

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Mgr. Jan Beneš

Procedural Modeling and Realism in Computer Graphics

Department of Software and Computer Science Education

Supervisor of the doctoral thesis:	doc. Ing. Jaroslav Křivánek, Ph.D.
Study programme:	Informatics
Study branch:	Computer Graphics and Image Analysis

Prague 2019

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....
signature of the author

Title: Procedural Modeling and Realism in Computer Graphics

Author: Mgr. Jan Beneš

Department: Department of Software and Computer Science Education

Supervisor: doc. Ing. Jaroslav Křivánek, Ph.D., Department of Software and Computer Science Education

Abstract: One of the long-standing goals of computer graphics is to generate realistic imagery. This realism is predicated, among other things, on having access to high-quality content to display, and being able to verify its realism. Unfortunately, realism and especially realism in procedural modeling are under-represented in current literature. Consequently, we address these two topics by investigating the definitions of general realism in computer graphics and by providing a basic framework for thinking and talking about realism in procedural modeling. In order to provide this framework with initial tangible data, we present a user study of the realism of procedural buildings. We find that realism of this specific class of procedural models is carried on different scales and conclude that authors of procedural rules for buildings need to beware of not only the fine details, but also the overall shape of the generated models. Finally, we present a method for growing procedural road networks in and close to cities. We use a traffic simulation and adapt existing research to grow cities over time. The cities are formed by letting several meaningfully positioned smaller settlements' minor roads grow in step with additional major roads. As a result, we are able to generate cities with rich, complex and realistic road networks.

Keywords: computer graphics, procedural modeling, realism, procedural buildings, procedural cities

Acknowledgements

First and foremost, I would like to thank my supervisor, Jaroslav Křivánek, for his unrelenting support, patience, and willingness to explore and wander off the beaten path. Aside from being a great supervisor, he has also always been very accepting and supportive of both my professional work and my personal growth. Without him, this chapter of my life would not be possible. Thank you!

The companionship and friendship of Ivo Pavlík had kept my spirits high throughout the many weekends and late evenings. I had also very much enjoyed sharing the office with and the friendship of many great people, especially Ondra Karlík, Petr Kmoch, Petr Kadleček, Martin Kahoun, Honza Horáček, Honza Kolomazník, Jirka Vorba, Martin Šik, and our great secretary, Radka Hacklová. Bernard Lidický had always been cheerful and supportive.

My 9 month stay at Yale was a once in a lifetime experience. It would not have been the same without Stephanie Sayson and Todd Harvey. In Zürich, I found great friends in Tom Kelly and Roman Miszkowicz. I will always cherish all of our common memories.

In the later years, Šimon Soták and Andrej Pančík have been very understanding friends and colleagues, enthusiastically cheering me on and always willing to work around my thesis's scheduling needs. Šimon also volunteered to provide feedback on the Chapters on realism.

Many other people have been my friends and have kept changing my life for the better during my thesis years. Thank you, Jirka N., Tomáš, Jirka H., Veronika, Oskar, Juan Carlos, Dan, Petra, Katka, Fanda, Patrik, and Bernard.

Last but not least, I'm indebted to my parents, Věra and Jiří, for their support, dedication, and sacrifice and my grandmother Věra for being the person she was.

This research was funded by the Charles University in Prague's Grant SVV-2016-260332, Charles University in Prague's project GA UK No. 581412, the Czech Science Foundation grant 16-18964S, and ESRI R&D Zürich.

Contents

1 Introduction **5**
- 1.1 Structure . . . 5
- 1.2 Contributions . . . 6
- 1.3 Publications . . . 7

2 Realism **8**
- 2.1 Elements of Realism . . . 9
- 2.2 Kinds of Realism . . . 12
  - 2.2.1 Full Realism . . . 12
  - 2.2.2 Functional Realism . . . 14
  - 2.2.3 Visual Equivalence . . . 14
  - 2.2.4 Plausibility . . . 15
  - 2.2.5 Summary . . . 16
- 2.3 Further Considerations . . . 17
  - 2.3.1 Pragmatic Choices in Realism . . . 17
  - 2.3.2 Culture, Individual, Historical Epoch, and Realism . . . 18
- 2.4 Photorealism . . . 21
  - 2.4.1 Photography and Photographs . . . 21
    - 2.4.1.1 Measurement Photography . . . 22
    - 2.4.1.2 Common Photography . . . 23
  - 2.4.2 Comparing Photographs . . . 23
    - 2.4.2.1 Comparing Common Photographs . . . 23
    - 2.4.2.2 Comparing Measurement Photographs . . . 24
  - 2.4.3 Realism in Photographs . . . 24
  - 2.4.4 Measurement-Photorealism . . . 25
  - 2.4.5 Perceptual-Photorealism . . . 26
  - 2.4.6 Summary . . . 27
- 2.5 Evaluation of Realism in Rendering . . . 28
  - 2.5.1 Comparing against Measurement Photographs . . . 28
  - 2.5.2 Path Tracing and Reference Images . . . 29
    - 2.5.2.1 Image Metrics . . . 30
  - 2.5.3 Evaluation of Rendering Algorithms . . . 31
- 2.6 A Neural Networks Perspective on Realism . . . 32
  - 2.6.1 Semantic and Stylistic Representations . . . 32
  - 2.6.2 Discriminative Networks as Realism Predictors . . . 34
- 2.7 Summary . . . 35

3 Realism in Procedural Modeling **37**
3.1 Procedural Modeling . . . 37
3.1.1 Goals of Procedural Modeling . . . 38
3.1.2 Types of Procedural Modeling . . . 40
3.1.3 Use Cases . . . 41
3.1.4 Procedural Rules and Methods . . . 42
3.1.5 Procedural Flow . . . 43
3.1.6 Representational and Full Models . . . 44
3.1.7 Generation of Models . . . 45
3.1.8 Summary . . . 47
3.2 Characteristics of Procedural Rules . . . 48
3.2.1 Procedural Range . . . 48
3.2.2 Desirability, Realism, and Plausibility . . . 49
3.2.3 Desirable Outputs . . . 50
3.2.4 Generative Power . . . 51
3.2.4.1 Precision . . . 51
3.2.4.2 Recall . . . 52
3.2.4.3 Discussion . . . 53
3.2.5 Summary . . . 53
3.3 Evaluation . . . 54
3.3.1 Rules and Methods . . . 54
3.3.2 Realism as a Score . . . 55
3.3.3 Content and Audience . . . 55
3.3.4 Asset Reuse . . . 55
3.3.5 Representational and Full Models . . . 56
3.3.6 Model-based and Image-based Evaluation . . . 57
3.4 Summary . . . 58
3.4.1 Contributions . . . 58
3.4.2 Future Work . . . 59

4 Realism of Architectural Procedural Models **60**
4.1 Introduction . . . 60
4.2 Previous Work . . . 61
4.3 Design Considerations . . . 61
4.3.1 Procedural Aspect . . . 62
4.3.2 No Reference vs. With Reference . . . 62
4.3.3 2D Images vs. 3D Models. . . . 64
4.3.4 Multiple Images at Once . . . 64
4.3.5 Structure and Fine Detail . . . 64
4.3.6 Datasets . . . 65
4.4 Method . . . 66
4.4.1 Stimuli – Image Sets . . . 66
4.4.1.1 Choosing Photographs . . . 66
4.4.1.2 Procedural Rules . . . 67
4.4.1.3 Generating CG Images . . . 67
4.4.1.4 Image Pre-processing . . . 68
4.4.2 Experiment . . . 69
4.4.2.1 Classification Screen . . . 70

4.4.2.2 Participants . . . 70
4.5 Data Analysis . . . 70
4.5.1 Quantitative Analysis . . . 70
4.5.1.1 Hypothesis H_{DIFF} . . . 71
4.5.1.2 Hypothesis H_{SCALE} . . . 71
4.5.1.3 Two- and Three-way Interactions . . . 72
4.5.1.4 Computer Generated to Photographs Ratio . . . 74
4.5.1.5 Summary . . . 74
4.5.2 Qualitative Analysis . . . 75
4.5.2.1 Debriefing . . . 75
4.5.2.2 Cut-Out Edges, Camera Angles, Background . . . 78
4.5.2.3 Other Factors. . . . 78
4.5.2.4 Images . . . 78
4.5.2.5 Participants . . . 79
4.5.2.6 Summary . . . 79
4.6 Results . . . 79
4.6.1 Applicability . . . 80
4.6.2 Recommendations . . . 80
4.7 Limitations and Future Work . . . 80
4.8 Acknowledgements . . . 81
4.9 Supplemental Material . . . 81

5 Procedural Generation of Urban Road Networks 82
5.1 Introduction . . . 82
5.2 Previous Work . . . 83
5.3 Our Method . . . 85
5.3.1 Overview . . . 85
5.3.2 Initialization and the Initial Road Network . . . 86
5.3.2.1 Rationale and Overview . . . 86
5.3.2.2 Technique . . . 88
5.3.3 Growing Minor Roads . . . 90
5.3.3.1 Rationale . . . 90
5.3.3.2 Technique . . . 91
5.3.4 Traffic Simulation and Growing Major Roads . . . 91
5.3.4.1 Rationale . . . 91
5.3.4.2 Technique . . . 92
5.3.4.3 Trade Routes over Existing Major Roads . . . 95
5.3.4.4 Trade Routes over New Major Roads . . . 95
5.3.5 User Input and Automation . . . 96
5.3.5.1 Traffic Function . . . 97
5.3.5.2 Neighboring Cities . . . 98
5.3.5.3 List of Pairs of Neighboring Cities . . . 98
5.3.5.4 Historical Context . . . 99
5.3.5.5 Direct Manipulation . . . 99
5.3.5.6 Summary . . . 100
5.4 Results and Discussion . . . 100
5.5 Limitations and Future Work . . . 103
5.6 Supplemental Material . . . 108

5.6.1 Parameters Used for Results 108
5.6.2 List of Symbols 108
5.6.3 Comparison of Existing Methods 113

6 Conclusion **114**
6.1 Summary 114
6.2 Future Work 115

Bibliography **116**

List of Figures **138**

List of Tables **141**

Chapter 1

Introduction

"I would have written a shorter letter,
but I did not have the time."
— Blaise Pascal

The objective of computer graphics is to display *content* on a computer screen or some other device. This content can be of various origins. It can be created manually, it can be measured, and it can be generated by a computer. This thesis is concerned with the last possibility, that is content that has been generated automatically using a computer program. The processes by which such content is generated and which we investigate are called *procedural modeling* or *procedural generation*.

Procedural modeling is peculiar in that it synthesizes new content with a partial lack of the human factor, possibly resulting in *idiosyncratic artifacts* that would not typically be present in man-made data. For example, a procedurally generated building or city will be unable to reflect all of the traditional and cultural aspects real buildings and cities would reflect, such as building codes, zoning decisions, or aesthetics. We postulate that with the exception of some unambiguous realms such as fluid simulation or plant growth, such artifacts are probably present in most procedurally generated content.

The nature of these artifacts and the breadth of the possible outputs of a procedural algorithm make it difficult to judge and validate the algorithm's *realism* and generative power. It is therefore somewhat unfortunate that only a limited amount of effort has been invested into understanding procedurally generated content, its realism, and its perception.

Our goal in this thesis is to address these above outlined issues and to contribute to the understanding of procedurally generated content, realism, and the understanding of their combination.

1.1 Structure

This thesis follows three main lines of investigation: 1) realism in computer graphics, 2) realism in procedural modeling, 3) realism of procedurally generated buildings in particular, and 4) procedural generation of urban road networks.

The topics are presented in what we believe is their most natural order. However, this order also happens to be the reverse chronological order to the one in which they were developed. This is because the chronologically first and second

topics – procedural road networks and realism of procedural buildings – both provoked new, more "basic" or "fundamental" directions of thought that we followed up on. Consequently, the later Chapters 4 and 5 do not always adhere to all of the earlier made observations and recommendations. However, the inverse chronological order in which the topics are presented should make the thesis and the topics easier to follow.

1.2 Contributions

The principal contributions of this thesis are as follows:

1. **Survey and discussion of realism in computer graphics.** We have surveyed existing definitions and approaches to realism in computer graphics and discussed their implications. We have also argued that there are often overlooked cultural elements to realism and have demonstrated that realism can be decomposed into three basic elements of realism, each of which is necessary for overall realism to be achieved. Previous work on realism in computer graphics – and to a degree, in general, is rather sparse, with the last notable discussions known to us being a paper by Ferwerda (2003) and an editorial by Reinhard et al. (2013).

2. **Survey and discussion of realism in procedural modeling.** To the best of our knowledge, we are the first to integrate and discuss the notions of realism and procedural modeling beyond the necessary minimum normally dedicated to this topic in the Introduction or Results sections of procedural modeling literature. We have also reviewed how realism in procedural modeling could be measured by adapting existing literature in other fields, how it could be evaluated in practice, and have established nomenclature.

3. **Evaluation of state of the art in procedurally modeled buildings.** We have created a state of the art dataset of procedurally generated buildings and proposed and used a novel methodology to conduct a user study that evaluated the perceived realism of these procedurally modeled buildings. This contribution is a practical implementation of the theoretical discussion from the previous contribution. Again, there is no previous work that we know of.

4. **A novel method for generating urban road networks.** We have adapted research from urbanism and sociology to devise a novel method that simulates the development of plausible road networks of cities and their surroundings. Unlike in previous methods, the resulting cities grow in the context of their neighboring cities, have minor roads that reflect historical epochs, grow meaningful new major roads, and focus growth around important areas of the road network, creating cities and urban areas with a more plausible structure than the previous state of the art.

1.3 Publications

Some of the content of this thesis is based on the two following publications:

- Jan Beneš, Alexander Wilkie, Jaroslav Křivánek. Procedural modeling of urban road networks. *Computer Graphics Forum*. Vol. 33. No. 6. 2014. doi: 10.1111/cgf.12283
- Jan Beneš, Tom Kelly, Filip Děchtěrenko, Jaroslav Křivánek, Pascal Müller, On Realism of Architectural Procedural Models. *Computer Graphics Forum*. Vol. 36, No. 2. 2017. doi: 10.1111/cgf.13121

The first publication is the basis of Chapter 5, whereas the second one is the basis of Chapter 4. Additionally, a very early version of the method presented in the first publication was submitted as the author's Master Thesis.

Chapter 2

Realism

> *"Painting is a science, and should be pursued as an inquiry into the laws of nature. Why, then may not landscape painting be considered as a branch of [physics], of which pictures are but the experiments?"*
>
> — John Constable, *Fourth Lecture at the Royal Institution, 1836*

Generating imagery that substitutes reality has been one of the longstanding goals of computer graphics. This pursuit of reality and realism is motivated by both intellectual curiosity and the need to present audiences that consume computer graphics with a real-world like impression of a virtual world. The successful results of this pursuit of reality are said to be "realistic" and to achieve "realism".

However, the meanings of the words "realism" and "realistic" are overloaded and mean different things in different contexts. As a result, it is *not always clear* what kind of realism is being discussed, achieved, or validated.

In this Chapter, we analyze how the notion of realism is used in the field of computer graphics. We limit our discussion to the effort of creating still 2D images of 3D scenes and 3D scenes that strive to appear realistic in one or more ways. Of special interest to us are the quality of scenes (geometry), materials (material appearance), and rendering. We will *not* discuss "non-realistic" rendering and realism in video, virtual reality, forensics, physical fabrication related material appearance, etc.

In this Chapter, our contributions are 1) identifying the three *Elements of Realism*: geometry, material appearance, and rendering, 2) analyzing the definitions of realism in existing computer graphics literature, 3) introducing and discussing concepts related to realism that are mostly left unmentioned and underdefined, especially the various cultural influences of realism, and 4) analyzing the approaches that current literature uses for validation and investigation of realism.

We hope the discussion in this chapter will provide a better framework for working with the notion of realism in computer graphics.

2.1 Elements of Realism

> *"[Color and fine modeling] always accompany each other, and it is these qualities which give to every masterpiece... the radiant appearance of living flesh."*
>
> — Auguste Rodin, *Art*

We will now discuss the individual factors behind realism in computer graphics. This decomposition helps create a mental framework for thinking about realism and allows us to discuss realism with added clarity.

In order to create an image that is *fully realistic* – one which observers agree is life-like – we have to have a realistic *scene* and a *rendering algorithm* capable of simulating the propagation of light in that scene. In practice, however, we cannot afford to have scenes with a microscopic level of detail (Dorsey et al. 2007, p. 6).

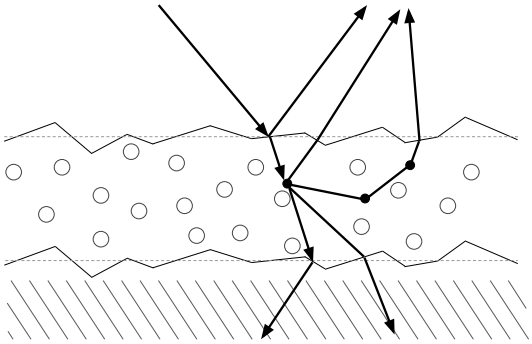
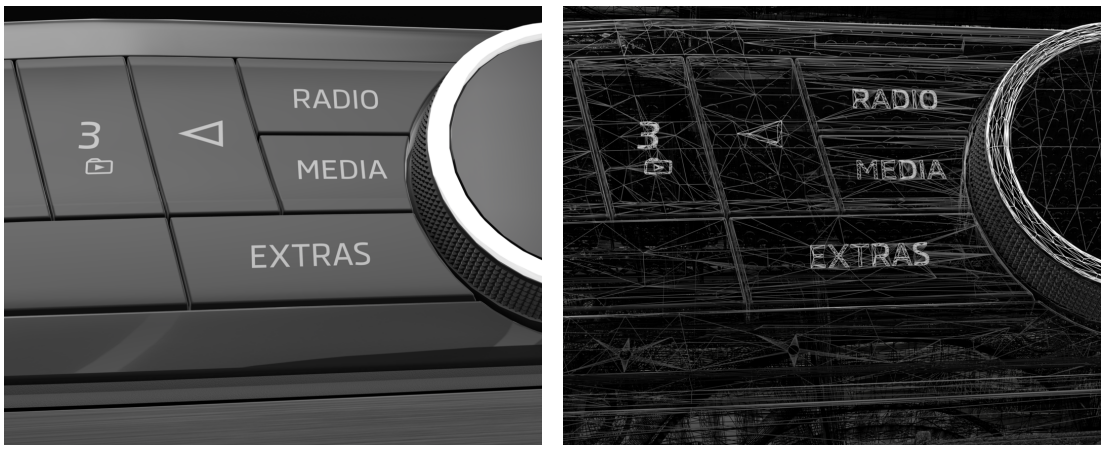


Figure 2.1: Geometry with a microscopic level of detail representing a surface with a thin layer of varnish. Arrows depict complex light propagation. Figure adapted from Hanrahan and Krueger (1993).



(a) Full rendering of the dashboard (b) Wireframe rendering of the dashboard

Figure 2.2: Full and wireframe renditions of the dashboard of Škoda Fabia. In this extreme example, even the small and flat details such as the anti-slip indentation and the button labels are represented in geometry rather than being represented using shaders and texture maps. Such a balance between geometry and material appearance is uncommon. By Škoda Auto, used with permission.

To this end, the scene is typically decomposed into two separate entities: *the high-level geometry* and its *material appearance* descriptions. For example, Figure 2.1 shows a scene with microscopic detail. Instead of allowing such a complex scene, a common approach would be to assume a smoother *high-level geometry* and approximate the microscopic detail with a possibly simplified *material appearance* description, e.g. Weidlich and Wilkie 2007. To contrast this, Figure 2.2 shows the opposite end of the spectrum, a rather extreme *practical* example of representing material appearance in geometry, with even the imprint on buttons represented by geometry.

With the above decomposition in mind, we can identify three *elements of realism*, each of which is crucial and needs to be present if *full realism* is to be achieved in practice:

1. *Geometry.* The *geometry* is the geometrical information representing a virtual collection of *shapes*, *objects* or *models*. Frequently, the geometry consists of triangles (faces) (Akenine-Möller et al. 2018, Section 1.2.2), but volumetric (Beyer et al. 2015) and parametric (Akenine-Möller et al. 2018, Chapter 13) representations are also commonly used.

2. *Material appearance.* Describes how light interacts with the geometry. Its purpose is to abstract away the geometric detail that cannot reasonably be modeled by the *geometry* itself – such as that in Figure 2.1 – and that influences the interactions between matter and light. Importantly, we can assign a specific material to any point or larger geometric entity on or in the *geometry*. Notably, we do not treat texture maps separately, but understand them to simply be a parametrization of material appearance. Note that in this setup, a light would be represented by its geometry and a material appearance representing its emissive properties. Also, we intentionally gloss over the rather complex technical details of material appearance modeling and refer the interested reader to Dorsey et al. (2007) for an in-depth treatment of the topic.

3. *Rendering algorithm.* An algorithm that combines the *geometry* and the *material appearance's* description of the geometry's interaction with light to generate an image.

If any of the above three elements of realism are not "realistic", we will end up with an image that is not *fully realistic*.

First, if we have geometry that represents an implausible object, no amount of realism in materials and in the rendering algorithm will help. An interesting illustration of this phenomenon can be found in the Velocipedia project (Gimini 2009). It explores how well people can draw simple sketches of bicycles from memory. Gimini recreated these sketches in the form of almost photorealistic photo-montages. Some of them, seen in Figure 2.3, show convincingly photograph-like images of impossible bike designs. Any sensible observer will, however, agree that these bike images are not realistic in their content and consequently are not fully realistic.

Similar arguments can be made for the combinations of a high quality scene and a realistic algorithm with just very basic materials, or a highly realistic scene with high quality materials, but with a poor rendering algorithm (Dorsey et al. 2007, pp. 77, 251). We illustrate those cases in Figure 2.4.



(a) Bike number 7

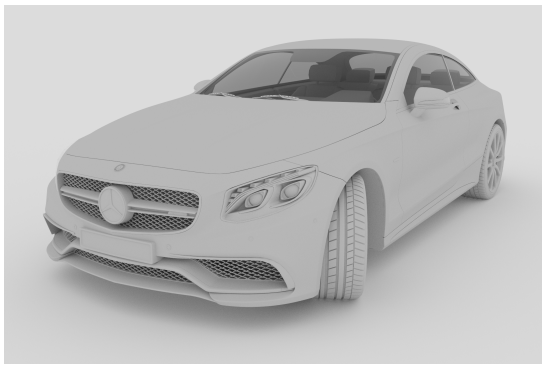


(b) Bike number 21

Figure 2.3: Convincing photo-montages depicting impossible bicycle designs. From Velocipedia, by Gimini (2009), used with permission.



(a) All three elements of realism present



(b) Only basic material appearance



(c) Only a basic rendering algorithm used

Figure 2.4: Comparison of (a) a scene with all three elements of realism present and the (b) same scene, with only a basic material appearance, and (c) the same full scene, rendered using a basic rendering algorithm. Rendered with help from Ondřej Karlík, used with permission.

2.2 Kinds of Realism

Above, we have defined *full realism* as an approach to realism that is concerned with how life-like or reality-like an image looks. For many people, a photograph, see Figure 2.5a, would be a typical example of an image that is fully realistic.

An additional way of looking at realism is by judging an image's *functional realism*, that is how well it is able to convey "*important properties of the scene*" (Ferwerda 2003, p. 5). A schematic diagram or a topological map, see Figure 2.5b, would be typical examples.



(a) Photograph (b) Topological map

Figure 2.5: (a) Photograph, an image most people would say has a high level of *visual realism*. Taken by the author. (b) Topological map, an image with low *visual realism* but high *functional realism*. Crop-out of an unofficial MBTA subway map by Kvrivishvili (2013), licensed under CC BY 2.0.

Full and functional realism are mostly found in different contexts within computer graphics. While full realism is more frequently used in rendering, functional realism is more relevant for procedural and manual modeling, where the information in the scene is more important than the way in which the scene is portrayed. Unfortunately, when presenting results, both these kinds of realism are often utilized at the same time. Specifically, when it is the content or geometry that is being evaluated for realism, e.g. a procedural model, it is often presented with highly realistic rendering and material appearance. Conversely, a rendering algorithm or a material appearance method would more often than not be presented on hand-modeled, functionally realistic scenes. This way, the *perceived* realism can leak between the individual elements of realism.

In order to better present this distinction, we provide the various definitions and more specific treatment for both full and functional realism in the two following Sections 2.2.1 and 2.2.2. We then extend the discussion by going over the related concept of *visual equivalence* in Section 2.7 and detail the distinction between realism and *plausibility* in Section 2.2.4.

2.2.1 Full Realism

The term *full realism* is an umbrella term that encompasses a variety of different but similar definitions. It is not well established and for the purposes of this thesis, we derive it from common English usage.

Dutré et al. (2006, p. 2) and Jensen (2001, p. 2) both define *realistic image synthesis* as generating images such that there is no noticeable difference between the generated image and a photograph of the scene the generated images are trying to reproduce. A similar definition is given by McNamara et al. (2000, p. 1), who defines the goal of realistic image synthesis as the generation of images that are perceptually indistinguishable from a real scene. Ramasubramanian et al. (1999), giving an equal definition, says an image and a photograph need to be "*visually*" indistinguishable, again emphasizing the perceptual nature of the definitions.

A more nuanced treatment of full realism is provided by Cohen and Wallace (1995, pp. 1–4). In their book, the goal of *image synthesis* is similarly defined as the re-creation of visual experience. Since the visual experience is a perceptual process in the human observer, Cohen and Wallace postulate that simply reproducing the physical stimulus should be enough, before conceding that no output device on which such computer values could be viewed exists and if it did, it might not be safe or practical. Finally, it is stated that the usual and practical goal of image synthesis is the reproduction of an image on a computer display, rather than the reproduction of the visual experience.

In Physically Based Rendering, the canonical contemporary book on algorithms for generating (fully) realistic images, Pharr et al. (2016, p. xx) say the book's accompanying implementation tries to achieve photorealism and state the term can be defined as either eliciting the same response from a human observer as looking at the actual scene would, or as generating images indistinguishable from a photograph. Finally, however, the perceptual approach is discounted and the authors limit themselves to "*an accurate simulation of the physics of light and its interaction with matter*" (Pharr et al. 2016, p. 4).

Another look at realism comes from Ferwerda (2003, p. 2), who provides two different definitions of full realism. The first one is *physical realism*, which should provide the same visual stimulation as the scene. In other words, physical realism is a recreation of the *spectral* irradiance incident at the observer's retina.

This definition places an interesting emphasis on the fact that to fully reproduce the realism for an individual observer, we need to reproduce the *spectral* irradiances, rather than just reproducing the RGB values that correlate with, but are going to be different from, any individual observer's cone response curves.

For *physical realism*, Ferwerda (2003, p. 5) also defines *accuracy* to be a measure of realism, expressed in terms of physical quantities describing the scene.

The second definition of realism provided by Ferwerda (2003, p. 3) is the definition of *photorealism*. Ferwerda says that photorealism tries to achieve the same visual response as the real scene, or that after processing by the visual system, but *before* cognitive processing, the same set of values is produced for both the original stimulus and the generated image.

Notably, working definitions of photorealism as well as other literature commonly understand photorealism to either measure the irradiance, or to account for full cognitive human perception, see Figure 2.8 for a visual comparison and Section 2.4 for more in-depth discussion. In this respect, Ferwerda's photometric approach differs from the way photorealism is commonly understood.

2.2.2 Functional Realism

Additionally, in a third definition of realism, Ferwerda (2003, p. 2) defines *functional realism* as providing the same visual information as the original scene. Here, the phrase "visual information" is to be understood as the semantic information conveyed in a visual manner, rather than the image's *full realism*, as defined in the previous Section. Ferwerda (2003, pp. 3–4) also provides an alternative definition, which defines an image as functionally realistic if it gives an observer enough information to perform a given task.

The *fidelity* of an image is then the degree to which an image conveys the semantic information necessary for a specific task or function (Ferwerda 2003, p. 4). Consequently, an image which provides enough semantic information is considered functionally realistic.



Figure 2.6: US Navy flight simulator, by Chris Howell (1998), in Public Domain.

In his paper, Ferwerda uses technical illustrations to convey the meaning behind functional realism, but also discusses flight simulators as examples. Following up on this example, Figure 2.6 shows an US Navy flight simulator from 1998. While the *full realism* (accuracy) is low, all of the required visual information is conveyed and allows the user to complete their piloting tasks, providing a high degree of *functional realism* (fidelity).

Ferwerda's definition of functional realism is also probably the most applicable to realism of generated and manually modeled content. For example, if we want to generate a piece of virtual terrain for simulation purposes, it will be the realism of the terrain's height-map (*functional realism*) that should be evaluated for realism, not the visual depiction (*full realism*) of the terrain itself. We discuss this further in Sections 2.1 and 3.1.6.

2.2.3 Visual Equivalence

Closely related to the concepts of functional realism and full realism is *visual equivalence* (Ramanarayanan et al. 2007, esp. Section 3.2), which adapts functional realism for use in rendering. Two rendered images, one correct and one with diminished accuracy, are considered. While they can be visibly different

when viewed side by side, the complexity of the optical effects involved precludes a human observer from telling which is which when viewed without reference. This is illustrated in Figure 2.7.



(a) (b) (c)

Figure 2.7: Visual equivalence. An image rendered with (a) a normal illumination map and (b) with a warped illumination map have the same appearance, therefore being *visually equivalent*, even though the Visible Differences Predictor (Daly 1992), seen in (c), reports there should be apparent differences. Reproduced from Ramanarayanan et al. (2007), used with permission.

With respect to the definition of functional realism, a *visually equivalent* image with diminished accuracy provides the observer with the same information as the original as far as telling the *full realism* of the scene is concerned.

2.2.4 Plausibility

At the beginning of Chapter 2, we have defined something to be *realistic* if it substitutes reality. However, such a definition is too narrow. It implies that the object or process we are trying to recreate needs to exist in the real world.

In contrast with this definition, we would oftentimes like to attribute realism to representations of objects and concepts that do not exist in the real world. The reasons for this non-existence could vary. These representations could depict things that have not happened but could have happened, for example, the Canadians being the first to the Moon in an alternative history movie. Or they depict things that could still happen, such as the Canadians visiting the Moon in the future in a movie set in the future. They could also represent things that happened but are no longer true, such as yesterday's weather and Sun position, or things that are completely imaginary, such as an alternate universe or a sci-fi world.

With this in mind, we define that something is *plausible* if it seems within reason, probable, and is hard to deny in a given context. Thus, instead of asking whether something exists in real life, we ask whether something could possibly exist considering the context in which an image or a scene are posed.

Plausibility mostly applies to an image's content and semantics. As can be deduced from the definition of full realism in Section 2.2.1, full realism implies the plausibility of geometry and material appearance. The inverse – that plausibility of all the three elements of realism: geometry, material appearance, and rendering, implies full realism – is, in the strictest sense, not true. For example, if a plausible but biased rendering algorithm is used, the result cannot be fully realistic. Nonetheless, we will use the term realism in a looser sense, where something that is plausible is also realistic.

Finally, while plausibility and functional realism can seem similar, they are not equivalent. With functional realism, we judge how well semantics and information is conveyed, while with plausibility, we judge whether something could exists. For example, an image of *today's* London – with that piece of context in mind – where cars drive on the right side might not be plausible, but it would be functionally realistic in that it conveyed the side on which cars drive. Conversely, an image without depth of field might be plausible in terms of all the three elements of realism, but not be functionally realistic with respect to determining the depth at which objects were placed.

2.2.5 Summary

The spectrum of relevant definitions of realism is quite broad. Additionally, the definitions are not always consistent, especially for *full realism*. Therefore, in Figure 2.8, we contrast the above discussed definitions in terms of the pieces of information being compared. For further information, see Reinhard (2008, Section 4, especially 4.6), who discusses in significant detail the various physiological and psychological representations of an image.

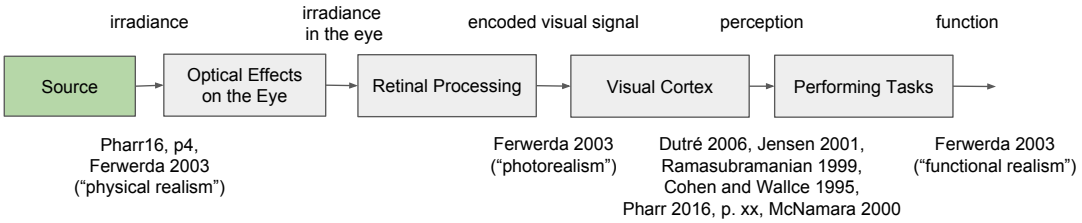


Figure 2.8: The different representations (arrows) and changes of representation (gray boxes) an image goes through. The various definitions of realism, cited at the bottom, differ in which representation of an image they compare. For example, "Retinal Processing" changes the representation from "irradiance in the eye" to "encoded visual signal".

Despite the differing definitions, three clear directions emerge. The first two are a result of decomposing *full realism*, the third one is *functional realism*:

- *Physically-based realism.* Recreation of the physical quantities incident on an image plane, without considering human perception or possible limits of the display medium.
- *Perceptually-based realism.* Recreation of the stimulus of the visual system, either before or after cognitive processing.
- *Functional realism.* Providing visual information necessary for a specific task or function, without emphasis on the form of visual depiction.

Notably, both *perceptually-based realism* and *functional realism* are challenging in that they encompass the human factor. We describe how perception of realism is influenced, among other factors, by experience and culture in more detail in Section 2.3.

2.3 Further Considerations

The purpose of this section is to discuss some of the external, non-technical effects that affect how realism in computer graphics is though of, treated, and how it is approached.

First, we discus the interplay between pragmatism and the ideal of perfect realism. We follow by postulating that there are inherent biases concerning realism in people whose work touches realism. Finally, we propose that there are several strong effects on realism in computer graphics that affect how realistic things appear across history, age, and culture.

This list of further things to consider is not exhaustive and perhaps not even very representative. Our discussion should be seen as a first attempt at a basic exposition of those issues within the field of computer graphics.

2.3.1 Pragmatic Choices in Realism

First, we return to the trade-off between geometry and material appearance. For ideal geometry and an ideal rendering algorithm, we would not require a material appearance description. In the most extreme case, the geometry would describe the individual atoms and its particles and the rendering algorithm would be able to simulate the underlying physical processes as the light propagates through the scene. Properties such as the index of refraction or whether the material behaves as a dielectric would naturally emerge from such a simulation.

However, such a scenario is all but impossible by today's standards. We therefore approximate a lot of the fine detail, including things such as color or roughness, using mathematical functions and texture maps. For most applications, this is a desirable trade-off. Rushmeier et al. (2000) discuss this point in more detail. The trend of replacing detail with texture maps, however, can also go the other way. Some materials make use of various deformations maps (Akenine-Möller et al. 2018, Section 6.7) that try to mimic small but not microscopic detail without changing the geometry. Further yet, there are techniques such as geometry and displacement shaders (Akenine-Möller et al. 2018, Section 3.7; Apodaca and Gritz 1999, Section 8.2) that can add actual geometric detail to the underlying model and make its level of detail more appropriate for the given resolution.

Next, an ideal rendering algorithm would try to reproduce the true results of light being propagated through the scene. However, it is quite common for rendering algorithms to adopt some simplifications. For example, the wave nature of light (Moravec 1981; Musbach et al. 2013) is frequently not simulated. Additionally, even though spectral rendering has been seeing increasing adoption, for example in Weta Digital's renderer (Fascione et al. 2017, Chapter 7), the images generated by most rendering algorithms are not spectral. Instead, they are usually in linear RGB space and are therefore of little use for predictive purposes (Wilkie et al. 2009) or to render specific optical effects (Sun et al. 2001, p. 2) such as polarization, interference, and diffraction (Weidlich and Wilkie 2008; Weidlich and Wilkie 2009; Wilkie et al. 2001).

Also, in order to render an image, we need to use a camera model. Simple pinhole cameras, as well as more advanced models (Pharr et al. 2016, Section 6.4; Kolb et al. 1995) are typically used where especially perceptual approaches

might prefer a simulation of the human eye. While there have been efforts to use the human eye as a camera (Barsky 2004; Deering 2005), their use seems to be in ophthalmology rather than in graphics.

All of these choices are driven by practical considerations. It makes little sense to invest human labour and computational time to model realism at a level of detail that is higher than the level of detail required for the intended use. Additionally, the human labour or computational time are constraints that are often of higher importance than the realism of the final work. Many of the works referenced in this thesis address these topics by studying the perceivable difference between images, see Sections 2.4.2 and 2.5.2, and trading realism for computation time and human labor, Sections 2.2.3 and 2.4.5.

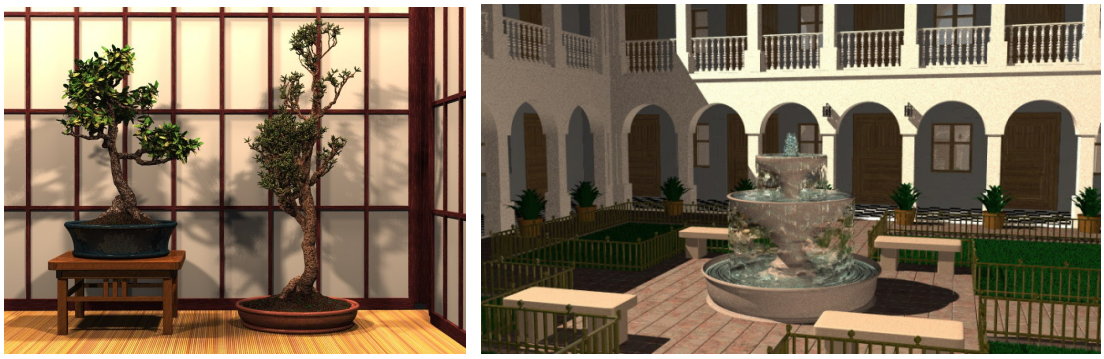
Finally, choosing the resolution of the image, whether to render spectrally or in RGB, and that we use a camera model instead of a model of the human eye, see also Section 2.4, might seem rather technical. These decisions, however, also have perceptual implications, changing the level of realism that can be achieved.

To summarize, this section demonstrated two things. First, there is always a trade-off between the level of perceived realism we want to achieve, computational resources, and human labor. And second, even results that people would consider highly realistic do in fact make numerous concessions to practicality.

2.3.2 Culture, Individual, Historical Epoch, and Realism

In this Section, we propose that culture, individual and collective experience, and historical epoch affect *perception* of realism.

First, we argue that perception of realism changes with *time*. In other words, people 100 years ago would understand realism differently than people nowadays. Lyu and Farid (2005, Figure 3) present a sample of photorealistic – although not "photographic", and perhaps closer to "computer generated" – images that, from today's perspective, are quite apparently computer generated. Similarly, the Photorealistic Computer Generated part of the Colombia Dataset (Ng et al. 2004) would not be considered photorealistic nowadays. In Figure 2.9, we illustrate both of these cases.



(a) Bonsais (b) Fountain

Figure 2.9: Images *illustrative* of those shown in Lyu and Farid (2005, Figure 3) and Ng et al. (2004). Bonsais by Jaime Vives Piqueres (1999), Fountain by Jaime Vives Piqueres (1997), both licensed under CC BY-SA 4.0. The images from the original publications could not be used due to unavailability and copyright.



(a) (b) (c)

Figure 2.10: Images from games that were advertised or reviewed as highly realistic. (a) Pete Rose Pennant Fever, 1988, (b) Wolfenstein 3D, 1992, (c) The Journeyman Project, 1993.

A similar phenomenon can be observed for video games. There, apparent visual realism is historically perceived as fairly high even though by today's standards, the imagery is clearly non-realistic.

For example, a 1988 review of Pete Rose Pennant Fever, a baseball game shown in Figure 2.10a, in Computer Gaming World magazine (CGW 52, p. 12) says "*the graphics... looks realistic*". Later, an advertisement in a different issue of the same magazine for the 1992 game Wolfenstein 3D: Spear of Destiny, see Figure 2.10b, says it brings "*virtual reality to the PC*" and that "*the visual action, is frighteningly realistic*" (CGW 113, p. 2). Its review simply says it is "*with Ultima Underworld, the first game... creating a sufficient element of disbelief-suspension*" (CGW 98, p. 52) about its *overall*, not necessarily visual, realism and immersion. Other advertisements are similarly bold. One, for The Journeyman Project, shown in Figure 2.10c, claims it is the "*World's First Photorealistic... Adventure*" (CGW 113, p. 135), while another for Earth Invasion touts "*incredible photorealistic 3D graphics*" (CGW 113, p. 137).

Naturally, the advertisements are more boastful than the reviews and the reviewers themselves would definitely be able to tell, say, a photograph of a baseball game from an in-game screenshot, but they still paint a picture of preconceptions and expectations. Unfortunately, there are no studies about the perceived visual realism of video games from that time that we are aware of, meaning this thesis will always remain an untestable conjecture.

Another example of the different perception of realism across *time* and *culture* concerns the paintings of Ancient Egypt, see Figure 2.11a. About the image, Gombrich (1995, pp. 51–52) says that "*[w]hat mattered most was... completeness*" and "*[t]heir method... resembled that of the map-maker*", indicating that the Egyptians were more concerned with something akin to *functional realism*, see Section 2.2.2, where the image enumerates the objects, rather than trying to achieve *full realism*, see Section 2.2.1. Furthermore, Gombrich (1961, Chapter VIII., Part X) also states that the cues that influence our "reading" of an image are considered to change between cultures.

Analogous processes are also associated with experience (Gombrich 1961, p. 217). Consequently, we are able to tell the realism of daily objects better and the realism of uncommon objects, for example exotic animals, with lower accuracy. Using an image of a whale washed ashore in the Netherlands in 1598, see Figure 2.11b, Gombrich (1961, p. 65) illustrates how experience influences



(a) (b)

Figure 2.11: Realism in different historical epochs. (a) The Garden of Nebamoun, a painting from Ancient Egypt, cca 1400 BC, by Yann Forget (2012), in Public Domain. (b) Cut out of an image of a whale washed ashore in the Netherlands in 1598. A copper engraving by William van der Gouwen (1684), after an 1598 engraving by Jakob Matham of a drawing by Hendrik Goltzius, in Public Domain.

perception of what is in front of us. In that Figure, the Dutch artists – whom Gombrich describes as "*those masters of realism*" – probably mistook a whale's flippers for an ear due to them not having been familiar with whales and due to the familiar schema of a head with ears.

It follows that this dependence of our ability to "read" images on learned, pre-existing schemata and experience also influences our ability to perceive – "read" – the world around us, thus changing our perception of what is and is not realistic depending on our experience and culture.

While one is usually interested in the perception of realism here, now, and for a specific demographic, it is good to keep in mind that *perception of realism* might change or be inconsistent with time, personal experience, and culture.

2.4 Photorealism

> *"Remember once more the photographer's troubles. If he wants us to admire the wonderful autumn tints, he photographed on his latest trip he will lure us into a darkened room where he displays his transparencies on a silver screen. Only the borrowed light of the projector lamp, aided by the adaptability of our eyes, will allow him to match the range of light intensities he had enjoyed in nature."*
>
> — Ernst Gombrich, *Art and Illusion, 1961*

> *"The camera cannot lie; but it can be an accessory to untruth."*
>
> — Harold Evans, *Pictures On A Page*

One of the commonly understood meanings of *realism*, especially in computer graphics, is *photorealism*. This meaning is particularly common in contexts where the goal is to simulate light propagation in a virtual scene. Being photorealistic, or, to paraphrase, being as close as possible to a photograph, then means that our algorithm provides us with results that are *indistinguishable from a photograph of an identical, real-life scene* or a plausible notional scene, see also the discussion on plausibility in Section 2.2.4. Indeed, to a computer graphics practitioner, using photorealism as a means of evaluating realism seems intuitive. Nevertheless, to understand what photorealism is and its variants mean and could mean, we need to better understand its two underlying terms: *photograph* and *indistinguishable*.

2.4.1 Photography and Photographs

If we were to ask a random person to provide us with as close as possible a representation of what they see in front of them, most would shoot a photograph and consider that the ultimate answer. They have taken pictures of family and friends and understand photography to be a process by which the reality imprints itself on film or a sensor, reassuring them that it captures the reality of what is in front of them. And if the process of taking a photograph is not the best solution, what is?

However, there is further nuance to photography. Its quality is really a spectrum. On its one end, we have highly accurate photographs whose goal is to measure the scene. On its other end, we have photographs that are technically imperfect, whether intentionally or accidentally. For the purposes of this discussion, we will divide this continuous spectrum into two discrete categories, illustrated in Figure 2.12:

1. *Measurement photography.* Photography that measures – as well as reasonably possible – the light projected onto a plane.

2. *Common photography.* The complement of measurement photography.



(a) (b)

Figure 2.12: Examples of a measurement photograph and a common photograph. (a)Measurement photograph of a bird (Cotinga maynana) for the ultraviolet 300.5–333.5 nm spectral band, including a white reference and a gray scale. By Todd Harvey, used with permission. (b) Common photograph, with a high number of technical imperfections. By author.

2.4.1.1 Measurement Photography

Measurement photography's explicit goal is to capture the quantities of light intersecting a projection plane as closely as possible. Some typical contemporary examples of measurement photography would be some forms of astrophotography and spectral photography (Kim et al. 2012; Striova et al. 2018). In fact, one can understand measurement photography to be what its name suggests: a number of scientific measurements where the density and number of these measurements became so high that the discipline started to coincide with "taking pictures".

Despite its attempt at perfection, measurement photography is subject to technical and practical limits. For example, sensors and films have limited resolution, lenses introduce distortion, parts of the visible spectrum might be filtered out by optical elements, many times, we can only reasonably be expected to capture RGB correlates of the incoming light, etc.

Our definition of measurement photography overlaps with forensic photography and scientific photography. However, different specific applications might differ in whether they strive to capture the incoming light from the scene as is, for example to compare against the results of a rendering algorithm, or whether they will take additional steps to make sure the scene is well lit or that as little light as possible is reflected off the environment.

For our purposes, a photograph in the measurement sense does not have to have the resolution or regular spacing of samples expected of a photograph. Instead, we understand a measurement photograph to be almost any set of measurements of light in a plane. While this decision might seem arbitrary, it allows us to simplify and generalize the discussion. In Section 2.5.1, we provide further discussion and specific examples of measurement photography in computer graphics.

2.4.1.2 Common Photography

In *common photography*, the motivation is mostly to take everyday amateur images, such as images of one's family, landscapes, etc. Additionally, common photography also includes most hobbyist and professional photography, including artistic photography or documentary photography. The fact that common photographs are not explicitly taken with measurement in mind makes it unlikely that the shot is properly set up and executed, see again Figure 2.12b, although there might be exceptions, e.g., where taking a measurement photography would be considered artistic. Consequently, incorrect exposure, dynamic range conversion, white balancing, compression, and other artifacts make these photographs inadequate as physical measurements. An interesting discussion of similar effects for paintings, with examples, can be found in Gombrich (1961, pp. 28–31).

2.4.2 Comparing Photographs

In the previous section, we have talked about *photographs*. In this Section, we discuss what it means for a generated image to be *indistinguishable* from a photograph of the scene it is trying to depict. The approach we take to comparing images and photographs differs based on whether the photograph is a common photograph or a measurement photograph.

2.4.2.1 Comparing Common Photographs

In the case of common photographs, we are working with photographs that have been taken without a well known setup. Their main advantage is the simplicity of their acquisition and availability. On the other hand, they do not do a particularly good job at capturing the exact qualities and quantities of the incoming light.

Since a common photograph is not well suited to exact comparisons, its best use seems to be as an intermediary medium by which humans can judge images and subjective impressions. Importantly, this subjective comparison of two images inextricably includes human perception, with its inability to perceive some of the differences. This drawback is, at the same time, also a feature. It allows use of common photographs for the purpose of verifying that a human would not be able to *perceive* a difference between a common photograph and a generated image – as long as the generated image is properly tone mapped and otherwise preprocessed to mimic the appearance of a photograph – and that the rendered image is therefore good enough. Among existing literature, McNamara and her colleagues are especially strong proponents of this approach (McNamara et al. 2000; McNamara 2001; McNamara 2005; McNamara 2006).

Therefore, for a common photograph and a generated image, the most appropriate mode of comparison is by a *human observer and their perceptual judgment*.

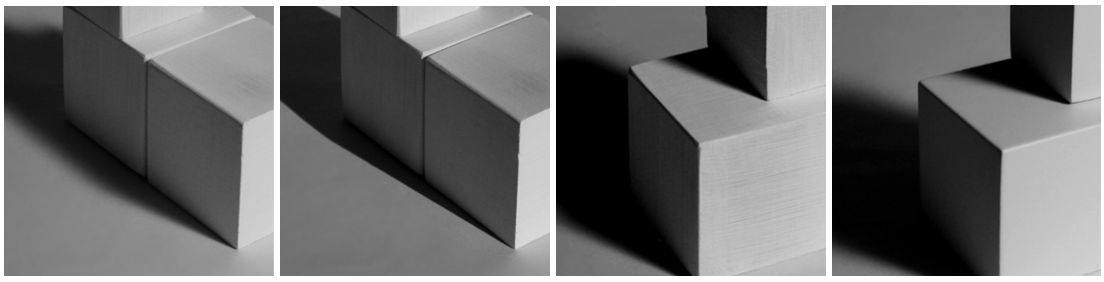
It should also be somewhat possible to replace human perception by an algorithm that tries to simulate it. Common approaches include Visual Differences Predictor (VDP) (Daly 1992) or the Structural Similarity Index (SSIM) (Wang et al. 2004). We review these methods in Section 2.5.2.1.

2.4.2.2 Comparing Measurement Photographs

For measurement photographs, the situation is somewhat more straightforward. The photograph has captured physical measurements of a scene. Consequently, we can compare these measurements against the result of an algorithm. Such a comparison of exact values is best done numerically and *without* human perception being a part of the equation, as perception will cause some of the differences to be ignored. We provide further discussion in Section 2.5.2.1.

2.4.3 Realism in Photographs

In his research, Rademacher (2002) sets out to establish some of the factors influencing the perception of realism in images. In one set of experiments, Rademacher took a number of real photographs of simple objects while varying the properties of the scene. He established that the perceived realism was lower for hard shadows and higher for soft shadows (Rademacher 2002, p. 78) and that is was also higher for rough surfaces and lower for smooth surfaces (Rademacher 2002, p. 79). The remaining factors – number of objects, mix of object shapes, and the number of light sources – did not influence perceived realism in a statistically significant way. The experiment's significant factors are shown in Figure 2.13.



(a) Soft shadow (b) Hard shadow (c) Rough surface (d) Smooth surface

Figure 2.13: Photographs used in Rademacher's experiment (Rademacher et al. 2001, Figures 2 and 4). Soft shadows were considered more realistic than hard shadows. Rough surfaces were considered more realistic than smooth surfaces. Additional photographs and generated images can be found in (Rademacher 2002). Adapted from (Rademacher et al. 2001), used with permission.

The takeaway from Rademacher's research is that even photographs, which are "*photo-real*" (Rademacher et al. 2001, p. 12) by definition, can be perceived as unrealistic. Or, in other words, that people have some preconceptions about what is and is not real and these preconceptions influence their judgment of realism, even for high quality photographs. This conclusion notwithstanding, we also have to consider the quite limited set of simple scenes Rademacher's research was conducted on and therefore its somewhat limited generalizability.

Another argument to be made, somewhat at odds with Rademacher's findings, is about the expectation of realism in photographs. As has already been shortly discussed in Section 2.4.1, people trust photography to recreate the scene in front of them. For example, Kember (1996, p. 152) talks about the authority of photographic realism and how it is grounded in Euclidean geometry. A fur-



(a) (b)

Figure 2.14: Historical color photographs. These were most likely considered highly realistic when shot, but might not be up to current expectations of realism. (a) Mulberry Street, New York City, approximately 1900s (Mulberry), in Public Domain. (b) Garden of the Gods Park, shot in Kodachrome, 1950s (Butterfield). Licensed under CC BY-SA 3.0.

ther illustration of this fact is that most people would consider black and white photographs as realistic, even though they fail to convey color.

One final possible effect to consider is that the perception of realism in photographs has changed over time. Consider, for example, the photographs in Figure 2.14. They were most probably considered highly realistic at the time they were taken. However, we can see distinct color distortion and if we had access to the photographs' originals, could probably identify other artifacts. To some degree, we could even consider some historical photographs border-line non-realistic. A similar discussion about how realism in general changes over time and with culture can be found in Section 2.3.2.

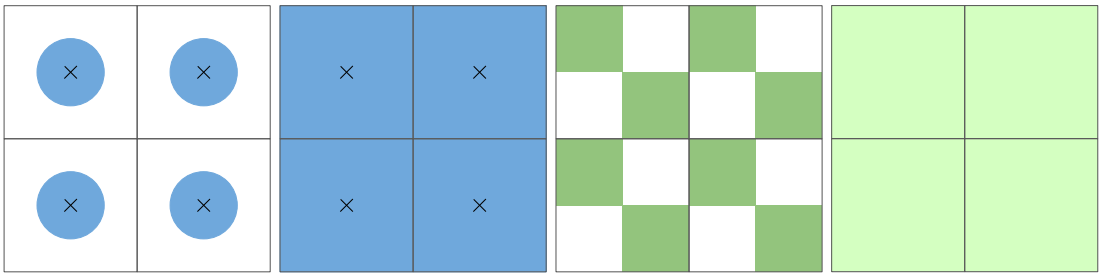
This, together with the above effect of expecting a photograph to be realistic shows that we cannot take perceived realism of photographs at face value.

2.4.4 Measurement-Photorealism

Based on the previous discussion, we now explore some of the possible definitions of photorealism. We start with *measurement-photorealism*. We say an image is *fully measurement-photorealistic* if its every value is identical to the corresponding value in a measurement photograph of the scene it is trying to reproduce.

The requirements of this definition are similar to those of physically-based realism, see Section 2.2.5. However, for two reasons, we still discuss measurement-photorealism separately. First, it provides us with a "photography" perspective of the problem and allows us to separate the discussion into a measurement-photorealism portion and the rest. Second, where the discussion in Section 2.2.5 was more theoretical, here we illustrate the practical difficulties of *collecting physical quantities on a plane*.

To start with, the above definition of an image being fully measurement-photorealistic means that resolution and sampling matters. If we assume perfect point sampling in both the photography and the generated image, as illustrated in Figures 2.15a and 2.15b, our definition could have an image come out as



(a) (b) (c) (d)

Figure 2.15: Scenes captured using a resolution of 2×2 pixels that would be considered equal under measurement-photorealism. In (a), (b) for point sampling, where crosses determine the positions of the samples and in (c), (d) for area sampling, black rectangles indicating the sampling areas.

measurement-photorealistic against a very different second image. This situation where not all of the detail in the image can be resolved is described by the Shannon-Nyquist sampling theorem (Shannon 1949). This ambiguity also implies, naturally, that achieving measurement-photorealism for a very high resolution image will be just as or more difficult, because the scene could be required to provide a higher level of fidelity. If we instead assume that each measurement and/or pixel perfectly samples its neighborhood, we can still get identical effects, albeit with a smaller probability, see Figures 2.15c and 2.15d.

Therefore, measurement-photorealism depends on resolution and sampling. Both of these choices are normally either due to the requirements of the intended output device and therefore implicit, or to be decided by other practical limitations. Next, one could also naturally extend measurement-photorealism to a continuous metric to distinguish the degree of realism, from not measurement-photorealistic to fully measurement photorealistic. The typical choices of metrics that could be used to assess measurement-realism between two images are discussed in Section 2.5.2.

In the end, however, measurement photography is difficult to do correctly and tedious to execute. Nevertheless, it can still be very useful in some contexts, especially where historical preservation or analysis of the photographed data is the objective (Kim et al. 2012; Striova et al. 2018).

2.4.5 Perceptual-Photorealism

If we try to amend the definition of *measurement-photorealism* to common photographs, we discover some additional implications. Let an image be *perceptually photorealistic* if a human observer cannot tell the difference between the image and a common photograph of the scene it is trying to reproduce.

If a human observer cannot tell the difference between an image and a common photograph, it means we have been able to recreate the scene to within perceptible differences. What these differences could be, if any at all, is not clear. However, several factors could play a role, including but not limited to differences:

- in complex reflections (Ramanarayanan et al. 2007),
- in lightness (McNamara et al. 2000),

- in arrangement of objects in the scene (Ramanarayanan et al. 2008), or
- below the perceptual threshold (Daly 1992; Wang et al. 2004).

While such a definition of perceptual photorealism is useful for validating perceptually driven rendering (Myszkowski 1998; Volevich et al. 2000; McNamara et al. 2000; McNamara 2001, Section 4) or the perceived full realism, it is not very useful for validating physical accuracy.

Additionally, common photographs are the most practical choice for *user studies*. Common photographs are also required where numerical comparison of measurements is not possible, for example in procedural modeling, see Section 3.1, or in evaluation without reference.

Further treatment of perceptual evaluation is available in (Meyer et al. 1986, Section 3), which describes an experiment where humans evaluate the realism of a rendered scene. McNamara (2001, Section 5) reviews human-based evaluation of rendered scenes, and a similar set of authors further argues for the utility of both perceptual rendering and human evaluation in rendering in McNamara (2005) and McNamara (2006).

2.4.6 Summary

In this Section, we have provided a short general discussion of photography and its realism. While most authors in computer graphics define photorealism – or equivalently, the goal of image synthesis – as generating images indistinguishable from a photograph (Ramasubramanian et al. 1999; Jensen 2001; Dutré et al. 2006; Pharr et al. 2016), we tried to show that photorealism is a complex notion with various possible meanings and implications to be aware of.

In the process, we have defined two kinds of photography: *measurement photography*, whose goal is to measure the incident light, and *common photography*, which encompasses the vast majority of amateur and professional photographs and which is *measurement photography*'s complement.

We have argued that for *measurement photography*, numerical comparison is the most sensible choice. Conversely, we have also argued that for *common photography*, human comparison is the best option.

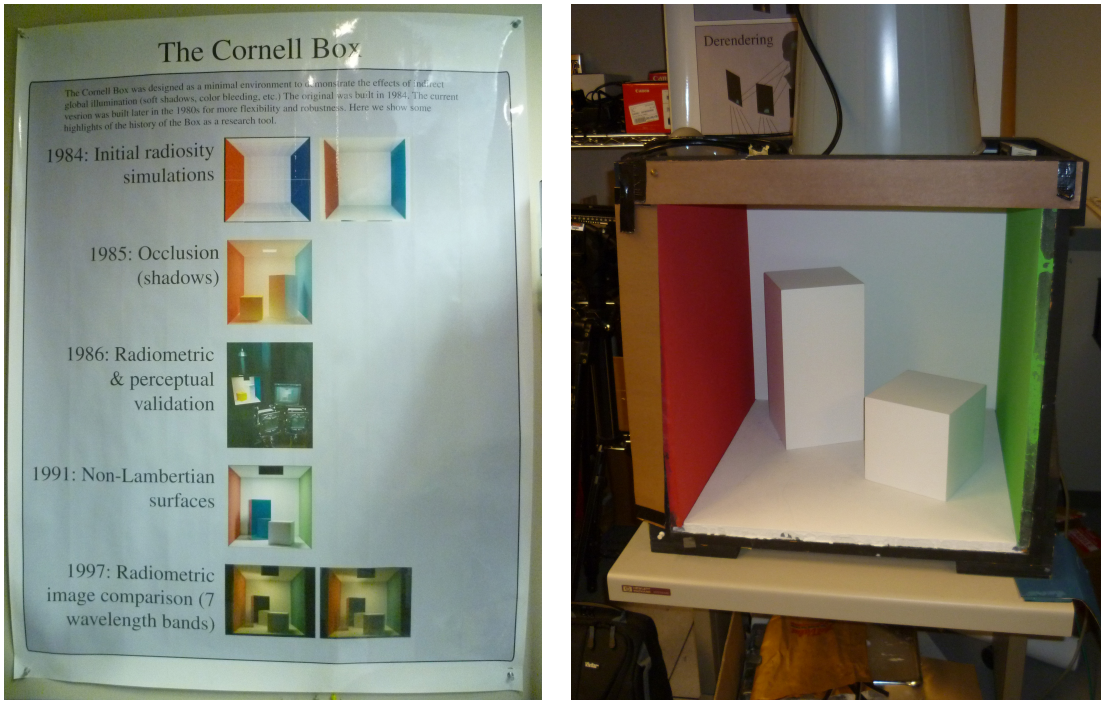
Next, we have proposed that people expect photographs to be realistic, that the realism of photographs is not well established in time and that even photographs can appear non-realistic. Based on these observations, we have discussed the implication of various possible definitions of photorealism.

2.5 Evaluation of Realism in Rendering

> *"Thus, what usually passes for validation of a simulation is just anecdotal evidence with a healthy dose of pretension, and you will find no exception here."*
>
> — Greg Ward (Ward 1990)

2.5.1 Comparing against Measurement Photographs

In evaluating realism, the use of measurement photographs had been anticipated, encouraged, and proposed (Ward 1990, pp. 2–3; Rushmeier et al. 1995, Section 3.2; Greenberg et al. 1997, Section 3.3). In a sense, the use of measurement photography as a means of measuring the scene we are trying to reproduce is probably the closest we can get to an idealized definition of photorealism. However, mostly for practical reasons, there has been little adoption. Below, we give an overview of ways in which measurement photographs have been compared against *reference images* to evaluate realism. A much more in-depth treatment can be found in the survey by Ulbricht et al. (2006).



(a) (b)

Figure 2.16: The Cornell Box. (a) Cornell Box poster, detailing its history. (b) The physical Cornell Box. Both images were shot by the author at the Cornell University in 2013.

In 1985, see Figure 2.16a, Meyer et al. (1986) devised a simple scene consisting of a box with two added cuboids, called the Cornell Box (Cornell University). The scene was built in real life, as shown in Figure 2.16b, and also recreated on a computer. However, only a small number of (spectral) samples was used to verify

the validity of the results, with just 5×5 regularly spaced measurement points being compared against the same locations on a rendered image (Meyer et al. 1986, Figure 2a and 5). While the reason for the small number of samples is not clear, it is probably due to the difficulty of executing the measurement procedure – it is mentioned that careful control of probe position is required (Meyer et al. 1986, Section 2.4) – and the fact that the measurements were made using a point probe.

A few years later, Ward (1990, pp. 2–3) mentions taking a "*dozen or so illumination values*" for his experiment and comparing these illumination values to a generated image, stating that a direct comparison is out of question due to, among other issues, registration problems.

Later, Rushmeier et al. (1995, Section 3.2) attempted to utilize per-pixel comparison for a common photograph and a rendered image and concluded, just like Ward (1990), that even with registration between the images, such an approach is impractical. Nonetheless, even as late as 1997, Greenberg et al. (1997, especially Figure 13 and Section 3.3) still see comparing spectral radiance measurements with results of a light transport simulation as an integral part of "A Framework for Realistic Image Synthesis".

Ultimately, due to the necessity of a physical setup, the difficulty of recreating virtual scene's exact materials in real life, or possibly vice versa, and the extremely difficult registration – or as Ulbricht et al. (2006) put it "*because currently there is no easy way of performing it*" – the direct comparison of values between a measurement or common photograph and a rendered image is nowadays rarely considered when verifying physical accuracy in computer graphics.

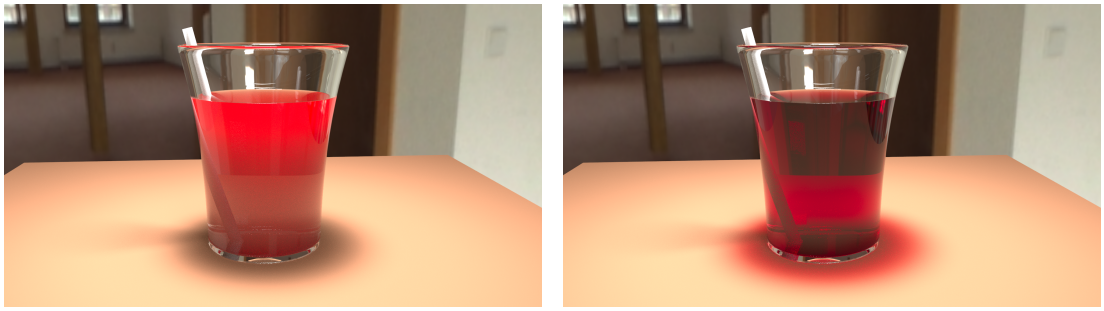
2.5.2 Path Tracing and Reference Images

In the Section above, we have all but dismissed the possibility of direct comparison of values between a measurement photograph and a rendered scene. What has instead become common practice in rendering research is to compare the image generated by a *new* rendering algorithm against a reference image that has been generated by another, trusted rendering algorithm. This less efficient, but more trusted and believed to be correct rendering algorithm is called *path tracing*. Path tracing, in turn, is a direct implementation of the rendering equation (Kajiya 1986) or its extended versions (Raab et al. 2008, Equation 4).

It should be noted that while comparing against a measurement photograph would ascertain the realism of all three elements of realism – that is of the geometry, the material appearance, and of the rendering algorithm, see Section 2.1 – the usual purpose of comparing the results of an algorithm to a path-traced image only lies in asserting that the new alternative algorithm generates the same results as the path tracing algorithm. In other words, only the validity of the rendering algorithm is verified.

By comparing against a path traced solution, all of the above outlined difficulties of comparing against a measurement-photograph or comparing without reference can be avoided.

However, the path tracing algorithm also trades its increased clarity and believed correctness for an increased computational time. For example, Whittle et al. (2017, p. 8) recommends that the reference image have at least an order of magnitude more samples per pixel. However, due to increases in computational



(a) With scattering (b) Without scattering

Figure 2.17: Scattering. The same scene with and without scattering. By Ondřej Karlík, used with permission.

resources (Pharr 2018) and its many beneficial qualities (Křivánek et al. 2018, Part 7), path-tracing has even re-established itself recently in many production renderers (Keller et al. 2015; Fascione et al. 2018).

Somewhat unfortunately, the results of the path tracing algorithm usually only include some of the terms of the rendering equation. Specifically, it only makes sense to include those terms that the new algorithm is trying to simulate and thereby shorten the time to convergence. For example, an algorithm that aims at simulating correct reflection off materials might not include atmospheric scattering in its path-tracing generated reference image, see Figure 2.17

While this is a logical and justifiable choice, only partial photorealism is achieved in both the result and the reference. Thus, the reference image is usually not perfectly measurement-photorealistic.

2.5.2.1 Image Metrics

There is a number of ways to compare a reference image and the rendered image using a computer, commonly called "full reference Image Quality Metrics", or FR-IQM. Lavoué and Mantiuk (2014, Section 2) and Mohammadi et al. (2014) provide surveys.

Two classic, strictly non-perceptual metrics are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) (Čadík 2008, Section 3.2.2.1). They work directly with pixel values and make no effort to involve human perception. In that respect, they might be best suited for comparing the physically accuracy of a rendered image against a reference.

Two of the most popular perceptual metrics are the Structural Similarity Index (SSIM) (Wang et al. 2004) and the Visible Differences Predictor (Daly 1992). The Structural Similarity Index works under the assumption that human visual system has adapted to extracting structural information from a scene (Wang et al. 2004, p. 1) and produces a single scalar. The Visible Differences Predictor (Daly 1992) implements a model of early vision (Čadík 2008, p. 22) to generate a map of detection probabilities. Many further variants of these methods exist (Aydın et al. 2008; Mantiuk et al. 2011; Ponomarenko et al. 2011). A perceptual comparison metric could be used to determine whether a human observer can see any differences. Nevertheless, it should not be used to determine physical accuracy.

However, whether a metric is or is not perceptual can be a point of contention. For example, Hore and Ziou (2010, p. 3) derive a mathematical relation between the perceptual SSIM and the non-perceptual PSNR and remark that one can be predicted from the other, with the difference being mainly in their sensitivity to certain artifacts. Similarly, some authors, e.g., Hošek and Wilkie (2012, Figure11) use a difference plot of perceptual color difference DeltaE (Sharma and Bala 2002, Section 1.7), which in itself is based on human color perception, but which does not take any locality or other aspects of human vision within the image into account.

Also, there are methods for detecting rendering errors without a reference image. These work by designing a system that is sensitive to the same kind of artifacts as humans (Čadík et al. 2012; Čadík et al. 2013). Lavoué and Mantiuk (2014, Section 4.4.1) provide further discussion. However, these approaches can produce false negatives – that is, they can label an image as problem-free when it is not. An interesting alternative approach is proposed by Herzog et al. (2012), who attempt to detect possible errors in rendering with information about the scene, but without a reference image.

Finally, in Section 2.6.1, we discuss how neural networks could be used to develop more advanced Image Quality Metrics.

2.5.3 Evaluation of Rendering Algorithms

Rendering algorithms are the final element of realism. Their goal is to take high quality geometry and material appearance and transform them into a realistic image by means of a *physical simulation* of light. As such, rendering algorithms can also be critical while evaluating the realism of the geometry or material appearance they are displaying and should be subject to a thorough evaluation themselves. In this Section, we therefore briefly focus on the possible biases in their evaluation.

As discussed in Section 2.5.1, comparing a rendered image against a photograph is an almost impossible task. Instead, a commonly used option is to use converged path-traced images as a trusted ground truth. Then, a value-by-value, non-perceptual comparison should be used to determine the algorithm's accuracy.

This approach presents one major possible bias: not fully demonstrating the drawbacks of the algorithm. This happens by consciously or unconsciously selecting or not selecting results that would demonstrate:

1. *Retained behavior.* Set of results that are comparable with previous state of the art, meaning the algorithm does as good a job as other algorithms.

2. *Resolved behavior.* Situations the previous state of the art had a problem with that have now been resolved, or an improvement on previous behavior with respect to, for example, performance. Usually, only this category is shown in papers.

3. *(Re-)introduced behavior.* An issue that was not common with recent state of the art, but that has now been introduced or re-introduced, or where the behavior worsened. One example of such behavior would be the Metropolis Light Transport algorithm by Veach and Guibas (1997) and its derivatives,

which introduced flickering (Woestijne et al. 2017, p. 1; Šik and Křivánek 2018, Section 11.5) and non-uniform convergence (Šik and Křivánek 2018, Section 6.2.3; Hachisuka et al. 2014, Figure 6.3). This behavioral category is almost never reported.

That is not to say that an approach that has certain drawbacks and a different set of benefits should not be presented. It may still stimulate the field and can be combined into new approaches, such as those by Georgiev et al. (2012) and Křivánek et al. (2014).

The preferential presentation of benefits is to a large degree due to only limited expectation on the side of the reviewers, the readers, or the authors to see comprehensive results. That is probably exacerbated by the understanding that thorough testing is a very tedious and demanding process.

This line of thought, however, still does not detract from the fact that all of the changes in behavior, positive as well as negative, should be documented.

One of the biggest enablers of such documented behavior would be the existence of a standardized test suite for rendering and diligently following a standardized testing routine (Beck 2002, pp. 84–85). Although there are some classic scenes such as the Teapot, the somewhat incomplete Rendering Resources collection of 32 tests scenes by Bitterli (2016), or the recently released Moana dataset (Disney 2019), to the best of our knowledge, no comprehensive test suite for rendering or one of its sub-fields is available.

Such a test suite would contain different highly validated scenes that would be *representative* of the known space of issues and behaviors within rendering and would serve as a baseline performance metric. It could be run automatically, with selected results being reported in the papers, and the *whole* set of results being a standard part of supplemental materials.

2.6 A Neural Networks Perspective on Realism

The field of neural networks enriches the discussion about realism with some interesting insights and ideas. This section provides a concise overview of these contributions. First, in Section 2.6.1, we show how neural networks can find representations of style and realism of content in an image and how this information can then be used to judge the amount of realism. Then, in Section 2.6.2 we describe neural network architectures that generate new content and in the process, judge its quality, much like some procedural modeling algorithms, see Section 3.2.3.

This Section is not meant to be an overview of neural networks themselves and assumes a basic working knowledge of the topic. An approachable exposition of the basics is presented by Nielsen (2015), while Goodfellow et al. (2016) provide a more theoretical, in-depth treatment. A good hands-on tutorial on neural networks is Howard and Thomas (2018).

2.6.1 Semantic and Stylistic Representations

A neural network consists of a number of layers. Each layer transforms its inputs, for example the pixel intensities of an image, yielding a new intermediate representation. It turns out that specifically for neural networks that classify

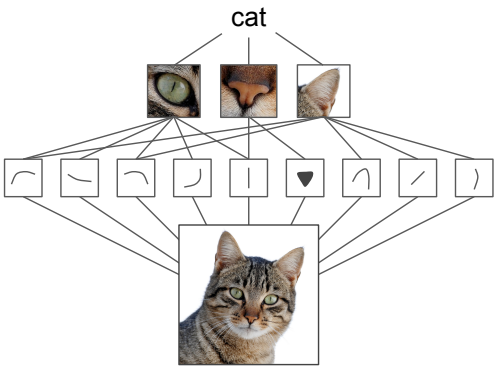


Figure 2.18: A neural network's representation of a cat. The image of a cat is first processed by layers, in third row from top, that identify simple features like edges. Signals for these simple features are then combined to identify higher-level concepts, such as an eye, a nose, or an ear, in second row. The presence of these higher-level features is then used to indicate that the image, in fact, contains a cat. Image after Chollet (2017, Figure 5.2). Cat image by Gaspar (2010), licensed under CC BY-SA 3.0.

images, these earlier, most commonly convolutional layers tend to yield a representation which describes the lower-level features (Yosinski et al. 2014, p. 1) of an image, such as edges from the input images. The following layers then use these lower-level features to build a representation consisting of higher-level features, for example an "eye" or a "nose". Eventually, these features get combined further into a representation where whole concepts, such as a "face" or a "dog" can be represented (Yosinski et al. 2015, Section 2, Figure 5; Zeiler and Fergus 2013, Figure 2; Chollet 2017, Figure 5.2). This is illustrated in Figure 2.18.

In other words, this means that neural networks tend to build a hierarchical and for a human increasingly more meaningful semantic representation of an input image. This ability to separate the higher-level, semantic information – the *content* – and the lower-level *style* of an image is used in various applications.

To give an example, Gatys et al. (2015a, p. 4) propose an algorithm for transferring the style from a painting onto a photograph. This algorithm works by iteratively changing the style of the photograph while preserving its content. In such an algorithm, judging whether the intermediate generated images preserve the *realism of the content* would be almost impossible using classical approaches (Johnson et al. 2016, pp. 1–2) such as Visual Differences Predictor (Daly 1992) or Structural Similarity Index (Wang et al. 2004). It is, however, quite straightforward using the above described semantic representation extracted by a neural network (Gatys et al. 2015a, Equation 1; Li et al. 2017, p1).

Other methods that make use of a neural network's abilities to extract semantic and stylistic attributes of an image include increasing the resolution of an image (Ledig et al. 2017; Johnson et al. 2016), finding a way to change the style of an image without changing its content (Gatys et al. 2015a; Johnson et al. 2016), and texture synthesis (Gatys et al. 2015b).

The above described ability to separate content and style outlines a possible direction of research into *realism* which would use an extracted representation – of content, style, or a newly developed one – from a reference image and compare

it against the same representation extracted from another image whose realism we want to verify. In principle, this approach is not different from the existing approaches discussed in Section 2.5.2.1 that make use of our knowledge about human vision to extract pertinent information. What is or would be different, however, is the mechanism by which this information is extracted. This extraction would no longer be based on pre-existing knowledge of the human visual system, but rather on knowledge encoded in the weights of neural networks by training.

A neural networks-based approach could facilitate automation and be especially useful for automated evaluation of realism for each of the three elements of realism – geometry, material appearance, and the quality of the rendering algorithm – and thereby also for procedural modeling.

Next, there currently does not seem to be any literature regarding the principles which a neural network could use to classify images as specifically realistic or unrealistic. However, some rudimentary hypotheses are proposed by Quan et al. (2018, Section 6).

Since neural networks are still a rapidly developing field, there is also an ongoing discussion about the ability of neural networks to represent some of features, and in turn the style and the content, accurately. For example, Brendel and Bethge (2019, pp. 1, 8) posit that most of the modern improvements in neural network performance on image tasks come from better fine-tuning rather than better concept representation, while Geirhos et al. (2018, p. 9) suggest that some of the state-of-the art image networks are biased towards deducing information from texture within the image rather than from the shape of the objects. This is in contrast with the above discussion.

2.6.2 Discriminative Networks as Realism Predictors

There is a number of approaches for generating content with neural networks. Probably the most popular one is called GANs – Generative Adversarial Networks (Goodfellow et al. 2014). In this approach, one *generative* neural network is used to "*[capture] the data distribution*" (Goodfellow et al. 2014, p. 1) of a set of reference examples. A second, *discriminative* neural network is then used to

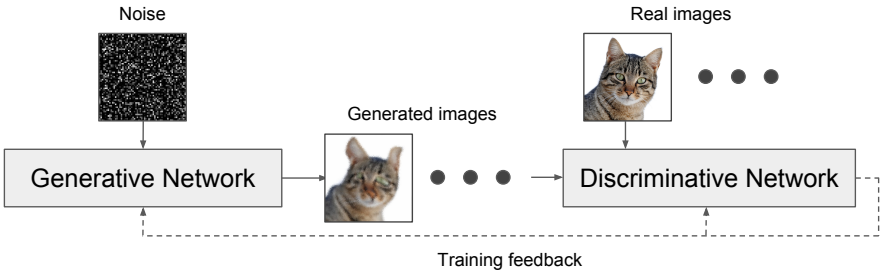


Figure 2.19: A generative network transforms noise into a batch of generated images. The discriminative network then tries to discriminate between this set of generated images and a set of real images. The ability of the generated images to convince the discriminative network of their realism or quality is used as feedback to the generative network. The ability of the discriminative network to distinguish between the generated and the real images is used as its own feedback. Cat image by Gaspar (2010), licensed under CC BY-SA 3.0.

distinguish the generative neural network's output from a set of reference inputs. These two neural networks are trained in step, effectively being made to compete against each other. This is illustrated in Figure 2.19.

This concept is interesting not only because it provides a way of generating new content, but because it also demonstrates that a neural network, here the discriminative neural network, can distinguish between good and bad images, or perhaps procedural models, see Section 3.1, and could be trained *and* used in practice as an automated "judge" of quality and perhaps, of some kinds of *realism*.



(a) (b) (c) (d)

Figure 2.20: The first four faces we generated using ThisPersonDoesNotExist, a website that implements the GAN-based algorithm by Karras et al. (2018). Computer generated images, assumed to be in Public Domain.

This is corroborated by a number of GAN flavors being used for both 2D and 3D generation. For example, Karras et al. (2018) use StyleGANs to generate faces, see Figure 2.20; Ledig et al. (2017) use SRGANs to generate higher-resolution images, Wu et al. (2016) use 3D-GANs to generate 3D objects, and Kelly et al. (2018) use FrankenGANs to add detail to geometrically reconstructed 3D buildings. Odena (2019) provides an interesting discussion about open questions for GANs.

Other generative approaches exist, mostly relying on learning an inner representation of objects and a generative step that can turn this inner representation into a full model. Examples include Dosovitskiy et al. (2014), Tatarchenko et al. (2017), Sharma et al. (2016), and Kingma et al. (2014).

2.7 Summary

In this Chapter, we have discussed realism in computer graphics. We have shown that three *elements of realism* – geometry, material appearance, and rendering algorithm – are required to get a life-like, *fully realistic* image.

We have defined *functional realism* as realism of the information an image conveys and reviewed *visual equivalence*, which combines both *full* and functional realism for the purposes of rendering. Next, we have discussed *plausibility*, the notion of something being believable or hard to deny, which allows us to talk about scenes that do not exist in real life, but have a realistic quality, mainly with respect to the depicted scenes. This part is summed up in more detail in Section 2.2.5.

Next, we talked about the pragmatic choices in realism and the trade-off between realism and cost and also discussed how culture, experience and historical period influence how people perceive realism of images.

Then, we discussed the term *photorealism*, a commonly used expression in computer graphics. We have suggested there are two kinds of photographs, *common* ones and *measurement* ones, and discussed their uses and the implications they carry. Finally, we have discussed how photographs can be perceived. This topic has its own, more detailed summary in Section 2.4.6.

In another Section, we talked about evaluation of realism in rendering and how a test-driven development inspired approach might alleviate some of its problems.

Lastly, we have briefly discussed how neural networks relate to realism. Specifically, we've shown how neural networks represent content and style and how this information could possibly be used for evaluating realism. We have also described GANs, a kind of neural network architecture that uses an internal neural network to classify images as, among other, references or generated.

Chapter 3

Realism in Procedural Modeling

> *"Even when a subject which I am working on compels me to ask a model for a certain fixed pose, I indicate it to him, but I carefully avoid touching him to place him in the position, for I will reproduce only what reality spontaneously offers me. I obey Nature in everything..."*
>
> — Auguste Rodin, *Art*

> *"There is a beautiful angel in that block of marble... [a]ll I have to do is to knock off the outside pieces of marble, and be very careful not to cut into the angel with my chisel."*
>
> — George F. Pentecost, *1883, The Angel in the Marble*

In the preceding text, we have looked at realism as it is commonly understood and treated in computer graphics. In this Section, we will discuss *realism in procedural modeling*, a much more sparsely explored area of computer graphics. We begin in the following Section by giving a brief overview of procedural modeling, after which we review its specifics with respect to realism. In Section 3.2, we then talk about what it means for a procedural generation algorithm to perform well and provide initial thoughts on how such an algorithm's performance can be measured. Finally, in Section 3.3, we talk about things that make evaluating procedural algorithms and the realism of their results different from evaluating the realism of other content.

3.1 Procedural Modeling

Procedural modeling is a set of approaches for automatic or semi-automatic generation of objects and scenes for computer graphics. In line with existing nomenclature, we call these objects and scenes *procedural models*. They consist of just *geometry*, just *material appearance*, or both, as shown in Figure 3.1. Commonly generated procedural models include textures, weathering, vegetation, landscapes, building facades, houses, or cities.

A recent general survey on procedural modeling in computer graphics is by Smelik et al. (2014). There are also three somewhat outdated, more specialized reports providing an overview of weathering effects (Mérillou and Ghazanfarpour 2008), procedural terrain modeling (Smelik et al. 2009) and procedural city



(a) (b) (c)

Figure 3.1: Examples of procedural models. (a) Terrain heightmap (Heightmap), a procedural model than consists of geometry only, in Public Domain. (b) Weathering, a procedural model of material appearance on a pre-existing geometry. By Oskar Elek, used with permission. (c) Building, a procedural model with both material appearance and geometry. Image by author.

modeling (Kelly 2006). A collection of classic procedural modeling methods, especially for textures, can be found in the seminal book by Ebert (1994). Togelius et al. (2011b) provide a taxonomy and outline theoretical concepts for search-based content generation in video games. In a more general way, Hendrikx et al. (2013) also provide a taxonomy and a short overview of generated content in video games, spanning from textures and objects to game levels and gameplay mechanics. Finally, in a related but distinct field, Olsen et al. (2009) have surveyed methods for generating 3D models from sketches.

In this Section, we are only interested in *material appearance* and *3D geometry* for computer graphics. With the exception of the occasional supporting argument, we will not discuss generation of noise, textures, or video game mechanics.

3.1.1 Goals of Procedural Modeling

From our point of view, there are two chief goals to procedural modeling. One is the ability to take either some or most of the *work away* from a human artist (Smelik et al. 2014, p. 1; Müller et al. 2006b, p. 9). The other is procedural modeling's ability to generate *novel*, previously unseen objects, also called database amplification (Smith 1984, p. 2; Smelik et al. 2014, p. 1).

To give an example of the first case, imagine a 3D artist tasked with generating the mesh of a highly detailed tree. With manual modeling, the artist would have to model the trunk, all of the differently sized branches, and the leaves. For a model of such complexity, the amount of pieces that would have to be modeled is unmanageable (Ebert 1994, p. 1). Additionally, the level of domain knowledge required for such a task might be difficult to attain for a non-expert, which would diminish the quality of the result. Here, procedural modeling can take over and do all of the tedious work, while ideally leaving artistic control of the generated model with the artist (Merrell and Manocha 2011, p. 1).

The second goal, *generation of novel objects*, is just as important. Here, *novelty* means not only that the object has not been previously seen, but that it *generalizes* the expected appearance or structure. For example, a building is

not a good novel procedural model for a procedural algorithm that is generating terrains. Or in a more specific example, a family villa is not a good Parisian apartment building.

A procedural method typically has some sort of in-built randomness (Ebert 1994, p. 65; Smelik et al. 2014, p. 1; Togelius et al. 2011a, p. 2), while also depending on the artist's input (Smelik et al. 2014, p. 13). This ability to take random decisions during the generation process, together with the per-model constraints imposed by user input (Merrell and Manocha 2011, pp. 1, 3), allow the procedural method to generate a wide range of outputs (Smelik et al. 2014,

(a) Dinosaur-like creatures



(b) Fish-like creatures

Figure 3.2: Some of the game objects that could be generated by the video game No Man's Sky, shown here with only basic materials and rendering. By Greg (3DGameDevBlog), used with permission.

p. 1). One recent application of this principle is in the 2016 video game No Man's Sky, see Figure 3.2, which used procedural modeling to generate novel variants of, among other, animals and spaceships.

Another benefit of procedural modeling is its ability to repeatably and in a deterministic way reproduce a large amount of data in the result using only a short description and an application of the appropriate algorithm (Smelik et al. 2014, p. 1). This was especially useful in older games such as the 1984 space trading video game Elite, where vast, repeatable universes could be generated, or in toy projects such a .kkrieger (Hendrikx et al. 2013, p. 3). Nowadays, it might be useful for populating large scenes with similar objects, such as landscapes with vegetation (Deussen et al. 1998, p. 6), or generating infinite terrains (Parberry 2015). However, in our experience, the vast majority of procedurally generated content is touched up by a human artist before being used. Due to a lack of appropriate tooling and limited literature (Smelik et al. 2010; Smelik et al. 2011), this feature of procedural modeling is only used in a limited way.

One facet of procedural modeling we will mostly omit from discussion is inverse procedural modeling (Šťava et al. 2010; Vanegas et al. 2012c; Martinović and Gool 2013). Its goal is to find a description or a set of parameters that would recreate a given existing geometry and/or material appearance using a given procedural approach.

3.1.2 Types of Procedural Modeling

There are two main kinds of procedural modeling approaches: Semi-automatic and fully-automatic.

With a *semi-automatic* approach, the way to go about modeling a virtual tree would be that the artist would choose the tree species, sketch out the shape of the main branches and then outline the expected shape of the crown, much like Chen et al. (2008b) or Longay et al. (2012), see Figure 3.3. The procedural algorithm would then fill in all the smaller branches and leaves based on its in-built expert knowledge, presenting the artist with a ready-made complex model.

This means the artist can *steer* the algorithm and thereby influence the results. Some more elaborate approaches would also provide *interactivity* (Smelik et al. 2014, p. 1; Chen et al. 2008b; Longay et al. 2012), where the artist would be allowed to interact with the model or its original specifications in real- or close to real-time.



Figure 3.3: Tree sketching. Based on a user sketch on the left, the algorithm fills in all the details of a tree. Adapted from Chen et al. (2008b), used with permission.

In contrast, a *fully automatic approach* would generate the whole tree, with the artist's input being limited to a bare minimum, such as the tree's species and age, but not, for example, its shape (Reffye et al. 1988, p. 9).

3.1.3 Use Cases

Both of the above approaches have their use cases. The semi-automatic one is especially useful where the scene designer has a specific vision for an object and its place in the scene. This can happen due to the object's aesthetic contribution to the scene or the scene's shot, or due the object's importance for the narrative.

For example, the famous Farnsworth House had for decades been sheltered by "Witness", a Black Maple tree that died, aged about 120 years, and was removed from the site in 2013 (Contract Magazine). A virtual recreation of the House would therefore require us to recreate that quite specific and identifiable tree, as shown in Figure 3.4. The scene designer and artist will therefore want to have fine control of the overall appearance, or perhaps even of the details, but will naturally want to avoid excessive manual modeling.



Figure 3.4: The Farnsworth House, built in a forested area, with a widely recognized Black Maple tree in the foreground. Photograph by Carol M. Highsmith (2006), in Public Domain.

The more *automatic* approach is useful where the exact variant of the object is less important as it is mostly to be used as a backdrop and among a high number of other objects. Coming back to our Farnsworth House example in Figure 3.4, most of the trees in the background could be generated in such a fashion, with the tree species specified but the particular shape of each individual tree being of little to no importance.

Another case where a high level of automation is desirable is for highly complex phenomena (Smelik et al. 2014, p. 1; Ebert 1994, p. 1), for example in simulation of fluid and gas flow (Stam 1999). There, human intervention might be detrimental to the realism of the scene or possibly put the simulation into an unstable state.

3.1.4 Procedural Rules and Methods

The terms *procedural rule*, *procedural method*, and *procedural approach* do not have an established usage and can take on different meanings. For the purposes of this thesis, we will assign each a fixed meaning. To do this, we first have to explain the possible meanings these terms can assume. We start with two examples.

First, consider the generation of a plant. Most frequently, plants are generated using a grammar, see Figure 3.5a, that specifies how the stems of a plant branch and how they grow leafs and buds (Reffye et al. 1988; Prusinkiewicz and Lindenmayer 2014). This grammar could be different for different species, but it can also be parametrized so that different but similar species can be generated. This grammar, a blueprint describing how the plant is to be generated, is then interpreted by an algorithm.

For another example, imagine we are generating different kinds of airplanes, see Figure 3.5b. One way to do this could be data-driven. Thousands of example airplane models would be provided as the training dataset on which an algorithm would learn. After the learning phase, the algorithm would then synthesize – "predict" – new airplanes based on the airplanes it has already seen (Merrell 2007; Talton et al. 2012; Kalogerakis et al. 2012; Ritchie et al. 2018), possibly reusing modified parts of the wings, stabilizers, or fuselages.

The two above examples allow us to distinguish two important parts of the procedural modeling process: the changeable part describing the desirable output, and the fixed algorithm that interprets the changeable part. We call them the *procedural rule* (changeable) and the *procedural method* (fixed algorithm):

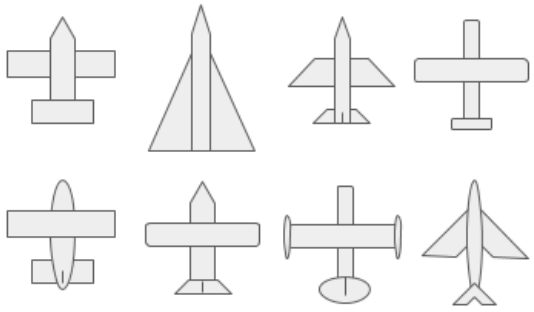
- *Procedural Rule.* A procedural rule is created by a programmer or an artist who also has expert knowledge about the desirable output. It contains *generic* and *abstracted* information about how a new instance should look like or be constructed, see Figure 3.5. By changing the procedural rule, new kinds of outputs can be described and subsequently generated.

  In the flowers example, see Figure 3.5a, the procedural rule would be the grammar defining how a flower's stems are to branch and where leafs and buds are to spring up. For the airplane example, see Figure 3.5b, the

iterations = 10, $\delta = 60°$

$\omega : A \sim$ petal
$p_1 : A \quad : * \quad \rightarrow [-/\sim K][+/\sim K]I(0)/(90)A$
$p_2 : I(t) : t \neq 2 \rightarrow FI(t+1)$
$p_3 : I(t) : t = 2 \rightarrow I(t+1)[-FFA][+FFA]$

(a)



(b)

Figure 3.5: Examples of procedural rules. (a) An L-system defining a Lilac flower head, adapted from Prusinkiewicz and Lindenmayer (2014, Figure 3.19). (b) A simplified example of a database of airplanes.

procedural rule would be the example airplane objects themselves, defining the parameters and bits and pieces that serve as the basis for any output.

Authoring a procedural rule is a non-trivial process and much effort has been dedicated to making the procedural rules easier and more user-friendly (Kelly and Wonka 2011; Patow 2012; Silva et al. 2013; Smelik et al. 2011).

- *Procedural Method.* A procedural method is an algorithm or formalism – possibly one of those enumerated above and by Smelik et al. (2014, Table 1) – that only serves as an interpreter of the *procedural rule* and generates the final object. It is *generic*, meaning it can process any *procedural rule* in the correct format.

  In the flowers example, the procedural method would be the parser and grammar interpreter that would also be able to piece together the resulting 3D flower. In the airplane generation example, the procedural rule would be the implementation of a machine learning algorithm that extracts the characteristics – "learns" – from the example airplanes and synthesizes – "predicts" – new airplanes with similar but different features. For a more in-depth discussion about the internals of procedural methods, see Smelik et al. (2014, Table 1).

In some cases, the procedural rule and the procedural method might get conflated into one. For example, we would understand an algorithm that generates landscapes by iteratively changing the values on a 2D height-map to represent both the procedural rule and the procedural method. There, the procedural rule would be understood to be hard-coded into the algorithm.

Finally, we use the term *procedural approach* to denote a procedural method in a general sense, without being interested in the details of the algorithm.

3.1.5 Procedural Flow

The diagram in Figure 3.6 shows the data flow for a procedural method. The procedural method (gray) takes three inputs (green): the *user input*, the *random generator state*, and the *procedural rule*. It then uses these three inputs to generate an *output model* (blue).

The (pseudo) *random generator state* determines all the random decisions taken by the procedural method. Specifically, if a procedural method makes use of a random generator, it is *stochastic*. As is to be expected, if we fix the remaining parameters and run the procedural method twice with the same random generator

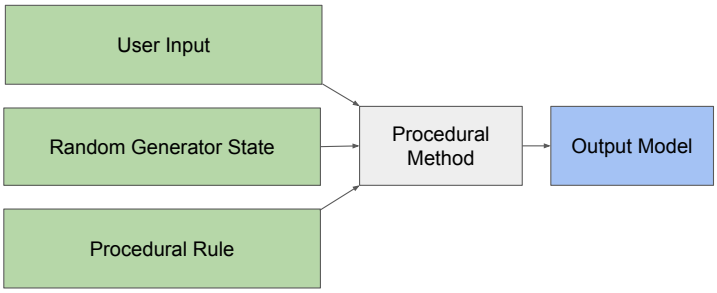


Figure 3.6: Data flow for a procedural model.

state, we will obtain the same results. Conversely, by supplying a new random state, we can make a stochastic procedural rule generate new output models. Through the rest of this text, we will assume all procedural rules are stochastic.

Next, the *user input* is a set of values supplied to each run of the procedural method. It typically constrains or *steers* the algorithm in some ways. In our tree example, user input would be the species, size, age, etc., or *steering input* such as the expected shape of the tree's crown, the layout of the branches, an so on. Such user input might also be given interactively in the middle of the generation process, but without loss of generality, we can still understand it as being given at the very beginning of the algorithm run.

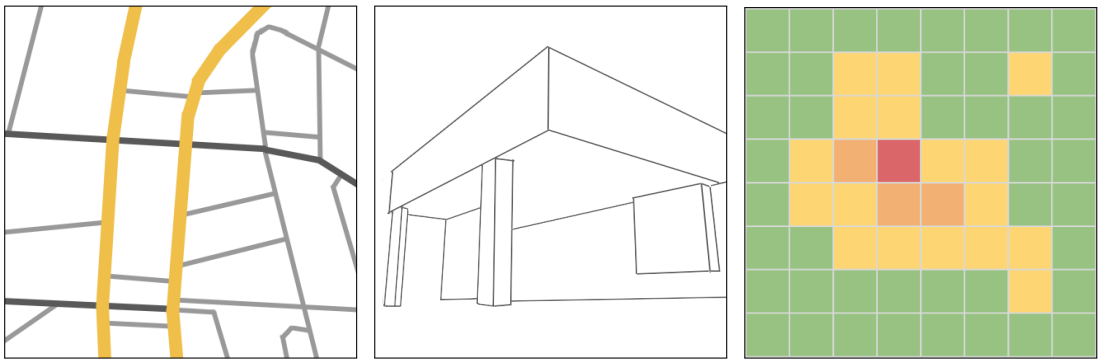
In later discussion, we will say that we are running the procedural rule instead of saying that we are running the procedural method with the procedural rule as a parameter. This is a convention that allows us to simplify discussion. In practice, this can be achieved using techniques from theory of computation (Chiusano and Bjarnason 2014, Section 2.6), namely by partially applying or fixing the procedural rule parameter for the procedural method and then referring to this newly created function as the procedural rule.

3.1.6 Representational and Full Models

We will now divide procedural models into two different categories: *representational models*, which aim for functional realism, and *full models*, which aim for more visual detail and full realism.

For a non-procedural, manually modeled geometry of a real object, we would call the real life object it was modeled after the *reference* object. However, procedurally modeled objects do not usually have a real-life *reference* object, see also Section 2.2.4. It is nonetheless practical to imagine that each procedural model mimics a would-be *imaginary reference* object that would exist in a real or imaginary world, with its implied real world or imaginary world detail. With this, we define *representational* and *full* models as follows:

1. *Representational Model.* A representation of the important features of an *imaginary reference*. It does not include geometric or visual detail that is irrelevant to the function of the *imaginary reference*.



(a) City road network (b) Building volume (c) Species in biotope

Figure 3.7: Representational models.

A typical example would be the layout of a city with the major and minor streets in different widths and colors, the crude shapes constituting the volume of a building, without geometric detail, or species distribution in a biotope. These examples are depicted in Figure 3.7.

A good representational model would achieve high *functional realism*, or fidelity (see Section 2.2.2).

2. *Full Model.* A model that attempts to provide all or some of the detail of an *imaginary reference* and that contains geometric or material appearance information that go beyond just providing information about the overall content or function of the object.

   The level of detail for full models can vary from application to application, dictated by technology and resources. Nonetheless, full models always try to provide additional geometrical or visual detail. Importantly, for a *full model* to be considered realistic, its underlying *representational model* also needs to be realistic, in line with the discussion from Section 2.1. A good *full model* would achieve a high level of full realism (accuracy).

As part of the generation process, a procedural modeling method might also generate or assign material appearance information to various parts of the geometry. For example, a building generation algorithm might assign brick, glass, and wood material appearances to walls, windows, and door frames, respectively. In other cases, the material appearance assignment could also be parametrized. For example, a weathering algorithm, as shown in Figure 3.1b, might want to continuously adjust material appearance parameters based on the position on the model.

Whether the procedural model represents detail as geometry or as material appearance, a duality discussed in Section 2.1, depends on the procedural rule and is not important for the discussion. Last, a geometrical model without material information can still be a useful step towards obtaining a full model. For further discussion of this topic with respect to evaluation see Section 3.3.

3.1.7 Generation of Models

In this Section, we talk about the mechanics of generating models and distinguish two basic cases. The discussion is based on previous work in (video) game studies, which defines *procedural content generation*, a term roughly equivalent to procedural modeling, but in the context of video games. See (Togelius et al. 2011a, p. 6) for an extended discussion. A more comprehensive review of procedural content generation can be found in the book of Shaker et al. (2016). Furthermore, Summerville et al. (2017) review machine learning applications in video games.

When generating new procedural models, two basic approaches can be taken: a *constructive* one, and a *search-based* one (Togelius et al. 2011b, p. 4; Togelius et al. 2012). This distinction has implications on how the generated models can be evaluated, see Section 3.3.

A *constructive approach* works by directly generating an output model. A typical example are L-systems, where a new output model is generated by simply iterating on the provided set of rules. Constructive approaches also often employ simulation, see also Section 3.3.5, to generate the result.

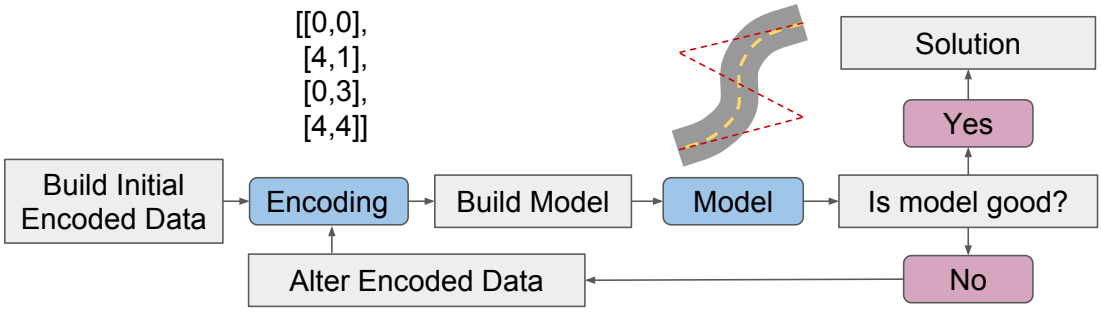


Figure 3.8: Search-based model generation.

A *search-based approach*, illustrated in Figure 3.8, works by devising an encoding that can represent *possibly* desirable output models (Togelius et al. 2011b, p. 4). An encoding instance is then generated and evaluated using an *evaluation function*, for example by examining whether a generated building's balconies are accessible, see Figure 3.9. If the encoding instance is seen as *acceptable*, it is returned as the generated output model. Otherwise, the encoding instance is modified and the process repeated (Togelius et al. 2011b, p. 3). Oftentimes, there is also a mechanism that allows the algorithm to keep exploring for some time after the first solution is found in hopes of finding a better solution.



(a) (b)

Figure 3.9: Inaccessible balconies in procedurally generated buildings.

To give an example, a track for a racing game can be represented using Bézier curves, see again Figure 3.8. The set of all lists of Bézier curves constitutes the encoding and each list of Bézier curves is an encoding instance. A list of random Bézier curves is then used as a starting encoding instance and transformed into an output model when a corresponding race track model is generated. On this track model, an evaluation function is run, evaluating, for example, curvature and expected time to finish. If the result of the evaluation is *acceptable*, we have a result. If it is not acceptable, we modify the random list of Bézier curves we started with, e.g., by perturbing the values of the control points slightly or by adding or removing curves, and repeat the process.

There is a distinction between the model being desirable and the model being acceptable, as determined by the evaluation function. This is because the evaluation function itself is not always perfect and will accept undesirable results,

much like a grammar in a grammar-based approach is usually not perfect and will also generate undesirable outputs.

Search-based approaches are quite frequently a reformulation of global optimization methods using metaheuristics (Georgios N. Yannakakis 2018, Section 4.3), such as gradient descent, simulated annealing, and others. Luke (2015) is an excellent, approachable introduction to such methods. Weise (2009) provides a more in-depth treatment.

As far as we know, computer graphics literature, unlike video game studies, has largely stayed away from formalizing the distinction between *constructive* and *search-based* procedural modeling, although both approaches have been used.

The choice of either a *constructive* or a *search-based* approach has various implications. Most importantly, a search-based approach requires an *evaluation function*, whereas a constructive one does not. Here, Togelius et al. (2011b, III.B) note that constructing an evaluation function so that it captures all of the wanted qualities, especially ones related to aesthetics and style, may be difficult. On the other hand, although examples to the contrary exist (Krecklau et al. 2012; Schwarz and Müller 2015), aesthetic requirements and similar constraints are usually not naturally and easily expressed in many of the constructive formalisms, including grammars. The choice of whether or not to use a search-based method should therefore be carefully considered. Finally, the constructive and the search-based approaches can also be combined, as discussed by Togelius et al. (2012).

3.1.8 Summary

Procedural modeling is a way of automatically or semi-automatically generating novel content. It is typically used to lower the workload on artists and to generate objects that are too vast or too complex to model by hand.

A *procedural modeling approach* normally consists of two parts: the *procedural rule*, which contains information about the content we want to generate, and the *procedural method*, a formalism and an algorithm that interprets the procedural rule and generates the new resulting objects called *procedural models*.

The procedural method takes the *procedural rule*, the *random generator state*, and the optional *user input* as its parameters. When we apply a procedural method on these inputs, we get a different *output model*. However, we will sometimes talk about about the procedural rule and its parameters as if they were the procedural method. This "trick" is detailed at the end of Section 3.1.4.

Next, a procedural method can be *constructive*, meaning it directly generates its output, or *search-based*. In the latter case, an *evaluation function* is used to judge the quality, or acceptability, of the proposed inputs.

Finally, the generated procedural models can be split into two groups: *representational models*, which roughly aim for functional realism, and *full models*, which try to achieve full realism.

In the following Section, we will talk about the outputs of procedural rules and their performance.

3.2 Characteristics of Procedural Rules

Probably the most important quality of a procedural rule is that its output matches the desires of its author. In other words, the set of all *output models* of the procedural rule – its *procedural range* – should match the outputs the author desires – the *desirable outputs*. In this thesis, we presume the author desires the results to be realistic.

In this Section, we discuss the *procedural range* and the *desirable outputs* and combine them to arrive at a definition of the *generative power* of a procedural rule. Aside from introducing this terminology, this Section also serves as a prelude to an upcoming Section 3.3 about evaluation of realism and plausibility in procedural modeling.

To the best of our knowledge, the procedural range, the desirable outputs, and the generative power of a procedural rule have only been mentioned in passing in procedural modeling and computer graphics. This is unfortunate, as better exposure and understanding of these factors would allow researches to better report realism and plausibility in procedural modeling.

In this Section, as well as in this Chapter, we are mostly aiming for *exposure* and basic discussion. In addition to that, we also propose some naive approaches that could be adopted as a baseline.

3.2.1 Procedural Range

First, we look at ways to understand the procedural range of a procedural rule. This is a challenging task, since we ideally want a procedural rule to be able to generate an infinite number of novel output models. We discuss two possible approaches: a black-box one, where the procedural range is described by sampling, and a white-box one, where the knowledge of the procedural method's and procedural rule's internals allow us to better understand the procedural rule's range.

First, we consider the procedural rule a black-box. If we understand the procedural rule to be a function of its random generator state and user input, as stated in Section 3.1.5, we can evaluate this function multiple times – yielding various results thanks to changes in the random generator state – to arrive at a subset of the procedural range. We can then analyze and evaluate this subset. This method of approximating a population by randomly selecting its subset – sampling – is commonly used and understood, e.g. (Kingdom and Prins 2016, p. 101; McClave and Dietrich 1988, p. 140; Field et al. 2012, Section 2.3), although the rigor with which it is used in different applications varies. In procedural modeling, sampling is in our experience commonly used where the user input can be generated automatically, e.g., building size, tree shape, or terrain type. It is also typically, knowingly or unknowingly, performed by the authors of the procedural rule when they run and re-run their rule and investigate the individual outputs.

With sampling, the biggest question is about the representativeness of the collected sample. In many cases, we do not know how to choose truly representative samples, as we do not understand in what way the inputs, that is both the random generator state and the user input, map to the outputs.

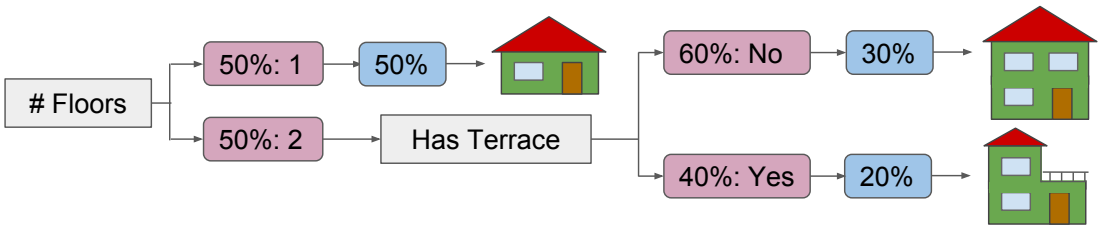


Figure 3.10: A Simple procedural rule for generating houses and its computational paths. Purple probabilities are relative chances of different computational paths being taken based on a random decision. Blue probabilities are probabilities for each path.

The most common strategy to evaluate the collected samples then seems to be to manually inspect the outputs, and perhaps investigate how the inputs map to outputs, and make a human judgment about its variation and representativeness.

For some procedural rules, we can gain further insight about the necessary sample size and an appropriate sampling algorithm by analysis – the white-box approach. For example, we can analyze the computational paths the procedural rule can take, as illustrated in Figure 3.10, learning the probability of each path and the total number of paths. In the depicted example, we would know that a sample of $5n$ single floor houses, $3n$ two-floor houses without a terrace, and $2n$ two-floor houses with a terrace, where $n \geq 1$, would be representative, as it represents all possible outputs proportionately. This example is very simplified and normally, we would also have to sample other parameters, such as the size or the color of the house, etc.

For *search-based* approaches and other approaches that use some sort of internal encoding, we can analyze this encoding and the evaluation function. If the "build model" step, see Figure 3.8, maps the encoding to the model in a roughly continuous way, and when we know the distribution of the encoding, we can just sample this encoding densely enough and filter out results rejected by the evaluation function, yielding the final sample. However, knowing what densely enough means, especially for continuous variables, requires human judgment. For non-continuous "build model" steps, difficult to analyze encodings, or otherwise hard-to-approach methods, a white-box approach does not seem appropriate.

Furthermore, for other procedural methods, such as those based on neural networks, the current state of the art gives us little hope of being able to analyze and fully understand the procedural rule and therefore the representativeness of the sample with respect to the procedural rule.

Without deeper understanding of the internals of the procedural rule and method at hand, see also (Smelik et al. 2014, Table 1), the above described naive sampling approach provides us with at least the basic tools to investigate the procedural range and its representativeness.

3.2.2 Desirability, Realism, and Plausibility

For the rest of Section 3.2, we will frequently refer to procedural models and outputs of a procedural rule that are *desirable*, that is those that the author of the procedural rule desires. As mentioned at the beginning of Section 3.2,

we assume authors want to generate realistic outputs, meaning that a desirable model should also be realistic or plausible.

3.2.3 Desirable Outputs

The set of *desirable outputs* is the set of all outputs the author of the procedural rule would desire.

If we knew the contents of this set of desirable outputs, we could compare it against the procedural range and establish a procedural rule's performance metrics. In some limited cases, such as inverse procedural modeling (Šťava et al. 2010; Martinović and Gool 2013; Vanegas et al. 2012c), where the desirable outputs are well defined, this is possible. However, in the general case, this is very difficult, as the desirable outputs are very loosely defined, for example as "Parisian looking houses" or "hilly terrain".

One might argue that the desirable outputs can be defined implicitly by the procedural rule. However, there is virtually always a *mismatch* between the procedural range and the actual desirable outputs. The procedural range alone is therefore not a good way of describing desirable outputs.

Therefore, in order see what outputs are and are not desirable, we actually need to be interested in a different and usually smaller set of *representative examples*. These representative examples are images or models that were used as motivation for the procedural rule. In practice, this set tends to be small in size. It may not even be concretized in actual models or images, being instead represented only in the mind of the author. Although the name implies otherwise, we do not consider these *representative* examples to be a sample of some set of all representative examples, but simply a set of images. We can then understand the desirable outputs as a *generalization* the procedural rule makes over these representative examples.

An important observation is that we cannot have a way of finding – rather than just identifying – the desired outputs for anything but trivial cases. If we did, we could just choose output models at random from this set of desired outputs, by which we would have solved the entirety of the procedural modeling problem. Similarly, since the desirable outputs are a generalization over the representative examples, even having the whole set of representative examples will not help us.

The above set of representative examples is best defined for database-driven approaches such as (Kalogerakis et al. 2012; Ritchie et al. 2018; Merrell 2007; Talton et al. 2012). There, it is equal to the respective databases that are used for training the procedural rules. For non-database-driven approaches, we could use the set of images or models the author of the procedural rule worked from, or possibly attempt to reconstruct it ex-post, like we did in Chapter 4.

We therefore conclude that the only apparent way of dealing with desirable outputs is by having *representative examples* and a way to tell whether an output model is its generalization. Typically, this is done by human evaluation. Computer-based evaluation could use methods such as the discriminative neural networks from GANs, see Section 2.6.2, or other neural-networks-based methods. However, it appears existing literature is limited to judging the visual realism of image patches (Wu et al. 2011; Rahmouni et al. 2017; Quan et al. 2018), rather than that of the overall content.

3.2.4 Generative Power

Above, we have talked about a procedural rule's *procedural range* and *desirable outputs*. We now very loosely define *generative power* as the ability of a procedural rule to generate *desirable outputs*. Intuitively, we understand that generative power should be defined in terms of a procedural rule's ability to generate all of the desirable outputs and not to generate the undesirable outputs.

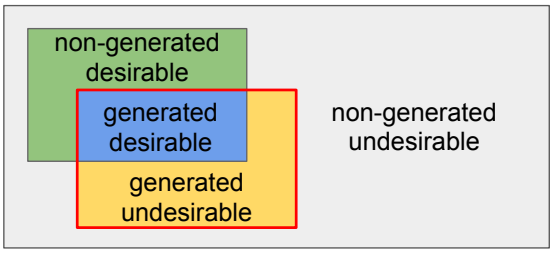


Figure 3.11: The space of all models. Red outline denotes the procedural range.

In Figure 3.11, we see the space of all possible models, both desirable and undesirable. In the red outline, we show the procedural range – the set of all procedural models a procedural rule generates. We also see that two kinds of error can occur. First, there can be outputs we desire, but cannot generate (non-generated desirable, in green), which make the procedural rule not expressive enough. Then there are outputs that we generate, but that are not desirable (generated undesirable, in yellow), which make our procedural rule not specific enough. Both of these errors can and typically do also combine. Finally, there is also the much larger set of non-generated undesirable models (gray).

The two error types mentioned above are similar to Type I and Type II errors in statistics (Field et al. 2012, Section 2.6.3) and *false negatives* and *false positives* in machine learning and information retrieval (Fawcett 2006, Figure 1). We illustrate this parallel in Table 3.1.

procedural model	is desirable	is undesirable
has been generated	true positive	false positive
has not been generated	false negative	true negative

Table 3.1: Confusion matrix of the space of all procedural models.

The relative sizes of categories defined by the rows, columns, and their intersections in Table 3.1 are our primary statistics. We hope to combine them to arrive at a meaningful statistic – metric – of *generative power*.

3.2.4.1 Precision

Borrowing further from machine learning, we define the *precision* (Fawcett 2006) of a procedural rule as

$$\text{precision} = \frac{\text{generated desirable}}{\text{all generated}} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}.$$

Precision is the proportion of desirable results in the generated results. It is also what one would intuitively evaluate when faced with the results of a new procedural rule. If, for example, a procedural rule generated 90 desirable results out for every 100 results it generates, its precision would be 0.9.

Because we normally cannot easily express the number of generated desirable models analytically, see Sections 3.2.1 and 3.2.3, we opt to estimate them using other means. We generate a fixed size *sample* of the results of the procedural rule and manually evaluate this sample in order to split it into the desirable subset and the undesirable subset. Assuming the sample is considered representative, see Section 3.2.1, we obtain the relatives sizes of *true positives* and *false positives* and can calculate the precision of the procedural rule.

3.2.4.2 Recall

Another useful metric would be

$$\text{recall} = \frac{\text{generated desirable}}{\text{all desirable}} = \frac{\text{true positives}}{\text{true positivies + false negatives}}, \tag{3.1}$$

which would tell us what proportion of all desirable models we can generate. As discussed in Section 3.2.4.1, we can obtain the relative number of generated desirable instances (true positives) as well as the number of all generated instances in a sample. This means that we now either have to compute the number of non-generated, desirable models (false negatives), or the relative size of all desirable models. We will illustrate that neither of those can be obtained in a practical way.

To obtain the relative number of all desirable instances out of *all models*, we would have to sample the set of all models and then determine what proportion of these models is desirable. While this could be done using human evaluation, the non-desirable models are expected to heavily outnumber the desirable models. This makes the problem of determining the relative number of all desirable instances almost intractable in practice.

Two possible solutions to this problem are automated evaluation of desirability, see also Section 3.4.2, and obtaining an analytical formula for the relative number of all desirable models. As discussed in Sections 3.2.3 and 3.2.1, those are hard problems.

The alternative is to obtain the number of non-generated, desirable models (false negatives). One way is to obtain all desirable models, the denominator in the middle expression of Equation 3.1, and identify those models that have not been generated, which requires solving the above problem of obtaining all desirable models and is thus at least as hard. Another way is to obtain all non-generated models, see the denominator in the right-most expression of Equation 3.1, and then see what number of those is desirable (false negative). Obtaining the relative number of non-generated models, however, again means sampling the set of *all* models and then determining which could possibly be generated. This would require being able to solve the inverse procedural modeling problem for said instances and procedural rule, which is non-trivial. It also assumes that those models that could be generated are the ones that would be generated.

3.2.4.3 Discussion

Since we cannot reasonably obtain the recall of a procedural rule, we also cannot define other metrics. One such metric is the F-score, a biased (Powers 2011, p. 1) but commonly used metric which requires knowledge of both precision and recall. Another, less well-know metric is the Matthews Correlation Coefficient (Powers 2011, Equation 19), which is unbiased, but also requires knowledge of false negatives. It expresses the correlation between observations and predictions and works better for imbalanced classes (Boughorbel et al. 2017, p. 4). Additional possible metrics can be found in (Powers 2011; Boughorbel et al. 2017).

Both of the above metrics would give us a better idea about the whole of the confusion matrix, in Table 3.1, and consequently about the performance of the whole procedural rule. Specifically, using just precision does not tell us anything about how many desirable models we do not generate.

For search-based methods, the situation is similar. While search-based methods do classify candidate procedural models as acceptable and unacceptable, this classification is not always equal to desirable and undesirable, as discussed in Section 3.1.7. Additionally, an evaluation function is usually created together with the corresponding encoding and is only expected to work and sometimes even only defined for models generated from that encoding. For example, an evaluation function that tries to judge whether a racing circuit is acceptable, as illustrated in the example in Figure 3.8, might fail or give nonsensical results if given the model of an airplane.

Fortunately, in procedural modeling, generating undesirable outputs is probably worse than only generating a subset of all the desired outputs. Therefore, while not ideal, having *precision* as a metric of generative power of a procedural rule seems adequate. Due to the unavailability of an automated approach to evaluation, however, determining the precision of a procedural rule still requires human evaluation, something that prevents wide-spread adoption.

In the above text, we assumed the decision about a procedural model's desirability is binary: the model either is desirable, or it is undesirable. However, common evaluation methodologies, see also Section 3.3, assign a score to each evaluated (procedural) model. For example, a completely realistic or desirable model would get a score of 1.0, and a completely non-desirable or non-realistic model would get a score of 0.0. We can easily convert these scores to the binary labels "desirable" and "undesirable" by applying a threshold, see also Section 3.3.2.

Note, however, that in such a situation, investigating the ROC curve (Fawcett 2006, Section 4) is not meaningful, as we would be investigating the performance of human test subjects, not the procedural rule.

Finally, it unfortunately seems that at the moment, the only practical way of improving the generative power of a procedural rule is to use human judgment, manually improve the procedural rule, and re-evaluate.

3.2.5 Summary

In Section 3.2, we have discussed the concepts of *desirable outputs*, *procedural range*, and *generative power*. Specifically, we have argued that a procedural rule is modeled after *representative examples*. The set of *output models* it can generate is called the *procedural range* and can contain both *desirable* and *undesirable output*

models. Similarly, the set of procedural models it does not generate can also contain both *desirable* and *undesirable output models*.

We have postulated that in general, one is unable to enumerate the set of all *desirable outputs*. Doing so would mean we already have a perfectly performing procedural rule, meaning we would have solved the procedural modeling problem for this specific rule.

Finally, we set out to measure the performance of a procedural rule – its *generative power*. Ideally, we would like to employ established metrics from statistics. Towards this end, we have adapted terminology from video game studies and machine learning to procedural modeling. However, since we are unable to enumerate and/or establish the relative size of *all desirable outputs* and/or *false positives*, we cannot define *recall* or more advanced metrics such as the F-score or the Matthews Correlation Coefficient. Consequently, we resort to using *precision* as the measure of a procedural rule's *generative power*.

3.3 Evaluation

In this Section, we will talk about evaluating realism in procedurally generated models. We will not provide a precise methodology, but will rather try to outline the relevant factors and topics. Some of these factors, such as those in Sections 3.3.3 or 3.3.6, are not specific to procedural models and apply equally to manually created models. Nonetheless, we include them here because of their relevance.

While evaluation of realism in procedurally generated models and their images is not the same as evaluation of realism in other models and images, a lot can be learned from existing literature. Mantiuk et al. (2012) describe and compare the efficacy of four different methods for subjective evaluation of Image Quality Assessment. Those could be easily adapted to evaluate the *realism* instead of the *quality* of images. Pairwise comparison-based experimental setups for computer graphics are discussed by Perez-Ortiz and Mantiuk (2017). Finally, a very detailed treatment of psychophysical experiments is given by Kingdom and Prins (2016).

Most of the remarks below are made with human evaluation in mind, but could plausibly be adapted to automated evaluation. Many of these factors are discussed in the context of an actual experiment in Sections 4.3 and 4.4.

Throughout this Section, we also assume the availability and use of a full global illumination enabled renderer, such as the Corona Renderer (Render Legion 2019). Only the other two elements of realism – geometry and material appearance – are considered.

3.3.1 Rules and Methods

It is important to understand that we want to evaluate the quality of the output models of the *procedural rule*, not the expressiveness of a *procedural method*. Therefore, such an evaluation has to be done for each and every newly authored procedural rule, even if the underlying formalism – the *procedural method* – remains unchanged.

Furthermore, the typically understood meaning of evaluating realism is to investigate whether the *generated* models are realistic. In terms of the terminology

used in Table 3.1 and Section 3.2.4, this means investigating the *precision* by determining the relative numbers of *true positives* and *false positives*.

To the best of our knowledge, the *false negatives*, that is the desirable but non-generated output models, are usually not a part of a quantitative investigation methodology, although they can be discussed in a qualitative sense.

3.3.2 Realism as a Score

As stated in the discussion on generative power in Section 3.2.4, a typical methodology for evaluating the realism of an image or a model gives us a score. This score can be converted to a probability (Fawcett 2006, Section 4). Such a probability then expresses an accumulation of various, typically human, judgments.

However, the question remains about what threshold should be used to convert these scores to binary labels. Assuming the probability expresses how likely the image is to be considered realistic, we can also reformulate the matter as: "What fraction of people do we want these images to convince of their realism?".

If we used references during the evaluation, another possible approach would be to also look at the probabilities for the reference images or models and use the median score or one of the other percentiles. This would mean that for a generated model to be considered realistic, it has to be better than half, or some other portion, of the considered references. Another approach is outlined by Čadík (2008, p. 75), who uses a probability of 0.75 as a standard threshold for discrimination tasks.

3.3.3 Content and Audience

When evaluating realism and plausibility, it is important to keep in mind what exactly it is we are trying to achieve. For example, the study in Chapter 4 evaluates realism of buildings. However, it is constrained to evaluating the realism and plausibility of the buildings as shown from the outside. The interior is only alluded to with window layout and textures.

Similarly, sometimes there is a difference between what is realistic and the amount of realism the target audience – expert or layman – can appreciate. For example, Gain et al. (2015, p. 9) found that non-experts preferred synthesized terrains over real ones. Similarly, the more realistic layout of roads in Chapter 5 might contribute more to a transport simulation video game where cities are viewed from above than to a first-person view game, where the difference is presumably harder to appreciate.

Naturally, we should also keep in mind whether we are evaluating representational models or full models, see Section 3.3.5, and instruct the test subjects accordingly. Here, comparisons with references that have been adjusted to a similar format, see Figure 3.7, might be useful in preventing biases.

3.3.4 Asset Reuse

Procedural rules contain biases where one or more of the features of the procedurally generated models are over-emphasized or under-emphasized. For example, a specific color or a constituent piece of geometry in a database-driven approach



(a) (b) (c) (d)

Figure 3.12: Example procedural models of Parisian buildings that share common mansard windows.

might be inadvertently overused, or there might be a tendency to assign excessive weights to certain octaves when using noise to generate height-maps.

To demonstrate, one of the procedural rules used in the realism experiment we introduce in Chapter 4 had a tendency to overuse the same type of mansard windows, see Figure 3.12. A little over a fifth (7 out of 33) of the generated procedural models used this type of mansard windows and there were only 5 mansard window types for the whole artificial dataset of 33 images. In contrast, no more than two buildings out of the 48 reference photographs shared the same type of mansard windows.

Although the results concerning the test participants' ability to notice this bias – see Chapter 4, especially Sections 4.3.2 and 4.6 – are mixed, we still believe further research into detecting or demonstrating this phenomena, with a more appropriate methodology and a large dataset, is warranted.

3.3.5 Representational and Full Models

There is always pressure in computer graphics to present visually pleasant images. In addition to that, representational models might not always be particularly eye-catching, see Figure 3.7. These two factors cause many results to be presented with the representational part augmented with realistic material appearance and possibly additional hand-crafted details, as if they were a full model.

In order to avoid bias when evaluating representational procedural models, for example models of road networks, as in Chapter 5, it is necessary to evaluate desirability and/or realism without added material appearance or geometric detail.

Next, one might be tempted to be more confident about the realism of procedural models generated using a simulation. However, not all representational models are the result of a perfect simulation. Shortcuts might be taken during the simulation to prevent excessive computational times and implementational complexity. Other times, only heuristics are used to generate results instead of a proper simulation.

For full models, we might face the opposite pressure: not to evaluate their representational part. One should also avoid this urge to prevent results such as those shown in Figure 3.9. Such a situation would mostly happen by not explaining well enough to the test subjects during evaluation what they should be looking for. For example, test subjects in Chapter 4 tended to focus on reflections

in windows, see Section 4.5.2.1. One unverified hypothesis is that users identified some of the most easily noticeable tell-tale signs and failed to follow-up on others.

By instructing the test subjects to focus not only on the visual appearance, but also on the plausibility, we might be able to achieve a more substantial evaluation. However, one might also argue that this already changes the quality that we are evaluating from realism to something else. An alternative could be to evaluate representation and full realism separately.

3.3.6 Model-based and Image-based Evaluation

Another decision to make is whether to evaluate the 3D models themselves, or whether to evaluate their images. A similar dilemma is faced by literature on assessing the visual quality of 3D models, surveyed by Corsini et al. (2013) and Lavoué and Mantiuk (2014). Rogowitz and Rushmeier (2001) demonstrated that there is indeed a difference between evaluating a 3D model and its projections.

However, the goal of this field is to understand the role that different transformations, such as 3D model compression or a decrease in a 3D model's level of detail (Lavoué and Mantiuk 2014, p. 15) play in perception and to build mathematical models that predict these changes. Therefore, these methods almost exclusively work in a with-reference setup (Lavoué and Mantiuk 2014, p. 29).

However, since for procedural modeling, there is no *imaginary reference* to evaluate against, see Section 3.1.6, not all of the knowledge can be directly used. Nevertheless, Lavoué et al. (2015) show that for some tasks, image-based comparison with reference and with only simplified rendering is meaningful. This indicates that using images to evaluate the fidelity of representational models without reference *might* also be meaningful, as it shows that the simplified rendering does not necessarily strip the information needed to judge the above mentioned fidelity. However, for more complex interactions, e.g., with textures Guo et al. (2016) or light (Vanhoey et al. 2017), that could be used for evaluating full models, the literature is sparser.

If an image-based comparison is to be employed, the number of views also has to be determined. Previous experiments range from using a single view (Watson et al. 2001, p. 2), to 15 (Rogowitz and Rushmeier 2001, p. 8), up to 42 views (Lavoué et al. 2015, Section 4.1). For comparison, our experiment in Section 4.4.2 only uses one camera position and lighting setup matched with a reference example.

For model-based comparisons, either animations and video (Rogowitz and Rushmeier 2001, p. 2) or interactive exploration (Corsini et al. 2007) have been used. For the evaluation of realism specifically, having the study participant manipulate a 3D object might be detrimental. As corroborated in Section 2.4.1, having a photograph-like comparison might be less disbelief-inducing. Also, with interactive evaluation, we face the additional challenge of interactive realistic rendering (Ritschel et al. 2012).

Additionally, for with-reference comparisons in procedural modeling, we also have to consider the existence of a reference image or model. In some cases, where the reference examples we would compare against come from real life, obtaining ground truth 3D models would need either manual modeling or a 3D acquisition with a tight quality control. In such cases, photographs are the most viable option. Another factor is that for image-based comparisons, a lot of tech-

nical details, such as the trade-off between material appearance and geometry are handled by existing literature (Lindstrom and Turk 2000, p. 15).

Whether we choose to compare models or images, we have to unify the representation of the representative examples and the generated models. For example, if we were to compare procedurally generated road networks such as those generated in Chapter 5 against GIS data, we would have to make sure they are shown in the same style.

In the end, there is a lot of factors at play and the choice of whether to compare the 3D models themselves or their images is largely specific to the application.

3.4 Summary

This Chapter has discussed realism in procedural modeling and was divided into three parts. The first two parts each have their own, more detailed summary available in Sections 3.1.8 and 3.2.5, respectively.

In the first part, we discussed what procedural modeling is, why it is useful, and what are its use cases. We then decomposed procedural modeling into two distinct parts: the *procedural rule* and the *procedural method*, and the generated models into two distinct categories: *representational models* and *full models*. Both of these distinctions enabled us to later discuss performance and evaluation with added clarity. Lastly, we discussed the internals of how procedural models might be generated, adapting previous research from game studies.

The second part of this Chapter was focused on formalizing the performance metrics of a procedural rule. We have categorized procedural models as *desirable* or *undesirable*, and *generated* or *non-generated*. This allowed us to adapt existing performance metrics from machine learning and retrieval theory and show that some of these commonly used metrics are not applicable for procedural modeling under the current state of the art.

Lastly, in the third part, we have outlined and discussed some principles related to evaluating realism in procedural modeling, some of which can also be applied to evaluating realism of generic models. Three of these principles stand out on their own. First, evaluation should be done for each new procedural rule, not each procedural method. Second, there are additional biases in procedural models we should be aware of. Finally, representational models should not be evaluated with added geometrical detail or material appearance.

3.4.1 Contributions

Our contributions in this Section, discussed in more detail above, are:

- Clarifying the nomenclature of procedural modeling. Specifically, making clear the distinction between a procedural method and a procedural rule and discussing the implications, and defining representational and full models.
- Adapting previous work from machine learning and game studies to describe and measure the performance of procedural rules.

- Discussing the need to measure performance of procedural rules, the possible biases, and the need to evaluate representational models in their representational form.

3.4.2 Future Work

The task of determining realism in procedural modeling has many open questions. We see two major directions of interest in future work.

The *first direction* is further investigation of realism in procedural modeling itself. This includes devising better and easier methodologies for evaluation and understanding the relationship between the results of a normal, pair-wise evaluation methodology and different methodologies tailored for procedural models. It would also be helpful to better understand the different biases, such as the one discussed in Section 3.3.4.

Furthermore, it would be beneficial to understand what people look for when judging desirability and realism, as it would allow the authors of procedural rules to focus their effort on specific areas, such as geometry of the generated models, material appearance, or the quality of rendering. Similarly, understanding the relationship between results for static images and results in other contexts and under different conditions, such as video or games, could also help direct effort to specific areas.

Finally, by better comprehending the space of models a formalism such as grammars can generate, we could enable easier generation of new candidate models for search-based approaches.

The *second direction* of future work concerns automation. One of the most tedious and unfortunate aspects of evaluating realism in procedural modeling is that it has to be done for every new procedural rule and that this evaluation is based on user studies. Any meaningful automation would therefore unlock new possibilities.

The biggest, but also the least approachable problem in this Chapter is automatic evaluation of desirability. With it, we could not only optimize for precision, but we could also attempt to optimize for recall. That is, we would see whether it is practical to generate models at random and see whether they are desirable.

Here, neural networks, especially GANs, see Section 2.6.2, seem to be the best approach for solving at least a meaningful subset of the problem.

However, one obstacle to a neural-networks-based approach is that such networks would have to be able to train on and work with a much smaller training set of representative examples.

Additionally, we believe being able to work with semantics is a pre-requisite for meaningful representational models and therefore also for full models. However, it is not clear how much high-level semantics current neural network approaches can decode and encode (Brendel and Bethge 2019; Geirhos et al. 2018).

Chapter 4

Realism of Architectural Procedural Models

4.1 Introduction

Procedural modeling is a set of methods for generating computer graphics content. These methods ordinarily strive to achieve a key factor – realism – to provide a more engaging experience for the user and to increase the value of the final product.

Procedural methods typically approach the assessment of realism by qualitatively evaluating a small number of results. Occasionally a user study that compares the method's results and real world exemplars is employed. However, in general, there is a lack of systematic treatment and understanding of what is considered realistic in procedural modeling, see also Chapter 3. Similarly, only a small number of studies, such as Rademacher et al. (2001) or Ramanarayanan et al. (2007), investigates the realism of content in computer graphics.

We contribute to the discussion about realism by conducting a user study that investigates the realism of *procedurally* generated buildings. Specifically, our study set out to determine:

1. the contribution of both fine and coarse details to the perception of realism of procedurally generated building, and

2. the qualitative factors that contribute to the perception of realism of procedurally generated buildings.

We have found that the perception of realism of procedural buildings depends on different scales of detail in statistically significant ways, detailed in Section 4.5.1. Additionally, we have identified a number of other factors, listed in Section 4.5.2, that contribute to realism of procedural buildings. We discuss how these results generalize to non-procedural buildings in Section 4.6.

Our contributions are 1) proving the importance of coarser scale structure on realism, 2) identifying features that increase or diminish the realism of procedurally and non-procedurally generated buildings, and 3) a novel methodology for investigating procedural models using cut-outs to compare generated objects with reference photographs.

4.2 Previous Work

To the best of our knowledge, there is no systematic treatment of realism for *procedural rules or models*. Many works on procedural modeling explicitly attempt to achieve realism, e.g. Deussen et al. (1998) or Ebert (1994), and numerous other mention it.

An editorial by Reinhard et al. (2013, p. 1) remarks that even low-resolution, "tourist" photographs preserve plausibility and proposes that it is the lack of fine detail that hinders the realism of synthetic, although not necessarily procedural, images. One approach to remedy this lack of detail, due to Johnson et al. (2011), is CG2Real, an image-based rendering method that replaces parts of generated images with parts of photographs. The work also presents a small study to evaluate the realism of its results that is similar to ours. An interesting study into the nature of realism, albeit for photographs in a rendering and setup-of-scene context, was presented by Rademacher et al. (2001), who found that shadow softness and surface roughness have a statistically significant contribution to the perception of realism. We have also discussed these results in more detail in Section 2.4.3.

The procedural models that we investigate are typically created by generative procedural modeling approaches, such as shape-grammars (Stiny et al. 1980), split-grammars (Wonka et al. 2003; Müller et al. 2006b; Schwarz and Müller 2015), data-driven techniques that synthesize buildings from real-world data (Fan and Wonka 2016), or from a set of pre-defined parts (Kalogerakis et al. 2012; Talton et al. 2012), see also Section 3.1.4. However, non-automatic, user-guided techniques, for example sketch-based building generation (Nishida et al. 2016), also make use of information on the realism of procedural buildings.

Closely related, although not directly applicable, are image processing and machine learning approaches (Doersch et al. 2012; Mathias et al. 2011) that could help identify features missing from, or misrepresented on, a procedural model. Furthermore, there is a body of work concerned with image equivalence. Image Quality Metrics (Čadík 2008) assess the arithmetic equivalence of images, whereas the Visible Differences Predictor (Daly 1992) is the primary method for perceptual equivalence. In contrast, we look for similarity in visual appearance, much like (Ramanarayanan et al. 2007; Ramanarayanan et al. 2008). Finally, the Structural Similarity (SSIM) metric (Wang et al. 2004) is used to assess the degradation of an image with respect to a reference image. We discuss these methods more thoroughly in Sections 2.4.2 and 2.5.2.

4.3 Design Considerations

Our experiment compares images of *procedurally* generated buildings to photographs. We aim to better understand the factors which contribute to the perceived realism of procedurally generated buildings, and buildings in general. We are not interest in photo-realism in an image equivalence sense, such as the Visible Differences Predictor (Daly 1992), but rather in the quality of the generated models and their materials.

Below, we explain the procedural aspect of our experiment, discuss whether to run the experiment with or without reference, whether to run it in 2D or 3D, and the effects our experiment investigates.

4.3.1 Procedural Aspect

In procedural modeling, authors create procedural rules (Müller et al. 2006b; Schwarz and Müller 2015) that are expected to generate a *wide* range of building style variations. One of the ways this is achieved is by using random numbers to influence the generation process and generate "unexpected" results. This makes procedural buildings susceptible to errors that are different from errors one would expect from manually modeled buildings. For example, those errors could be incorrectly repeated textures, misplaced geometries, or repetition between models. Additionally, a procedural rule's output relies on the quality and variety of assets – one cannot expect to generate a realistic procedural city with only a limited number of textures – and authors may underestimate the difficulty of attaining realism across a rule. Other factors, such as how consumers would perceive procedurally generated buildings over prolonged periods, or when seen in aggregates (Ramanarayanan et al. 2008), also play a role.

The importance of these factors is an insight that studying solely the realism of perfect, manually modeled buildings could not provide. Consequently, performing this study with manually modeled buildings may result in different factors being detected.

While our design does not aim to investigate all of the above mentioned factors, it is nonetheless designed for procedural rules and conducted on procedurally generated buildings. Consequently, its results are more readily applicable to procedural rules. Applicability of the results to non-procedural buildings is discussed in Section 4.6

4.3.2 No Reference vs. With Reference

We chose to run our experiment with reference. While simply asking how realistic a single image is is tempting, it can also be misleading, as no base-line is given and because participants might operate with different notions of realism (Rademacher et al. 2001, Section 3.1). Other formulations, such as asking how photographic or synthetic an image appears, or explaining what to look for, could introduce preconceptions and bias the experiment.

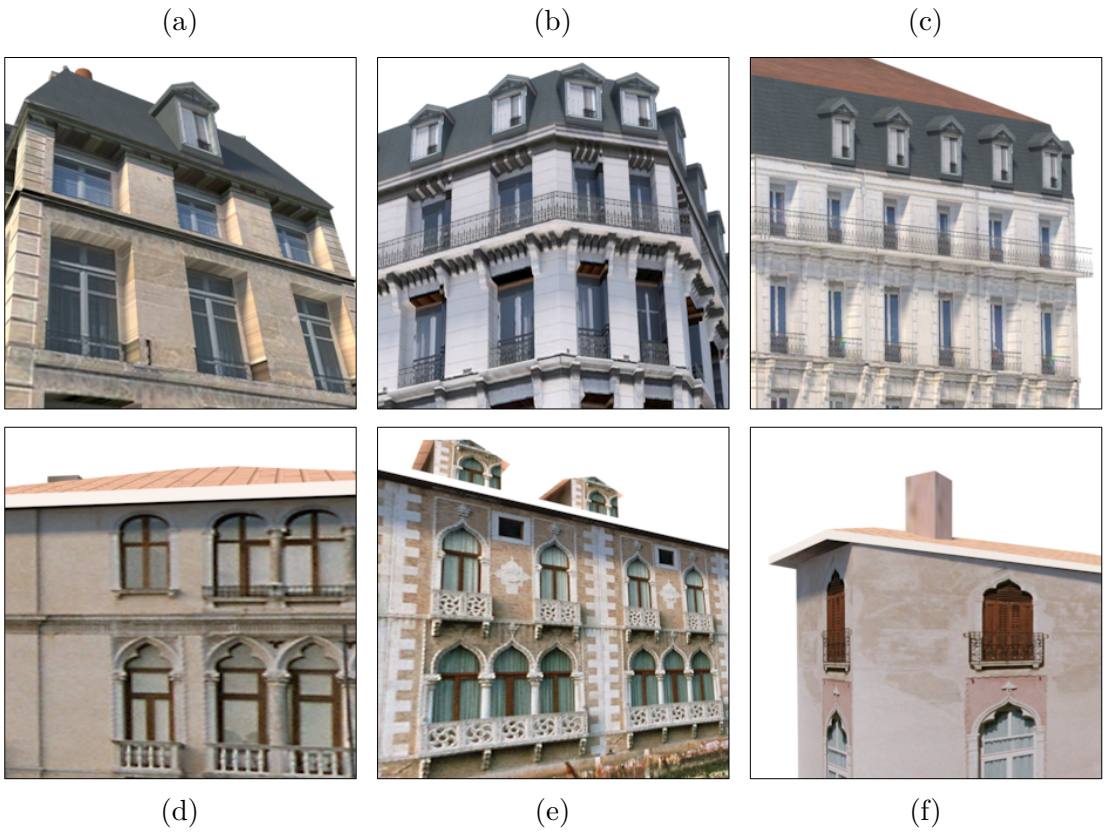


Figure 4.1: Shared features and lack of variation in procedural models. *Top row*: three out of six procedurally generated images of Paris, while different in style, share the same roof windows. *Bottom row*: Three Venetian buildings sharing the same flaw, a white rim around the roof.

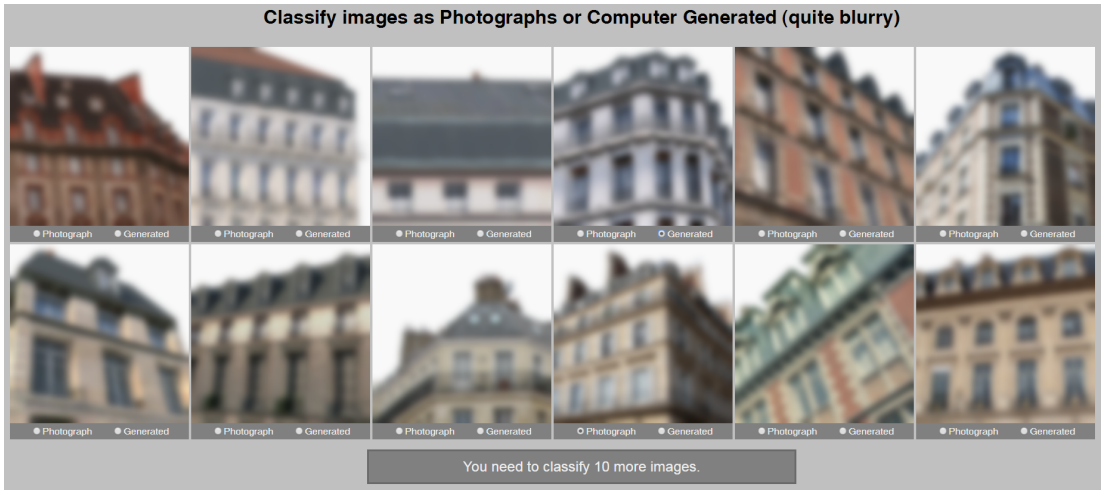


Figure 4.2: Classification screen, shown for the Paris dataset at 37 px Gaussian blur, with 1 image classified as computer generated, 1 classified as a photograph, and 10 images left unclassified.

4.3.3 2D Images vs. 3D Models.

Instead of comparing procedural rules and their generated models in 3D, we compare their *images*, that is photographs and computer generated renders. The main benefit of this approach is that it is considerably easier to acquire and display the reference – in our case, photographs.

The main benefit of choosing to compare 3D models would be that the participants could view the objects from arbitrary angles, but there would also be several drawbacks. The acquisition of a *real world* reference – probably 3D scanning – would be significantly more difficult and perhaps imperfect. It is also not clear how the occluded parts of objects would be acquired and displayed. The 3D reference model could be presented to the participant as a diffusely lit, solid color object. Though, by discarding texture and material information, we would also lose a large part of what makes a 3D model realistic. Alternatively, the 3D model could be presented with its acquired texture, displayed using the same rendering method we would use to display the procedural model. However, this would limit us to real-time rendering methods with a stronger bias, or force us to revert back to 2D. Additional discussion about this topic can be found in Section 3.3.6.

For the above reasons, we consider 2D comparison to be the better approach.

4.3.4 Multiple Images at Once

Our goal is to evaluate the realism of *whole* procedural rules, not the individual models it generated – or models manually created by artists – see Section 4.3.1 above. For such an evaluation, we have to consider not only the quality of the individual models, but also the amount of feature variance between the models, see Figure 4.1. Similarly to Wallraven et al. (2009), we therefore allow participants to see all the images at the same time, in our case in a 6×2 grid design, showing 12 images at once. The number of images was chosen as trade-off between screen size, how well the images, in our opinion, represented the dataset, and expected participant fatigue. Refer to Section 4.7 for further discussion. The layout of our experiment's screen can be seen in Figure 4.2.

4.3.5 Structure and Fine Detail

It is commonly alluded that realism is especially influenced by the presence or absence of fine detail (Reinhard et al. 2013, p. 1; Johnson et al. 2011, p. 1; Guérin et al. 2016, p. 1). Since the layout of a procedural model is often stochastically determined, we also wished to investigate the role of the coarser features and of the structure of a building.

To do this, we assume a roughly similar feature size across images and apply a Gaussian blur to filter out the higher-frequency detail at six different blur levels: 1 px (identity – no blur), 7 px, 13 px, 25 px, 37 px, and 55 px, with the blur level in pixels converted to the Gaussian standard deviation σ using

$$\sigma = 0.3\left(\frac{\text{blur size in px} - 1}{2} - 1\right) + 0.8,$$

see Figure 4.3. These values are based on the results of a pilot study and range from a small 7 px blur that only removes very fine detail to a 55 px blur level

that some study participants considered made the study task impossible, see Section 4.5.2, but that still preserves information about realism, see Section 4.5.1.

The assumption about uniform feature size is a strong one. However, an inspection of the images used in the experiment, see Supplemental Material in Section 4.9, shows that the size of features, e.g. windows and floors, varies only slightly between images. Additionally, sizes of features in both photographs and procedural models are not easily available. Our approach is not ideal and does introduce a bias, but we consider it an acceptable compromise.



(a) 55 px blur (b) 37 px blur (c) 25 px blur

(d) 13 px blur (e) 7 px blur (f) 1 px blur, identity

Figure 4.3: The 6 different Gaussian blurs, applied to a Paris dataset image.

4.3.6 Datasets

For our experiment, we have used four *pre-existing* architectural procedural rules, namely Favela, Medieval, Paris, and Venice, see Figure 4.4. These procedural rules were created by advanced practitioners, matching the profile of the vast majority of procedural rule authors. For each procedural rule, see Section 4.4.1.2, we generated the corresponding images and gathered photographs, as described in Section 4.4, creating four *datasets*. The procedural rules used in our experiment have differing characteristics, see Table 4.1, and provided us with a reasonable range for the experiment. Unfortunately, we were unable to acquire any procedural rules that would generate other kinds of buildings, such as family houses, industrial buildings, or commercial buildings.

(a) Favela (b) Medieval (c) Paris (d) Venice



(e) Favela (f) Medieval (g) Paris (h) Venice

Figure 4.4: Sample images from our datasets. Top row contains generated images, bottom row consists are photograph cut-outs.

Quality	Textures	Geometry	Overall
Favela	Very High	High	Very High
Medieval	Medium	Medium	Low/Medium
Paris	High	Medium	Very High
Venice	Low	Low	Low

Table 4.1: Subjective pre-experiment assessment of procedural rule quality.

4.4 Method

In this section, we first describe the dataset creation process and then describe the experiment itself.

4.4.1 Stimuli – Image Sets

For each dataset, we need to create two sets of images to present to participants for comparison – one set of photographs and one set of generated images. Each of these two image sets needs to a) be representative of the underlying data, and b) provide the smallest possible number of clues as to the origin of the images. There is a trade-off between these two qualities and human judgment is required.

4.4.1.1 Choosing Photographs

If the photographs that were used in creating the procedural rule were available, we could just reuse them. However, in our case, original photographs are not available, therefore we have to collect and process a new set. We chose publicly available photographs with permissive licenses, see the Supplemental Material in Section 4.9, and conjectured that people tend to photograph and know an object from the most representative viewpoints.

Furthermore, we selected our photographs to match the range of models the respective procedural rules typically produce.

Next, we removed photographs with too many occlusions from objects such as cars and people. There were two exceptions to the rule. First, for some sets of buildings, it is practically impossible to find images without people present in those images. Therefore, those images cannot be excluded. We deal with this issue later. Second, we have not excluded photographs with artifacts that could plausibly be a part of the procedural model, such as wall-mounted street signs or lamps. Ideally, those would be generated by the procedural rule. Finally, we remove images with *extreme* angles, exposures, graining, or which are otherwise severely defective.

4.4.1.2 Procedural Rules

In order to create computer generated images, we first have to generate the 3D models. For all four of our datasets, we generated these 3D models using CGA procedural rules (Müller et al. 2006b) interpreted by CityEngine, a state of the art city and building modeling software by ESRI R&D Zürich. The Favela procedural rule was created by Matthias Bühler, an employee of ESRI R&D Zürich, for a personal project. The Medieval, Paris, and Venice datasets were created by unknown employees of ESRI R&D Zürich and are supplied as example procedural rules with the CityEngine software.

4.4.1.3 Generating CG Images

First, we generated a number of 3D models from each rule, described in the above Section 4.4.1.2. We then manually positioned the camera for each model to roughly match the camera angles and focal lengths between the above set of photographs and the set of generated images we are creating.

This manual process seems to be relatively easy to execute in practice and makes sure we do not introduce any *new* bias into the experiment. Nonetheless, our preliminary experiments had shown that people are not good at judging the plausibility of camera angles for neither photographs, nor computer generated images. For that reason, we have instructed participants not to pay attention to the camera angle. Refer to Section 4.5.2 for further discussion.

For each image, we then set the Sun's elevation to a random angle between 0 and 60 degrees and the Sun's azimuth to be behind the camera, with a deviation of ±60 degrees. Next, we rendered our images against a white background, at a resolution of 4000×3000 pixels, until convergence.

To render our images, we used Corona Renderer (Render Legion 2019). We rendered with full global illumination and used the Hošek and Wilkie (2012) sky model. A white ground plane has been used to approximate light interactions with the environment. Finally, we manually adjusted each image's exposure to prevent over-exposure and under-exposure. This process, together with a cropping step discussed below, is shown in Figure 4.5.

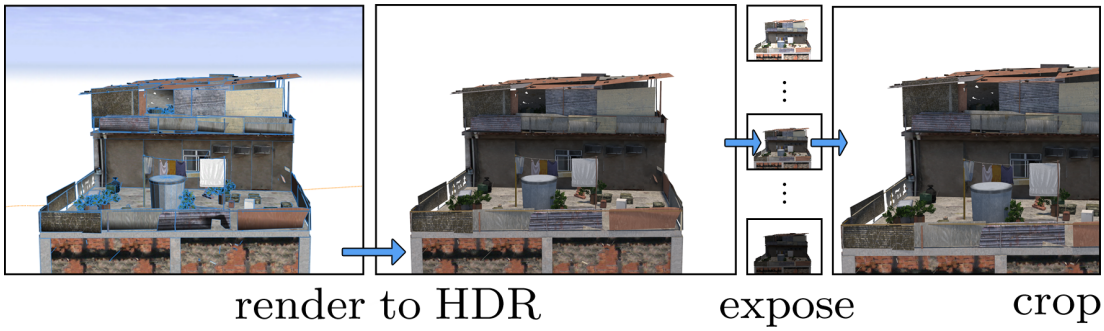


Figure 4.5: Computer generated image preparation workflow. A 3D model is rendered to an HDR file with full global illumination, the exposure is manually adjusted, and the image is cropped.

4.4.1.4 Image Pre-processing

At this stage, we have a set of photographs and a set of computer generated images ready. Before they can be used in the experiment, both sets need to be pre-processed further.

First, for both photographs and generated images, we always attempt to isolate single buildings, even if they stand in a row of other buildings. This is done to present the participant with only one building in each image and to remove bias from the experiment. To this end, we crop the images to a square so that regions with people or other occluders can be avoided, and all images have an identical aspect ratio.

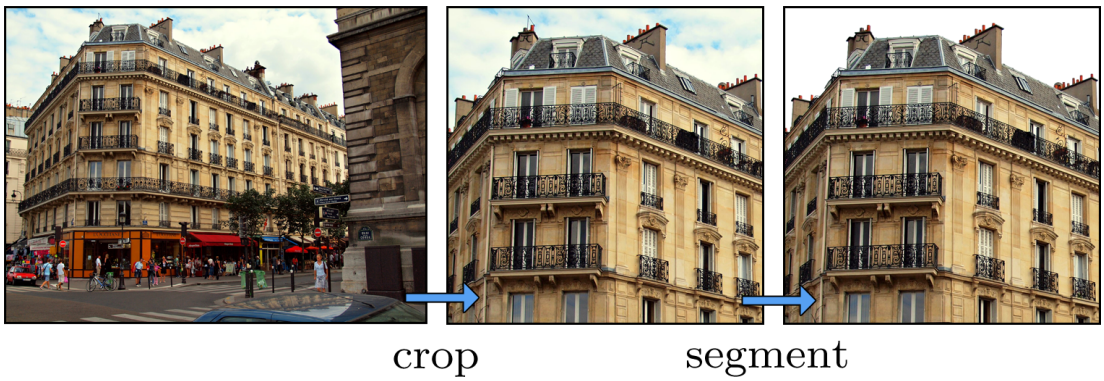


Figure 4.6: Photograph preparation workflow. A photograph is first cropped so that people and other biases are removed. It is then segmented to remove the background. The cropping process is nearly identical for the computer generated images.

Second, we remove the background from our *photographs* and replace it with a solid white color, see Figure 4.6. This is a crucial concept in our experiment, as it eliminates a large quantity of bias. For buildings, segmentation is a relatively easy task, as the majority of the edges are straight.

While putting photograph cut-outs against a white background will introduce some bias, alternatives, such as replacing all backgrounds with a procedurally generated scene or a collage of a photograph and a rendering, would introduce an even stronger bias. We think the white background approach is the least objectionable

one. Additionally, laymen are familiar with photo cut-outs as they are a popular desktop publishing technique, frequently used in magazines and textbooks.

Finally, we resize all images to a common size of 300×300 px. At this resolution, reasonable detail can be discerned on both photographs and generated images without making it prohibitively difficult to prepare high quality data.

Another source of bias that could possibly be dealt with is the color-tone of the images. However, with the exception of removing extremely discolored images, which we addressed in a previous step, we do not adjust color-tone over our datasets.

Finally, to prevent ourselves from unconsciously introducing further bias by choosing and/or crafting the best/worst images for our experiment, we created a larger amount of data than we needed: 21, 29, 49, and 33 candidate photographs and 24, 32, 34, and 33 candidate computer generated images for the Favela, Medieval, Paris, and Venice datasets respectively. From those, 6 computer generated images and 6 photographs were *randomly* chosen for each dataset.

4.4.2 Experiment

Algorithm 1: Experiment structure as pseudo-code.

```
for dataset in datasets do
    dataset = random.pick6CG6Real(dataset);
blurs = List(55 px, 37 px, 25 px, 13 px, 7 px, 1 px);
for each participant do
    showPersonalQuestionnaire();
    showTrainingScreen();
    datasets = random.permute(datasets);
    for blur in blurs do
        for dataset in datasets do
            images = applyBlur(blur, dataset);
            images = random.permute(images);
            showClassificationScreen(images);
    showDebriefingScreen();
```

We now describe the experiment itself. An overview of the procedure is given in Algorithm 1. Our experiment was conducted in person. However, the methodology could easily be used online.

The participants started by filling out a personal questionnaire and familiarizing themselves with the UI on a training screen. The individual classification screens, described below, followed. Finally, the participants were debriefed.

To reduce variance, we randomly chose 6 generated images and 6 photographs from each dataset once, and reused them for *all* participants. We started from the strongest blurs, hypothesizing that this minimizes the learning effect, as the participants cannot be sure about their previous – "more blurry" – choice. We interleaved the datasets to further minimize how well participants recognize and remember images and their associated classification choices. Finally, we randomized the position of images on the 6×2 grid and the order in which the datasets were interleaved to prevent spatial and temporal biases.

4.4.2.1 Classification Screen

On the classification screen, the participants choose an *assigned class* for each image: either *generated* or *photograph*. They are unaware of the image's *original class*. The classification screen, depicted in Figure 4.2, showed 12 images presented in a 6×2 matrix against a gray background. The participants were instructed to take as much time as they needed, and told that usually the test takes around 20 minutes, or under a minute per screen. The participants then used a radio button to pick an assigned class for each image. Once all images were classified, the participant would confirm the selection using an on-screen button. A slideshow walk-through of the whole experiment can be found in the Supplemental Material, see Section 4.9.

4.4.2.2 Participants

In total, 52 participants, mostly university students, on average 24.33 years old ($SD = 4.83, \min = 20, \max = 43$), of which 11 were female, took part in our experiment. All had normal or corrected to normal vision, and normal color vision. Reported experience with related fields is given in Table 4.2.

Experience	None	Casual Interest	Professional
Computer Graphics	27	23	2
Photography	22	30	0
Architecture	41	11	0

Table 4.2: Experience in related fields, as reported by the participants.

4.5 Data Analysis

Below, we analyze the quantitative (classification) and qualitative (questionnaire) data gathered in our experiment. Source code for our quantitative analysis, as well as the experiment data and additional analysis data, are provided in the Supplemental Material, see Section 4.9.

4.5.1 Quantitative Analysis

One of the main goals of our study is to investigate the role of various scales (levels) of detail on perceived generated building realism. Specifically, we postulate the two following hypotheses:

- H_{DIFF}: participants can tell photographs and generated buildings apart, and
- H_{SCALE}: the detail that allows participants to tell photographs and generated buildings apart is present at various scales.

We test these hypotheses by studying the classification accuracy of building images subjected to different levels of blur. Since our hypotheses only concern one-way interactions, we limit the post-hoc tests reported here to the main effects. The rest of the analysis is exploratory. We therefore only present descriptive statistics for the two and three-way interactions.

4.5.1.1 Hypothesis H_{DIFF}

To test the hypothesis H_{DIFF}, we use the Student's t-test to compare mean classification accuracy at the highest blur level, 55 px, and random choice (= 0.5). We define *accuracy* as the fraction of classifications in which the assigned class equals the original class. The difference for the overall accuracy at a 55 px blur is statistically significant ($t(2495) = 10.2, p < .001$). This demonstrates statistically significant differences in the images that allow the participants to distinguish between computer generated images and photographs, even at the highest blur level. The same holds for each individual dataset with the exception of Favela, for which the accuracy at a 55 px blur is not different from $\mu = 0.5$ in a statistically significant way.

A weaker but more relatable variant of the hypothesis is that there is a statistically significant difference between photographs and generated images at no blur ($t(2495) = 42.25, p < .001$). This means that users can differentiate between photographs and generated images without blur. For this variant, the statement holds for all individual datasets.

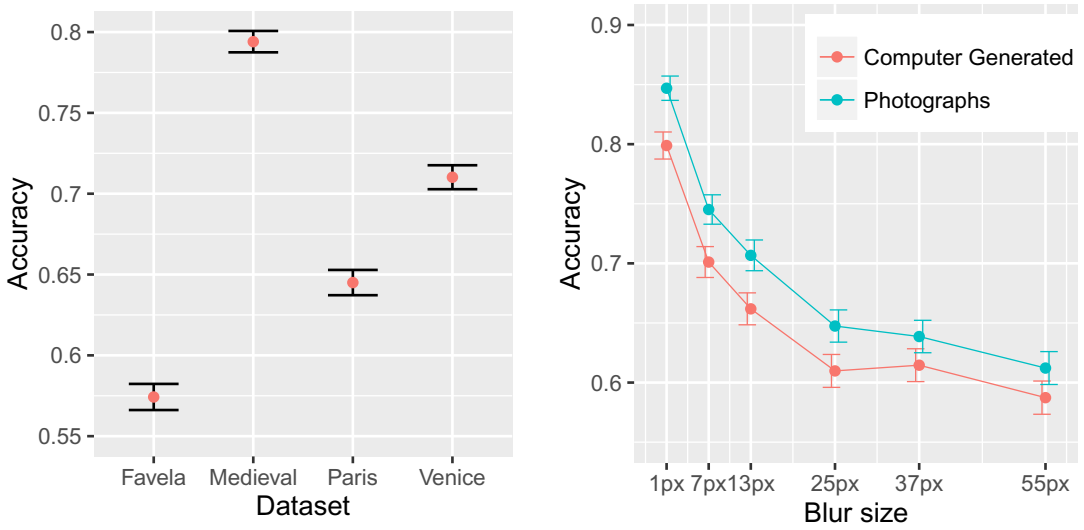
4.5.1.2 Hypothesis H_{SCALE}

To test the hypothesis H_{SCALE} and further explore the data, we use three-way, within-subject, ANOVA (Bakeman 2005) to model differences in classification accuracy, the dependent variable, with blur level, dataset, and an image's original class as factors, that is the independent variables. There are six blur levels (1 px, 7 px, 13 px, 25 px, 37 px, 55 px), four datasets (Favela, Medieval, Paris, Venice), and two original classes (computer generated, photograph).

Blur level ($F(5, 255) = 93.3$, $p < .001$) and dataset ($F(3, 153) = 82.6$, $p < .001$) are the main statistically significant effects impacting accuracy. In other words, classification accuracy is significantly different for some blur levels than for others. Similarly for datasets. The image's original class (*computer generated* or *photograph*) does not have a significant effect on accuracy ($F(1, 51) = 2.57$, $p = .11$). Because the effect of blur size is statistically significant, hypothesis H_{SCALE} can be accepted. The relationships between the significant effects and classification accuracy are shown in Figure 4.7.

Furthermore, post-hoc tests – Tukey's HSD for each factor individually – reveal significant differences between all pairs of blur levels ($p < .001$; for 7 px and 13 px, $p = .014$) with the exception of 25 px and 37 px blur levels ($p = 1.0$), 25 px and 55 px blur levels ($p = .15$) and 37 px and 55 px blur levels ($p = .21$). Therefore, we can conclude that there is not enough difference between these three blur levels pairs for them to be considered different. However, that does not imply that no further features would be removed by more extreme blurs or that there is no change between these blurs, only that the change is not statistically significant.

For the dataset effect, the post-hoc tests show that the difference in accuracy is only significant between the Medieval and the Favela datasets ($p = .009$). Other differences are not significant.



(a) (b)

Figure 4.7: (a) Average classification accuracy for each dataset. (b) Average classification for computer generated images and for photographs. The accuracy plots show the relationship between blur level and accuracy for both original classes. The similar shape and position of the plots illustrates the lack of a statistically significant effect of an image's original class on accuracy. For both plots, the error bars show the standard error.

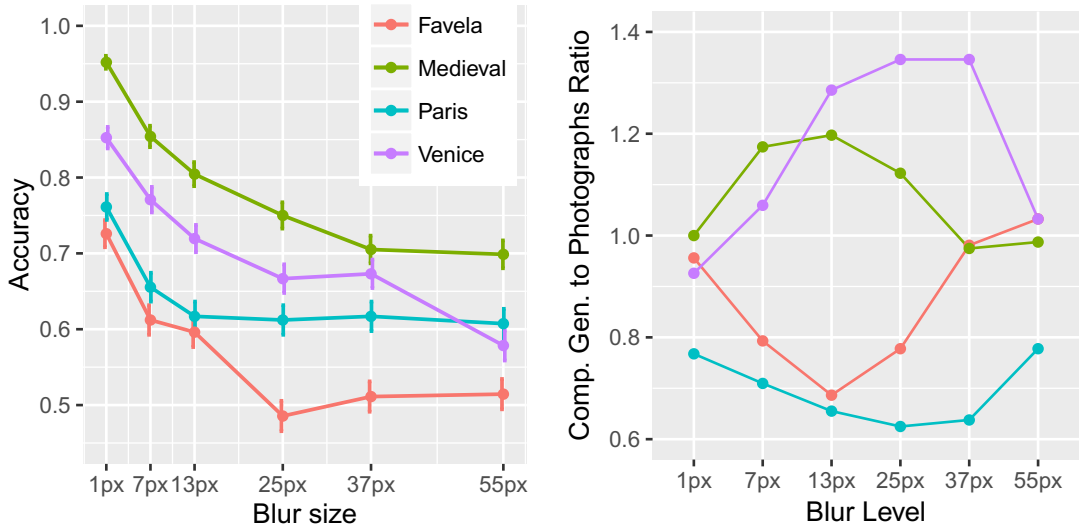
Dataset	All		Favela		Medieval	
Assigned Class ⇒	CG	Photo	CG	Photo	CG	Photo
Original Class: CG	79.89%	20.11%	71.47%	28.53%	95.19%	4.81%
Original Class: Photo	15.30%	84.70%	26.28%	73.72%	4.81%	95.19%

Dataset	Paris		Venice	
Assigned Class ⇒	CG	Photo	CG	Photo
Original Class: CG	69.55%	30.45%	83.33%	16.67%
Original Class: Photo	17.31%	82.69%	12.82%	87.18%

Table 4.3: Contingency tables for the 1 px blur. For all combinations of original image classes and datasets, we give the percentage of images that were assigned the classification computer generated (CG) and photograph (Photo). The equal values for the Medieval dataset are not an error.

4.5.1.3 Two- and Three-way Interactions

There are two significant two-way interactions. The first is between images' original class and dataset ($F(3, 153) = 21$, $p < .001$). This signifies that the difference in accuracy between computer generated images and photographs changes between datasets. This is illustrated in Table 4.3, which shows the percentages with which each original class is classified as either a photograph or as computer generated. While it is clear that the classification percentages differ, it is not clear whether this is caused by a different quality of photographs or a different quality of generated images. We speculate that the quality of photographs, influenced by factors such as lens type, composition, experiment pre-processing, etc., could be



(a) (b)

Figure 4.8: (a) The accuracy of each dataset across blurs. Notice the dip at 25 px for the Favela dataset and the decrease in accuracy between 37 px and 55 px for the Venice dataset. (b) For each blur level and dataset, the ratio of images classified as computer generated and images classified as photographs. If the ratio is greater than 1, more images were classified as computer generated than as photographs. If the ratio is smaller than 1, more images were classified as photographs than as computer generated (same legend as (a)).

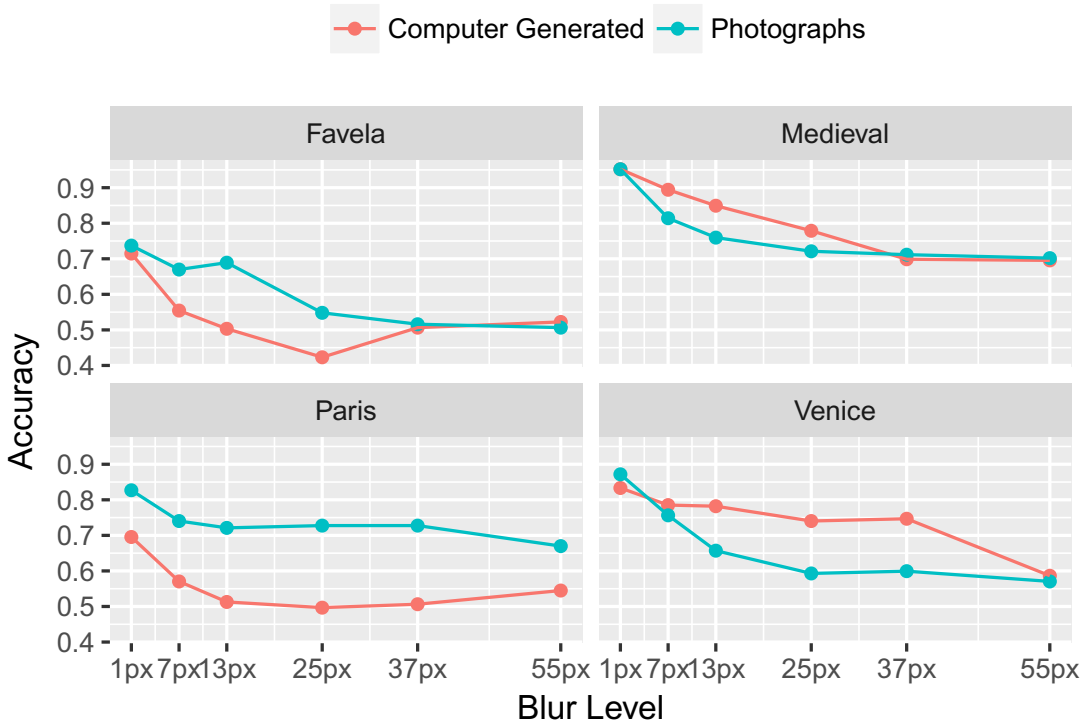


Figure 4.9: Three-way interaction between original class, dataset, and blur level. For each dataset, the way blur behaves with respect to the original class and the dataset differs.

assumed to be *approximately* equal or at least have less variance than the quality of the generated images. We therefore stipulate that it is mainly the quality of the generated images and therefore the quality of the individual datasets that changes and implies the change in accuracy.

The second significant two-way interaction is between blur level and dataset ($F(15, 765) = 3.96$, $p < .001$).

This implies that the blur behaves differently for each dataset. We speculate that this result is most likely an artifact. As visualized in Figure 4.8a, the differences in behavior manifest themselves in two places: in the Favela dataset at a 25 px blur, where it is likely the result of insufficient sample size, and at 55 px for the Venice dataset. For the latter, one plausible explanation is that the Venice dataset has less detail at that resolution than the other datasets.

The third two-way interaction, between the image's original class and the blur, is not significant ($F(5, 255) = 0.34$, $p = .89$). See also Figure 4.7b.

Finally, the three-way interaction between all of the factors is statistically significant ($F(15, 765) = 6.49$, $p < .001$). Figure 4.9 illustrates this, showing that blur behaves differently between computer generated images and photographs for each dataset.

4.5.1.4 Computer Generated to Photographs Ratio

For each dataset, Table 4.3 shows the contingency table for the image classes. As mentioned above, our experiment's design does not allow us to verify whether the differences in the percentages between datasets are solely due to the quality of the generated images, or whether the quality of photographs plays a role.

Figure 4.8b shows the ratio of the number of images classified as computer generated versus the number of images classified as photographs. Both the Medieval and Venice datasets have a number of blur levels for which the number of images classified as computer generated is higher than the number of those classified as photographs (ratio > 1). In other words, photographs are being classified as computer generated. For the Medieval dataset, this might be due to the relative obscurity and perhaps implausibility, from today's perspective, of such buildings. For the Venice dataset, the fact that many of the photographs were taken almost perpendicularly to the facade might have played a role. However, no conclusive evidence is available.

For the Paris dataset, the participants overestimated the realism of the images throughout the experiment. Interestingly, they thought the dataset contained more photographs in roughly the middle of the experiment (37 px–13 px) than at the beginning and at the end. A similar, more pronounced effect, was recorded for the Favela dataset.

Overall, datasets that we assessed as "more realistic" before the experiment, i.e., Paris and Favela, see Table 4.1, better convince participants of their realism – ratio < 1, meaning some of the generated images were classified as photographs – even when shown at higher blurs.

4.5.1.5 Summary

We accept the premise of hypothesis H_{DIFF}: that *overall*, participants perform better than random choice even at the highest blur level ($p < .001$). A three-way,

within-subject ANOVA was used to fit a model to our data. Because ANOVA shows that the blur level has a statistically significant effect on classification accuracy ($p < .001$), we can accept hypothesis H_{SCALE}. We therefore conclude that there are certain image features that are removed or diminished by each additional blur level that contribute to the perception of realism, up to and including the 25 px blur, as shown by the post-hoc tests. Higher blurs have not been shown by the post-hoc tests to have a statistically significant impact on accuracy.

Finally, we have conducted further exploratory analysis and discussed the other statistically significant effects in our experiment.

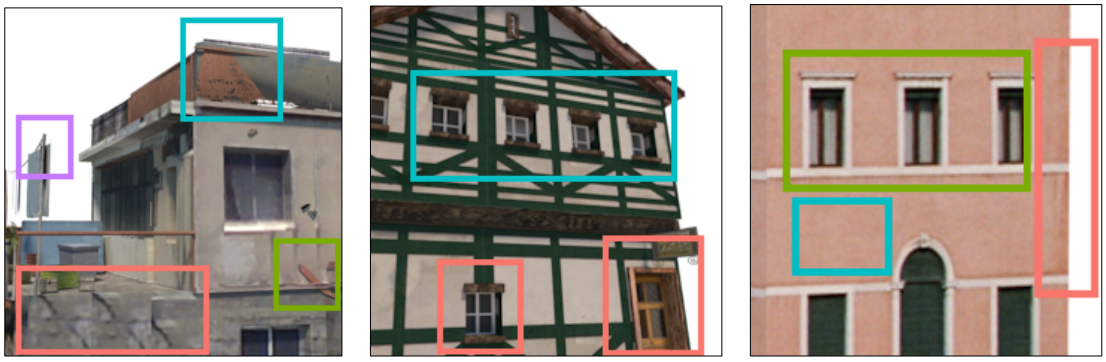
4.5.2 Qualitative Analysis

In this subsection, we detail the participants' debriefings, analyze the outlier images, and discuss individual datasets.

4.5.2.1 Debriefing

After the test, each participant was debriefed, see the Supplemental Material in Section 4.9. First, we enquired about the factors that influenced their decisions. We then asked if there were any tell-tale signs, or *indicators*, which influenced their classification of the images.

By manually tallying the number of times certain topics were discussed in the debriefings, we identified 9 indicators that influenced the 52 participants' decisions the most, namely: imperfections and/or small detail (30 participants $\approx$ 58%), texture (19 $\approx$ 37%), reflections in windows (18 $\approx$ 35%), "weird" or uniform color (17 $\approx$ 33%), things in, behind, or around windows (16 $\approx$ 31%), model structure (14 $\approx$ 27%), lighting (12 $\approx$ 23%), shadow (12 $\approx$ 23%), and regularity (11 $\approx$ 21%).



(a) (b) (c)

Figure 4.10: Frequently recognized computer generated images with artifacts; not to scale. (a) Texture repetition (red), incorrect geometry (purple, green, cyan); (accuracy $a = 0.88$), (b) model structure (red), regularity and window detail (cyan); ($a = 0.96$), (c) linear edges (red), lack of window reflections (green), uniform color and lack of detail (cyan), ($a = 0.98$).

Overall, participants looked for, and saw, imperfections and small detail, such as stains, dirt, wall cracks, etc., as indicators of realism. On the other hand, the lack of detail, for example large areas of uniform color, see Figure 4.13c, was considered unrealistic, see Figure 4.10.

Texture quality, especially inadequate resolution, inadequate detail, or incorrect repetitions, for example these shown in the bottom row of Figure 4.1, were also identified as indicators of a generated model. Some participants also mentioned such terms as "plasticity", referring to the overuse of bump-mapping, as illustrated in Figure 4.1e.

Reflections in windows were another important factor in perceived realism. The consistency of reflections, the curtain position, and the interiors were all indicators discussed by participants. However, analysis of the images suggests that when participants though they saw a real reflection in a photograph, they really just saw a texture on the window's model.

Scene lighting, color, and shadows were another often indicated factor. A subset of the 17 participants ($\approx 33\%$) that mentioned color remarked that a real building could not have such colors, one remarking that the colors were "kitchy". Some considered the lighting overall to be unrealistic, often mentioning shadow sharpness as an indicator of a computer generated image. See Rademacher et al. (2001) and Section 2.4.3 for further discussion of how shadow sharpness influences perception of realism, possibly misguiding participants. Some additional participants noticed apparent over-exposure and light bleeding around the edges of the photographs.



(a) (b) (c) (d)

Figure 4.11: Computer generated images with implausible structures. (a) Balcony and roof without any access points ($a = 0.87$); (b) Window placed in a support pillar, door too close to edge ($a = 0.96$); (c) Wooden balcony support inside stone/concrete support ($a = 1.0$) (d) Deformed roof structure and incorrectly applied texture ($a = 1.0$).

Finally, model structure and regularity had an effect. Participants were sensitive to repeated identical instances of windows, e.g., see top row of Figure 4.1, irregularities, and were sometimes uncertain whether such a building was structurally plausible, see Figures 4.10 and 4.11. However, we did not observe participants noticing repetitions between models, such as those in Figure 4.1, beyond the above mentioned instance.



Figure 4.12: Most confounding buildings at the highest, 55 px level blur. *Top row*: photographs that were classified as computer generated images the most. With fine detail missing, participants had to rely on coarser structure to make their judgments. Note how the structure seems implausible in (a), (b), and (c), the building improbable in (c), and the color too strong in (d). *Bottom row*: computer generated images that were classified as photographs the most. The plausible structure in (e), (f), and (g) contributes to realism. For (h), the variance in the window shutter positions and the fine detail in the top right corner are the most likely factors.

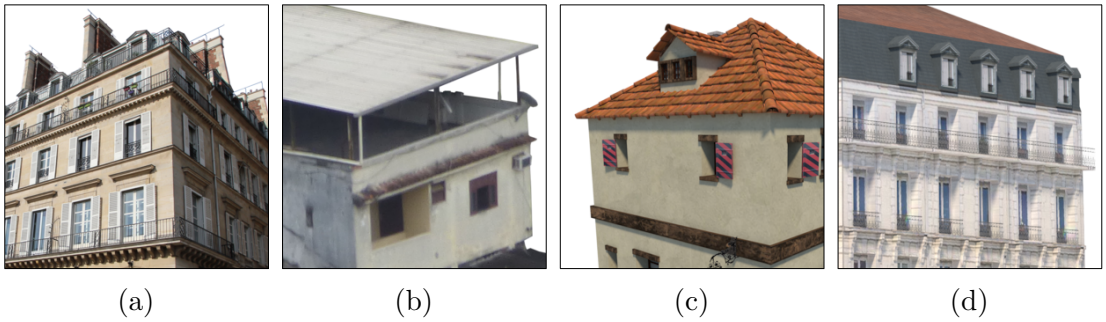


Figure 4.13: Unblurred (1 px blur) images. (a) and (b) Photographs that were often classified as computer generated ($a = 0.67$ for (a) and $a = 0.5$ for (b)). For (a), the high regularity of the unusual chimneys might be the cause. For (b), the implausible layout of the building is the probable cause. (c) and (d) Computer generated images that were very often classified as such ($a = 1.0$ and $a = 0.98$, respectively). For (c), the simple structure, uniform walls, and lack of detail on both edges and surface, and the very strong roof color seem to be the decisive factors. For (d), the large quantity of repetition, e.g., windows and roof windows, the unusual roof shape, and perhaps lighting are the most likely factors diminishing realism.

4.5.2.2 Cut-Out Edges, Camera Angles, Background

During debriefing, we asked the participants whether the edge between the building and the white background influenced their choices. Of the 52 participants, 21 ($\approx$ 40%) remarked that the cut-out edge played a role. Some mentioned that generated images had a sharper edge. This, however, is unlikely considering the generated images were significantly downscaled after segmentation. A small number mentioned that the complexity of the silhouette influenced their decisions. Notably, the sharpness and fine detail of edges are lost even at a low, 7 px blur level, meaning these observations relate mostly to images without blur.

During debriefing, 8 participants ($\approx$ 15%) said that the white background made them choose computer generated more often. This clearly biased our experiment. We believe this bias is hard to avoid, see discussion in Section 4.4.1.

In a pilot experiment, we observed that participants are not good at determining the realism of camera angles. We therefore instructed participants not to consider camera angles. After the experiment, we asked the participants whether the camera angles influenced their decision, with 18 participants ($\approx$ 35%) admitting they have been influenced for at least one image.

4.5.2.3 Other Factors.

The majority of the 52 participants found the higher blur levels quite challenging and some found it almost impossible. Many commented that the experiment was easy or not too difficult for the lower blur levels. Two participants ($\approx$ 4%) considered the experiment too long and tiring.

Five people commented that they were or might had been influenced by their previous image choices. We suspect this had a small or negligible overall effect, as accuracies of individual images changed steadily between blurs.

4.5.2.4 Images

Some of the factors that affect realism were already discussed above and illustrated in Figure 4.10. Here, we take a further look at some individual images and blur levels.

Overall, at high blur levels, see Figure 4.12, participants seemed to look for two factors: plausible building structure and irregularity. Sometimes, these indicators can be misleading, e.g., Figure 4.12e was thought to be very realistic at a 55 px blur ($a = 0.25$), but was readily recognized as computer generated ($a = 0.88$) when shown without blur, see Figure 4.10a. Other times, these factors can diminish realism, e.g. a photograph, shown in Figure 4.12a, with a relatively high amount of detail – that is fine structures, stained roof, and very irregular – only had an accuracy of $a = 0.23$ at the highest blur, and remained on par with random choice ($a = 0.5$) even when blur was removed, see Figure 4.13b.

For images without blur, fine detail, or lack thereof, becomes an additional factor. However, a high level of fine detail can be overridden by other factors, see Figure 4.13.

Overall, realism is a multi-faceted problem where a qualitative study might provide guidelines, but cannot be relied upon to easily predict performance.

4.5.2.5 Participants

The participants were mostly university students, some with casual interest in photography, computer graphics, and architecture, see Table 4.2. Therefore, our study best represents a non-expert audience similar to the one that is the final consumer of procedurally generated buildings. As corroborated by Gain et al. (2015, p. 9), there might be differences in how expert and non-expert participants perceive realism. It is therefore important to match study participants with the intended audience.

Refer to the Supplemental Material in Section 4.9 for additional statistics about the participants.

4.5.2.6 Summary

We have collected and discussed factors that participants subjectively considered most important for the perception of realism, and illustrated them with images from the experiment. Additionally, we have analyzed individual images with both subjectively and objectively interesting features, illustrating the complexity of the interplay between the above factors.

4.6 Results

Above, we have presented a study that investigates the perceived realism of procedural buildings. The study has a quantitative and a qualitative part.

In the quantitative part, we have shown that even at the highest level of blur, participants on the whole still perform above random choice – hypothesis H_{DIFF} – and that blur has a statistically significant effect on the resulting accuracy – hypothesis H_{SCALE}. This shows that realism in procedural models is carried not only by fine detail, but also by coarser detail and structure at various scales.

The qualitative part has indicated that quality of textures, colors, presence of reflections, presence of model irregularity, and structural plausibility, see Figures 4.10a and 4.10b, and lack of silhouette detail, see Figure 4.10c, were important factors. Of the above, irregularity, plausibility, and quality of texture mapping are especially relevant to procedurally generated buildings.

Contrary to our expectations, participants seemed mostly unable to spot repeated features shared between different buildings from the same dataset, such as those in Figure 4.1. We suspect this is due to the relatively low sampling power of our experiment, as only six generated images are shown. A different design would help to answer this question.

Lastly, we have analyzed the individual images. The findings are in agreement with the above and show that realism needs to be achieved on both fine and coarse scales.

Our experiment tried to answer a question in a difficult context, where any possible design would have a number of biases, such as the rendering method, cut-out quality, lighting, or selection of data. We attempted to minimize the number and effect of these biases. The biggest biases in our study were the effect of the cut-out background, lighting control, and a slight memory effect. Those

biases, however, are not strong enough to invalidate the results of our study, see Section 4.5.2 and Section 4.3 for an analysis.

Below, we discuss the applicability of our results and summarize them into recommendations.

4.6.1 Applicability

Among other information, our study of procedural buildings gives us insights into how well *procedural rules* deal with structure of buildings, feature repetitions across instances, etc. Such insights could not be revealed by studying manually reviewed or modeled buildings, where conceivably, the artist would remove many of the imperfections. A study of that kind would only provide data about the influences of different rendering factors or errors that were intentionally included in the model.

In that sense, we surmise our hypotheses H_{DIFF} and H_{SCALE} might provide us with results that differ from results one would get from a study of manually modeled buildings, as we do not avoid or choose our errors, but instead sample them from procedural rules to identify what they are.

Consequently, our results are most readily applied to procedural buildings. However, they are also at least partially applicable to non-procedural buildings. Both desirability of various details and structural plausibility can be considered sensible guidelines for any virtual buildings. Others results, especially those related to shadows, colors, and other rendering aspects, can be applied to non-procedural buildings without caveats.

4.6.2 Recommendations

On the whole, participants could differentiate between generated and photographic images – H_{DIFF}. The quantitative analysis shows that there is information that contributes to the participants' perception of realism even if the small and very small detail gets blurred out – H_{SCALE}. Authors should therefore focus not only on this detail, but also on what is left when the small and very small detail is taken away: the overall structure.

The debriefings suggest that both manually and procedurally modeled buildings should provide enough content to convince their users that the building interacts with the real world. Such content could include dirt, weathering or cracks, etc. Another, relatively cheap, way to boost realism is by adding reflections to windows, either from a texture, or from an environment map.

Additionally, procedurally generated buildings seem to suffer the most from poor structural plausibility and too much regularity. Users notice misaligned, misplaced, or repetitive geometry and texture, both for smaller regions as well as for the overall structure. A good procedural model should strive to provide plausible structure and placement of its components and textures at all scales.

4.7 Limitations and Future Work

With this study we want to stimulate further research into the realism of procedural modeling in general. We hope our results will motivate research into

procedural rules with improved plausibility and correctness. For example, the ability to generate meaningful interiors for a given exterior, creating procedural models that are plausible from a navigation and plumbing perspective, or a system that is able to detect and reject implausible, or poor quality outputs.

As for realism itself, we hope to see work that would study procedurally generated buildings and their perception in context of other buildings; As Ramanarayanan et al. (2008) have shown, there could be several further effects at play. Such an experiment could also help with better answering the question of how much participants notice and are influenced by common features and regularity in a set of procedural buildings.

Additionally, the recent advent of easily accessible image classification using neural networks (Szegedy et al. 2016) could open doors to novel groups of procedural algorithms that replace the human participants in our study with artificial participants. This would allow procedural model synthesis to be guided by a learned concept of realism. We also discuss this topic in Chapter 3.

Our technique could be adapted and applied to large scale urban models from any source, including those which were manually created, scanned, reconstructed, or found in large databases. The only general requirement is that a photographic ground-truth is available. It may also be possible to apply our technique to domains other than urban environments.

As in all user studies, the main limitation of our work is that our results are, in the strictest sense, valid only for our datasets and our setup. We realize other kinds of datasets, such as family houses, industrial buildings, commercial buildings, etc., might behave differently and that we have only explored a small fraction of the buildings even our datasets can generate. Other setups, where the buildings are seen in aggregates, in motion, or under different conditions could provide different insights.

4.8 Acknowledgements

The computer generated images were rendered using the Corona Renderer, kindly donated by Render Legion. The Favela dataset was graciously provided by Matthias Bühler of `vrbn.io`.

4.9 Supplemental Material

The supplemental material, referenced as "Supporting Information", can be obtained from the publisher of the original paper, see doi: 10.1111/cgf.13121. Alternatively, it can also be obtained from the project's homepage: `https://cgg.mff.cuni.cz/~benes/ORAPM/`.

Chapter 5

Procedural Generation of Urban Road Networks

In the previous Chapters, we have talked at length about procedural modeling. We have discussed its role in computer graphics in Section 3.1, talked about its realism in Chapter 3, and finally, presented a study of realism for procedurally generated buildings in Chapter 4.

In this Chapter, we conclude the topic of procedural modeling by presenting a procedural method for generating roads in and around cities and towns.

5.1 Introduction

> *"[Settlements] have grown rapidly, and in a great many cases they are running together; at the present rate our not very distant successors will see an almost continuous town, and of one monotonous type as far as man can make it, for a couple of hundred miles."*
>
> — Sir Patrick Geddes, 1915, *Cities in Evolution*, p. 47

Cities have been an important phenomenon in civilized society. In fact, it is when a society transforms from barbaric to civilized that cities and writing start to appear (Childe 1950, p. 1). The physical locations where both the first and the contemporary settlements tend to grow are largely driven by the availability of resources, processing facilities, trade (Johnson 1976, p. 82) and the induced transportation (Geddes 1915, p. 34; Johnson 1976, pp. 14, 39–45; Hoyt 1939, p. 98). As the settlements grow, they form "*vast city-regions*" (Geddes 1915, 18, also 34) called *conurbations*, forming an *urban landscape* of smaller and larger settlements, connected by roads and other means of transportation. See Figure 5.1 for an illustration.

Our method recreates the gist of the above outlined process on a computer. Its results are road networks in and around cities that develop over time. Due to our method's first-principles approach in which smaller settlements grow together to form larger settlements, its results have a more realistic shape and are able to form a more plausible and richer major and minor road structure than previous work.

Additionally, our method has minimal required *per-city* input and only has a small, well defined set of user parameters – a *historical context* – that can be

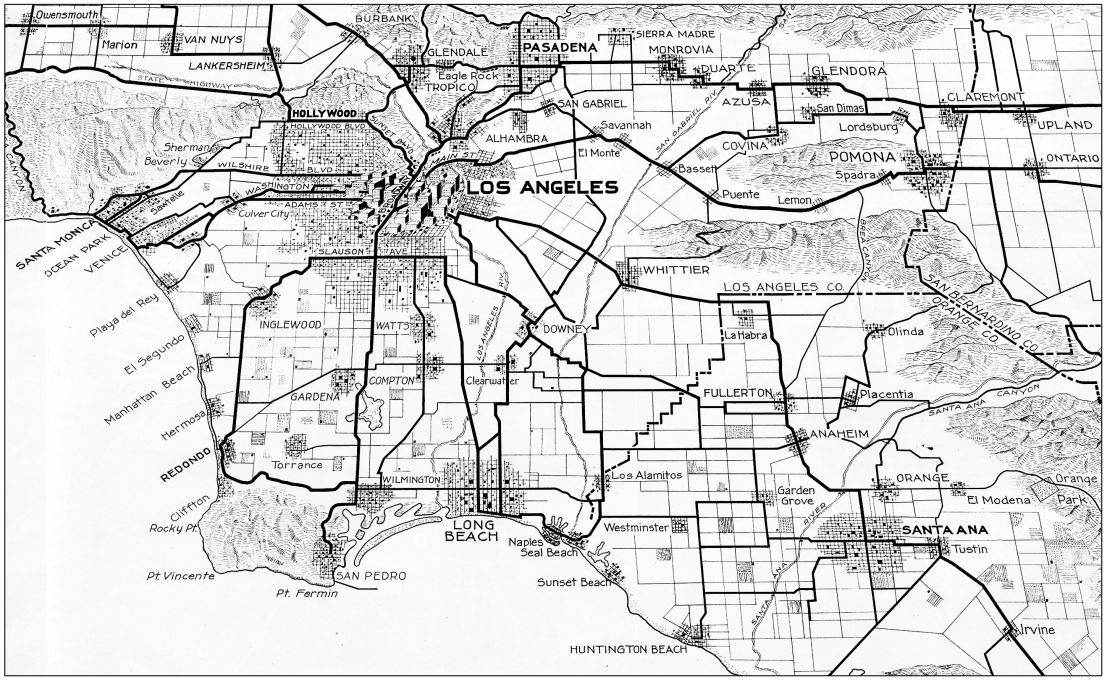


Figure 5.1: The Los Angeles area and Orange County on a motorist map from 1915, before the various constituent settlements grew together to form one continuous conurbation, the Greater Los Angeles of today. Image adapted from (ACoSC), and believed to be in Public Domain.

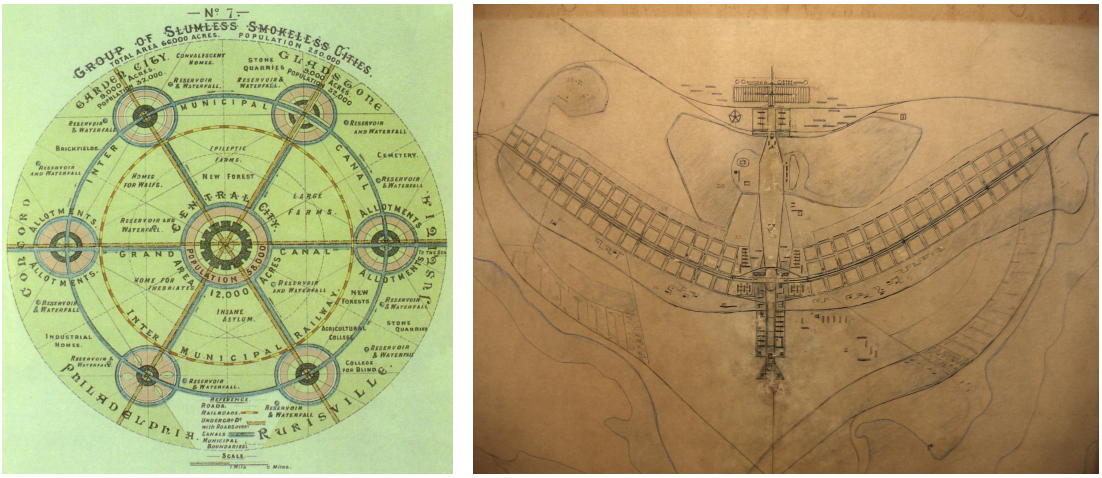
reused by different cities that share the same traits. In contrast, the current methods for cities require various maps (Parish and Müller 2001; Vanegas et al. 2009) or a growth center distribution and are more interaction oriented (Weber et al. 2009; Vanegas et al. 2009). Therefore, our method is better suited to applications that require the generation of city assets at run-time.

There are two factors that we consider to be outside of scope. First, we only generate the road network of a city, but no buildings or other geometry. Those can be generated by, e.g., Müller et al. (2006a) or Galin et al. (2010). We also do not attempt to replicate roads patterns with a strong individualistic design, such as those seen in Figure 5.2, as we consider them to be a non-simulatable artifact that could perhaps be replicated on a per-culture basis, but is unlikely to *emerge* from any simulation.

5.2 Previous Work

The 2001 paper by Parish and Müller (2001) is probably the first published effort to generate realistic cities for use in computer graphics. It is a hierarchical, L-system-like method to generate both major and minor roads that follow one of a set of predefined road pattern styles – grid, radial, least-elevation – over a given population density and road pattern map. In Weber et al. (2009), this approach was augmented with a custom traffic simulation and extended by a highly configurable land use simulation generating cities that evolve over time.

In another work, Vanegas et al. (2009) proposed an interactive city design method based on geometrical and behavioral modeling that also generates a road



(a) (b)

Figure 5.2: Examples of planned cities. (a) The proposed structure of a garden city (Howard). Image in Public Domain. (b) The plan of Brasilia, the capital of Brazil - inscribed in the outline of a bird with spread wings (Costa 1957). Image by (Rosenheck), CC-SA 3.0.

network and supports grid and radial patterns. An early attempt at agent-based modeling of cities is described in Lechner et al. (2003) and an interesting method where minor and major roads are traced in a sketched tensor field is given in Chen et al. (2008a). More recently, Yang et al. (2013) proposed a new hybrid streamline tracing and template matching method for generating high-quality, neighborhood-scale urban layouts by means of hierarchical domain splitting, aimed primarily at urban design.

Also, methods for example-based synthesis of road networks (Aliaga et al. 2008; Yu and Steed 2012), and for interpolation of given historical urban road networks (Krecklau et al. 2012) have been suggested.

To the best of our knowledge, Groenewegen et al. (2009) were the first to point out the differences between various land use models, using two of them in determining land use for Western European and North American cities. Relatively recently, Vanegas et al. (2012b) proposed a method for "inverse design" of cities, an intuitive high-level method of controlling existing urban procedural models. In the same year, Emilien et al. (2012) presented a method for generating various types of villages.

Of all these methods, only Weber et al. (2009), Krecklau et al. (2012) and perhaps Emilien et al. (2012) model the *evolution* of a city over time, while the other generate cities and their respective road networks at a *fixed* point in time.

Finally, Galin et al. published two works concerning generation of roads over terrain. The first one (Galin et al. 2010) proposes a method for generating single roads and their geometry over terrains using a discretization scheme, while the latter (Galin et al. 2011) extends it to allow for generation of plausible major road networks that connect (unsimulated) cities and other settlements. For completeness, a comparison of the main features of closely related methods is given in Table 5.8 of the Supplemental Materials.

Since this Chapter has been published in 2014, additional literature has appeared. Garcia-Dorado et al. (2014) presented an urban design tool that allows for

the analysis and generation of cities based on desired traffic parameters. Nishida et al. (2015) proposed a method that takes existing road networks and is able to generate, with possible user steering, new road networks that are statistically similar. Hartmann et al. (2017) proposed StreetGAN, a method based on deep-learning that also synthesizes road networks. Williams and Headleand (2017) use historical context, in a fashion similar to our method, to drive architectural and road styles within smaller settlements. The importance of assigning semantic tags to various parts of an urban model for further simulation is discussed and a model is proposed by Rogla et al. (2017). Finally, Kim et al. (2018) discuss the current state of procedural city generation in a wider context.

Our contribution to previous work is: 1) a simulation that accounts for neighboring cities, 2) having settlements grow together to form an urban landscape, 3) taking into account water transportation and its highly important implications on settlement positioning, 4) changing minor road patterns with time, and 5) minimal per-city input. A thorough discussion of our contributions, their novelty, and their implications can be found in Sections 5.4 and 5.5.

5.3 Our Method

In accordance with previous work, we call the results of our algorithm *cities*. We also understand the word city the way it is commonly used. Note, however, that the notion of a city is vague, as discussed by, e.g., Frey and Zimmer (2001), and we therefore sometimes use the word *settlement* to describe a populated place. When we want to emphasize that a city is the result of several settlements growing together, we might refer to it as a *conurbation*. We also use the phrase *urban landscape* to denote all the settlements that are the result of our method. We reiterate that an *urban landscape* consists of possibly many settlements of various sizes and the road networks connecting them. For convenience, a list of used symbols is provided in the Supplemental Materials, Section 5.6.2.

5.3.1 Overview

At a coarse level of detail, see Algorithm 2, our method consists of an initialization phase and a main loop. In the initialization phase, we use the positions of the neighboring cities, as mentioned in Section 5.1, to generate an initial road network on which our city will grow. In the main loop, we then let the city develop by building new minor roads, simulating traffic, and building new major roads, in that order, repeating for as long as desired.

Algorithm 2: High-level pseudo-code of our algorithm.

```
Initialization();                // see Section 5.3.2
while t < t_max do
    GrowMinorRoads();            // see Section 5.3.3
    TrafficSimulation();         // see Section 5.3.4.2
    GrowMajorRoads();            // see Section 5.3.4.2
    Increase(t);
end
```

More precisely, our algorithm takes as its input 1) a terrain elevation map $\boldsymbol{m}$, classified into land, water, and forbidden areas, 2) a list of neighboring cities $\boldsymbol{nc}$, which we do not simulate except for their contributions to the traffic simulation, 3) a set of control functions $\boldsymbol{t}(t), \boldsymbol{G}(t), \boldsymbol{R}(t)$ that determine the level of traffic, the rate of city growth, and major road construction parameters with respect to simulation time t, 4) a time-dependent sequence of minor road pattern definitions $\boldsymbol{K}(t)$, and 5) time-independent coefficients $\boldsymbol{O}_1, \ldots, \boldsymbol{O}_4$ which we explain later.

While the terrain $\boldsymbol{m}$, the traffic function $\boldsymbol{t}(t)$, and the list of neighboring cities $\boldsymbol{nc}$ are required for each new city (lower-case bold font), we provide a way of generating them automatically, see Section 5.3.5. The growth and major road construction parameters functions $\boldsymbol{G}(t)$ and $\boldsymbol{R}(t)$, coefficients $\boldsymbol{O}_1, \ldots, \boldsymbol{O}_4$, and the sequence of minor road patterns $\boldsymbol{K}(t)$ form a "*historical context*" (upper-case bold font) which several different cities can share. With the historical context and per-city input, we proceed as follows:

1. **Initialization.** One by one, the algorithm builds major roads that connect pairs of neighboring cities in $\boldsymbol{nc}$, resulting in an initial road network, see Figure 5.3b. We find all intersections in this initial road network and use them as the initial nuclei of city growth.

2. **Growing minor roads.** With the positions of nuclei of city growth in mind, we select positions out of which new minor roads can grow. We then grow these minor roads using the appropriate minor road pattern $\boldsymbol{K}(t)$.

3. **Traffic simulation.** For each pair of neighboring cities, we keep track of the total traffic between them. This traffic can flow across one or more different trade routes, which are paths running entirely over major roads. If the available traffic capacity between a pair of neighboring cities is greater or equal to the increase $\boldsymbol{t}(t) - \boldsymbol{t}(t-1)$ in traffic, no new major road is required and we only increase the traffic. Otherwise, a new path to accommodate the increase the traffic has to be found. If there is no such path, a new major road that we can route the traffic over has to be created, see the following point.

4. **Growing major roads.** If there is not enough available capacity to accommodate an increase in traffic, a new major road might have to be built. We therefore propose (a) a new major road that is subject to the major road construction parameters $\boldsymbol{R}(t)$. This road might consist entirely of new major roads, reuse existing minor and major roads, or both, and (b) as an alternative, we consider re-routing the surplus traffic over already existing major roads. Out of these two options, we choose the better one and add a new trade route that runs along it, thus accommodating for the increase in traffic. Finally, we increase the simulation time t and go back to step 2.

5.3.2 Initialization and the Initial Road Network

5.3.2.1 Rationale and Overview

According to Johnson (1976, p. 82), cities can be classified into four generic types based on the reason for their existence:

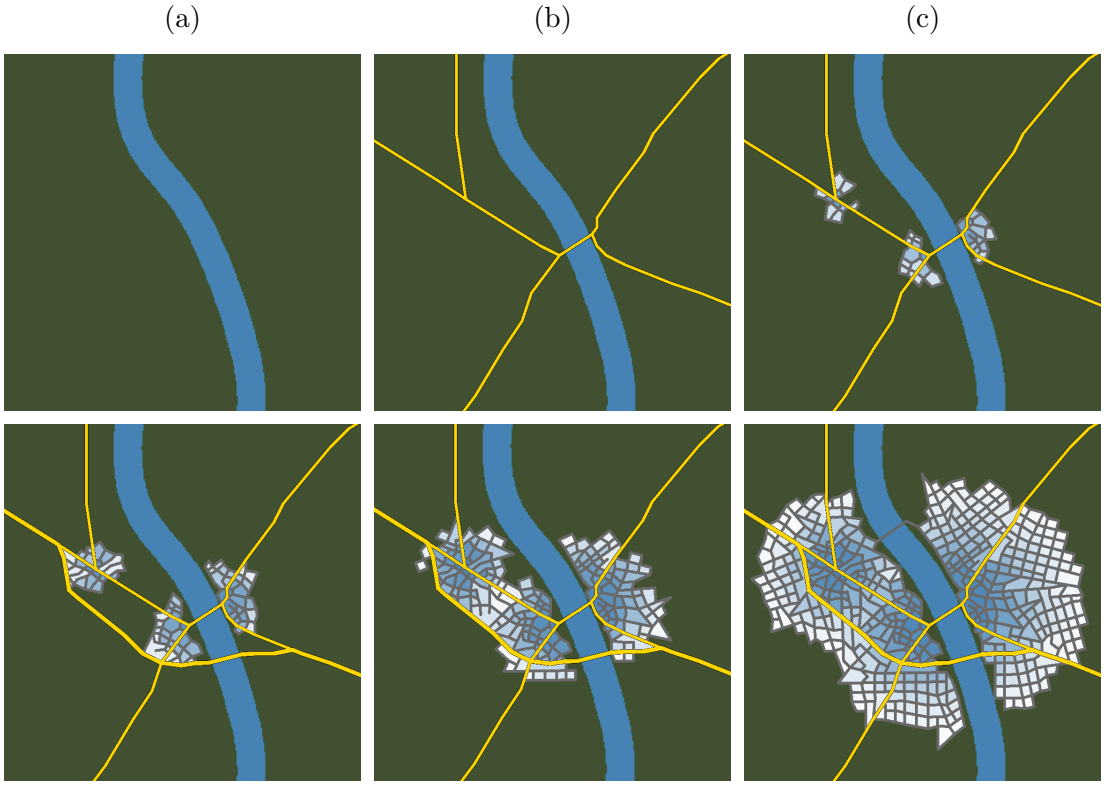


Figure 5.3: A typical progression of our algorithm. On an empty map (a), an initial road network (b) is grown and nuclei are identified. Initially, the city consists of several smaller settlements (c). As it grows and traffic increases, new major roads (yellow) are built (d) and a conurbation is formed (e). Eventually, even the new major roads are absorbed (f) by the growing city.

1. **Resource Cities.** Generate a resource, for example by mining, crop growing, or fishing. Their position is determined by the location of the resource and they are likely to be directly connected to several other cities, as they need to trade their products.

2. **Processing Cities.** Handle a resource, including people, by providing services such as raw resource processing, assembly of goods from parts, changing mode of transportation, splitting trade routes, or providing housing for travelers. Thus, they grow on intersections of pre-existing routes.

3. **Market Cities.** Provide local access to an (imported) resource. They are therefore cities with very high accessibility and can be thought of as being located at the intersection of roads connecting several neighboring cities.

4. **Other Cities.** Such as fortresses or artificial capitals. Here, the existence of a city is not implied by the road network's structure. However, even these cities eventually develop connections to other settlements or cease to exist.

As shown above, virtually all cities are to be thought of as being located on an intersection of trade routes and have a moderate or higher degree of connectivity. For these reasons, our simulation does not start on an empty map. It instead

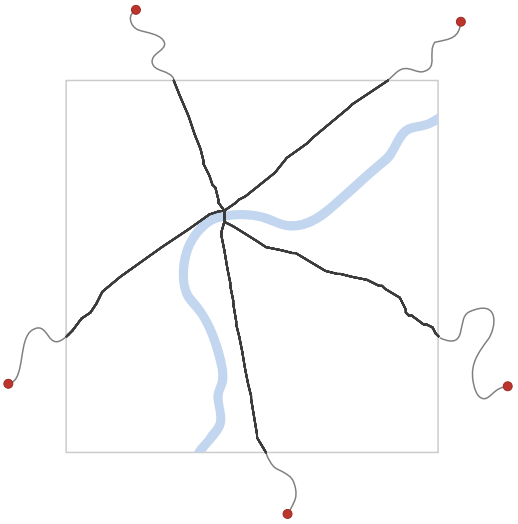


Figure 5.4: The map $\boldsymbol{m}$ (light gray rectangle) with a river (blue), initial road network (thick black), neighboring cities $\boldsymbol{nc}$ beyond the map's border (red); the gray roads are just schematic and a non-simulated continuation of the map's road network.

starts by constructing an initial road network. On this road network, roads connect the notional, unsimulated neighboring cities, see Figure 5.4, providing a road network structure on which to start our simulation, mimicking the situation at a very early point in the city's history, with the neighboring cities already in existence and the road network connecting them in place. This road network contains a small number of intersections that will be the nuclei of the city's future growth.

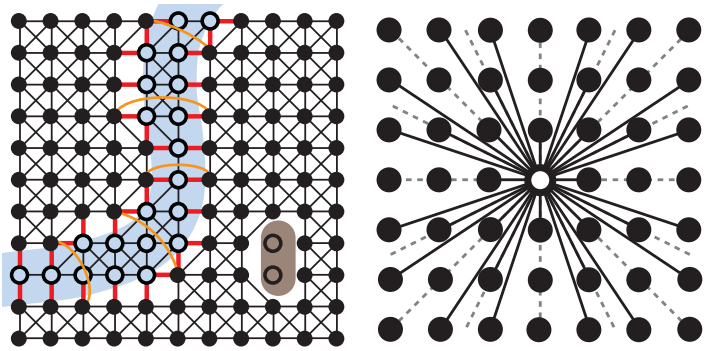
5.3.2.2 Technique

We now discuss how the initial road network is constructed. Galin et al. (2010) suggested the use of a pathfinding graph to generate roads over complex terrains. Below, we extend their algorithm for finding paths (roads) over land so as to *also* allow for paths *over water*.

Our pathfinding graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$ is defined over a regular lattice $\mathcal{L}$ spaced at $\ell = 20$ m, which we classify into three mutually exclusive sets $\mathcal{L}_L$, $\mathcal{L}_W$, and $\mathcal{L}_F$ for lattice vertices over land, water, and forbidden areas respectively. See Figure 5.5a for an illustration. First, we construct the land graph $\mathcal{G}_L = (\mathcal{L}_L, \mathcal{E}_L)$, where the edges connect vertices in $\mathcal{L}_L$ for which their distances along both axes of the lattice are less than or equal to a radius r on the lattice, see Figure 5.5b. In our simulation, we use $r = 8$.

For each (undirected) edge $e \in \mathcal{E}_L$, its weight $\textsc{Weight}(e)$ represents the difficulty of getting from one of its ends to the other. We calculate this weight by marching along the edge and keeping track of the presence of water, forbidden areas and the maximum edge slope $E_{L,\max}$. If an edge runs over water, forbidden areas, or has $E_{L,\max}$ greater than a cut-off constant $E_{L,\text{cut}}$, we discard it. For all non-discarded edges, their weight is computed as

$$\textsc{Weight}(e) = \textsc{Length}(e) \cdot \max(1, \boldsymbol{O}_1 \cdot E_{L,\max}),$$



(a) (b)

Figure 5.5: (a) Schematic depiction of the pathfinding graph $\mathcal{G}$, with land vertices $\mathcal{L}_L$ and edges $\mathcal{E}_L$ (black dots), water vertices $\mathcal{L}_W$ (black and blue) and edges $\mathcal{E}_W$ (between vertices $\in \mathcal{L}_W$), forbidden vertices (brown-black) in the forbidden area (brown), bridge edges $\mathcal{E}_B$ (orange arcs), and harbor edges $\mathcal{E}_H$ (red). (b) The way edges (black) connect a vertex (white) on the lattice with the nearest other vertex for each direction (dashed line), here shown for a radius of $r = 3$.

where $\boldsymbol{O}_1$ is a scalar constant. This heuristic works well in practice. We then construct the water graph $\mathcal{G}_W$ in a similar fashion, requiring the edges not to pass over land or forbidden areas. In our implementation, water elevation is constant and no edge slope cut-off constant is used.

As we want the land graph $(\mathcal{L}_L, \mathcal{E}_L)$ and the water graph $(\mathcal{L}_W, \mathcal{E}_W)$ to be connected, we "stitch" them together using harbor edges $\mathcal{E}_H$ and bridge edges $\mathcal{E}_B$. The harbor edges connect adjacent land and water vertices, allowing for a change of transportation mode, whereas the bridge edges connect not necessarily adjacent land vertices and have at least one point on their arc located over water. Details of bridge edge generation can be found in Galin et al. (2010, Sections 4.2 and 5.3). For vertices in forbidden areas, no edges are generated.

The complete pathfinding graph $\mathcal{G}$, as shown in Figure 5.5, is therefore given as the union of the land component $(\mathcal{L}_L, \mathcal{E}_L)$, the water component $(\mathcal{L}_W, \mathcal{E}_W)$, the harbor edges $\mathcal{E}_H$ and the bridge edges $\mathcal{E}_B$:

$$\mathcal{G} = (\mathcal{L}, \mathcal{E}) = (\mathcal{L}_L \cup \mathcal{L}_W \cup \mathcal{L}_F, \mathcal{E}_L \cup \mathcal{E}_W \cup \mathcal{E}_H \cup \mathcal{E}_B).$$

Next, we describe the construction of roads that connect neighboring cities. Each neighboring city is represented by a distinct vertex on the edge of either the water or the land lattice $(\mathcal{L}_L \cup \mathcal{L}_W)$. We are then given or generate, see Section 5.3.5 for an algorithm, an ordered list $\boldsymbol{y'}$ of neighboring city pairs p_i. This list $\boldsymbol{y'}$ is normally a carefully selected *subset* of all possible city pairs $\boldsymbol{y}$.

For each pair $p_i \in \boldsymbol{y'}$, we use Dijkstra's algorithm to find a minimum weight path $\mathcal{P}_i$ in the pathfinding graph $\mathcal{G}$ that connects the neighboring cities in the pair p_i. We then multiply the edge weights along this path $\mathcal{P}_i$ by a reuse coefficient $\boldsymbol{u}$ to *stimulate edge reuse* by subsequent paths. The constant $\boldsymbol{u}$ is in $(0, 1\rangle$ and can be randomly generated at the algorithm's start or optionally set by the user.

As is to be expected, each path $\mathcal{P}_i$ is a sequence of edges in the pathfinding graph $\mathcal{G}$. However, we additionally represent the built roads and waterways in

a separate planar graph $G = (J, S)$, the *segment graph*. In the segment graph, each (straight) edge $s \in S$ is called a *segment* and the points where two or more segments meet are called *nodes* $j \in J$.

As a next step, we convert each found path $\mathcal{P}_i$ of edges in the pathfinding graph $\mathcal{G}$ into a sequence $P_{i,1}$ of land, water, bridge, and harbor segments – a *trade route* – where i denotes that the trade route runs between the cities of the neighboring cities pair p_i and the index 1 means that it is the first of potentially many trade routes for that pair, see Section 5.3.4. During the conversion from $\mathcal{P}_i$ to $P_{i,1}$, we add new segments and nodes to the segment graph G where necessary and *reuse* existing segments and nodes where possible, keeping the graph planar.

Now, the initial road network is built. That is, all of the roads and waterways between the pairs p_i in the ordered list $\boldsymbol{y'}$ of neighboring city pairs are added and represented in the segment graph G. We also have representations of the trade routes between the neighboring cities $p_i \in \boldsymbol{y'}$. As a next step, we find trade routes $P_{i,1}$ in the segment graph G for all the remaining pairs of neighboring cities, that is, $p_i \in (\boldsymbol{y} \setminus \boldsymbol{y'})$. If that is not possible, $\boldsymbol{y'}$ needs to be extended by additional city pairs from $\boldsymbol{y}$. As a result, we now have a trade route $P_{i,1}$ for every pair of distinct cities in the list of neighboring cities $\boldsymbol{nc}$.

We continue the initialization step by finding all the nodes in the segment graph G with a degree > 2 – the intersections – and all the nodes from which bridges and harbors emanate. We then add them to a set of *city growth nuclei* Q. The role of nuclei in our algorithm is discussed in more detail in Section 5.3.3. Finally, we set the current simulation time $t = 0$.

5.3.3 Growing Minor Roads

In the previous Section, we have described the algorithm's initialization step, namely the construction of the pathfinding graph $\mathcal{G}$, the initial road network, and the set of city nuclei Q. We continue by describing how our algorithm grows minor roads, the first step of the simulation's main loop.

5.3.3.1 Rationale



(a) (b) (c)

Figure 5.6: Various minor road patterns found in the historical center (a) and in newer parts (b), (c) of Avignon, France. Images generated from data licensed under ODbL, by OpenStreetMap and contributors.

As can be seen from looking at the maps of various old – especially European – cities, see Figure 5.6, there is a strong dependency between historical period and

the exhibited minor road pattern, although other dependencies, such as use of land or majority culture, could be observed. We make the simplifying assumption that time is the only important quantity and, by means of the control function $\boldsymbol{K}(t)$, assign to each simulation time t a specific minor road pattern to be used when growing new minor roads.

5.3.3.2 Technique

The location of new minor roads is heavily influenced by the position of nuclei of city growth $q \in Q$. These nuclei represent places that attract city growth, such as intersections, workplaces, or public amenities, see Section 5.3.4.1. The amount of growth stimulus for each nucleus q is expressed as $\textsc{Strength}(q) \in \langle 0, 1 \rangle$.

The nodes from which new minor roads may be grown form a set Z. This set consists of nodes $j \in J$ of the segment graph that have not spawned streets yet. Additionally, to facilitate growth from the inside parts of long segments, we temporarily subdivide them before each round of minor road growth. Specifically, for each segment $s \in S$ whose $\textsc{Length}(s) \geq 2 \cdot \ell$, where ℓ is the spacing of vertices in the lattice $\mathcal{L}$ of the pathfinding graph $\mathcal{G}$, we add $\lfloor \textsc{Length}(s)/(1.25 \cdot \ell) \rfloor$ equidistant temporary nodes along each such segment.

To actually grow new minor streets, we order the nodes $z \in Z$ from the set of nodes from which new streets can grow into a list Z'. The order depends on the distance of the node z to the nearest nucleus q_N and this nucleus's strength:

$$\textsc{Rank}(z) = \textsc{EuclideanDistance}(q_N, z) \cdot (1 - \textsc{Strength}(q_N)) \,.$$

We then select the first $\boldsymbol{G}(t)$ nodes from the ordered list Z' as the nodes from which new minor roads will be grown.

In our implementation, we use the algorithm presented by Parish and Müller (2001) and extended by Weber et al. (2009) to generate minor road patterns and consequently, new individual minor roads. Other algorithms, such as those by Aliaga et al. (2008) or Yu and Steed (2012) might conceivably be used.

To prevent artifacts near map borders, we select the nucleus q_C corresponding to the initial road network intersection with the highest degree as the center of the city that we are growing, let its $\textsc{Strength}(q_C) = 1$, and attenuate the strength of other nuclei as they approach the map's borders. This attenuation is a heuristic and other approaches or variants could be used.

As mentioned in Section 5.3.2.2, the first nuclei are added at the intersections of the initial road network. Later, nuclei corresponding to intersections of the new *major* roads are added to the list of all nuclei Q. Additional nuclei are placed at positions randomly selected from the set Z of all nuclei that can grow new streets to account for chance.

5.3.4 Traffic Simulation and Growing Major Roads

5.3.4.1 Rationale

For automated major road growing, if any, current city growth simulations such as Weber et al. (2009) usually depend on local traffic simulations. These simulations in turn depend on land use simulations. However, current land use simulations, such as Lechner et al. (2006), as well as other popular city models – such as

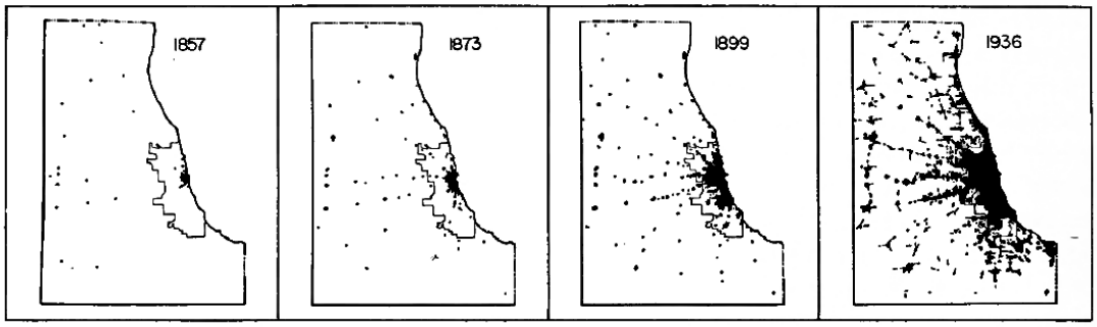


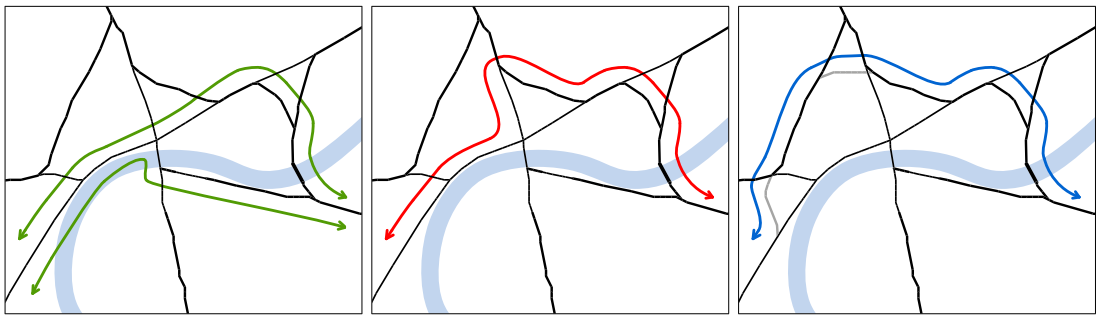
Figure 5.7: Time progression of settled areas within the Chicago metropolitan region between 1857 and 1936. A tentacle-like sprawl along traffic (railway) routes is clearly apparent. The dot-like pattern reflects the positions of train stations. Reproduced from Hoyt (1939, p. 98), in Public Domain.

the concentric, sector, and multiple nuclei models (Johnson 1976, Chapter 9) – are results of analyzes of then-current city growths. In these time contexts, these can be readily applied, as demonstrated by Groenewegen et al. (2009). The land use in the concentric model has, for example, been shown to be inverted in pre-industrial (Sjoberg 1960) and 20th century cities (Park and Burgess 1984, Charts 1 and 2). We therefore argue that models involving land use simulations are not easily applicable to simulations over longer periods of time or to different cultures (Knox and Pinch 2010, e.g., Section 1.3), motivating the development of our method. Further complexity is added by the changing understanding of cities (Frey and Zimmer 2001), with new urban structures such as megacities, edge cities, or outer cities appearing relatively recently (Garreau 1992; Soja 2000).

We attempt to overcome these problems by identifying trade and the resulting traffic as the most important, stable and simulatable factor for determining the shape and growth rate of a city across virtually all cultures. In addition to the discussion in Section 5.3.2.1, the importance of traffic and trading is also addressed by classic literature, e.g., by Childe (1950, pp. 15–16) or by Johnson (1976, Chapter 1). Next, the importance of transportation route configuration on city shape is treated by Hoyt (Hoyt 1939, Chapter 2), see Figure 5.7, who shows that city growth is related to accessibility and therefore, the presence of transportation routes. Similar observations in the context of highways have later been made for example in Tolley and Turton (1995) and Garreau (1992). Overall, accessibility attracts growth and increases near intersections. We therefore use a simple traffic simulation between the neighboring cities to simulate the need for new major roads, which we use to drive the shape of our cities by placing additional growth nuclei at the newly created major road intersections and nearby, see Section 5.3.3.2.

5.3.4.2 Technique

When building new major roads at simulation time t, we optimize their length and cost with respect to a set of major road construction parameters expressed by an umbrella function $\boldsymbol{R} \rightarrow \boldsymbol{P}, \boldsymbol{A}, (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_6)$. For parameters (t, α), this user-defined umbrella function defines:



(a) (b) (c)

Figure 5.8: Traffic simulation overview. (a) There can be more than one trade route (green) between two neighboring cities. When they cannot accommodate the increase in traffic between the cities, an alternative trade route (red) over existing major roads only (b) and a trade route (blue) that can build new major roads (gray) are proposed (c) and the better one is built. Only major roads (black) are shown.

1. a function $\boldsymbol{P}(t, \alpha)$, see Figure 5.9a, that relates angle α between consecutive segments to a cost penalty,

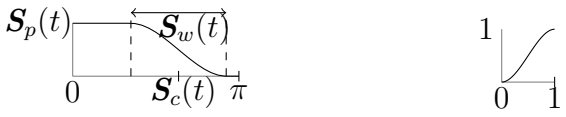
2. a tabulated function $\boldsymbol{A}(t)$ that defines the recommended traffic capacity for newly built major roads, and

3. six cost coefficients $\boldsymbol{C}_1(t), \ldots, \boldsymbol{C}_6(t)$, which will be detailed later and which influence edge costs when new major roads are being proposed.

Using $\boldsymbol{R}$, we can control the shape of new major roads, allowing historical roads to be more curvy and to reuse existing minor roads more easily, while forcing newer roads to be smoother and stay clear of built-up areas.

For the purposes of the traffic simulation, we keep a list of trade routes $P_i = (P_{i,1}, \ldots, P_{i,|P_i|})$ for each pair $p_i \in \boldsymbol{y}$ of neighboring cities. The first trade route $P_{i,1}$ for each pair has been described and found in Section 5.3.2.2. We denote the traffic over a trade route $P_{i,j}$ as $T(P_{i,j})$ and always initially set it to 0.

Since a trade route is a list of segments in the segment graph G, each segment can be shared by several trade routes, see Section 5.3.2.1 and Figure 5.8a. We therefore also have to keep track of traffic on a per-segment basis, with the traffic $T(s)$ over a segment $s \in S$ being the sum of traffic over all trade routes that make



(a) (b)

Figure 5.9: (a) The turn angle penalty function $\boldsymbol{P}(t, \alpha)$ at a fixed time t is a scaled version of the SmoothStep function, where $\boldsymbol{S}_c(t)$, $\boldsymbol{S}_w(t)$, and $\boldsymbol{S}_p(t)$ are the center, width, and height of the step. The angle α between two consecutive segments is on the x-axis, ranging from 0 (turn back) to π (no turn). (b) The $\text{SmoothStep}(x) = 3x^2 - 2x^3$ function (Cortes and Raghavachary 2007).

use of that segment. The *available* traffic capacity $T_A(s)$ over a segment $s \in S$ is given as

$$T_A(s) = \boldsymbol{O}_2 \cdot \textsc{Width}(s) - T(s),$$

where $\boldsymbol{O}_2$ is a scalar constant that relates segment width and the segment's capacity to carry traffic. Using this formula, we can define the available traffic capacity over a trade route $P_{i,j}$ as $T_A(P_{i,j}) = \min_{s \in P_{i,j}} T_A(s)$. Finally, due to the possibility of several trade routes sharing a segment and therefore also available capacity, saturating a trade route $P_{i,j}$ – that is making $T_A(P_{i,j}) = 0$ – might change available capacity T_A of other trade routes.

We compute the available capacity $T_A(P_i)$ for a list P_i of trade routes between a pair of neighboring cities using a simple iterative algorithm. One-by-one, we saturate the trade routes $P_{i,j} \in P_i$ and sum the traffic required to saturate all of them. After the available traffic is calculated, we restore the traffic values to their previous values.

Algorithm 3: Traffic simulation and major road growing.

```
foreach (p_i, P_i) do
    T(P_i) := 0;              // set the traffic for all trade routes ∈ P_i to 0
    if T_A(P_i) < t(t) ;                    // is the new traffic too high?
     then
        P_A := ProposeFromMajor();                          // see Sec. 5.3.4.3
        P_B := ProposeFromNew();                            // see Sec. 5.3.4.4
        P_W := Winner(P_A, P_B);                        // choose the winner
        P_i := P_i ∪ P_W;                                   // add the winner
        TradeRouteToSegmentGraph(P_W, G(S, J));
    end
    GreedilyRedistribute(t(t), P_i);
end
```

With the above definitions, we can proceed to describe the traffic simulation itself, see Algorithm 3 for pseudo-code. We use the historical context's function $\boldsymbol{t}(t)$ to determine the new amount of traffic flowing between a pair p_i of cities. We then reset the traffic for all trade routes in P_i to zero and check whether there is enough available capacity $T_A(P_i)$ along all trade routes in P_i to accommodate the new traffic $\boldsymbol{t}(t)$. If there is enough available traffic capacity, we greedily – with respect to trade route length – redistribute it among all trade routes in P_i and no new trade routes are created.

If, however, there is not enough available traffic capacity, we need to find a completely new trade route. We do this by evaluating two possibilities: 1) a new trade route P_A, see below, over existing major roads, or 2) a trade route P_B that uses existing minor roads or requires new major roads, see below.

We pick the better candidate by choosing the one with the smaller $\boldsymbol{O}_3 \cdot \textsc{Length}(P_k) + \boldsymbol{O}_4 \cdot \textsc{Cost}(P_k)$, $k \in \{A, B\}$ and add it to the list P_i of trade routes for the i-th neighboring city pair, adding and upgrading segments in the segment graph G as necessary. Finally, we again redistribute the traffic $\boldsymbol{t}(t)$ between all trade routes in P_i greedily.

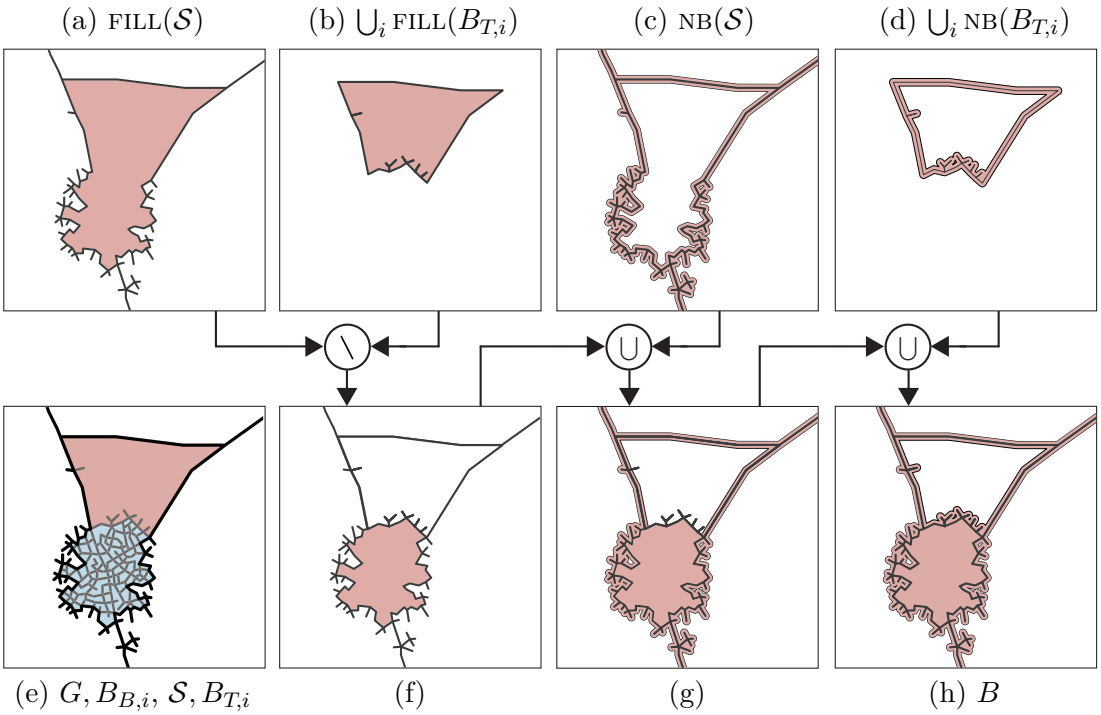


Figure 5.10: Illustration of Equation 5.1. First, we realize that the inside of the city shell $\mathcal{S}$ (a) contains all of the city's blocks, both built-up and terrain, as shown in (e). Since we are only interested in the built-up areas, we subtract the inside of the terrain blocks (b), yielding the built-up blocks (f). Next, we want to add the neighborhood of all roads. The neighborhood of most is already contained in the intermediate result (f). Those still partially missing are the ones on the city shell (c) and the ones on terrain blocks (d). Adding these first yields (g) and then (h), which is our definition of the built-up area B. In (e), city shell $\mathcal{S}$ (thick line), built-up blocks $B_{B,i}$ (blue), terrain block(s) $B_{T,i}$ (red), and segment network G (gray) are shown. In all other diagrams, the shell and the outlines are shown for clarity only.

5.3.4.3 Trade Routes over Existing Major Roads

Finding a new trade route P_A over existing roads is simple. We construct a temporary graph from all existing segments that represent major roads. To segments that cannot accommodate the increase in traffic, we assign an infinite weight. To other segments, we assign a weight as described in Section 5.3.2.2. In this temporary graph, we then find the minimum weight path.

5.3.4.4 Trade Routes over New Major Roads

Proposing a trade route P_B that requires an upgrade of existing minor roads or construction of new major roads is more complex. We start by defining the built-up area of a city and then use this definition to construct a new, auxiliary path-finding graph $\mathcal{G}'$.

The segment graph G cuts up the map into pieces – faces of the *planar* segment graph G – that we call blocks. These are classified into 1) *built-up blocks* $B_{B,i}$, and 2) *terrain blocks* $B_{T,i}$. The first kind of blocks represents areas that do

contain buildings, while the second represents larger undeveloped areas, such as fields or forests.

We distinguish between them using a simple block-area classifier, which in our experiments has performed on par with more computationally demanding, e.g., shape-aware, classifiers.

Next, we define the city shell $\mathcal{S}$ to be the "outer block", that is the outline of G, shown as a thick black line in Figure 5.10e. Using these, we give the built-up area B as

$$B = \left(\textsc{fill}(\mathcal{S}) \setminus \bigcup_i \textsc{fill}(B_{T,i}) \right) \cup \textsc{nb}(\mathcal{S}) \cup \left(\bigcup_i \textsc{nb}(B_{T,i}) \right). \tag{5.1}$$

The above Equation 5.1 is illustrated in detail in Figure 5.10. The FILL operator represents the inside of a block or of the shell and the NB operator represents the neighborhood of a block's or the shell's outline.

With a definition of the built-up area B in place, we proceed to construct, as illustrated in Figure 5.11, a pathfinding graph $\mathcal{G}'$ that will allow us to *find paths over both segments and terrain*. Since the built-up area B can only expand and can never shrink, we first take the pathfinding graph $\mathcal{G}$ and permanently remove from it all land edges $e \in \mathcal{E}_L$ that intersect the built-up area B. Next, we create a temporary copy $\mathcal{G}'$ of the pathfinding graph $\mathcal{G}$ and add to it all segments and nodes from the segment graph G. We continue by "stitching" these two components together by taking nodes along the city shell $\mathcal{S}$ together with all nodes along all terrain blocks $B_{T,i}$. We then connect them to nearby vertices in the set $\mathcal{L}_L$ of land vertices, provided that the new edge does not intersect or get too close to existing segments in G. Additionally, "stitches" connecting points in the shell $\mathcal{S}$ alone are generated to allow for smoother roads in some cases, not shown in Figure 5.10. Using these "stitches", we connect the disconnected components – copies of the pathfinding graph $\mathcal{G}$ and the segment graph G – in $\mathcal{G}'$.

In order to control the characteristics of new major roads, such as the tendency to reuse existing major and minor roads, we multiply weights of edges in $\mathcal{G}'$ by coefficients $\boldsymbol{C}_1(t), \ldots, \boldsymbol{C}_6(t)$. In addition to edges representing major roads ($\boldsymbol{C}_1(t)$), minor roads ($\boldsymbol{C}_2(t)$), unbuilt harbors ($\boldsymbol{C}_3(t)$), built harbors ($\boldsymbol{C}_4(t)$), and (unbuilt) bridges ($\boldsymbol{C}_5(t)$), we also define an additional edge type – *extendible edges* ($\boldsymbol{C}_6(t)$) – which are not surrounded by built up blocks and can therefore be easily widened to allow for more traffic. This way, one can for example ask that a new major road reuse existing major roads heavily by setting $\boldsymbol{C}_1(t)$ to a small value or that minor roads be avoided if possible by setting $\boldsymbol{C}_2(t)$ to a high value.

Finally, to propose a new trade route, we use the resulting pathfinding graph $\mathcal{G}'$ as input to a modified Dijkstra's algorithm (Volker 2008) which penalizes turns in paths using $\boldsymbol{P}(t, \alpha)$. After the new major road is constructed, its extendible pieces are widened to the recommended width $\boldsymbol{A}(t)$ of a new major road.

5.3.5 User Input and Automation

So far, the ways the user can intervene in our simulation and its input have been left largely unmentioned. As has been described in Section 5.3.1, our algorithm's input can be divided into two sets: a historical context that is common to a larger group of cities, and per-city input that needs to be generated to keep our

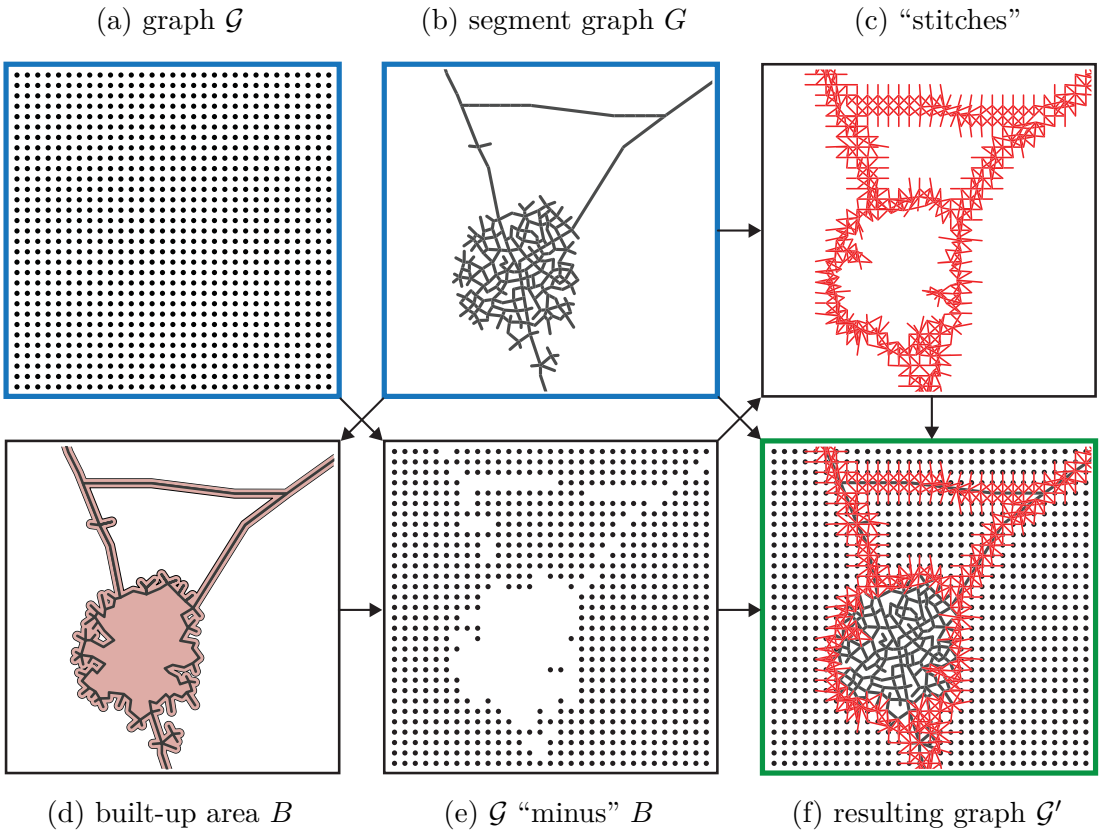


Figure 5.11: We start by generating the built-up area B (d) from the segment graph G (b), as described in Section 5.3.4.4 and Figure 5.10. We take the pathfinding graph $\mathcal{G}$ (a) – for clarity, only the vertices of $\mathcal{G}$ are shown – and update it by removing all edges and vertices that intersect the built-up area B, resulting in (e). Then we generate the edges – "stitches" (c); shell to shell "stitches" not shown – that connect the updated pathfinding graph $\mathcal{G}$ (e) and the segment graph G (b). Finally, we combine the "stitches" with the segment graph G (b) and the updated pathfinding graph $\mathcal{G}$ (e) to yield the resulting graph $\mathcal{G}'$. The input is shown in a blue border, the result in a green one.

method automatic within the scope of a historical context. In contrast to previous methods that concentrated mostly on user-aided generation (Vanegas et al. 2009) or required per-city datasets (Weber et al. 2009; Parish and Müller 2001), this allows us to generate batches of cities with similar traits.

Below, we detail automatic generation of per-city data, that is 1) the traffic function $\boldsymbol{t}(t)$, 2) the neighboring city list $\boldsymbol{nc}$, and 3) the neighboring city set $\boldsymbol{y}$ and its sublist $\boldsymbol{y'}$. As all of these are input for the algorithm itself, they need to be generated before the algorithm begins. Lastly, we discuss how the user can manipulate the segment graph G and the list of nuclei Q.

5.3.5.1 Traffic Function

There are many sensible choices for generating the traffic function $\boldsymbol{t}(t)$. Since growth and trade are intertwined, as discussed in Section 5.3.4.1, we choose the function to be a slightly randomized copy of the growth per simulation step

function $\boldsymbol{G}(t)$, which is a part of the historical context. Specifically, $\boldsymbol{t}(t) = \boldsymbol{b} \cdot \boldsymbol{G}(t)$, where $\boldsymbol{b}$ is uniformly selected from $\langle 0.7, 1.3 \rangle$.

5.3.5.2 Neighboring Cities

We generate the set of neighboring cities $\boldsymbol{nc}$ using a heuristic. First, we generate the number $\boldsymbol{c} = |\boldsymbol{nc}|$ of neighboring cities. We restrict the minimal and maximal number of cities to $\boldsymbol{c}_{min} = 4$ and $\boldsymbol{c}_{max} = 10$, respectively.

We choose the number $\boldsymbol{c}$ of neighboring cities from a binomial distribution with $n = \boldsymbol{c}_{max} - \boldsymbol{c}_{min} = 3$ trials, $p = 0.5$, and shift the result by $\boldsymbol{c}_{min}$ so that we yield values between $\boldsymbol{c}_{min} = 4$ and $\boldsymbol{c}_{max} = 10$ with a mean of $\mu = \frac{\boldsymbol{c}_{min} + \boldsymbol{c}_{max}}{2} = 7$.

Next, we generate the vertices that represent the neighboring cities. All of these vertices lie on the edges of the lattice $\mathcal{L}$. We do this by generating $\boldsymbol{c}$ equally spaced rays emanating from the center of the map X at angles $i\frac{2\pi}{\boldsymbol{c}} + \phi + \theta_i$, where $\phi \in \langle 0, 2\pi)$ is the uniformly distributed global angular offset and $\theta_i \in \langle -\frac{\pi}{2\boldsymbol{c}}, \frac{\pi}{2\boldsymbol{c}} \rangle$ are per-ray perturbations. For each ray, we then find its intersection with the edge of the map. The nearest vertex on the edge of the lattice to that intersection is the vertex representing the respective neighboring city. If two different neighboring cities are represented by the same vertex, we rerun the above process.

5.3.5.3 List of Pairs of Neighboring Cities

In this Section, we generate the list of initial neighboring city pairs $\boldsymbol{y'}$. Using a fixed list of neighboring cities $\boldsymbol{nc}$, we sort all unique pairs $\boldsymbol{y} = \{(C_u, C_v) \mid u < v \text{ and } C_u, C_v \in \boldsymbol{nc}\}$ of neighboring cities by the angle $C_u X C_v$, where X is the center of the map, removing pairs with the smallest angle until the removal of a further pair would result in a disconnected initial road network. We then permute the pairs that we have not removed, yielding a list of neighboring city pairs $\boldsymbol{y'}$.

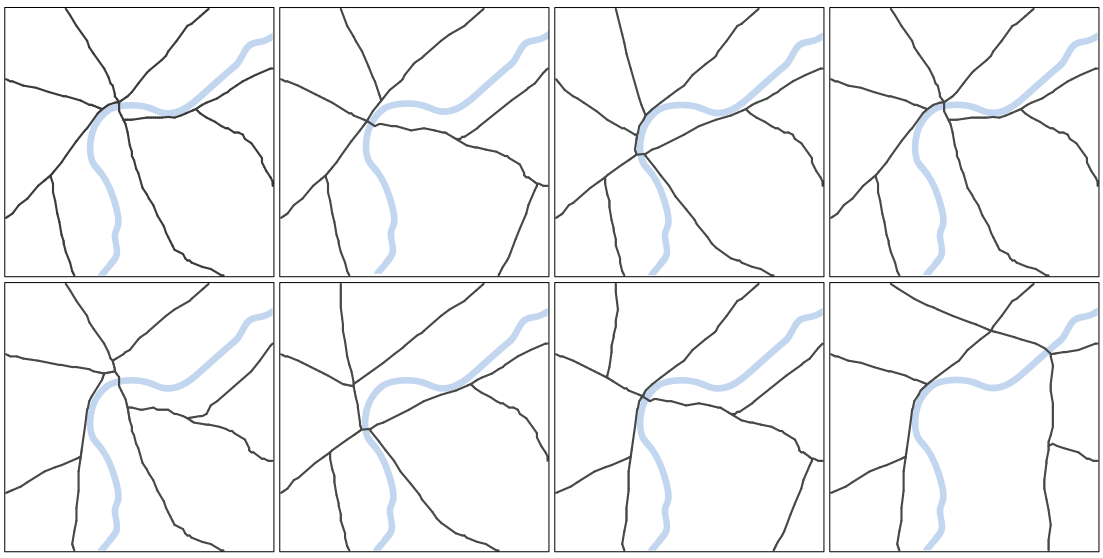


Figure 5.12: Initial road networks corresponding to various automatically generated sublists $\boldsymbol{y'}$ of the set of all neighboring city pairs $\boldsymbol{y}$ for a fixed list of initial neighborhood city pairs $\boldsymbol{nc}$.

As a result of this permutation, we get a variety of initial road networks, as shown in Figure 5.12. Further variation is achieved by adding or substituting the removed pairs into the list $\boldsymbol{y'}$.

5.3.5.4 Historical Context

Before automatic generation can start, the user is required to provide a set of values representing a historical context. In our experience, the system is stable and minor changes in parameter values cause only minor changes to the result's *generic appearance*.

The result itself, however, might change significantly due to, e.g., a major road being built in a different place and influencing the position of growth centers Q. Most of the values constituting the historical context have a well defined meaning. For example, the built harbor edge coefficient $\boldsymbol{C}_4(t)$ expresses how expensive changing the mode of transportation between land and sea would be. For some, such as the turn angle penalty function $\boldsymbol{P}(t, \alpha)$, a reasonable amount of experimentation is required.

5.3.5.5 Direct Manipulation

We optionally let the user intervene in the algorithm by letting them add or remove nuclei of city growth during the simulation and also by allowing them to change the nuclei's strengths. This, combined with the possibility of manipulating the growth and traffic functions $\boldsymbol{G}(t)$, $\boldsymbol{t}(t)$, and the segment graph G, allows the user to modulate the locality and strength of city growth, see Figure 5.13. The user is also allowed to paint new roads directly.



Figure 5.13: Demonstration of the intervention capabilities of our method. Nuclei (colored dots) and their strengths were manipulated to influence the locations of growth and the city's shape, and minor roads were added.

5.3.5.6 Summary

We described methods for fully automatic generation of all per-city input, with the exception of terrain $\boldsymbol{m}$, which can be generated by published methods, e.g., Musgrave et al. (1989). The definition of minor road patterns $\boldsymbol{K}(t)$ depends on the method used. In our case, the method of Parish and Müller (2001) and Weber et al. (2009), and therefore a L-system equivalent, is used.

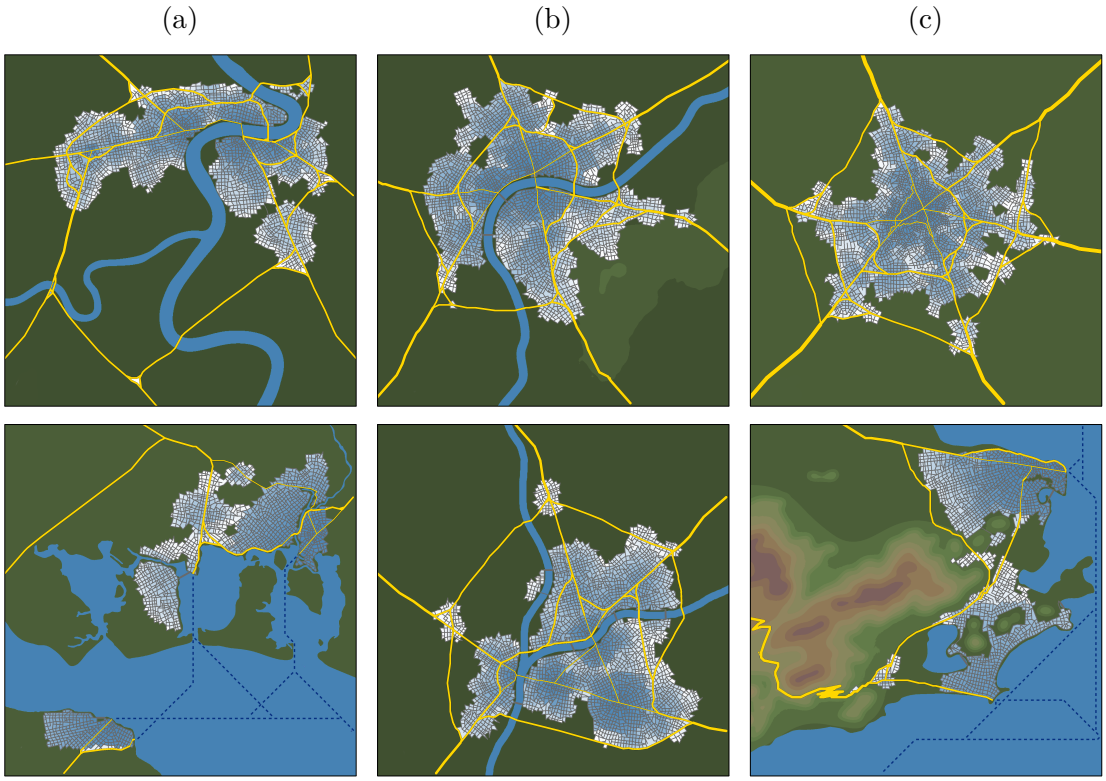


Figure 5.14: (a) An eccentric city, (b) a normal city, (c) a city with a more classic major road structure, (d) a coastal city spanning several pieces of land, (e) a city with two isolated satellite suburbs, and (f) a city in a difficult, mountainous terrain. Each quantization level represents a 60m vertical difference. Block color indicates age. Major and minor roads are yellow and gray, respectively. For a discussion of the results, see Section 5.4. Note that minor roads that do not enclose a block are not shown in the visualization.

5.4 Results and Discussion

The algorithm was implemented in C# and uses CUDA to compute the *initial* pathfinding graph $\mathcal{G}$. The generation times for 8×8 km maps for a lattice $\mathcal{L}$ with a spacing of $\ell = 20$ m were between 2.5 and 4.5 minutes running on a single core of an Intel i7-2600K 3.4GHz CPU, with the construction of the pathfinding graph $\mathcal{G}$ run on a GeForce 480 GTX card. Average times for individual results are given in Table 5.1.

Average running time over 5 runs, in seconds.						
Figure						
5.14a	5.14b	5.14c	5.14d	5.14e	5.14f	5.15
272.01 s	150.53 s	166.58 s	157.256 s	209.505 s	149.753 s	237.68 s

Table 5.1

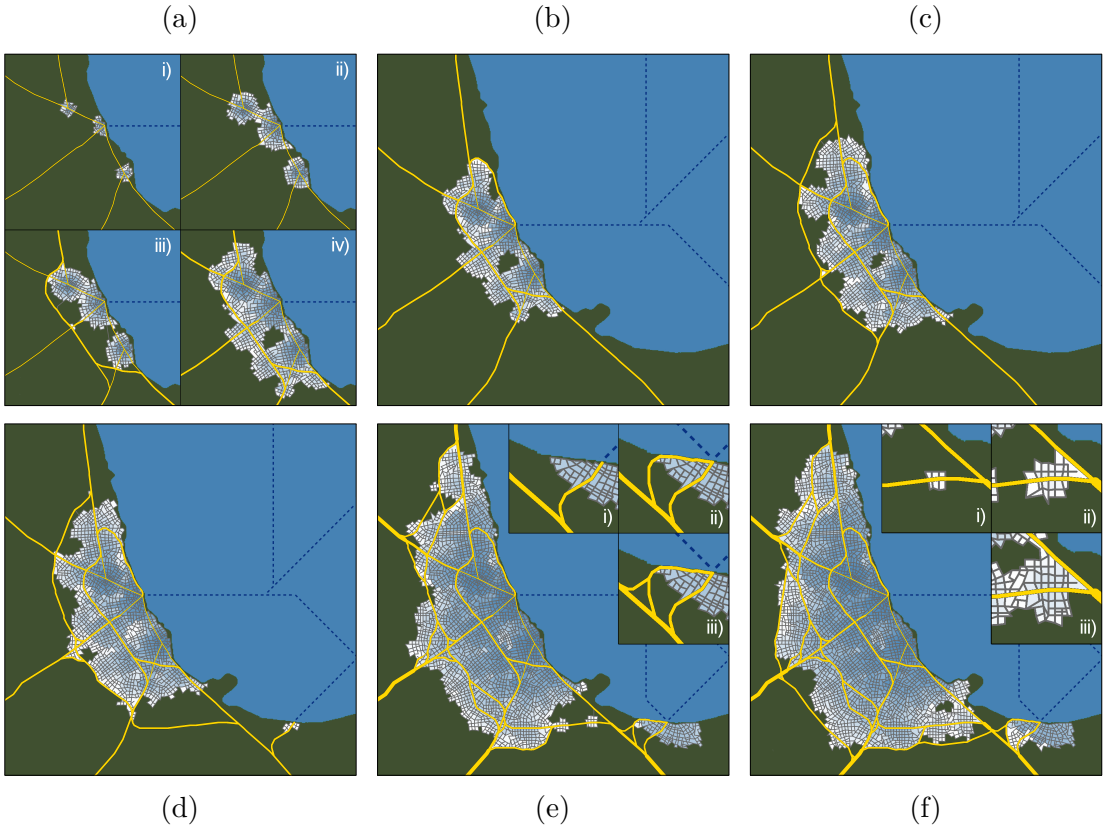


Figure 5.15: Evolution of a lake-side city with a harbor. Water trade-routes are dashed and in blue. See Section 5.4 for a discussion.

The parameter values for all presented results, together with images of all simulation steps, are available in the Supplemental Materials in Section 5.6. Below, we discuss the results in the context of our contributions:

- *Positioning of settlements and water transportation.* Unlike previous work, we put a strong emphasis on the placement of both initial and later settlements on the provided map. We place those settlements – "nuclei of city growth" – at the intersections of roads and back this approach by available literature, see Section 5.3.2.1. The results of this strategy can be seen in the positioning of the initial settlements in Figure 5.15a,(i) and in the isolated settlements generated later, as seen in Figures 5.14a, 5.14c, and 5.14e. The positioning of nuclei of city growth is an important aspect of city development that is strongly influenced by trade *between* cities, something that has not been addressed by previous work. We further extend this approach by being the first to consider a *new* mode of transportation: *water*. By treating places where waterways turn into roads as intersections (Johnson 1976, p. 82), we allow the simulation to automatically create and position

ports, that is cities such as New York, Hong Kong, or Hamburg. This is shown in Figure 5.15a,(i) by generating a settlement on the coast and in Figure 5.15d, where a further port is generated to ease the pressure on the first port. Additional ports are also generated in Figures 5.14d and 5.14e.

- *Neighboring cities and traffic simulation.* Our approach is the first to consider the influence of trade between neighboring cities. Here, our contribution is not in using a traffic simulation, but in using it to simulate *trade between cities* and identifying its impact on major road and city shape. See Section 5.3.4.1 for a thorough discussion. While previous work either requires the user to sketch major roads (Vanegas et al. 2009) or determines major road shape using *predefined* patterns (Weber et al. 2009; Parish and Müller 2001), our trade simulation grows major roads *automatically*, without patterns, and *in a meaningful way*. Depending on historical era, the major roads either need to navigate, see Figure 5.14b, or circum-navigate, see Figures 5.14 and 5.15, the built-up areas over large portions of terrain. This influences major road shape. In turn, with each new major road or waterway, new intersections, and with them, nuclei of city growth, are generated, influencing the city shape This can be seen on the kidney like shape in Figure 5.14a, the rugged city edges in Figure 5.14c, and the city's position close to water in Figures 5.15, 5.14d, and 5.14f. This results in a relatively non-deterministic, but plausible major road structure with ring-roads, see Figures 5.14b, 5.15d, 5.15e, and bypasses, see Figure 5.14a. In the inset of Figure 5.15e, the development of major roads close to the second harbor is shown. The insets in Figure 5.15f show the progression of a suburb, itself originally a few isolated blocks (*i*), to a compact settlement (*ii*), to when it is about to grow together with the city itself (*iii*).

- *Settlements growing together and urban landscape.* Due to our way of positioning settlements, see above, we grow cities over time in the same way *real* cities are formed, see Figure 5.15, that is by having several settlements (cities) grow together, see Section 5.3.4. This is an original contribution, not found in previous work. The spectrum of urban forms includes smaller and larger isolated suburbs, see Figure 5.14e, larger gaps within the cities, see Figure 5.14b, and increases realism for applications where the city is seen as a whole, or where the user can navigate through the generated city, transitioning from outside of the city, through sparser areas and gaps, down into the inner city. Such applications include flight simulators, tycoon games, and some RPGs, and driving simulations and open world games respectively.

  The way the urban landscape transforms over time – sparser areas becoming dense, suburbs forming conurbations – adds further realism to games that take place over long periods of time, e.g., open world games and tycoon games. Previous methods were unable to generates such cities without chance or user intervention.

- *Changing minor road patterns with time.* We observe that the minor road patterns have a tendency to change with time and use this observation to grow our minor roads. The change is visible in both Figure 5.14 and

Figure 5.15. In those Figures, the older parts of the city are darker and the newer parts are lighter. This approach is a new alternative to the existing methods where the spatial distribution of minor road patterns was either user-defined (Parish and Müller 2001) or where a land-use like simulation was used (Vanegas et al. 2009; Weber et al. 2009). Our results show that minor road growth is possible even without a land-use model.

- *Minimal per-city input.* Whereas recent previous work (Vanegas et al. 2009; Weber et al. 2009) is geared towards user-aided generation, our method is designed to require minimal input. This means it is maximally automatic, allowing for unsupervised generation of cities with similar traits. As a result, replay value is increased in games where content regeneration is desirable, such as tycoon games, MMORPGs, and some open world games.

Growth progression for the cities from Figure 5.14 are shown in Figures 5.16 through 5.21.

To conclude, we present a novel, maximally automatic method with several original contributions that generates cities with a rich urban structure (suburbs, gaps) and automatically generated and meaningful major roads over time by means of a trade simulation between neighboring cities.

5.5 Limitations and Future Work

While we hope to have provided a new perspective on procedural city modeling, our method has its limitations.

First, our prototype implementation is not real-time. This is less of a problem where cities can be generated in the background as game-time progresses or where multiple cities are generated at once on several cores. For games that require developed cities quickly, a more production ready implementation would be necessary.

Second, since our method does not simulate land-use, see Section 5.3.4.1, it has to be approximated using one of the existing methods before different types of buildings and other geometry can be generated. Also, if differing minor road patterns were to be generated for different land uses as in (Vanegas et al. 2009), a land use simulation would have to be incorporated as a sub-step in each step of our algorithm.

Next, the traffic $\boldsymbol{t}(t)$ is currently equal for all city pairs without significantly impairing the results. Different $\boldsymbol{t}(t)$ for each city pair could provide further improvements, but is outside of the scope of our paper.

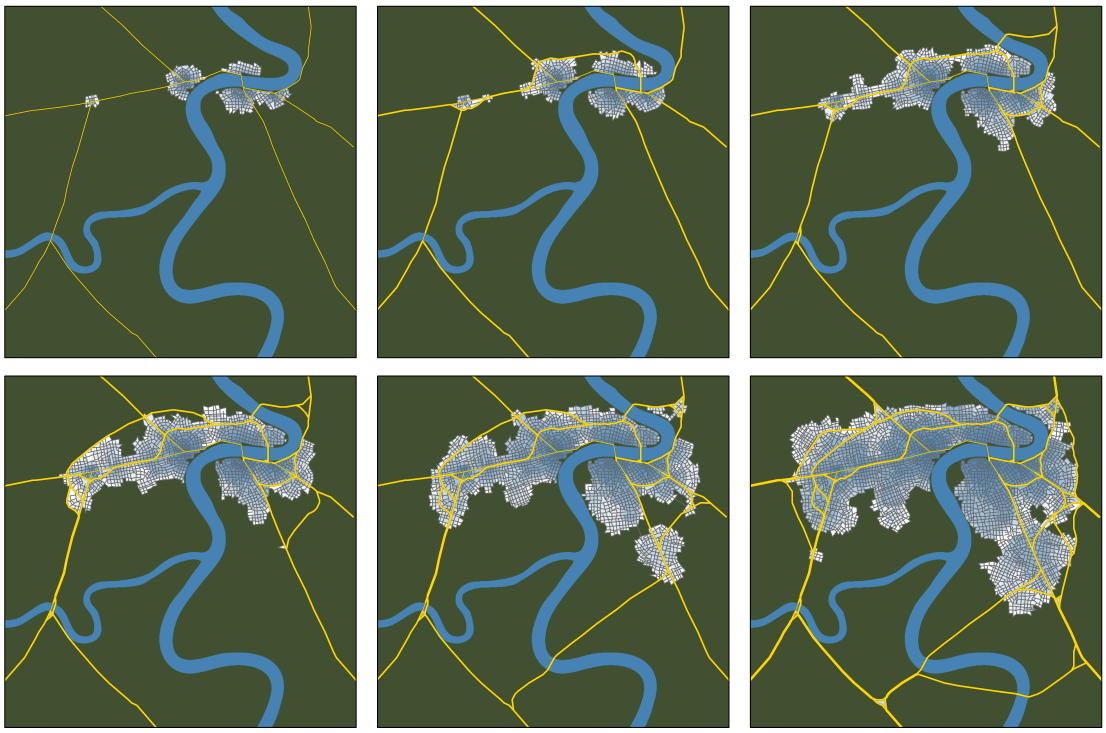
In contrast to existing methods (Weber et al. 2009; Parish and Müller 2001), we are not able to consistently simulate culturally influenced road patterns, e.g., perfect ring roads, which are mostly result of global planning. Nevertheless, partial ring roads do emerge naturally from the simulation, see Figures 5.14c and 5.15f.

We also believe our method could benefit from also considering an intra-city traffic simulation such as the one in Weber et al. (2009) while generating major roads. Furthermore, no current city simulation that develops cities over time has investigated the possibility of redevelopment within cities, including the change

in road structure over time. Such redevelopments, e.g., the Hausmann renovation of Paris, are a non-trivial matter that remains largely unexplored.

Lastly, including further modes of transportation and their "intersections", e.g., subway, train, and bus stops, as well as airports, could further enhance future results.

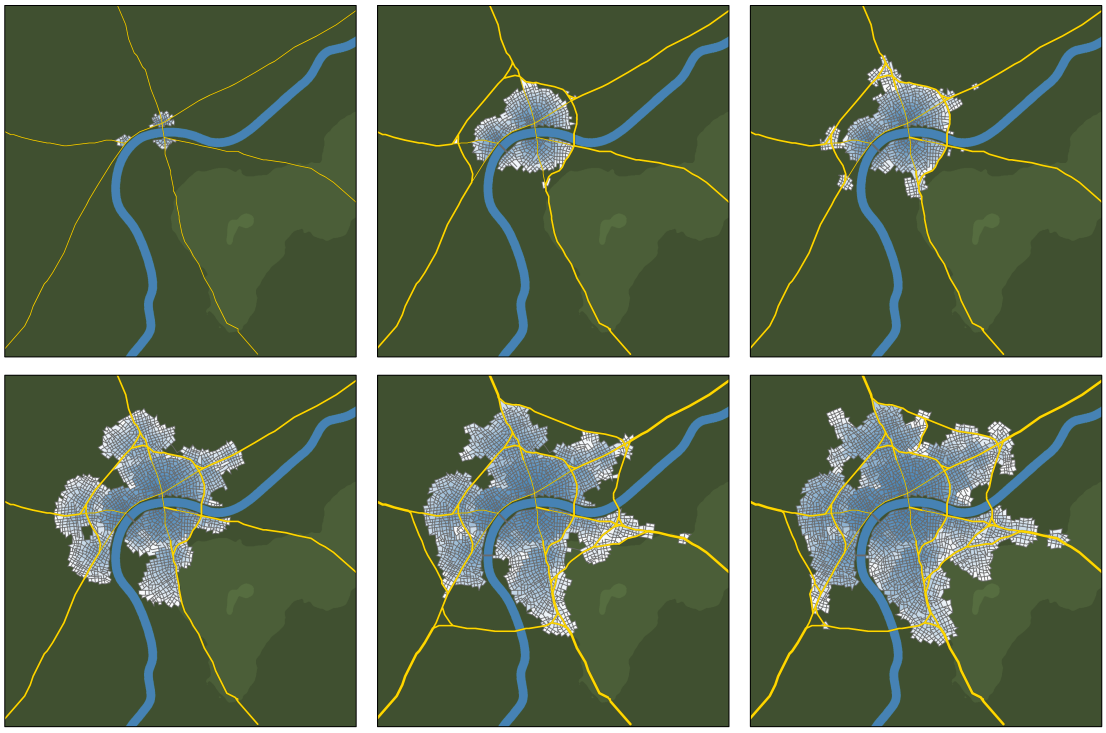
(a) Step 35 (b) Step 49 (c) Step 91



(d) Step 119 (e) Step 175 (f) Step 300

Figure 5.16: Growth of the result from Figure 5.14a.

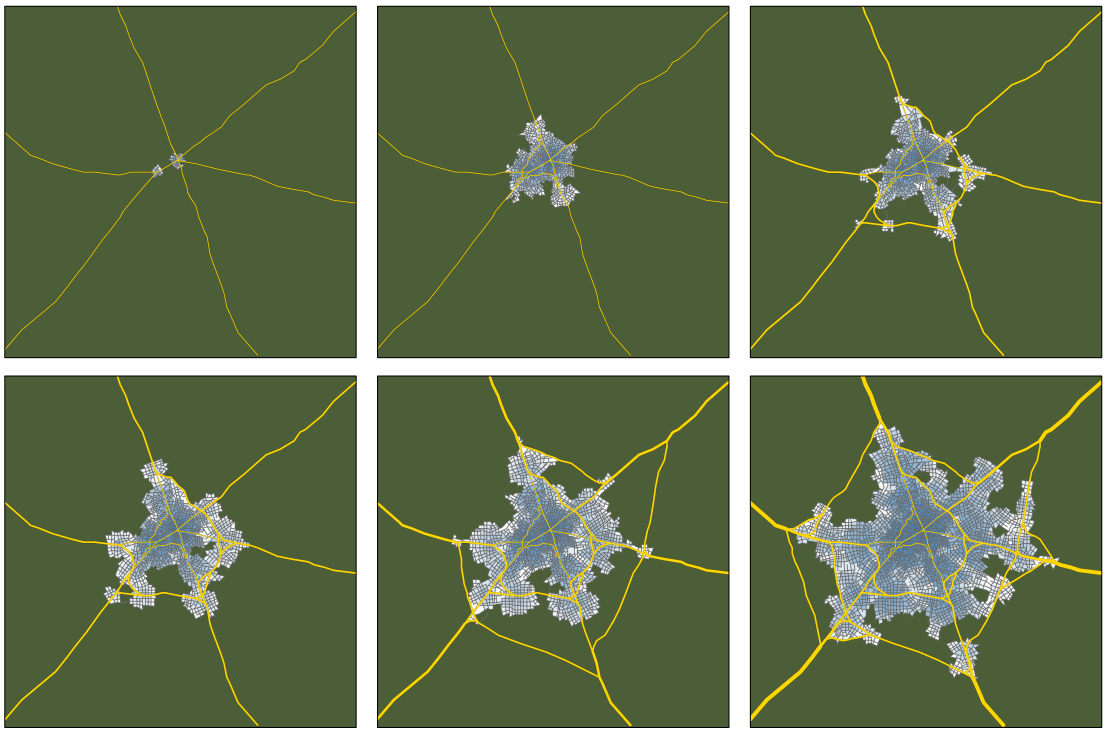
(a) Step 12 (b) Step 55 (c) Step 76



(d) Step 160 (e) Step 230 (f) Step 300

Figure 5.17: Growth of the result from Figure 5.14b.

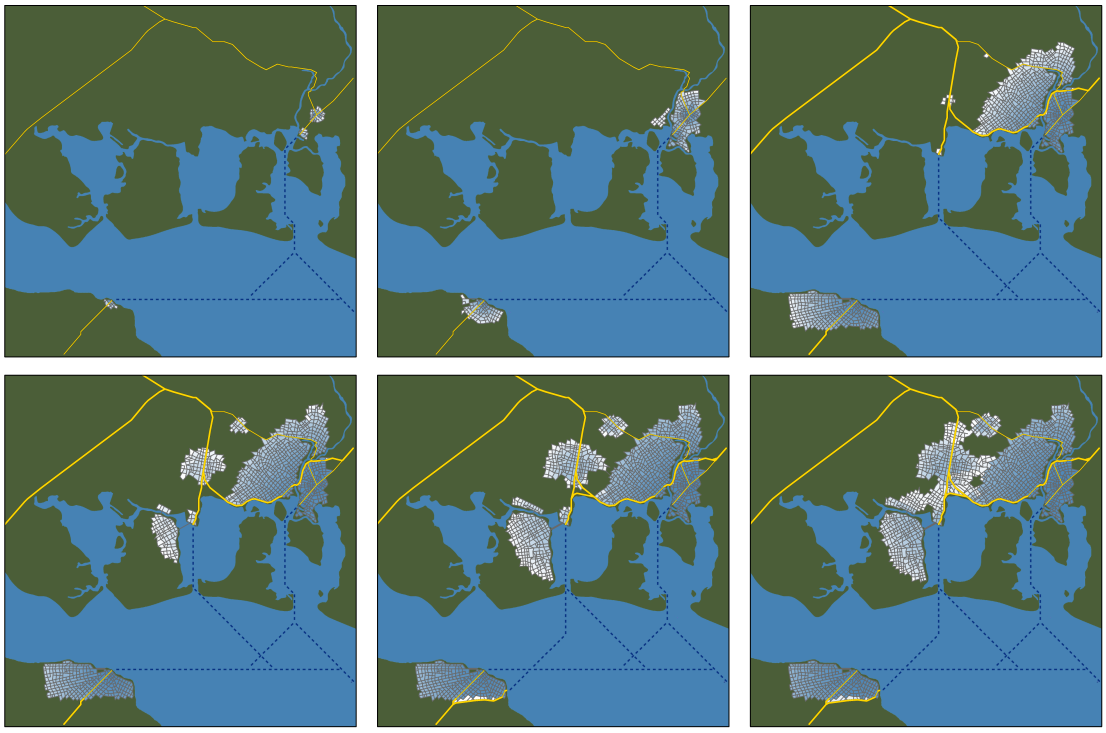
(a) Step 7 (b) Step 63 (c) Step 90



(d) Step 118 (e) Step 160 (f) Step 300

Figure 5.18: Growth of the result from Figure 5.14c.

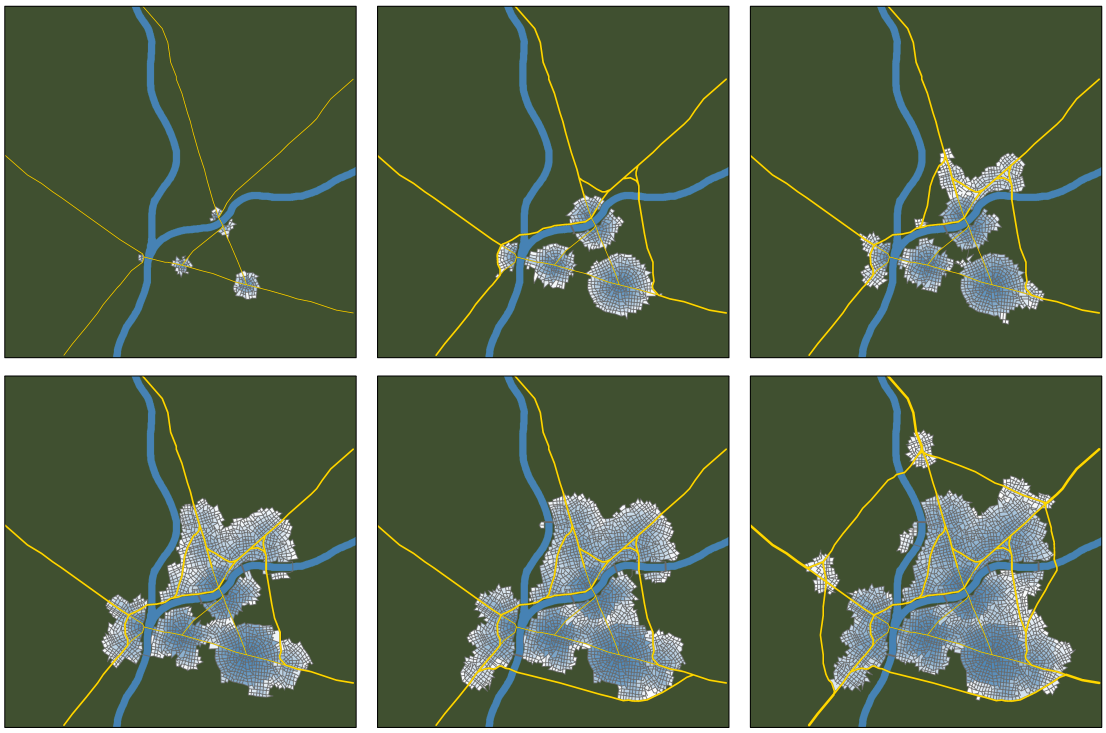
(a) Step 9 (b) Step 31 (c) Step 135



(d) Step 189 (e) Step 259 (f) Step 300

Figure 5.19: Growth of the result from Figure 5.14d.

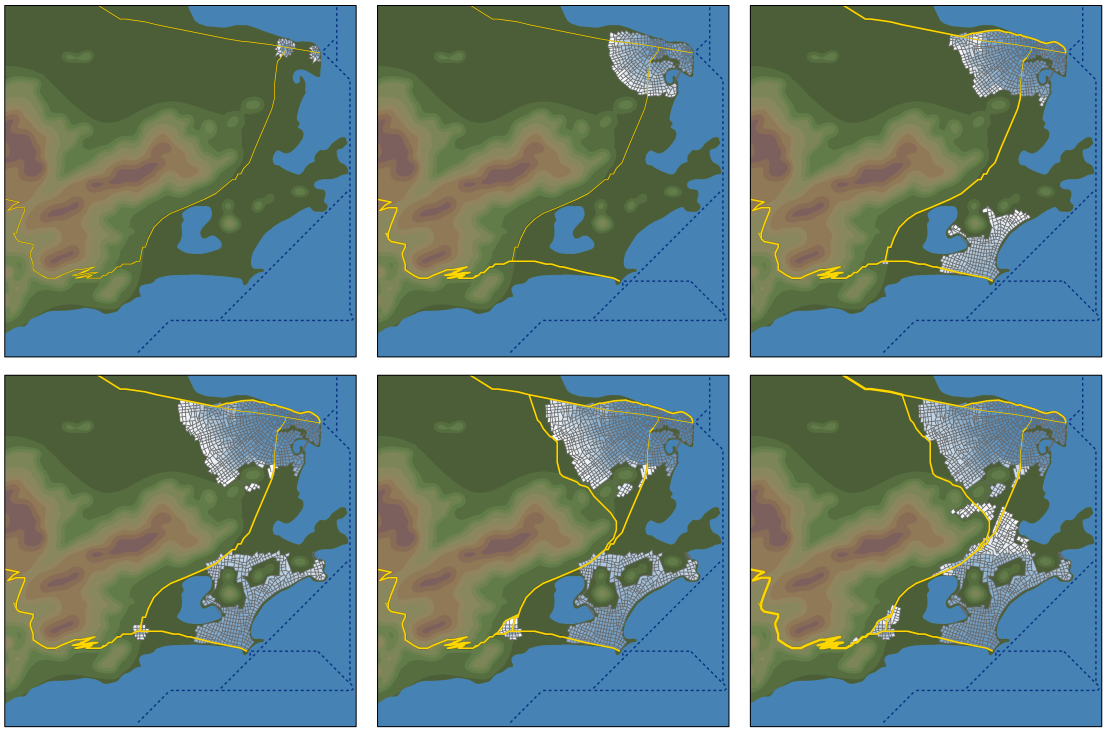
(a) Step 18 (b) Step 69 (c) Step 104



(d) Step 174 (e) Step 230 (f) Step 300

Figure 5.20: Growth of the result from Figure 5.14e.

(a) Step 9 (b) Step 50 (c) Step 120



(d) Step 217 (e) Step 241 (f) Step 300

Figure 5.21: Growth of the result from Figure 5.14f.

5.6 Supplemental Material

5.6.1 Parameters Used for Results

The parameters used for all the presented results are given in Table 5.2. The numeric per-city parameters are given in Table 5.3. The historical context parameters for the generated cities are given in Table 5.4.

Values of the time independent scalar constants $\boldsymbol{O}_1, \ldots, \boldsymbol{O}_4$.			
$\boldsymbol{O}_1$	$\boldsymbol{O}_2$	$\boldsymbol{O}_3$	$\boldsymbol{O}_4$
$0.3 \cdot (180/\pi)$	50	0.5	0.5

Table 5.2

Per-City Parameter Values.		
	$\boldsymbol{b}$ - Reuse coefficient for paths/roads during initial road network construction	$\boldsymbol{u}$ - Coefficient that relates traffic $\boldsymbol{t}(t)$ and growth $\boldsymbol{G}(t)$ by $\boldsymbol{t}(t) = \boldsymbol{b} \cdot \boldsymbol{G}(t)$
Figure 5.14a	0.084	0.55
Figure 5.14b	0.111	0.55
Figure 5.14c	0.14	0.7
Figure 5.14d	0.09	0.45
Figure 5.14e	0.12	0.55
Figure 5.14f	0.2	0.35
Figure 5.15	0.108	0.55

Table 5.3

5.6.2 List of Symbols

The symbols are split into three groups:

1. All functions and coefficients related to the *historical context*, that is those reused for a set of generated cities, are in upper-case bold, for example $\boldsymbol{G}(t), \boldsymbol{K}(t)$, and $\boldsymbol{O}_1$. These are given in Table 5.5.

2. Values that are supplied or *generated for each city* are in lower case bold, for example $\boldsymbol{t}(t)$ and $\boldsymbol{m}$, are given in Table 5.6.

3. Other, *non-input symbols*, mostly used in the description of the algorithm itself, are in a normal, non-bold font, for example t, G, α, and $\textsc{Cost}(P)$. See Table 5.7.

In many cases, e.g. s and S, a lower-case letter will denote members of set denoted by the same upper-case letter. All of the pathfinding related symbols are in calligraphic letters. Functions used in pseudo-code are in small-caps. Where possible, the first letter of the described term is used as a symbol, for example B for the *b*uilt-up area.

Historical Context Parameters for Figures 5.14a, 5.14b, 5.14d, 5.14e, 5.14f, 5.15.											
Time	Step Center	Step Width	Step Height	Recommended Road Capacity	Major Road Coefficient	Minor Road Coefficient	Unbuilt Harbor Coefficient	Built Harbor Coefficient	Bridge Coefficient	Extendible Road Coefficient	Street Pattern
t	$\boldsymbol{S}_c(t)$	$\boldsymbol{S}_w(t)$	$\boldsymbol{S}_p(t)$	$\boldsymbol{A}(t)$	$\boldsymbol{C}_1(t)$	$\boldsymbol{C}_2(t)$	$\boldsymbol{C}_3(t)$	$\boldsymbol{C}_4(t)$	$\boldsymbol{C}_5(t)$	$\boldsymbol{C}_6(t)$	$\boldsymbol{K}$
1–8	2.3	0.5	100	300	0.85	0.4	50000	16666	50000	0.3	Medieval
9-34	2.3	0.5	100	800	0.25	6.3	5000	1666	50000	0.3	Grid
35-95	2.3	0.5	100	1100	0.25	6.3	5000	1666	50000	0.3	Grid
96-300	2.3	0.5	100	1100	0.25	6.3	5000	1666	50000	0.3	Erratic Grid
Historical Context Parameters for Figure 5.14c.											
t	$\boldsymbol{S}_c(t)$	$\boldsymbol{S}_w(t)$	$\boldsymbol{S}_p(t)$	$\boldsymbol{A}(t)$	$\boldsymbol{C}_1(t)$	$\boldsymbol{C}_2(t)$	$\boldsymbol{C}_3(t)$	$\boldsymbol{C}_4(t)$	$\boldsymbol{C}_5(t)$	$\boldsymbol{C}_6(t)$	$\boldsymbol{K}$
1–43	2.3	0.5	100	300	0.85	0.4	50000	16666	50000	0.3	Medieval
44–60	1.9	1.5	40	300	0.125	0.1	5000	1666	50000	0.3	Medieval
61–300	2.3	0.5	100	1100	0.25	6.3	5000	1666	50000	0.3	Grid

Table 5.4: Historical context parameters for presented results. Note that $\boldsymbol{P}(t, \alpha)$ is *fully defined* by means of $\boldsymbol{S}_c(t)$, $\boldsymbol{S}_w(t)$, $\boldsymbol{S}_p(t)$, as mentioned in the caption of Figure 5.9b. We use a step of our algorithm as the basic unit of time. All results were generated over the course of 300 steps.

Historical Context (upper-case bold).		
	$\boldsymbol{G}(t)$	Growth per simulation step, expressed in number of nodes $j \in J$ that will spawn new streets.
	$\boldsymbol{K}(t)$	Sequence of minor road pattern definitions.
	$\boldsymbol{R}(t)$	Major road construction parameters; an umbrella function, see below for a complete list.
$\boldsymbol{R}(t)$	$\boldsymbol{P}(t, \alpha)$	Turn angle penalty function, expressed using $\boldsymbol{S}_c(t)$, $\boldsymbol{S}_w(t)$, $\boldsymbol{S}_p(t)$.
	$\boldsymbol{S}_c(t)$	Center of the step function.
	$\boldsymbol{S}_w(t)$	Width of the step function.
	$\boldsymbol{S}_p(t)$	Height of the step function.
	$\boldsymbol{A}(t)$	Recommended major road capacity.
	$\boldsymbol{C}_1(t)$	Major road edge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{C}_2(t)$	Minor road edge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{C}_3(t)$	Unbuilt harbor edge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{C}_4(t)$	Built harbor edge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{C}_5(t)$	(Unbuilt) bridge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{C}_6(t)$	Extendable road edge cost coefficient for $\mathcal{G}'$.
	$\boldsymbol{O}_1$	Edge slope coefficient.
	$\boldsymbol{O}_2$	Segment traffic capacity per unit WIDTH.
	$\boldsymbol{O}_3$	Trade route length coefficient.
	$\boldsymbol{O}_4$	Trade route cost coefficient.

Table 5.5

Per-City Input (lower-case bold).	
$\boldsymbol{t}(t)$	Traffic for any city pair at time t.
$\boldsymbol{m}$	Terrain elevation map.
$\boldsymbol{nc}$	List of neighboring cities.
$\boldsymbol{y}$	List of all possible neighboring city pairs.
$\boldsymbol{y'}$	An ordered list that is a subset of $\boldsymbol{y}$.
$\boldsymbol{u}$	Reuse coefficient for paths/roads during initial road network construction.
$\boldsymbol{c}$	Number of neighboring cities, $\boldsymbol{c} = \lvert\boldsymbol{nc}\rvert$.
$\boldsymbol{b}$	Coefficient that relates traffic $\boldsymbol{t}(t)$ and growth $\boldsymbol{G}(t)$ by $\boldsymbol{t}(t) = \boldsymbol{b} \cdot \boldsymbol{G}(t)$

Table 5.6

Non-input symbols.	
t	Time, in steps (iterations) of the algorithm.
$T(\ldots)$	Traffic over a segment or a trade route.
$T_A(\ldots)$	Available traffic capacity for a segment or a trade route.
G	Planar graph representing the road network, $G = (J, S)$.
j	Node, an individual vertex of G.
J	Set of all nodes, that is the vertices of G.
s	Segment, an individual edge of G.
S	Set of all segments, that is the edges of G.
$\mathcal{G}$	Pathfinding graph, $\mathcal{G} = (\mathcal{L}, \mathcal{E})$.
$\mathcal{L}$	Regular lattice, that is the set of vertices of $\mathcal{G}$. Spaced at ℓ units (meters). Subscripts L, W, F denote lattice vertices over land, water, and forbidden areas respectively.
$\mathcal{E}$	Set of edges of $\mathcal{G}$. Subscripts L, W, H, B denote edges over land, water, harbor edges, and bridge edges respectively.
e	Edge in $\mathcal{G}$.
ℓ	Spacing of the regular lattice $\mathcal{L}$, in meters.
$\mathcal{G}'$	Temporary copy of $\mathcal{G}$ that allows for pathfinding over both the road network and terrain.
$\mathcal{P}$	Path (sequence of edges) found in $\mathcal{G}$ or $\mathcal{G}'$.
r	"Square" radius up to which the vertices in $\mathcal{L}$ are connected by edges.
p_i	Pair of (distinct) cities in $\boldsymbol{nc}$.
C	Neighboring city $C \in \boldsymbol{nc}$.
P	Trade route, represented as a sequence of segments s, that is edges in G.
P_i	Set of all trade routes that connect the cities of the neighboring city pair p_i.
$P_{i,j}$	The j-th trade route that connect the cities of the neighboring city pair p_i.
P_A	Trade route over existing roads proposal, running exclusively over segments, that is edges, in G.
P_B	Trade route over both existing roads and free terrain, that is segments in G and edges in the pathfinding graph $\mathcal{G}$, after they are copied to $\mathcal{G}'$.
P_W	The better of P_A, P_B.
α	Angle between two consecutive segments, ranging from 0 (turn back) to π (no turn).
$\mathcal{S}$	City shell, the outer face of the planar graph G.
B	Built-up area of the city.
$B_{B,i}$	Built-up block.
$B_{T,i}$	Terrain block.

Non-input symbols, continued.	
FILL	Fill operator that returns the set of points on the inside of a block or the city shell.
NB	Neighborhood operator that returns the set of points in the neighborhood of a block or the city shell.
LENGTH(e)	Length of and edge $e \in \mathcal{E}$.
COST(P)	Cost of a trade route P.
WEIGHT(e)	Weight of an edge $e \in \mathcal{E}$, that is the difficulty of getting from one vertex of the edge to another.
WIDTH(s)	Width of a road/segment $s \in S$, in meters.
EUCLIDEANDISTANCE(a, b)	Euclidean distance between the positions of two objects a and b.
RANK(z)	Ranking function of nodes z that can grow new minor roads.
SMOOTHSTEP(x)	Smooth step function $3x^2 - 2x^3$.
Z	Set of nodes from which new minor roads may be grown.
Z'	List of nodes in Z ordered by RANK(z).
z	Nodes in either Z or Z'.
q	Nucleus of city growth.
q_N	Nuclei nearest to a given node z.
q_C	Nuclei on the initial road network that is the closest to geometrical center of the map X.
Q	Set of city growth nuclei.
STRENGTH(q)	Strength of nucleus, STRENGTH$(q) \in \langle 0, 1\rangle$, with 0 for no influence at all and 1 for maximum influence.
μ	Expected value of the number of cities $\boldsymbol{c}$.
ϕ	Global angular offset of neighboring cities. $\phi \in \langle 0, 2\pi)$.
θ_i	Per-ray perturbation of the i-th neighboring city. $\theta_i \in \langle -\frac{\pi}{2\boldsymbol{c}}, \frac{\pi}{2\boldsymbol{c}}\rangle$.
$E_{L,\max}$	Maximum edge slope for an edge $e \in \mathcal{E}$.
$E_{L,\text{cut}}$	Maximum allowed edge slope for any edge $e \in \mathcal{E}$.
X	Geometrical center of the map.

Table 5.7

5.6.3 Comparison of Existing Methods

Method	Focus	TP	Minor Roads	Major Roads	Land Use	Contributions to road network modeling
Our method	Automatic	Yes	Rule-based pattern expansion	Inter-city traffic simulation	None	Influence of neighboring cities, formation of urban landscape, changing minor road patterns with time, effects of water transportation.
Aliaga et al. 2008	Automatic User-aided	No	Random walk over set of intersections		None	Synthesizes road networks from example fragments, introduces random walk road network synthesis.
Chen et al. 2008a	User-aided	No	Tensor field tracing	Tensor field tracing	None	Tensor field driven minor and major road generation, easily steerable road network generation.
Emilien et al. 2012	Automatic	Yes	Shortest path	None	None	Scattered settlements, meaningful initial settlement positioning, roads connecting to neighboring cities.
Galin et al. 2011	Automatic User-aided	No	None	Graph skeletons	None	First to generate large-scale major road networks, graph-pruning-based road generation.
Krecklau et al. 2012	User-aided	Yes	Building/destroying roads between keyframes	Building/destroying roads between keyframes	Input map interpolation	Interpolation of historical map "key-frames" by construction/destruction events, uses a scheduling algorithm.
Lechner et al. 2003	Automatic	Yes	Agent-based	None	Agent-based	Extender and connector agents generate minor roads, one kind of developer agent for each land use generates parcels.
Parish and Müller 2001	Automatic	No	Rule-based patterns	Follow population gradient	None	First to generate urban road networks, introduced rule-based pattern expansion for generating minor roads.
Vanegas et al. 2009	User-aided	No	Enhanced rule-based pattern expansion	Sketched by user, possibly expansion	Behavioral modeling	Agent-based behavioral modeling, integration with geometric modeling, road generation adapts to land use. Highly interactive.
Vanegas et al. 2012a	User-aided	No	Rule-based pattern expansion	Rule-based pattern expansion	User-defined	Ability to model using target indicators, coupling of existing forward procedural modeling methods and inverse modeling to generate desired cities.
Weber et al. 2009	Automatic	Yes	Rule-based pattern expansion	Pattern expansion	Dynamic disequilibrium	First to generate cities that grow over time, highly configurable land-use simulation.
Yang et al. 2013	Automatic User-aided	No	Templates and/or streamline tracing	Templates and/or streamline tracing	none	Generates optimized urban layouts suitable for urban planning, hierarchically constructs layouts using either templates or streamline tracing.
Yu and Steed 2012	Automatic	Yes	Pattern matching	Pattern matching	No	Propose a measure for finding best intersection/outgoing roads match in example data.

Table 5.8: Basic comparison of methods that support road network synthesis.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we addressed issues in procedural modeling and realism in computer graphics. We provided an in-depth discussion of both general realism and realism in procedural modeling, extending the latter by conducting a user-study on realism of procedurally generated buildings. We also proposed a method for generation of representational models of urban road networks.

In Chapter 2, we identified three *elements of realism* – geometry, material appearance, and rendering algorithm – that are required for full realism. We discussed that even things that do not exist in real life can be realistic and pointed out how culture, experience, and historical period influence perception of realism. We also provided a different, photography-centric perspective of realism, making clear the distinction between common photography and measurement photography and the biases in perception of photographs. Next, we proposed rendering algorithm research could adapt a test-driven approach to keep track of changes in its quality and level of realism. Finally, we sketched the possibilities the adoption of neural networks might bring to evaluation and computer representation of realism.

In Chapter 3, we showed that generated models can be representational or full. This allowed us to point out that these two groups of models need to be evaluated in different ways so as to not introduce biases. In order to arrive at a definition of generative power, we then established the concepts of desirable outputs and procedural range. We concluded that while not perfect, precision is the only most easily applicable metric of generative power and that other metrics, such as recall or F-score, are hard to establish for procedural rules. Finally, we discussed additional possible biases and things to be aware of during evaluation of procedurally generated models.

We followed up with Chapter 4, which builds on the previous discussion by investigating realism of procedurally generated buildings. We showed that realism in procedural buildings is not only due to fine detail, but also due to coarser detail and structure at various scales. Consequently, we recommended that authors focus not only on small detail, but also on what is left when the small and very small detail is taken away. Among qualitative factors, irregularity, plausibility, and quality of texture mapping were shown to be especially relevant to procedur-

ally generated buildings. Contrary to our expectations, we were unable to verify that repeated features, as discussed in Section 3.3.4, play a significant role.

Lastly, in Chapter 5, we demonstrate how to grow meaningfully structured cities that exist in the context of their surroundings. In order to do this, we use the above mentioned context to determine where trade routes would exist on a map and where city growth should happen, using a traffic simulation to determine the saturation of existing trade routes and the need for new ones. Unlike previous work, our cities have a plausible urban structure with suburbs, ports, irregular fringes and urban gaps.

We believe we have contributed to computer graphics by reviving and extending previous discussion on realism and by pointing out novel issues in realism of procedurally modeled content. In the long term, we hope this will lead to both more research on realism and desirability, as well as adoption and adherence to better and more diligent evaluation and validation protocols in both procedural modeling and, for example, rendering. We hope our user-study will be seen as a stepping stone that others can follow up on. Finally, we hope our road network generation algorithm, aside from generating more plausible results, illustrates that reasonable middle ground between a complex simulation and building on first principles can be found even for complex man-made artifacts

6.2 Future Work

Most of the current procedural modeling literature relies on qualitative evaluation of desirability and/or realism. We nevertheless believe that the future lies in adoption of more rigorous, quantitative evaluation. This evaluation, in turn, depends on easy to execute evaluation methodologies for procedurally generated content, similar in nature to those suggested by Mantiuk et al. (2012) for Image Quality Assessment. This means that we need to develop a deeper understanding of the various biases and problems in evaluating procedurally generated content so that these methodologies can be meaningful. Such progress would also allow authors of procedural rules to better direct effort where it is needed.

While the above progress would and the current state of the art does allow us to measure the precision of a procedural rule, it would be very helpful if we could also measure the recall or some other metrics which would tell us how much desirable content is not being generated by a procedural rule. The discussion in Section 3.2.4 suggests that for this to happen, significant advances in automated evaluation would have to take place.

A further future direction concerning automated evaluation is in being able to extract and better represent and separate visual realism (Wu et al. 2011; Rahmouni et al. 2017; Quan et al. 2018) and the semantics (Gatys et al. 2015a; Johnson et al. 2016) of an image or a model. While we do not believe the advances could be good enough to allow for full automation – especially for complex models of man-made objects where domain knowledge would be required – it could allow for further development of overall understanding and partial automation. This could be in the form of evaluation functions for search-based methods, or in supplementing human evaluation.

Bibliography

Akenine-Möller, Tomas, Eric Haines and Naty Hoffman (2018). *Real-Time Rendering, Fourth Edition*. A K Peters/CRC Press. ISBN: 978-1-1386-2700-0 (pages 10, 17).

Aliaga, Daniel G, Carlos A Vanegas and Bedřich Beneš (2008). 'Interactive example-based urban layout synthesis'. In: *ACM Transactions on Graphics* 27.5. DOI: 10.1145/1409060.1409113 (pages 84, 91, 113).

Apodaca, Anthony A. and Larry Gritz (1999). *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann. ISBN: 1558606181 (page 17).

Automobile Club of Southern California (1915). *Untitled motorist map of the Los Angeles area*. URL: http://www.bigmapblog.com/2011/map-of-los-angeles-and-the-san-gabriel-mountains-1915 (visited on 31st Mar. 2019) (page 83).

Aydın, Tunç O., Rafał Mantiuk and Hans-Peter Seidel (Feb. 2008). 'Extending quality metrics to full luminance range images'. In: *Human Vision and Electronic Imaging XIII*. Ed. by Bernice E. Rogowitz and Thrasyvoulos N. Pappas. SPIE. DOI: 10.1117/12.765095 (page 30).

Bakeman, Roger (2005). 'Recommended effect size statistics for repeated measures designs'. In: *Behavior research methods* 37.3, pp. 379–384. DOI: 10.3758/BF03192707 (page 71).

Barsky, Brian A. (2004). 'Vision-realistic rendering'. In: *Proceedings of the 1st Symposium on Applied perception in graphics and visualization - APGV '04*. ACM Press. DOI: 10.1145/1012551.1012564 (page 18).

Beck, Kent (2002). *Test Driven Development. By Example*. Addison Wesley. 192 pp. ISBN: 0321146530 (page 32).

Beyer, Johanna, Markus Hadwiger and Hanspeter Pfister (May 2015). 'State-of-the-Art in GPU-Based Large-Scale Volume Visualization'. In: *Computer Graphics Forum* 34.8, pp. 13–37. DOI: 10.1111/cgf.12605 (page 10).

Bitterli, Benedikt (2016). *Rendering resources*. URL: https://benedikt-bitterli.me/resources/ (visited on 16th June 2019) (page 32).

Boughorbel, Sabri, Fethi Jarray and Mohammed El-Anbari (June 2017). 'Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric'. In: *PLOS ONE* 12.6. Ed. by Quan Zou. DOI: 10.1371/journal.pone.0177678 (page 53).

Brendel, Wieland and Matthias Bethge (2019). 'Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet'. In: *CoRR* abs/1904.00760. URL: http://arxiv.org/abs/1904.00760 (visited on 16th June 2019) (pages 34, 59).

Butterfield, Chalmers (1950). *Garden*. URL: https://commons.wikimedia.org/wiki/File:Garden_of_the_Gods_Park_,_Colorado_,_1950%27s_,_Kodachrome_by_Chalmers_Butterfield.jpg (visited on 6th July 2018) (page 25).

Čadík, Martin (Jan. 2008). 'Perceptually Based Image Quality Assessment and Image Transformations'. PhD thesis. CTU Prague. URL: http://cadik.posvete.cz/diss/ (visited on 18th June 2019) (pages 30, 55, 61).

Čadík, Martin, Robert Herzog, Rafał Mantiuk, Radosław Mantiuk, Karol Myszkowski and Hans-Peter Seidel (Oct. 2013). 'Learning to Predict Localized Distortions in Rendered Images'. In: *Computer Graphics Forum* 32.7, pp. 401–410. DOI: 10.1111/cgf.12248 (page 31).

Čadík, Martin, Robert Herzog, Rafał Mantiuk, Karol Myszkowski and Hans-Peter Seidel (Nov. 2012). 'New measurements reveal weaknesses of image quality metrics in evaluating graphics artifacts'. In: *ACM Transactions on Graphics* 31.6, p. 1. DOI: 10.1145/2366145.2366166 (page 31).

Carol M. Highsmith (2006). *Farnsworth House*. URL: https://commons.wikimedia.org/wiki/File:Farnsworth_House_2006.jpg (visited on 10th Feb. 2019) (page 41).

Chen, Guoning, Gregory Esch, Peter Wonka, Pascal Müller and Eugene Zhang (Aug. 2008a). 'Interactive Procedural Street Modeling'. In: *ACM Transactions on Graphics* 27.3, 103:1–103:10. DOI: 10.1145/1360612.1360702 (pages 84, 113).

Chen, Xuejin, Boris Neubert, Ying-Qing Xu, Oliver Deussen and Sing Bing Kang (Dec. 2008b). 'Sketch-based tree modeling using Markov random field'. In: *ACM Transactions on Graphics* 27.5. DOI: 10.1145/1409060.1409062 (page 40).

Childe, Vere Gordon (Apr. 1950). 'The Urban Revolution'. In: *The Town Planning Review* 21.1. DOI: 10.3828/tpr.21.1.k853061t614q42qh (pages 82, 92).

Chiusano, Paul and Rúnar Bjarnason (2014). *Functional Programming in Scala*. Manning Publications. 320 pp. ISBN: 1617290653 (page 44).

Chollet, Francois (2017). *Deep Learning with Python*. Manning Publications. ISBN: 9781617294433 (page 33).

Cohen, Michael F. and John R. Wallace (1995). *Radiosity and Realistic Image Synthesis*. Academic Press, Inc. ISBN: 0124121500 (page 13).

Computer Gaming World, Issue 113 (1993). URL: http://www.cgwmuseum.org/galleries/issues/cgw_113.pdf (visited on 24th Sept. 2018) (page 19).

Computer Gaming World, Issue 52 (1988). URL: http://pdf.textfiles.com/zines/CGW/1988_10_issue52.pdf (visited on 4th Feb. 2019) (page 19).

Computer Gaming World, Issue 98 (1992). URL: http://www.cgwmuseum.org/galleries/issues/cgw_98.pdf (visited on 24th Sept. 2018) (page 19).

Cornell University Program of Computer Graphics (1998). *History of the Cornell Box*. URL: http://www.graphics.cornell.edu/online/box/history.html (visited on 6th July 2018) (page 28).

Corsini, Massimiliano, Elisa Drelie Gelasca, Touradj Ebrahimi and Mauro Barni (Feb. 2007). 'Watermarked 3-D Mesh Quality Assessment'. In: *IEEE Transactions on Multimedia* 9.2, pp. 247–256. DOI: 10.1109/tmm.2006.886261 (page 57).

Corsini, Massimiliano, Chaker Larabi, Guillaume Lavoué, Oldřich Petřík, Libor Váša and Kai Wang (Jan. 2013). 'Perceptual Metrics for Static and Dynamic Triangle Meshes'. In: *Computer Graphics Forum* 32.1, pp. 101–125. DOI: 10.1111/cgf.12001 (page 57).

Cortes, Rudy and Saty Raghavachary (2007). *The RenderMan Shading Language Guide*. Cengage Learning. ISBN: 1598632868 (page 93).

Costa, Lúcio (1957). *Report of the Pilot Plan of Brasilia*. Competition Entry (page 84).

Daly, Scott J. (Aug. 1992). 'Visible differences predictor: an algorithm for the assessment of image fidelity'. In: *Human Vision, Visual Processing, and Digital Display III*. Ed. by Bernice E. Rogowitz. SPIE. DOI: 10.1117/12.135952 (pages 15, 23, 27, 30, 33, 61).

Deering, Michael F. (July 2005). 'A photon accurate model of the human eye'. In: *ACM Transactions on Graphics* 24.3, p. 649. DOI: 10.1145/1073204.1073243 (page 18).

Deussen, Oliver, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr and Przemysław Prusinkiewicz (1998). 'Realistic modeling and rendering of plant ecosystems'. In: *Proceedings of SIGGRAPH '98*. ACM, pp. 275–286. DOI: 10.1145/280814.280898 (pages 40, 61).

Disney (2019). *Moana Island Scene – Data Set*. URL: https://www.technology.disneyanimation.com/islandscene (visited on 16th Feb. 2019) (page 32).

Doersch, Carl, Saurabh Singh, Abhinav Gupta, Josef Sivic and Alexei A. Efros (2012). 'What Makes Paris Look like Paris?' In: *Proceedings of SIGGRAPH '12* 31.4. DOI: 10.1145/2185520.2185597 (page 61).

Dorsey, Julie, Holly Rushmeier and François Sillion (2007). *Digital Modeling of Material Appearance*. Morgan Kaufmann. ISBN: 978-0-12-221181-2 (pages 9, 10).

Dosovitskiy, Alexey, Jost Tobias Springenberg, Maxim Tatarchenko and Thomas Brox (2014). 'Learning to Generate Chairs, Tables and Cars with Convolutional Networks'. In: arXiv: 1411.5928v4. (Visited on 23rd June 2019) (page 35).

Dutré, Philip, Philippe Bekaert and Kavita Bala (2006). *Advanced Global Illumination*. A K Peters/CRC Press. ISBN: 9781568813073 (pages 13, 27).

Ebert, David S (1994). *Texturing & modeling: a procedural approach*. AP Professional. ISBN: 0122287606 (pages 38, 39, 41, 61).

Emilien, Arnaud, Adrien Bernhardt, Adrien Peytavie, Marie-Paule Cani and Eric Galin (2012). 'Procedural Generation of Villages on Arbitrary Terrains'. In: *The Visual Computer*. DOI: 10.1007/s00371-012-0699-7 (pages 84, 113).

Fan, Lubin and Peter Wonka (2016). 'A Probabilistic Model for Exteriors of Residential Buildings'. In: *ACM Transactions on Graphics* 35.5, p. 155. DOI: 10.1145/2910578 (page 61).

Fascione, Luca, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley and Brian Green (2017). 'Path tracing in production - part 1'. In: *ACM SIGGRAPH 2017 Courses on - SIGGRAPH '17*. ACM Press. DOI: 10.1145/3084873.3084904 (page 17).

Fascione, Luca, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich and Johannes Meng (Aug. 2018). 'Manuka'. In: *ACM Transactions on Graphics* 37.3, pp. 1–18. DOI: 10.1145/3182161 (page 30).

Fawcett, Tom (June 2006). 'An introduction to ROC analysis'. In: *Pattern Recognition Letters* 27.8, pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010 (pages 51, 53, 55).

Ferwerda, James A. (June 2003). 'Three varieties of realism in computer graphics'. In: *Human Vision and Electronic Imaging VIII*. Ed. by Bernice E. Rogowitz and Thrasyvoulos N. Pappas. SPIE. DOI: 10.1117/12.473899 (pages 6, 12, 13, 14).

Field, Andy, Jeremy Miles and Zoe Field (2012). *Discovering Statistics Using R*. SAGE Publications Ltd. 992 pp. ISBN: 1446200450 (pages 48, 51).

Frey, William H. and Zachary Zimmer (2001). 'Defining the City and Urbanization'. In: *Handbook of Urban Studies*. Ed. by R. Paddison and W.F. Lever. Sage Publications, pp. 14–35. ISBN: 080397695X (pages 85, 92).

Gain, James, Bruce Merry and Patrick Marais (May 2015). 'Parallel, Realistic and Controllable Terrain Synthesis'. In: *Computer Graphics Forum* 34.2, pp. 105–116. DOI: 10.1111/cgf.12545 (pages 55, 79).

Galin, Eric, Adrien Peytavie, Eric Guérin and Bedřich Beneš (2011). 'Authoring Hierarchical Road Networks'. In: *Computer Graphics Forum* 30.7. DOI: 10.1111/j.1467-8659.2011.02055.x (pages 84, 113).

Galin, Eric, Adrien Peytavie, Nicolas Maréchal and Eric Guérin (2010). 'Procedural Generation of Roads'. In: *Computer Graphics Forum* 29.2. DOI: 10.1111/j.1467-8659.2009.01612.x (pages 83, 84, 88, 89).

Garcia-Dorado, Ignacio, Daniel G. Aliaga and Satish V. Ukkusuri (May 2014). 'Designing large-scale interactive traffic animations for urban modeling'. In: *Computer Graphics Forum* 33.2, pp. 411–420. DOI: 10.1111/cgf.12329 (page 84).

Garreau, Joel (1992). *Edge City: Life on the New Frontier*. Anchor. ISBN: 9780385424349 (page 92).

Gaspar, Joaquim Alves (2010). *A young male cat*. URL: https://commons.wikimedia.org/wiki/File:Cat_March_2010-1.jpg (visited on 21st Apr. 2019) (pages 33, 34).

Gatys, Leon A., Alexander S. Ecker and Matthias Bethge (2015a). 'A Neural Algorithm of Artistic Style'. In: arXiv: 1508.06576v2. (Visited on 23rd June 2019) (pages 33, 115).

Gatys, Leon A., Alexander S. Ecker and Matthias Bethge (2015b). 'Texture Synthesis Using Convolutional Neural Networks'. In: arXiv: 1505.07376v3. (Visited on 23rd June 2019) (page 33).

Geddes, Patrick (1915). *Cities in evolution: an introduction to the town planning movement and to the study of civics*. Williams & Norgate. URL: https://archive.org/details/citiesinevolutio00gedduoft (visited on 18th June 2019) (page 82).

Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann and Wieland Brendel (2018). 'ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness'. In: arXiv: 1811.12231v2. (Visited on 23rd June 2019) (pages 34, 59).

Georgiev, Iliyan, Jaroslav Křivánek, Tomáš Davidovič and Philipp Slusallek (Nov. 2012). 'Light transport simulation with vertex connection and merging'. In: *ACM Transactions on Graphics* 31.6, p. 1. DOI: 10.1145/2366145.2366211 (page 32).

Georgios N. Yannakakis, Julian Togelius (2018). *Artificial Intelligence and Games*. Springer International Publishing (page 47).

Gimini, Gianluca (2009). *Velocipedia*. URL: http://www.gianlucagimini.it/prototypes/velocipedia.html (visited on 17th Nov. 2018) (pages 10, 11).

Gombrich, E. H. (1961). *Art and Illusion: A Study in the Psychology of Pictorial Representation (The A. W. Mellon Lectures in the Fine Arts)*. Princeton University Press. ISBN: 9780691097855 (pages 19, 21, 23).

Gombrich, E.H. (1995). *The Story of Art*. Phaidon Press. ISBN: 0714832472 (page 19).

Goodfellow, Ian J., Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press. (Visited on 23rd June 2019) (page 32).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014). 'Generative Adversarial Networks'. In: arXiv: 1406.2661v1. (Visited on 23rd June 2019) (page 34).

Greenberg, Donald P., Kenneth E. Torrance, Peter Shirley, James Arvo, Eric Lafortune, James A. Ferwerda, Bruce Walter, Ben Trumbore, Sumanta Pattanaik and Sing-Choong Foo (1997). 'A framework for realistic image synthesis'. In: *Proceedings of SIGGRAPH '97*. ACM Press/Addison-Wesley Publishing Co., pp. 477–494. DOI: 10.1145/258734.258914 (pages 28, 29).

Greg (2016). *No Man's Sky – Procedural Content*. URL: http://3dgamedevblog.com/wordpress/?p=836 (visited on 6th Feb. 2019) (page 39).

Groenewegen, Saskia A., Ruben M. Smelik, Klaas Jan de Kraker and Rafael Bidarra (2009). 'Procedural City Layout Generation Based on Urban Land Use Models'. In: *Eurographics Short Papers*. DOI: 10.2312/egs.20091045 (pages 84, 92).

Guérin, Eric, Eric Galin, François Grosbellet, Adrien Peytavie and Jean-David Geneveaux (2016). 'Efficient modeling of entangled details for natural scenes'. en. In: *Computer Graphics Forum (Proceedings of Pacific Graphics 2016)* 35.7. DOI: https://doi.org/10.1111/cgf.13023 (page 64).

Guo, Jinjiang, Vincent Vidal, Irene Cheng, Anup Basu, Atilla Baskurt and Guillaume Lavoué (Oct. 2016). 'Subjective and Objective Visual Quality Assess-

ment of Textured 3D Meshes'. In: *ACM Transactions on Applied Perception* 14.2, pp. 1–20. DOI: 10.1145/2996296 (page 57).

Hachisuka, Toshiya, Anton S. Kaplanyan and Carsten Dachsbacher (July 2014). 'Multiplexed metropolis light transport'. In: *ACM Transactions on Graphics* 33.4, pp. 1–10. DOI: 10.1145/2601097.2601138 (page 32).

Hanrahan, Pat and Wolfgang Krueger (1993). 'Reflection from layered surfaces due to subsurface scattering'. In: *Proceedings of SIGGRAPH '93*. ACM Press. DOI: 10.1145/166117.166139 (page 9).

Hartmann, Stefan, Michael Weinmann, Raoul Wessel and Reinhard Klein (2017). 'StreetGAN: Towards Road Network Synthesis with Generative Adversarial Networks'. In: *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2017*, pp. 133–142. ISBN: 9788086943442 (page 85).

Heightmap created with Terragen (2006). URL: https://commons.wikimedia.org/wiki/File:Heightmap.png (visited on 10th Feb. 2019) (page 38).

Hendrikx, Mark, Sebastiaan Meijer, Joeri Van Der Velden and Alexandru Iosup (Feb. 2013). 'Procedural content generation for games'. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 9.1, pp. 1–22. DOI: 10.1145/2422956.2422957 (pages 38, 40).

Herzog, Robert, Martin Čadík, Tunç O. Aydın, Kwang In Kim, Karol Myszkowski and Hans-Peter Seidel (May 2012). 'NoRM: No-Reference Image Quality Metric for Realistic Image Synthesis'. In: *Computer Graphics Forum* 31.2pt3, pp. 545–554. DOI: 10.1111/j.1467-8659.2012.03055.x (page 31).

Hore, Alain and Djemel Ziou (Aug. 2010). 'Image Quality Metrics: PSNR vs. SSIM'. In: *2010 20th International Conference on Pattern Recognition*. IEEE. DOI: 10.1109/icpr.2010.579 (page 31).

Hošek, Lukáš and Alexander Wilkie (2012). 'An analytic model for full spectral sky-dome radiance'. In: *ACM Transactions on Graphics* 31.4, p. 95. DOI: 10.1145/2185520.2185591 (pages 31, 67).

Howard, Ebenezer (2019). *Group of Slumless Smokeless Cities*. URL: https://en.wikipedia.org/wiki/Garden_city_movement#/media/File:Garden_City_Concept_by_Howard.jpg (visited on 11th Apr. 2019) (page 84).

Howard, Jeremy and Rachel Thomas (2018). *FastAI*. URL: http://www.fast.ai (visited on 21st July 2018) (page 32).

Howell, Chris (1998). *980310-N-7355H-003 ABOARD USS INDEPENDENCE*. URL: https://commons.wikimedia.org/wiki/File:980310-N-7355H-003_Simulator_Training.jpg (visited on 10th Feb. 2019) (page 14).

Hoyt, Homer (1939). *The structure and growth of residential neighborhoods in American cities*. Washington, D.C.: US Government Printing Office. URL: https://archive.org/details/structuregrowtho00unit (visited on 23rd June 2019) (pages 82, 92).

Jensen, Henrik Wann (2001). *Realistic Image Synthesis Using Photon Mapping*. A K Peters/CRC Press. ISBN: 9781568814629 (pages 13, 27).

Johnson, James H. (1976). *Urban Geography, An Introductory Analysis, 2nd Edition*. Oxford, UK: Pergamon Press. ISBN: 0080169287 (pages 82, 86, 92, 101).

Johnson, Justin, Alexandre Alahi and Li Fei-Fei (2016). 'Perceptual Losses for Real-Time Style Transfer and Super-Resolution'. In: *Computer Vision – ECCV 2016*. Springer International Publishing, pp. 694–711. DOI: 10.1007/978-3-319-46475-6_43 (pages 33, 115).

Johnson, Micah K., Kevin Dale, Shai Avidan, Hanspeter Pfister, William T. Freeman and Wojciech Matusik (2011). 'CG2Real: Improving the Realism of Computer Generated Images using a Large Collection of Photographs'. In: *IEEE Transactions on Visualization and Computer Graphics* 17.9, pp. 1273–1285. DOI: 10.1109/TVCG.2010.233 (pages 61, 64).

Kajiya, James T. (Aug. 1986). 'The rendering equation'. In: *ACM SIGGRAPH Computer Graphics* 20.4, pp. 143–150. DOI: 10.1145/15886.15902 (page 29).

Kalogerakis, Evangelos, Siddhartha Chaudhuri, Daphne Koller and Vladlen Koltun (July 2012). 'A Probabilistic Model for Component-based Shape Synthesis'. In: *ACM Transactions on Graphics* 31.4, 55:1–55:11. ISSN: 0730-0301. DOI: 10.1145/2185520.2185551 (pages 42, 50, 61).

Karras, Tero, Samuli Laine and Timo Aila (2018). 'A Style-Based Generator Architecture for Generative Adversarial Networks'. In: arXiv: http://arxiv.org/abs/1812.04948v2. (Visited on 23rd June 2019) (page 35).

Keller, Alexander, Luca Fascione, Marcos Fajardo, Iliyan Georgiev, Per Christensen, Johannes Hanika, Christian Eisenacher and Greg Nichols (2015). 'The path tracing revolution in the movie industry'. In: *ACM SIGGRAPH 2015 Courses on - SIGGRAPH '15*. ACM Press. DOI: 10.1145/2776880.2792699 (page 30).

Kelly, George (2006). 'A Survey of Procedural Techniques for City Generation'. In: *The ITB Journal* 7.2. DOI: 10.21427/d76m9p (page 38).

Kelly, Tom, Paul Guerrero, Anthony Steed, Peter Wonka and Niloy J. Mitra (2018). 'FrankenGAN: Guided Detail Synthesis for Building Mass-Models Using Style-Synchonized GANs'. In: *ACM Transactions on Graphics*. DOI: 10.1145/3272127.3275065 (page 35).

Kelly, Tom and Peter Wonka (Apr. 2011). 'Interactive architectural modeling with procedural extrusions'. In: *ACM Transactions on Graphics* 30.2, pp. 1–15. DOI: 10.1145/1944846.1944854 (page 43).

Kember, Sarah (Mar. 1996). 'The shadow of the object: Photography and realism'. In: *Textual Practice* 10.1, pp. 145–163. DOI: 10.1080/09502369608582242 (page 24).

Kim, Joon-Seok, Hamdi Kavak and Andrew Crooks (Nov. 2018). 'Procedural city generation beyond game development'. In: *SIGSPATIAL Special* 10.2, pp. 34–41. DOI: 10.1145/3292390.3292397 (page 85).

Kim, Min H., Todd Alan Harvey, David S. Kittle, Holly Rushmeier, Julie Dorsey, Richard O. Prum and David J. Brady (July 2012). '3D imaging spectroscopy for measuring hyperspectral patterns on solid objects'. In: *ACM Transactions on Graphics* 31.4, pp. 1–11. DOI: 10.1145/2185520.2185534 (pages 22, 26).

Kingdom, Frederick A. A. and Nicolaas Prins (2016). *Psychophysics*. Elsevier Science. 346 pp. ISBN: 9780080993812 (pages 48, 54).

Kingma, Diederik P., Danilo J. Rezende, Shakir Mohamed and Max Welling (2014). 'Semi-Supervised Learning with Deep Generative Models'. In: arXiv: http://arxiv.org/abs/1406.5298v2. (Visited on 23rd June 2019) (page 35).

Knox, Paul and Steven Pinch (2010). *Urban Social Geography, An Introduction, 6th Edition*. United Kingdom: Pearson Education. ISBN: 9780273717638 (page 92).

Kolb, Craig, Don Mitchell and Pat Hanrahan (1995). 'A realistic camera model for computer graphics'. In: *Proceedings of SIGGRAPH '95*. ACM Press. DOI: 10.1145/218380.218463 (page 17).

Krecklau, Lars, Christopher Manthei and Leif Kobbelt (2012). 'Procedural Interpolation of Historical City Maps'. In: vol. 31. 2pt3, pp. 691–700. DOI: 10.1111/j.1467-8659.2012.03048.x (pages 47, 84, 113).

Křivánek, Jaroslav, Christophe Chevallier, Vladimir Koylazov, Ondřej Karlík, Henrik Wann Jensen and Thomas Ludwig (2018). 'Realistic rendering in architecture and product visualization'. In: *ACM SIGGRAPH 2018 Courses on - SIGGRAPH '18*. ACM Press. DOI: 10.1145/3214834.3214872 (page 30).

Křivánek, Jaroslav, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai and Wojciech Jarosz (July 2014). 'Unifying points, beams, and paths in volumetric light transport simulation'. In: *ACM Transactions on Graphics* 33.4, pp. 1–13. DOI: 10.1145/2601097.2601219 (page 32).

Kvrivishvili, Michael (2013). *2013 unofficial MBTA subway map by Michael Kvrivishvili*. URL: https://commons.wikimedia.org/wiki/File:2013_unofficial_MBTA_subway_map_by_Michael_Kvrivishvili.png (visited on 10th Feb. 2019) (page 12).

Lavoué, Guillaume, Mohamed Chaker Larabi and Libor Váša (Aug. 2015). 'On the Efficiency of Image Metrics for Evaluating the Visual Quality of 3D Models'. In: *IEEE Transactions on Visualization and Computer Graphics* 22.8, pp. 1987–1999. DOI: 10.1109/tvcg.2015.2480079 (page 57).

Lavoué, Guillaume and Rafał Mantiuk (Oct. 2014). 'Quality Assessment in Computer Graphics'. In: *Visual Signal Quality Assessment*. Springer International Publishing, pp. 243–286. DOI: 10.1007/978-3-319-10368-6_9 (pages 30, 31, 57).

Lechner, Thomas, Pin Ren, Ben Watson, Craig Brozefski and Uri Wilenski (2006). 'Procedural modeling of urban land use'. In: *SIGGRAPH '06 posters*. DOI: 10.1145/1179622.1179778 (page 91).

Lechner, Thomas, Ben Watson and Uri Wilensky (2003). 'Procedural city modeling'. In: *1st Midwestern Graphics Conference* (pages 84, 113).

Ledig, Christian, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang and Wenzhe Shi (July 2017). 'Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2017.19 (pages 33, 35).

Li, Yanghao, Naiyan Wang, Jiaying Liu and Xiaodi Hou (2017). 'Demystifying Neural Style Transfer'. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. DOI: 10.24963/ijcai.2017/310 (page 33).

Library of Congress (2018). *Mulberry Street, New York City*. URL: https://commons.wikimedia.org/wiki/File:Mulberry_Street_NYC_c1900_LOC_3g04637u_edit.jpg (visited on 6th July 2018) (page 25).

Lindstrom, Peter and Greg Turk (July 2000). 'Image-driven simplification'. In: *ACM Transactions on Graphics* 19.3, pp. 204–241. DOI: 10.1145/353981.353995 (page 58).

Longay, Steven, Adam Runions, Frédéric Boudon and Przemysław Prusinkiewicz (2012). *TreeSketch: Interactive Procedural Modeling of Trees on a Tablet*. DOI: 10.2312/sbm/sbm12/107-120 (page 40).

Luke, Sean (2015). *Essentials of Metaheuristics*. URL: https://cs.gmu.edu/~sean/book/metaheuristics/ (visited on 2nd July 2019) (page 47).

Lyu, S. and H. Farid (Feb. 2005). 'How realistic is photorealistic?' In: *IEEE Transactions on Signal Processing* 53.2, pp. 845–850. DOI: 10.1109/tsp.2004.839896 (page 18).

Magazine, Contract (2013). *"Witness"*. Facebook Album. URL: https://www.facebook.com/ContractMag/posts/522307757810060?__xts__%5B0%5D=68.ARAx-DQvCBr6v9Zwie9lVW_Ltvw1For0WVBjuI_COBTMDkOngbBi0xC5zdD0jq0kbsMEdF3csC5zsAmbrlT45S2qEMvhaiTnYyTJmGv1KEw0-ao_wrA9nA5rGrjOBh_yC0m534c6tnU0Y-96KkYdtMFPk_K5Vzt6WUfbBS-qsaocOiJ7uVdY&__tn__=-R (visited on 26th Sept. 2018) (page 41).

Mantiuk, Rafał, Kil Joong Kim, Allan G. Rempel and Wolfgang Heidrich (July 2011). 'HDR-VDP-2'. In: *ACM Transactions on Graphics* 30.4, p. 1. DOI: 10.1145/2010324.1964935 (page 30).

Mantiuk, Rafał, Anna Tomaszewska and Radosław Mantiuk (Aug. 2012). 'Comparison of Four Subjective Methods for Image Quality Assessment'. In: *Computer Graphics Forum* 31.8, pp. 2478–2491. DOI: 10.1111/j.1467-8659.2012.03188.x (pages 54, 115).

Martinović, Anđelo and Luc Van Gool (June 2013). 'Bayesian Grammar Learning for Inverse Procedural Modeling'. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2013.33 (pages 40, 50).

Mathias, Markus, Anđelo Martinović, Julien Weissenberg, Simon Haegler and Luc Van Gool (2011). 'Automatic Architectural Style Recognition'. In: *ISPRS* XXXVIII-5/W16, pp. 171–176. DOI: 10.5194/isprsarchives-XXXVIII-5-W16-171-2011 (page 61).

McClave, James T. and Frank H. Dietrich (1988). *Statistics, 4th Edition*. Dellen Publishing. ISBN: 0023792604 (page 48).

McNamara, Ann (Dec. 2001). 'Visual Perception in Realistic Image Synthesis'. In: *Computer Graphics Forum* 20.4, pp. 211–224. DOI: 10.1111/1467-8659.00550 (pages 23, 27).

McNamara, Ann (2005). 'Exploring perceptual equivalence between real and simulated imagery'. In: *Proceedings of the 2nd symposium on Appied perception in graphics and visualization - APGV '05*. ACM Press. DOI: 10.1145/1080402.1080425 (pages 23, 27).

McNamara, Ann (July 2006). 'Exploring visual and automatic measures of perceptual fidelity in real and simulated imagery'. In: *ACM Transactions on Applied Perception* 3.3, pp. 217–238. DOI: 10.1145/1166087.1166092 (pages 23, 27).

McNamara, Ann, Alan Chalmers, Tom Troscianko and Iain Gilchrist (2000). 'Comparing Real & Synthetic Scenes using Human Judgements of Lightness'. In: *Rendering Techniques 2000*. Springer, pp. 207–218. DOI: 10.1007/978-3-7091-6303-0_19 (pages 13, 23, 26, 27).

Mérillou, Stephane and Djamchid Ghazanfarpour (Apr. 2008). 'A survey of aging and weathering phenomena in computer graphics'. In: *Computers & Graphics* 32.2, pp. 159–174. DOI: 10.1016/j.cag.2008.01.003 (page 37).

Merrell, Paul (2007). 'Example-based model synthesis'. In: *Proceedings of the 2007 symposium on Interactive 3D graphics and games - I3D '07*. ACM Press. DOI: 10.1145/1230100.1230119 (pages 42, 50).

Merrell, Paul and Dinesh Manocha (June 2011). 'Model Synthesis: A General Procedural Modeling Algorithm'. In: *IEEE Transactions on Visualization and Computer Graphics* 17.6, pp. 715–728. DOI: 10.1109/tvcg.2010.112 (pages 38, 39).

Meyer, Gary W., Holly E. Rushmeier, Michael F. Cohen, Donald P. Greenberg and Kenneth E. Torrance (1986). 'An experimental evaluation of computer graphics imagery'. In: *ACM Transactions on Graphics* 5.1, pp. 30–50. DOI: 10.1145/7529.7920 (pages 27, 28, 29).

Mohammadi, Pedram, Abbas Ebrahimi-Moghadam and Shahram Shirani (2014). 'Subjective and Objective Quality Assessment of Image: A Survey'. In: arXiv: http://arxiv.org/abs/1406.7799v1. (Visited on 23rd June 2019) (page 30).

Moravec, Hans P. (Aug. 1981). '3D graphics and the wave theory'. In: *ACM SIGGRAPH Computer Graphics* 15.3, pp. 289–296. DOI: 10.1145/965161.806817 (page 17).

Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer and Luc Van Gool (2006a). ' Procedural Modeling of Buildings'. In: *ACM Transactions on Graphics* 25.3. DOI: 10.1145/1141911.1141931 (page 83).

Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer and Luc Van Gool (July 2006b). 'Procedural modeling of buildings'. In: *ACM Transactions on Graphics* 25.3, p. 614. DOI: 10.1145/1141911.1141931 (pages 38, 61, 62, 67).

Musbach, Avery, Gary W. Meyer, Fernando Reitich and Sang-Hyun Oh (Feb. 2013). 'Full Wave Modelling of Light Propagation and Reflection'. In: *Computer Graphics Forum* 32.6, pp. 24–37. DOI: 10.1111/cgf.12012 (page 17).

Musgrave, F. Kenton, Craig E. Kolb and Robert S. Mace (1989). 'The synthesis and rendering of eroded fractal terrains'. In: *Proceedings of SIGGRAPH '89*. DOI: 10.1145/74334.74337 (page 100).

Myszkowski, Karol (1998). 'The Visible Differences Predictor: applications to global illumination problems'. In: *Rendering Techniques '98*. Springer Vienna, pp. 223–236. DOI: 10.1007/978-3-7091-6453-2_21 (page 27).

Ng, Tian-Tsong, Shih-Fu Chang, Jessie Hsu and Martin Pepeljugoski (2004). *Columbia Photographic Images and Photorealistic Computer Graphics Dataset*. Tech. rep. 205-2004-5. ADVENT, Columbia University. URL: http://www.ee.columbia.edu/~dvmmweb/dvmm/downloads/PIM_PRCG_dataset/techreport_CG.html (visited on 23rd June 2019) (page 18).

Nielsen, Michael A. (2015). *Neural Networks and Deep Learning*. Determination Press. URL: http://neuralnetworksanddeeplearning.com/ (visited on 21st July 2018) (page 32).

Nishida, Gen, Ignacio Garcia-Dorado and Daniel G. Aliaga (Sept. 2015). 'Example-Driven Procedural Urban Roads'. In: *Computer Graphics Forum* 35.6, pp. 5–17. DOI: 10.1111/cgf.12728 (page 85).

Nishida, Gen, Ignacio Garcia-Dorado, Daniel Aliaga, Bedřich Beneš and Adrien Bousseau (2016). 'Interactive Sketching of Urban Procedural Models'. In: *ACM Transactions on Graphics*. DOI: 10.1145/2897824.2925951 (page 61).

Odena, Augustus (Apr. 2019). 'Open Questions about Generative Adversarial Networks'. In: *Distill* 4.4. DOI: 10.23915/distill.00018 (page 35).

Olsen, Luke, Faramarz F. Samavati, Mario Costa Sousa and Joaquim A. Jorge (Feb. 2009). 'Sketch-based modeling: A survey'. In: *Computers & Graphics* 33.1, pp. 85–103. DOI: 10.1016/j.cag.2008.09.013 (page 38).

Parberry, Ian (May 2015). 'Modeling Real-World Terrain with Exponentially Distributed Noise'. In: *Journal of Computer Graphics Techniques (JCGT)* 4.2, pp. 1–9. ISSN: 2331-7418. URL: http://jcgt.org/published/0004/02/01/ (page 40).

Parish, Yoav I. H. and Pascal Müller (2001). 'Procedural Modeling of Cities'. In: *Proceedings of SIGGRAPH '01*. DOI: 10.1145/383259.383292 (pages 83, 91, 97, 100, 102, 103, 113).

Park, Robert E. and Ernest W. Burgess (1984). *The City*. The University of Chicago Press. 250 pp. ISBN: 0226646114 (page 92).

Patow, Gustavo A. (Mar. 2012). 'User-Friendly Graph Editing for Procedural Modeling of Buildings'. In: *IEEE Computer Graphics and Applications* 32.2, pp. 66–75. DOI: 10.1109/mcg.2010.104 (page 43).

Perez-Ortiz, Maria and Rafał Mantiuk (2017). 'A practical guide and software for analysing pairwise comparison experiments'. In: arXiv: http://arxiv.org/abs/1712.03686v2. (Visited on 23rd June 2019) (page 54).

Pharr, Matt (July 2018). 'Guest Editor's Introduction'. In: *ACM Transactions on Graphics* 37.3, pp. 1–4. DOI: 10.1145/3212511 (page 30).

Pharr, Matt, Wenzel Jakob and Greg Humphreys (2016). *Physically based rendering: From theory to implementation*. Morgan Kaufmann. ISBN: 9780128006450 (pages 13, 17, 27).

Ponomarenko, Nikolay, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian and Marco Carli (2011). 'Modified image visual quality metrics for contrast change and mean shift accounting'. In: *2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*. IEEE, pp. 305–311 (page 30).

Powers, David Martin (2011). 'Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation'. In: *Journal of Machine Learning Technologies*. DOI: 10.9735/2229-3981 (page 53).

Prusinkiewicz, Przemysław and Aristid Lindenmayer (2014). *The algorithmic beauty of plants*. Springer Science & Business Media. URL: http://algorithmicbotany.org/papers/abop/abop.pdf (visited on 23rd Sept. 2018) (page 42).

Quan, Weize, Kai Wang, Dong-Ming Yan and Xiaopeng Zhang (Nov. 2018). 'Distinguishing Between Natural and Computer-Generated Images Using Convolutional Neural Networks'. In: *IEEE Transactions on Information Forensics and Security* 13.11, pp. 2772–2787. DOI: 10.1109/tifs.2018.2834147 (pages 34, 50, 115).

Raab, Matthias, Daniel Seibert and Alexander Keller (2008). 'Unbiased Global Illumination with Participating Media'. In: *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, pp. 591–605. DOI: 10.1007/978-3-540-74496-2_35 (page 29).

Rademacher, Pablo Mauricio (2002). 'Measuring the Perceived Visual Realism of Images'. PhD thesis. University of North Carolina at Chapel Hill (page 24).

Rademacher, Pablo Mauricio, Jed Lengyel, Edward Cutrell and Turner Whitted (2001). 'Measuring the perception of visual realism in images'. In: *Rendering Techniques*. Springer, pp. 235–247. DOI: 10.2312/EGWR/EGWR01/235-248 (pages 24, 60, 61, 62, 76).

Rahmouni, Nicolas, Vincent Nozick, Junichi Yamagishi and Isao Echizen (Dec. 2017). 'Distinguishing computer graphics from natural images using convolution neural networks'. In: *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. IEEE. DOI: 10.1109/wifs.2017.8267647 (pages 50, 115).

Ramanarayanan, Ganesh, Kavita Bala and James A Ferwerda (2008). 'Perception of complex aggregates'. In: *ACM Transactions on Graphics*. Vol. 27. ACM, p. 60. DOI: 10.1145/1399504.1360659 (pages 27, 61, 62, 81).

Ramanarayanan, Ganesh, James Ferwerda, Bruce Walter and Kavita Bala (2007). 'Visual equivalence: towards a new standard for image fidelity'. In: *ACM Transactions on Graphics*. Vol. 26. ACM, p. 76 (pages 14, 15, 26, 60, 61).

Ramasubramanian, Mahesh, Sumanta N. Pattanaik and Donald P. Greenberg (1999). 'A perceptually based physical error metric for realistic image synthesis'. In: *Proceedings of SIGGRAPH '99*. ACM Press. DOI: 10.1145/311535.311543 (pages 13, 27).

Reffye, Phillippe de, Claude Edelin, Jean Françon, Marc Jaeger and Claude Puech (Aug. 1988). 'Plant models faithful to botanical structure and development'. In: *ACM SIGGRAPH Computer Graphics* 22.4, pp. 151–158. DOI: 10.1145/378456.378505 (pages 41, 42).

Reinhard, Erik (2008). *Color Imaging*. A K Peters/CRC Press. 1074 pp. ISBN: 1568813449 (page 16).

Reinhard, Erik, Alexei A. Efros, Jan Kautz and Hans-Peter Seidel (2013). 'On Visual Realism of Synthesized Imagery'. In: *Proceedings of the IEEE* 101.9, pp. 1998–2007. DOI: 10.1109/JPROC.2013.2260711 (pages 6, 61, 64).

Render Legion (2019). *Corona Renderer*. URL: https://corona-renderer.com/ (visited on 11th Feb. 2019) (pages 54, 67).

Ritchie, Daniel, Sarah Jobalia and Anna Thomas (May 2018). 'Example-based Authoring of Procedural Modeling Programs with Structural and Continuous Variability'. In: *Computer Graphics Forum* 37.2, pp. 401–413. DOI: 10.1111/cgf.13371 (pages 42, 50).

Ritschel, Tobias, Carsten Dachsbacher, Thorsten Grosch and Jan Kautz (Feb. 2012). 'The State of the Art in Interactive Global Illumination'. In: *Computer Graphics Forum* 31.1, pp. 160–188. DOI: 10.1111/j.1467-8659.2012.02093.x (page 57).

Rogla, Otger, Nuria Pelechano and A. Gustavo Patow (2017). *Procedural Semantic Cities*. DOI: 10.2312/ceig.20171217 (page 85).

Rogowitz, Bernice E. and Holly E. Rushmeier (June 2001). 'Are image quality metrics adequate to evaluate the quality of geometric objects?' In: *Human Vision and Electronic Imaging VI*. Ed. by Bernice E. Rogowitz and Thrasyvoulos N. Pappas. SPIE. DOI: 10.1117/12.429504 (page 57).

Rosenheck, Uri (2006). *One of the first plans of Brasília*. URL: https://commons.wikimedia.org/wiki/File:Brasilia_-_Plan.JPG (visited on 11th Apr. 2019) (page 84).

Rushmeier, Holly E., Bernice E. Rogowitz and Christine Piatko (June 2000). 'Perceptual issues in substituting texture for geometry'. In: *Human Vision and Electronic Imaging V*. Ed. by Bernice E. Rogowitz and Thrasyvoulos N. Pappas. SPIE. DOI: 10.1117/12.387174 (page 17).

Rushmeier, Holly, Greg Ward, Christine Piatko, Phil Sanders and Bert Rust (1995). 'Comparing real and synthetic images: Some ideas about metrics'. In: *Rendering Techniques' 95*. Springer, pp. 82–91. DOI: 10.1007/978-3-7091-9430-0_9 (pages 28, 29).

Schwarz, Michael and Pascal Müller (July 2015). 'Advanced procedural modeling of architecture'. In: *ACM Transactions on Graphics* 34.4, 107:1–107:12. DOI: 10.1145/2766956 (pages 47, 61, 62).

Shaker, Noor, Julian Togelius and Mark J. Nelson (2016). *Procedural Content Generation in Games*. Springer-Verlag GmbH. ISBN: 3319427148 (page 45).

Shannon, C.E. (Jan. 1949). 'Communication in the Presence of Noise'. In: *Proceedings of the IRE* 37.1, pp. 10–21. DOI: 10.1109/jrproc.1949.232969 (page 26).

Sharma, Abhishek, Oliver Grau and Mario Fritz (2016). 'VConv-DAE: Deep Volumetric Shape Learning Without Object Labels'. In: arXiv: http://arxiv.org/abs/1604.03755v3. (Visited on 23rd June 2019) (page 35).

Sharma, Gaurav and Raja Bala (2002). *Digital color imaging handbook*. CRC press. ISBN: 978-0849309007 (page 31).

Šik, Martin and Jaroslav Křivánek (2018). 'Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation'. In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: 10.1109/tvcg.2018.2880455 (page 32).

Silva, Pedro Brandão, Pascal Müller, Rafael Bidarra and António Coelho (2013). 'Node-based shape grammar representation and editing'. In: *Proceedings of the Workshop on Procedural Content Generation in Games (PCG'13)* (page 43).

Sjoberg, Gideon (1960). *The Preindustrial City: Past and Present*. New York: The Free Press. DOI: 10.1177/000271626234100172 (page 92).

Smelik, Ruben M., Klaas Jan De Kraker, Tim Tutenel, Rafael Bidarra and Saskia A. Groenewegen (2009). 'A survey of procedural methods for terrain modelling'. In: *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, pp. 25–34 (page 37).

Smelik, Ruben M., Tim Tutenel, Rafael Bidarra and Bedřich Beneš (Jan. 2014). 'A Survey on Procedural Modelling for Virtual Worlds'. In: *Computer Graphics Forum* 33.6, pp. 31–50. DOI: 10.1111/cgf.12276 (pages 37, 38, 39, 40, 41, 43, 49).

Smelik, Ruben M., Tim Tutenel, Klaas Jan de de Kraker and Rafael Bidarra (Apr. 2011). 'A declarative approach to procedural modeling of virtual worlds'. In: *Computers & Graphics* 35.2, pp. 352–363. DOI: 10.1016/j.cag.2010.11.011 (pages 40, 43).

Smelik, Ruben M., Tim Tutenel, Klaas Jan de Kraker and Rafael Bidarra (2010). 'Integrating procedural generation and manual editing of virtual worlds'. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games - PCGames '10*. ACM Press. DOI: 10.1145/1814256.1814258 (page 40).

Smith, Alvy Ray (1984). 'Plants, fractals, and formal languages'. In: *ACM SIGGRAPH Computer Graphics* 18.3, pp. 1–10. DOI: 10.1145/800031.808571 (page 38).

Soja, Edward W. (2000). *Postmetropolis, Critical Studies of Cities and Regions*. Malden, MA, USA: Blackwell Publishers. ISBN: 1577180011 (page 92).

Stam, Jos (1999). 'Stable fluids'. In: *Proceedings of SIGGRAPH '99*. ACM Press. DOI: 10.1145/311535.311548 (page 41).

Šťava, Ondřej, Bedřich Beneš, Radomír Měch, Daniel G. Aliaga and Peter Krištof (May 2010). 'Inverse Procedural Modeling by Automatic Generation of L-systems'. In: *Computer Graphics Forum* 29.2, pp. 665–674. DOI: 10.1111/j.1467-8659.2009.01636.x (pages 40, 50).

Stiny, George et al. (1980). 'Introduction to shape and shape grammars'. In: *Environment and planning B* 7.3, pp. 343–351. DOI: 10.1068/b070343 (page 61).

Striova, Jana, Chiara Ruberto, Marco Barucci, Jan Blažek, Diane Kunzelman, Alice Dal Fovo, Enrico Pampaloni and Raffaella Fontana (May 2018). 'Spectral Imaging and Archival Data in Analysing Madonna of the Rabbit Paintings by Manet and Titian'. In: *Angewandte Chemie International Edition* 57.25, pp. 7408–7412. DOI: 10.1002/anie.201800624 (pages 22, 26).

Summerville, Adam, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen and Julian Togelius (2017). 'Procedural Content Generation via Machine Learning (PCGML)'. In: *CoRR* abs/1702.00539. eprint: 1702.00539. URL: http://arxiv.org/abs/1702.00539 (visited on 23rd June 2019) (page 45).

Sun, Yinlong, F. David Fracchia, Mark S. Drew and Thomas W. Calvert (Oct. 2001). 'A spectrally based framework for realistic image synthesis'. In: *The Visual Computer* 17.7, pp. 429–444. DOI: 10.1007/s003710100116 (page 17).

Szegedy, Christian, Sergey Ioffe and Vincent Vanhoucke (2016). 'Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning'. In: *CoRR* abs/1602.07261. URL: http://arxiv.org/abs/1602.07261 (page 81).

Talton, Jerry, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman and Radomír Měch (2012). 'Learning Design Patterns with Bayesian Grammar Induction'. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, pp. 63–74. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380127 (pages 42, 50, 61).

Tatarchenko, Maxim, Alexey Dosovitskiy and Thomas Brox (2017). 'Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs'. In: arXiv: http://arxiv.org/abs/1703.09438v3. (Visited on 23rd June 2019) (page 35).

This Person Does Not Exist (2019). URL: https://thispersondoesnotexist.com (visited on 9th Mar. 2019) (page 35).

Togelius, Julian, Tróndur Justinussen and Anders Hartzen (2012). 'Compositional procedural content generation'. In: *Proceedings of the The third workshop on Procedural Content Generation in Games - PCG'12*. ACM Press. DOI: 10.1145/2538528.2538541 (pages 45, 47).

Togelius, Julian, Emil Kastbjerg, David Schedl and Georgios N. Yannakakis (2011a). 'What is procedural content generation?' In: *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games - PCGames '11*. ACM Press. DOI: 10.1145/2000919.2000922 (pages 39, 45).

Togelius, Julian, Georgios N. Yannakakis, Kenneth O. Stanley and Cameron Browne (Sept. 2011b). 'Search-Based Procedural Content Generation: A Taxonomy and Survey'. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3, pp. 172–186. DOI: 10.1109/tciaig.2011.2148116 (pages 38, 45, 46, 47).

Tolley, Rodney and Brian Turton (1995). *Transport Systems, Policy and Planning: A Geographical Approach*. Longman (page 92).

Ulbricht, Christiane, Alexander Wilkie and Werner Purgathofer (June 2006). 'Verification of Physically Based Rendering Algorithms'. In: *Computer Graphics Forum* 25.2, pp. 237–255. DOI: 10.1111/j.1467-8659.2006.00938.x (pages 28, 29).

Vanegas, C. A., I. Garcia-Dorado, D. Aliaga, B. Benes and P. Waddell (2012a). 'Inverse Design of Urban Procedural Models'. In: *ACM Transactions on Graphics* (page 113).

Vanegas, C., T. Kelly, B. Weber, J. Halatsch, D. Aliaga and P. Muller (2012b). 'Procedural Generation of Parcels in Urban Modeling'. In: *EG 2012*. Eurographics Association (page 84).

Vanegas, Carlos A., Daniel G. Aliaga, Bedřich Beneš and Paul A. Waddell (2009). 'Interactive design of urban spaces using geometrical and behavioral modeling'. In: *SIGGRAPH Asia '09*. DOI: 10.1145/1618452.1618457 (pages 83, 97, 102, 103, 113).

Vanegas, Carlos A., Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedřich Beneš and Paul Waddell (Nov. 2012c). 'Inverse design of urban procedural models'. In: *ACM Transactions on Graphics* 31.6, p. 1. DOI: 10.1145/2366145.2366187 (pages 40, 50).

Vanhoey, Kenneth, Basile Sauvage, Pierre Kraemer and Guillaume Lavoué (Oct. 2017). 'Visual Quality Assessment of 3D Models'. In: *ACM Transactions on Applied Perception* 15.1, pp. 1–18. DOI: 10.1145/3129505 (page 57).

Veach, Eric and Leonidas J. Guibas (1997). 'Metropolis light transport'. In: *Proceedings of SIGGRAPH '97*. ACM Press. DOI: 10.1145/258734.258775 (page 31).

Vives Piqueres, Jaime (1997). *The Peristence of Ignorance – Fountain*. URL: http://www.ignorancia.org/index.php/galleries/archaic-images/fountain/ (visited on 10th Feb. 2019) (page 18).

Vives Piqueres, Jaime (1999). *The Peristence of Ignorance – Bonsais*. URL: http://www.ignorancia.org/index.php/galleries/archaic-images/bonsais/ (visited on 10th Feb. 2019) (page 18).

Volevich, Valdimir, Karol Myszkowski, Andrei Khodulev and Edward A. Kopylov (Apr. 2000). 'Using the visual differences predictor to improve performance of progressive global illumination computation'. In: *ACM Transactions on Graphics* 19.2, pp. 122–161. DOI: 10.1145/343593.343611 (page 27).

Volker, Lars (2008). *Route Planning in Road Networks with Turn Costs*. Tech. rep. Universität Karlsruhe. URL: http://lekv.de/pub/lv-turns-2008.pdf (visited on 23rd June 2019) (page 96).

Wallraven, Christian, Roland Fleming, Douglas Cunningham, Jaume Rigau, Miquel Feixas and Mateu Sbert (2009). 'Categorizing art: Comparing humans and computers'. In: *Computers & Graphics* 33.4, pp. 484–495. DOI: 10.1016/j.cag.2009.04.003 (page 64).

Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh and Eero P. Simoncelli (2004). 'Image quality assessment: from error visibility to structural similarity'. In: *IEEE transactions on image processing* 13.4, pp. 600–612. DOI: 10.1109/TIP.2003.819861 (pages 23, 27, 30, 33, 61).

Ward, Greg (1990). *Notes on SIGGRAPH '90 course on Radiosity, partially recreated from a troff document.* personal communication (pages 28, 29).

Watson, Benjamin, Alinda Friedman and Aaron McGaffey (2001). 'Measuring and predicting visual fidelity'. In: *Proceedings of SIGGRAPH '01*. ACM Press. DOI: 10.1145/383259.383283 (page 57).

Weber, Basil, Pascal Müller, Peter Wonka and Markus Gross (2009). 'Interactive Geometric Simulation of 4D Cities'. In: *Computer Graphics Forum* 28.1. DOI: 10.1111/j.1467-8659.2009.01387.x (pages 83, 84, 91, 97, 100, 102, 103, 113).

Weidlich, Andrea and Alexander Wilkie (2007). 'Arbitrarily layered micro-facet surfaces'. In: *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07*. ACM Press. DOI: 10.1145/1321261.1321292 (page 10).

Weidlich, Andrea and Alexander Wilkie (Mar. 2008). 'Realistic rendering of birefringency in uniaxial crystals'. In: *ACM Transactions on Graphics* 27.1, pp. 1–12. DOI: 10.1145/1330511.1330517 (page 17).

Weidlich, Andrea and Alexander Wilkie (June 2009). 'Anomalous Dispersion in Predictive Rendering'. In: *Computer Graphics Forum* 28.4, pp. 1065–1072. DOI: 10.1111/j.1467-8659.2009.01483.x (page 17).

Weise, Thomas (2009). *Global Optimization Algorithms, Theory and Application*. URL: http://www.it-weise.de/projects/book.pdf (visited on 2nd July 2019) (page 47).

Whittle, Joss, Mark W. Jones and Rafał Mantiuk (May 2017). 'Analysis of reported error in Monte Carlo rendered images'. In: *The Visual Computer* 33.6-8, pp. 705–713. DOI: 10.1007/s00371-017-1384-7 (page 29).

Wilkie, Alexander, Robert F. Tobler and Werner Purgathofer (2001). 'Combined Rendering of Polarization and Fluorescence Effects'. In: *Eurographics*. Springer Vienna, pp. 197–204. DOI: 10.1007/978-3-7091-6242-2_18 (page 17).

Wilkie, Alexander, Andrea Weidlich, Marcus Magnor and Alan Chalmers (2009). 'Predictive rendering'. In: *ACM SIGGRAPH ASIA 2009 Courses*. ACM, p. 12. DOI: 10.1145/1665817.1665829 (page 17).

William van der Gouwen (1684). *Een Walvisch*. URL: https://commons.wikimedia.org/wiki/File:Im_Februar_1598_an_der_holl%C3%A4ndischen_K%C3%BCste_gestrandeter_Walfisch.jpg (visited on 10th Feb. 2019) (page 20).

Williams, Benjamin and Christopher J. Headleand (Sept. 2017). 'A Time-Line Approach for the Generation of Simulated Settlements'. In: *2017 International Conference on Cyberworlds (CW)*. IEEE. DOI: 10.1109/cw.2017.32 (page 85).

Woestijne, Joran Van De, Roald Frederickx, Niels Billen and Philip Dutré (2017). *Temporal Coherence for Metropolis Light Transport*. DOI: 10.2312/sre.20171194 (page 32).

Wonka, Peter, Michael Wimmer, François Sillion and William Ribarsky (2003). *Instant architecture*. Vol. 22. ACM. DOI: 10.1145/1201775.882324 (page 61).

Wu, Jiajun, Chengkai Zhang, Tianfan Xue, William T. Freeman and Joshua B. Tenenbaum (2016). 'Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling'. In: arXiv: http://arxiv.org/abs/1610.07584v2. (Visited on 23rd June 2019) (page 35).

Wu, Ruoyu, Xiaolong Li and Bin Yang (Sept. 2011). 'Identifying computer generated graphics VIA histogram features'. In: *2011 18th IEEE International Conference on Image Processing*. IEEE. DOI: 10.1109/icip.2011.6115849 (pages 50, 115).

Yang, Yong-Liang, Jun Wang, Etienne Vouga and Peter Wonka (2013). 'Urban Pattern: Layout Design by Hierarchical Domain Splitting'. In: *ACM Transactions on Graphics* 32 (6). DOI: 10.1145/2508363.2508405 (pages 84, 113).

Yann Forget (2012). *Garden of Nebamun*. URL: https://commons.wikimedia.org/wiki/File:Le_Jardin_de_N%C3%A9bamoun.jpg (visited on 10th Feb. 2019) (page 20).

Yosinski, Jason, Jeff Clune, Yoshua Bengio and Hod Lipson (2014). 'How transferable are features in deep neural networks?' In: *Advances in Neural Information Processing Systems 27, pages 3320-3328. Dec. 2014*. arXiv: 1411.1792v1. (Visited on 23rd June 2019) (page 33).

Yosinski, Jason, Jeff Clune, Anh Nguyen, Thomas Fuchs and Hod Lipson (2015). 'Understanding Neural Networks Through Deep Visualization'. In: arXiv: 1506.06579v1. (Visited on 23rd June 2019) (page 33).

Yu, Qizhi and Anthony Steed (2012). 'Example-based Road Network Synthesis'. In: *EG 2012 - Short Papers*. Eurographics Association. DOI: 10.2312/conf/EG2012/short/053-056 (pages 84, 91, 113).

Zeiler, Matthew D. and Rob Fergus (2013). 'Visualizing and Understanding Convolutional Networks'. In: arXiv: 1311.2901v3. (Visited on 23rd June 2019) (page 33).

List of Figures

2.1 Geometry with a microscopic level of detail 9
2.2 Car dashboard 9
a Fully rendered dashboard 9
b Wireframe dashboard 9
2.3 Impossible bike designs 11
a Bike number 7 11
b Bike number 21 11
2.4 Elements of realism 11
a All three elements of realism 11
b Only basic material appearance 11
c Only a basic rendering algorithm 11
2.5 Illustration of visual realism and functional realism 12
a Photograph 12
b Topological map 12
2.6 Flight simulator 14
2.7 Visual equivalence 15
2.8 Representations of realism 16
2.9 Images like those previously considered photorealistic 18
a Bonsais 18
b Fountain 18
2.10 Images from games that were considered highly realistic 19
a Pete Rose Pennant Fever 19
b Wolfenstein 3D 19
c The Journeyman Project 19
2.11 Realism in different historical epochs 20
a Garden of Nebamoun 20
b Whale washed ashore 20
2.12 Examples of measurement and common photographs 22
a Cotinga maynana 22
b Street at night 22
2.13 Rademacher's experiment 24
a Soft shadow 24
b Hard shadow 24
c Rough surface 24
d Smooth surface 24
2.14 Historical color photographs 25
a Mulberry Street 25
b Garden of the Gods Park 25

2.15 Measurement-photorealism equality . . . 26
2.16 Cornell Box . . . 28
a Cornell Box poster . . . 28
b Physical Cornell Box . . . 28
2.17 The importance of scattering . . . 30
a With scattering . . . 30
b Without scattering . . . 30
2.18 Neural network's representation of a cat . . . 33
2.19 GAN architecture . . . 34
2.20 GAN-generated faces . . . 35

3.1 Examples of procedural models . . . 38
a Terrain heightmap . . . 38
b Weathering . . . 38
c Building . . . 38
3.2 Examples of procedural content in No Man's Sky . . . 39
a Dinosaur-like creatures . . . 39
b Fish-like creatures . . . 39
3.3 Tree sketching . . . 40
3.4 Farnsworth House with a black maple tree . . . 41
3.5 Examples of procedural rules . . . 42
a Lilac flower head L-system . . . 42
b Database of airplanes . . . 42
3.6 Data flow for a procedural model . . . 43
3.7 Representational models . . . 44
a City road network . . . 44
b Building volume . . . 44
c Species in biotope . . . 44
3.8 Search-based model generation . . . 46
3.9 Inaccessible balconies in procedurally generated buildings . . . 46
3.10 Computational paths of a procedural model . . . 49
3.11 The space of all models . . . 51
3.12 Parisian buildings sharing the same mansard windows . . . 56

4.1 Shared features and lack of variation in procedural models . . . 63
4.2 Classification screen . . . 63
4.3 Different blur levels applied to a Paris dataset image . . . 65
4.4 Sample images from our datasets . . . 66
4.5 Computer generated image preparation workflow . . . 68
4.6 Photograph preparation workflow . . . 68
4.7 Relationship between blur level and accuracy . . . 72
a Average accuracy per dataset . . . 72
b Average accuracy per class . . . 72
4.8 Accuracies and class ratios per dataset . . . 73
a Accuracy of each dataset across blurs . . . 73
b Ratio of classified as photographs and computer generated 73
4.9 Three-way interaction between original class, dataset, blur level . 73
4.10 Frequently recognized computer generated images with artifacts . 75
4.11 Procedurally generated images with implausible structures . . . 76

a Inaccessible balcony . . . 76
b Window in support beam . . . 76
c Wood in concrete . . . 76
d Deformed roof . . . 76
4.12 Most confounding buildings at the highest blur . . . 77
4.13 Images often classified as computer generated . . . 77

5.1 A 1915 motorist map of the Los Angeles area . . . 83
5.2 Examples of planned cities . . . 84
a Garden city . . . 84
b Brasilia . . . 84
5.3 Typical progression of the algorithm . . . 87
5.4 Map $\boldsymbol{m}$ with a river . . . 88
5.5 Pathfinding related illustrations . . . 89
a Schematic depiction of the pathfinding graph $\mathcal{G}$ . . . 89
b Vertex connections on a lattice . . . 89
5.6 Minor road patterns in Avignon, France . . . 90
5.7 Progression of settled areas within the Chicago area . . . 92
5.8 Traffic simulation overview . . . 93
a Multiple trade routes between cities . . . 93
b Alternative trade route over existing major roads . . . 93
c New trade route making use of new major roads . . . 93
5.9 Illustration of the turn angle penalty function . . . 93
a Turn angle penalty function . . . 93
b SmoothStep function . . . 93
5.10 Built-up area construction . . . 95
5.11 Pathfinding graph construction . . . 97
5.12 Initial road networks . . . 98
5.13 User steerability . . . 99
5.14 Various results of our algorithm . . . 100
a Eccentric major road structure . . . 100
b Normal city . . . 100
c City with a more classical major road structure . . . 100
d Coastal city . . . 100
e City with suburbs . . . 100
f City in a mountainous region . . . 100
5.15 Evolution of a lake-side city . . . 101
5.16 Growth of result from Figure 5.14a . . . 105
5.17 Growth of result from Figure 5.14b . . . 105
5.18 Growth of result from Figure 5.14c . . . 106
5.19 Growth of result from Figure 5.14d . . . 106
5.20 Growth of result from Figure 5.14e . . . 107
5.21 Growth of result from Figure 5.14f . . . 107

List of Tables

3.1 Confusion matrix of the space of all procedural models 51

4.1 Subjective pre-experiment assessment of procedural rule quality . 66
4.2 Reported experience in related fields 70
4.3 Contingency tables for 1 px blur 72

5.1 Average running times 101
5.2 Values of the time independent $\boldsymbol{O}_1, \ldots, \boldsymbol{O}_4$ scalar constants 108
5.3 Per-city parameter values 108
5.4 Historical context parameters for presented results 109
5.5 Notation for historical context 110
5.6 Per-city input notation 110
5.7 Non-input symbols 112
5.8 Comparison of methods that support road network synthesis . . . 113