is not supported in the current version of the application.

Table 13 shows the means and standard deviations of the questionnaire’s results and the completion times. For the SUS questionnaire part, we used the scoring described by Brooke et al. [1996] to compute the SUS score. However, because we removed the first question, we scored this question 3 out of 5, which is the average of the Likert scale, for every participant. Based on the results from the table, we present a number of findings:

- The mean of the SUS score for all participants is 67, which is slightly lower than the usual SUS score mean (68). We employed a one sample t-test and found a p-value of 0.8344, which means that the difference is considered not to be statistically significant. Employing a one sample t-test on the SUS score of each background against the mean, we found a p-value of 0.3185 and 0.5111 for psychology and computer science respectively, and neither shows a significant difference. Comparing the backgrounds
Table 13: Test results

<table>
<thead>
<tr>
<th>Background</th>
<th>SUS</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
<th>Q15</th>
<th>Q16</th>
<th>Time A</th>
<th>Time B</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>67</td>
<td>4.5</td>
<td>4.4</td>
<td>4.6</td>
<td>3.9</td>
<td>4.6</td>
<td>4.3</td>
<td>4.4</td>
<td>537</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>18.2</td>
<td>0.52</td>
<td>0.93</td>
<td>0.63</td>
<td>1.07</td>
<td>0.51</td>
<td>0.91</td>
<td>0.65</td>
<td>154.53</td>
<td>62.66</td>
</tr>
<tr>
<td>Psychology</td>
<td>62</td>
<td>4.3</td>
<td>4.0</td>
<td>4.5</td>
<td>3.9</td>
<td>4.6</td>
<td>4.3</td>
<td>4.5</td>
<td>620</td>
<td>264</td>
</tr>
<tr>
<td></td>
<td>16.1</td>
<td>0.46</td>
<td>1.06</td>
<td>0.76</td>
<td>0.99</td>
<td>0.52</td>
<td>1.04</td>
<td>0.53</td>
<td>119.58</td>
<td>44.12</td>
</tr>
<tr>
<td>CS</td>
<td>74</td>
<td>4.8</td>
<td>4.8</td>
<td>4.8</td>
<td>4</td>
<td>4.6</td>
<td>4.3</td>
<td>4.3</td>
<td>427</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.41</td>
<td>0.41</td>
<td>0.41</td>
<td>1.26</td>
<td>0.55</td>
<td>0.82</td>
<td>0.82</td>
<td>128.64</td>
<td>29.76</td>
</tr>
</tbody>
</table>

This table shows the mean (upper) and standard deviation (lower) scores of all participants and participants with a specific background. The second column is the SUS score, the third to sixth columns are the scores (from a minimum of 1 and maximum of 5) of the corresponding questionnaire questions that can be found in Appendix A.4.2 and the last two columns are the completion times of the assignments in seconds.

using an unpaired t-test, a p-value of 0.2406 is calculated, which is considered to be not statistically significant. These results show that there is no significant differences between the SUS scores.

- The mean scores of generating a road network easily (Q10) and quickly (Q11), the handiness of presets (Q12), modifying a road network easily (Q13) and creating landmarks easily, such as rivers (Q14), hills (Q15) and parks (Q16), are relatively high and the spread of the scores, e.g. coefficient of variation (ratio between the standard deviation and mean), is relatively low. This shows that even though the application’s usability is close to the mean, road networks generation and modification, and landmarks creation are perceived to be easy.

- Comparing the mean SUS score and the mean completion time using the Pearson correlation coefficient between the participant’s SUS score and the completion time of assignment A and assignment B (Table 14). We found that in general, there is a weak correlation between the SUS score and completion time of assignment A and a moderate correlation for assignment B. However, these correlations are not significant according to the p-value, thus the SUS score and the completion times are not significantly correlated. What we noticed during the observation, which might explain this weak correlation, is that some participants have better insight into how to solve the assignment than other participants and this is unrelated to the usability of the application.
Table 14: Pearson correlation coefficient between SUS score and completion times

<table>
<thead>
<tr>
<th>Participants</th>
<th>Variable A</th>
<th>Variable B</th>
<th>Pearson coefficient</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>SUS score</td>
<td>Time A</td>
<td>-0.11</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>SUS score</td>
<td>Time B</td>
<td>-0.28</td>
<td>0.33</td>
</tr>
<tr>
<td>Psychology</td>
<td>SUS score</td>
<td>Time A</td>
<td>-0.08</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>SUS score</td>
<td>Time B</td>
<td>-0.40</td>
<td>0.43</td>
</tr>
<tr>
<td>Cs</td>
<td>SUS score</td>
<td>Time A</td>
<td>0.37</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>SUS score</td>
<td>Time B</td>
<td>0.60</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Based on the findings above, we can conclude that the usability test indicates that the developed application has approximately an average usability score (according to the SUS score), but a part of its intuitive mechanisms, such as high-level settings and tools, to easily create landmarks and generate and modify road networks are perceived well by the participants. Besides this conclusion, we have a number of points based on our observations that can be used in the future to improve our application, e.g. allowing the placement of a selected patch on a specific position and allowing different steepness thresholds for each cluster of hills. We also found out that there lacks significant differences between participants with different background, while there are differences observed. One possible reason is due to the low number of participants, however significant differences might be revealed when more participants are involved.
Chapter 8

Conclusion

Coming back to the research question we formulated in Chapter 1:

- How can procedural parametric-based and patch-based methods be integrated into a controllable and intuitive approach to generate fictive urban road networks?

We presented a semantic approach, which combines the advantages of a patch-based method with those of a parametric-based method, to generate fictive road networks with plausible complex and simple structures. For our patch-based method, the semantics of each individual patch is identified, represented, stored and utilized during the generation. The process of first trying to propagate the road network by appending patches using the patch-based method, and employing parametric-based method as a fall-back mechanism, when no suitable patch can be found, combines both methods into an integrated solution.

Using the application which integrates our approach, patches can be extracted from an example map or even manually created. Our patch-based generation is controllable by selecting the desired patch semantics, e.g. by excluding one or multiple patch tags, and by limiting the number of occurrences a patch can be used. Our parametric-based generator is configurable using various parameters, which allow for the generation of a variety of road network styles. The generator has been shown to have a wide expressive range. In order to facilitate its use by non-experts, one can define any number of presets that can be used to easily generate different types of road networks. Furthermore, to avoid requiring input maps, such as heightmap or watermap, tools are provided to easily create landmarks. The high level settings, such as presets, and tools are perceived to be useful and easy to use for non-experts based on the usability test. Hence, our contributions are: (i) the use of patch semantics to help guiding a patch-based road network generator, (ii) a controllable parametric-based road generation method with a high expressive power, and (iii) a high level settings scheme that allows non-experts to easily create and modify road networks.

Currently, our patch-based generation approach has two limitations. First, it is not possible to place a patch at a certain location, nor to assure that a patch is used at least once. Second, despite the two patch selection methods and rating mechanisms to find a suitable patch, it still sometimes happens that quite large and complex local cells are generated.
Conclusion

Although, there are a few directions for future work. First, in addition to using patterns for initializing a main cell, it would be interesting to explore how they could also be used during the propagation process. Second, the breadth-first method currently only seeks one connectable vertex when searching for a suitable patch. However, because some patches have multiple connectable vertices, it might be worth exploring how one could seek and select a patch that connects multiple connectable vertices.
Bibliography


Appendix A

Glossary

A.1 Parameters

Table 15 shows a list of parameters that the user can specify.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>The area of the road network, which is a squared area</td>
<td>int</td>
</tr>
<tr>
<td><strong>Hill</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxMainSteepness</td>
<td>The maximum steepness of a main street in %</td>
<td>int</td>
</tr>
<tr>
<td>maxLocalSteepness</td>
<td>The maximum steepness of a local street in %</td>
<td>int</td>
</tr>
<tr>
<td>maxHeight</td>
<td>The maximum height of a hill, 0 means no maximum</td>
<td>int</td>
</tr>
<tr>
<td>hillRadius</td>
<td>The radius of a hill</td>
<td>int</td>
</tr>
<tr>
<td>brushStrength</td>
<td>The strength of the brush</td>
<td>int</td>
</tr>
<tr>
<td><strong>River</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>riverRadius</td>
<td>The radius of the river</td>
<td>int</td>
</tr>
<tr>
<td>numberBridges</td>
<td>The maximum number of bridges that the river can have</td>
<td>int</td>
</tr>
<tr>
<td><strong>Patch-based (rating)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>patchMinimumRating</td>
<td>The lowest patch rating allowed</td>
<td>double</td>
</tr>
<tr>
<td>patchDeadEndIncrease</td>
<td>The rating increment value for a dead-end vertex or edge</td>
<td>double</td>
</tr>
<tr>
<td>patchDeadEndDecrease</td>
<td>The rating reduction value for a dead-end vertex or edge</td>
<td>double</td>
</tr>
<tr>
<td>patchNearbyDecrease</td>
<td>The rating reduction value for a non-dead-end vertex or edge</td>
<td>double</td>
</tr>
<tr>
<td><strong>Patch-based (special cases)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>patchMaxTerminalRatio</td>
<td>The maximum terminal ratio</td>
<td>int</td>
</tr>
<tr>
<td>patchRoundabout</td>
<td>If true, allow the use roundabout patch whenever it is possible</td>
<td>bool</td>
</tr>
<tr>
<td>patchTerminal</td>
<td>If true, allow the use terminal patch whenever it is possible</td>
<td>bool</td>
</tr>
<tr>
<td><strong>Patch-based generation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>patchMirror</td>
<td>If true, also use the mirror of each patch to find a patch with higher rating</td>
<td>bool</td>
</tr>
<tr>
<td>patchBreadthRotationThreshold</td>
<td>The maximum deviation from the propagation angle.</td>
<td>int</td>
</tr>
<tr>
<td>patchDepthRotationThreshold</td>
<td>Used for phase 1: Breadth-propagation.</td>
<td>int</td>
</tr>
<tr>
<td>patchMinWorkAreaSize</td>
<td>The minimum size of work area approximation</td>
<td>int</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>patchMinDeltaDegree</td>
<td>The minimum angle between two adjacent edges. Used to check if the largest adjacent degree is large enough to find the propagation angle.</td>
<td>int</td>
</tr>
<tr>
<td>patchRatingPhase1</td>
<td>If true, also apply phase 2 rating system for phase 1 to find a better fitting patch</td>
<td>bool</td>
</tr>
<tr>
<td>repairMaxConvexRatio</td>
<td>The maximum convex ratio</td>
<td>double</td>
</tr>
<tr>
<td>repairMaxAreaRatio</td>
<td>The maximum area ratio</td>
<td>double</td>
</tr>
</tbody>
</table>
A.2 Patches

The figures below show some of the used patches ordered by the tag.

Figure 34: Roundabout tag patches. Blue dots indicate connectable vertices, while no dot denotes dead-end vertex.
Figure 35: Loop tag patches.
Figure 36: Cul-de-sac tag patches
Figure 37: Straight tag patches
Figure 38: Curved tag patches
A.3 Sceelix

Figure 39: The Sceelix' node graph.
A.4 Usability test

A.4.1 Assignments

Assignment A - using parameters/presets

Create a street network of 3000x3000m on a terrain that:

1. Has a cluster of hills where local streets can pass through (streets are placed on the hills)

2. Has a cluster of hills where local streets cannot pass through (streets are not placed on the hills, with exception to the borders of the hills)

3. Has a river with at least two bridges. This river divides the area in two smaller areas of comparable sizes.
   
   a) One region contains grid local street pattern styles and without any dead-ends.
   
   b) The other region contains organic local street pattern styles and dead-ends.

4. Has any main street pattern.

5. Has a park nearby a bridge

Assignment B - using patches

Create a street network of 1000x1000m that has a square/uniform grid main street pattern with 500 street length resulting in 4 regions. Within each resulting region, create the following local street networks:

1. One region contains no straight patches

2. One region contains no curved patches

3. One region contains no loop patches

4. One region contains any kind of patches
A.4.2 Questionnaire

<table>
<thead>
<tr>
<th>Application</th>
<th>strongly disagree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The application was unnecessarily complex</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2. The application was easy to use</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3. I would need the support of a technical person to be able to use this application</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>4. The various functions in this application were well integrated</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5. There was too much inconsistency in this application</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>6. Most people would learn to use this application very quickly</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>7. The application was very cumbersome to use</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>8. I felt very confident using the application</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>9. I needed to learn a lot of things before I could get going with this application</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

| Street generation (specific)                                                                                                             | 1                 | 5              |
| 10. It was easy to generate a street network                                                                                        | 1                 | 5              |
| 11. I am able to quickly generate a street network                                                                                     | 1                 | 5              |
| 12. The presets were handy to use                                                                                                     | 1                 | 5              |
| 13. It was easy to modify the street network                                                                                        | 1                 | 5              |

| Landmark (specific)                                                                                                                    | 1                 | 5              |
| 14. It was easy to create a river                                                                                                     | 1                 | 5              |
| 15. It was easy to create hills                                                                                                       | 1                 | 5              |
| 16. It was easy to create a park                                                                                                      | 1                 | 5              |

<table>
<thead>
<tr>
<th>Assignments</th>
<th>very difficult</th>
<th>very easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>17. Overall, how difficult or easy did you find assignment A</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>18. Overall, how difficult or easy did you find assignment B</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 40: The questionnaire