

Billboarded Cutaway Visualization for Subsurface Urban Sewer Networks

Flood Simulation embedded in Unity3D

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Manuel Matusich

Matrikelnummer 0806024

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Ph.D. Ivan Viola, TU Wien, Vienna, Austria

Mitwirkung: Dipl.-Ing. Dr.techn. Jürgen Waser, VRVis Research Center, Vienna, Austria

Daniel Cornel, MSc, VRVis Research Center, Vienna, Austria

Wien, 2. Mai 2017

Manuel Matusich

Ivan Viola

Billboarded Cutaway Visualization for Subsurface Urban Sewer Networks

Flood Simulation embedded in Unity3D

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Manuel Matusich

Registration Number 0806024

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Ph.D. Ivan Viola, TU Wien, Vienna, Austria

Assistance: Dipl.-Ing. Dr.techn. Jürgen Waser, VRVis Research Center, Vienna, Austria

Daniel Cornel, MSc, VRVis Research Center, Vienna, Austria

Vienna, 2nd May, 2017

Manuel Matusich

Ivan Viola

Erklärung zur Verfassung der Arbeit

Manuel Matusich
Alte Gasse 20/2/9, 2552 Hirtenberg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. Mai 2017

Manuel Matusich

Acknowledgements

I want to express my gratitude to my supervisor Ivan Viola for the great support and feedback during the work on this thesis. Secondly I want to thank the visualization group of the Institute of Computer Graphics and Algorithms at the University of Vienna for their interest and feedback. I also want to thank Jürgen Waser and Daniel Cornel for the excellent cooperation and their feedback. Especially I want to thank my girlfriend for her love and support.

Kurzfassung

Das Visualisieren von Hochwassersimulation in Echtzeit ist ein notwendiger Schritt, um bestens für immer häufiger vorkommende Hochwasser vorbereitet zu sein. Eine wichtige Komponente von Hochwassersimulationen ist es, den Einfluss des Hochwassers auf das Kanalnetz der betroffenen Stadt zu berechnen. Da für Hochwassersimulationen nur sehr primitive Datenstrukturen des Kanalnetzes benötigt werden, stellt diese Arbeit eine innovative und effiziente Methode vor, um Kanalrohre und deren Wasserdurchfluss, auf Basis solcher Daten in Echtzeit zu visualisieren. Die zugrundeliegende Datenstruktur basiert dabei auf Knotenpunkten die lediglich eine Linie beschreiben. Mit diesen Daten berechnet der Geometrie Shader die Eckpunkte von Vierecken die angeordnet werden, um ein Rohr darzustellen. Weiters wird ein künstliches Beleuchtungsmodell berechnet, welches den Eindruck erweckt, als ob die Vierecke eine zylindrische Geometrie hätten. Um den Wasserdurchfluss zu sehen, wird das Rohr mit Hilfe einer Proxygeometrie aufgeschnitten. Dabei wird die Proxygeometrie benutzt, um im Stencil Buffer Werte zu setzen, welche definieren, ob das Beleuchtungsmodell und die Farbe der Vierecke, für das Äußere oder Innere der Kanalrohre angenommen werden muss.

Zusätzlich wird eine weitere Methode vorgestellt, mit der es möglich ist den Boden aufzuschneiden, welcher das Kanalnetz überdeckt. Dafür wird eine Distanztransformation der zweidimensionalen Repräsentation des Kanalnetzes generiert. Mit dem resultierenden Distanzfeld kann abgefragt werden, wie nahe ein Pixel des Bodens an einem Kanalrohr liegt, und entschieden, ob es verworfen werden soll. Zusätzlich wird dem aufgeschnitten Boden analytisch im Bild-Raum eine künstliche Schnittfläche hinzugefügt.

Die Methoden werden detailliert beschrieben und deren Implementierung mittels Unity3D demonstriert. Die Ergebnisse werden mit gängigen, wissenschaftlich belegten, Konventionen für solche Anwendungen verglichen und damit evaluiert. Zusätzlich wurden Visualisierungsexperten mit der Implementierung vertraut gemacht und deren Feedback trägt ebenfalls zur Evaluierung bei. Die Ergebnisse zeigen, dass die Methode sehr effizient ist und wenig Speicherplatz beansprucht im Vergleich zu herkömmlichen Rendering-Methoden mittels Polygonnetzen. Zusätzlich erfüllt die Methode beinahe alle Konventionen und begeisterte die Visualisierungsexperten.

Abstract

Visualizing the results of a flood simulation interactively in real-time has become very important due to the increasing number of flood hazards in the recent past. An important part of a flood simulation is the impact on the sewer system of the flooded city. For the simulation itself only a very primitive data structure of a sewer system is required. Therefore this thesis proposes an innovative and efficient technique to render sewage pipes and to visualize the water flow inside of them. With a simple underlying data structure of node positions, billboarded sewage pipes are constructed in the geometry shader. With a proxy geometry the stencil buffer is utilized to distinguish outside and inside of the pipe and the billboards are drawn accordingly.

In addition another technique is introduced to cut-open the occluding terrain to expose the sewage pipes to the viewer. Therefore a two dimensional distance transform of the sewer network is generated. The created distance field is used to discard fragments of the terrain, which are relatively close to a pipe. Further an artificial thickness is added to the cut-open terrain analytically in image-space.

The techniques are presented in detail and their implementation is demonstrated in Unity3D. It is evaluated by comparing the result to multiple conventions for computer generated cutaways. Additionally the feedback from visualization experts and the technical performance is presented. The results show that billboarded and cut-open tubular geometry can be rendered fast, efficient and with a low memory footprint. Furthermore the technique fulfills many of the design principles for cutaways and got positive feedback from visualization experts.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Background & Motivation	1
1.2 Specification of Provided Data	2
1.3 Goal of the Thesis	3
1.4 Structure of the Thesis	6
2 State of the Art	7
2.1 Flood Risk Management	7
2.2 Occlusion Management Techniques	7
2.3 Billboarding	10
3 Methodology	13
3.1 Terrain Cutaway	13
3.2 Drawing the Pipe	16
3.3 Cutting the Pipe	19
3.4 Cutting the Shaft	24
4 Implementation	25
5 Demonstration	27
6 Critical Reflection	33
6.1 Evaluation	33
6.2 Performance	34
6.3 Feedback from Domain Experts	34
6.4 Discussion of Open Issues	36
7 Summary and Future Work	39
	xiii

Introduction

1.1 Background & Motivation

Scientific visualization and illustration of three-dimensional objects has the purpose to enable scientists and domain experts to give better insight to their data. Moreover visualizations in the domain of flood risk management enhance responsible authorities to better understand what would naturally be hidden. This is especially useful when dealing with floodwater that affects the subsurface sewer system. In urban areas and major cities it is advisable to be prepared for floodwater if any rivers are located in the surrounding area. To reduce potential harm, flood risk management should evolve strategies on how to deal with various cases of different floodwater scenarios. Recent events of flood disasters show how much they can damage the surrounding area. The European Union has established a solidarity fund of up to EUR 500 million per year that is reserved for natural disasters in member states [Eur12]. On the 26 of January 2017 the European Commission allocated EUR 60.3 million to support the United Kingdom in recovery actions for the floods in 11 regions [Eur17]. Authorities in the United Kingdoms estimate a total direct damage of EUR 2 412 million due to floods caused by storms and heavy rainfalls. Investing research into advanced flood simulation is therefore reasonable. Luckily flood simulations are already computer assisted. Simulating emergency cases oftentimes produces vast amounts of data. The visualization of this data is essential to visually conceive such simulated disaster events. In major cities the subsurface infrastructure has an impact on how disaster flood events pan out. Therefore it is important for flood simulations to calculate how floodwater will interact with the sewer system of urban areas. To visualize such generated data in a non-photo-realistic environment, it is necessary to reveal the sewer system. Therefore a strategy needs to be developed to somehow remove the occluding terrain and to expose the water flow in the sewer systems underneath.

The VRVis Research Center, located in Vienna, Austria, is working on a software - Visdom - that combines visualization, simulation and analysis techniques to assist decision making

[WFRB11]. This software is enhanced with a fast flood simulation technique that works on modern graphics hardware. Visdom lets the user test out strategies on how to deal with floodwater scenarios interactively. For example it enables the user to place sandbags in certain areas to change the outcome of the flood simulation. This feature supports responsible authorities to prepare their flood defense strategies more accordingly. A mobile device with remote access to Visdom enhances the user with on-site decision making support [SRCW13]. This enables the user to sketch changes on the touch-sensitive device and test out multiple strategies to get useful feedback from the interactive flood simulation. In Visdom the water levels and amounts of discharge of the sewer system are calculated and interact with the simulation. Unfortunately the sewer system is not made visible to the viewer. Any objects underneath the terrain would not be visible and therefore it was unnecessary for the subsurface infrastructure to be rendered yet. For this reason this thesis will develop a applicable solution to extend Visdom with an innovative visualization technique for the subsurface urban infrastructure.

The structure of the provided data impacted our decisions and the course of how to develop a new visualization technique for the subsurface infrastructure. Therefore we first need to take a closer look on the provided data to define the goals of this thesis particularly.

1.2 Specification of Provided Data

VRVis provided various flood simulation data, and city model data, that was exported from Visdom. All the data appertains to Worringen, a district of Cologne. Cologne is a large city in the north of Germany and is located on both sides of the river Rhine. Cologne has a population density of 2500 inhabitants per km^2 and was flooded in 1993 and 1995 [Bir08]. Since floodwater is an endangerment to the citizen of Cologne, authorities trouble to design the city to be prepared for a 100 years flood event. Statistically a 100 years flood event should in general have a recurrence interval of at least 100 years. Some parts of Cologne are even prepared to withstand a 500 years flood event.

The provided data can be separated into two groups by their data structure. Surrounding objects that contribute to the overall impression of the city and support the user to orient in the environment are considered the first group and labeled as city model. All the static surroundings of the city model have been exported and provided as 3D-Models in the OBJ format. The city model includes terrain, bridges, vegetation and buildings. Each of them has been packed into one OBJ file, respectively. The second group of data was provided as binary files. The term sewer system is considered to include the structure of the sewer network and flood simulation data. The sewer network can be separated into two types of objects, pipes and shafts. Both are represented by the same data structure. Each pipe is defined by a list of three-dimensional nodes. The first node defines the starting position in world space coordinates and each successive node defines the spatial direction. Additionally for each pipe the radius is defined. In general the start and end points of pipes are connected to sewage shafts. Sewage shafts are

defined by two three-dimensional nodes in world space coordinates. These two nodes define the upper and lower point of the shaft. Since each sewage shaft and pipe has been adequately supplied with an identification number, the binary files contained information of their mutual dependencies. For each pipe it is defined, which sewage shaft(s) it is connected to. Conversely for each shaft it is defined, which pipe(s) it is connected to. Further on to augment the visual experience VRVis provided binary data that contains color-code information for the water inside of the sewer network. Based on water depth and discharge velocity the water is color-coded to convey this information. Additionally the water levels of the sewer system have been provided as binary files. Flood simulation data containing 721 consecutive water levels, with a time step of 10 seconds, has been received as well. This made it possible to facilitate this project with a timeline.

Important to note is that polygon meshes for the sewer network or the water inside of it, are nonexistent and would have to be generated somehow. Now that the data structure is set out, we can define the goals of this thesis specifically.

1.3 Goal of the Thesis

The goal of this thesis is to remove occluding terrain and to look inside the sewer network. The object of interest is the water flow in the sewer system. To get some inspiration on how to visualize the sewer system illustrative artwork regarding subsurface infrastructure was reviewed. Most of the sighted artwork was drawn computer assisted and therefore rather modern. All of them had the purpose to clearly convey the subsurface infrastructure. We reviewed several architectural visualizations that dealt with visualization of sewer systems. All these architectural visualizations depicted the subsurface infrastructure in a similar manner. In general a box was cut out of the terrain just deep enough to expose objects of interest. Another finding is that the illustrators always made use of the new cut surface which has been created by the cutaway. Each exposed element casted a shadow onto this artificial surface level or the artificial walls. The shadows are a great help to understand the spatial relationship of the exposed objects. Another feature that assists the viewer to mentally auto-complete the cut open parts of the scene are horizontal walls of the cut-shape. Pipes, for example that emerge out of this walls, give a spatial clue to the user and how they are arranged in 3D space.

Another architectural visualization that has been sighted was about subway networks. The depicted scene in Figure 1.1 was provided by the visualization experts of Office Le Nomade. Additionally the two founders, Clemens Kraigher and Markus Stöger, provided insight on the development process of this visualization. Working together on architectural visualizations, they can look back on 18 years of experience. Important to them was to include well known buildings on the surface as a point of reference. Regarding the cutaways Kraigher and Stöger claimed that they chose a very simple structure as a cutaway, a simple box that is cut out of the subway hoistways, because it is easier to understand for the viewer. This way the viewer can easily comprehend the spatial relationship. Assisting this purpose, the terrain is partly visualized by a plane

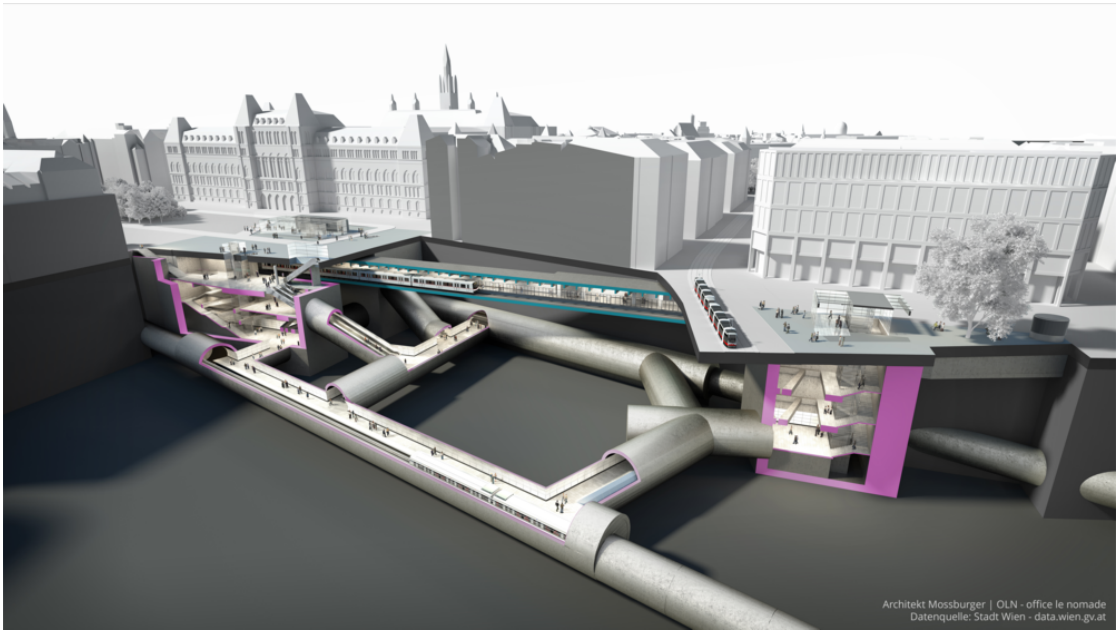


Figure 1.1: Architectural visualization of the subway station *U5 Rathaus* in Vienna provided by Office Le Nomade [OLN].

with a rather small artificial thickness. Shadows of this plane drop on structures below and help to understand the depth of the scene. Colors of the cut surface are related to the Vienna subway system and encode which subway line is driving the hoistway. To emphasize the cut surfaces, anything else was made less prominent and colored in grayscale. Below all objects of interest there is a second artificial surface to cast shadows on, which again, assists to comprehend the spatial relationship.

Inspired by many illustrations of sewer systems, we decided to use cutaways to look inside pipes and shafts of the sewer network. Since the approach by Office Le Nomade shows an innovative and very appealing way to do cutaways on terrain, we decided to adopt their approach. Furthermore the simple structure of subway hoistways is similar to sewage pipes and it seemed natural to apply the same cutaway style on the pipes of the sewer systems.

The criteria for the cutaways are to utilize the cognitive capabilities of humans to auto-complete the information that has been cut away. Additionally cutaways should be rendered in real-time and therefore they need to be rendered fast. Additionally the cutaways should be view-dependent. Furthermore the technique should be interactive and as such should have a minimal amount of steering parameters. To support the rising field of flood simulations this thesis aims to invent a new and efficient rendering technique for this domain. To generate meshes for each pipe and shaft and to deal with their intersections seemed challenging. Moreover it would be problematic to recalculate them each time the data is changed. For instance if another set of data from a different part

of the city has to be visualized, it has to be swapped out. An efficient way to generate tubes in real-time is with the use of billboards (also known as impostors). Billboarded tubes were used in other contexts but not in combination with cutaways. Image space techniques are used more often lately and seem to become a standard approach rather than a niche.

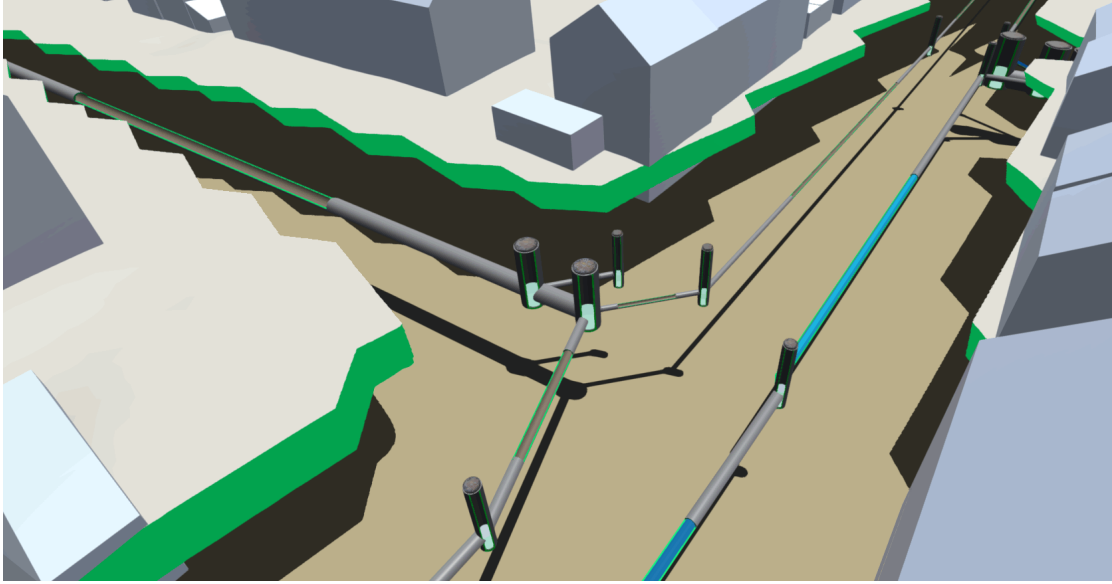


Figure 1.2: Preview of terrain cutaway and billboarded cut-open pipes.

Our approach is to develop an efficient implementation with procedural billboarded tubes which are cut open analytically in image space. This kind of technique could be a great contribution to the domain of flood risk management simulations. The increasing use of visual simulations for flood defense, which utilize the visualization of subsurface infrastructure, will certainly benefit from new techniques. In general this technique has potential to be used for visualizing the interior of any tubular geometry.

1.4 Structure of the Thesis

1. Introduction

This very chapter. Specifies the provided data and proposes the goals of the thesis.

2. State of the Art

Points out the must of flood risk management and reviews state of the art occlusion management techniques and billboard techniques.

3. Methodology

Introduces the routines and algorithms used to realize the approach of analytically and cut-open terrain and billboarded sewage pipes.

4. Implementation

Highlights important facts about how the method is implemented in the Unity3D engine.

5. Demonstration

Showcases several images of the implementation complemented by a verbal explanation describing individual aspects of the scene.

6. Evaluation

Compares the result with cutting conventions and design principles for cutaways. Further evaluates the implementation by providing performance data and feedback of visualization experts on the visual aspects. Further discussing open issues of the technique.

7. Summary and Future Work

A summary and conclusion of the entire work is given. Further possible future work is proposed.

State of the Art

This chapter aims to provide an overview of what has been studied and what was useful and contributed to this work. First the chapter introduces the must of flood risk management. It will point out reasons for the increasing use of visualizing simulations in order to develop defensive strategies in the domain of flood risk management. After that the chapter will examine the state of the art occlusion management techniques with a special focus on cutaway techniques. The last section of this chapter is about billboarding and the different use cases for this technique.

2.1 Flood Risk Management

Disasters in Europe have in general become more frequent and cause more damage in the period 1998-2009 [EEA11]. In this period flooding was the most costly hazard and added up to about EUR 52 billion. In 2007 the European Parliament published a directive on the assessment and management of flood risks [Eur07]. It states that human activities and climate change contributed to an increased likelihood and adverse impacts of flood events. Further it aims to reduce and manage the risk that floods pose to human health, life and infrastructure. This is mostly effective if it is coordinated throughout a river basin and requires the member states to work together if they share any basins. They state that flood risk management plans should focus on prevention protection, preparedness, flood forecasts and early warning systems. With this in mind it is very important to be prepared for flood events and flood simulation software is a key tool in the process of preparing.

2.2 Occlusion Management Techniques

Illustrations are often used to explain the rather complex interior nature of an opaque entity in a visual simplified way. Areas of application arise in many domains, including

medicine, mechanical processes in engineering, architectural objects or (micro-)biological objects. Such illustrations can be found in almost every book about technical or biological processes. What complex opaque objects have in common is the need to look inside of them to understand their functionality or inner structure. Their complexity derives from different nested parts that may interact with each other. Thus the objects of interest are occluded by intertwined parts or covered by a girdling surface and thereby hidden anyhow from an outside point of view. To clearly convey the inner structure and the interaction of various parts, it is important to remove occluding entities. To do so, many illustrators made use of transparency, also known as ghosting, as a visual effect, to show spatial relationship.

However, illustrators often prefer to use cutaway techniques to enable the user to perceive objects of interest inside of a solid opaque object. Instead of using transparency, and letting the inner object shine through, parts of a girdling surface or parts of other occluding object are simply removed. As a result it looks like a piece had been cut out of the occluding objects. Cutaways often facilitate a better understanding of spatial ordering than transparency, especially when sharp contrast is used between foreground and background objects. Diepstraten et al. [DWE03] extracted a small and effective set of rules from traditional cutaway techniques for an automatic generation process on a computer used by illustrators. These rules were further used to provide and compare methods of cutaway drawings that achieve interactive frame rates on modern graphics hardware. Diepstraten et al. state that it is hard to formalize where an automatically computer-generated cutaway should be placed. This depends on many different factors like the size and shape of the inner and exterior object, the semantics of the objects and even personal taste, for example. Nevertheless, five quite convincing rules for fully automatic cutaway drawings are provided. Note that the following rules are quotations.

- (a) Inside and outside objects have to be distinguished from each other
- (b) The cutout geometry is represented by the intersection of (a few) half spaces
- (c) The cutout is located at or around the main axis of the outside object
- (d) An optional jittering mechanism is useful to allow for rough cutouts
- (e) A possibility to make the wall (cut surface) visible is needed

Table 2.1: Set of rules for computer generated cutaways by Diepstraten et al. [DWE03]

Another mentioned technique is a cutout via Stencil Test. Thereby convex exterior objects can be cut open. This image-space technique uses the stencil test and stencil buffer to represent the cutout geometry.

Basically the goal of illustrations is to emphasize important structures and to maximize the information content of the represented entity. In importance-driven volume rendering, presented by Viola et al. [VKG04], the object importance is assigned to the data. Usually

each voxel of the volumetric data set contains information about density that is mapped to the opacity and color. In the importance-driven approach, an additional property encodes the visibility priority and is assigned to each voxel. In traditional volume rendering a transfer function, based on density, is used to compose a final image. With additional information of importance and various composing schemes using this new dimension in the rendering pipeline, this technique makes the object of interest clearly visible. In case a less important object occludes more important structures it is displayed more sparsely in the obscuring area. Further readings on visualizing medical volume data can be found by Burns et al. [BHW⁺07] and Bruckner et al. [BGKG05].

Interactively generated cutaways of complex three-dimensional models are discussed by Li et al. [LRA⁺07]. Their paper includes a great overview of cutting conventions found in traditional scientific and technical illustrations. Further they introduced an approach how this conventions are incorporated into an interactive visualization system for static scenes. In this system the user controls auxiliary parameters to define how the cutaways are structured and arranged. Additionally the user can define sets of objects of interest and the system will expose those parts automatically.

"The best cutaways clearly expose the target parts but also preserve some context from occluding structures so that viewers can better understand the spatial relationships between all parts" - Li et al.

To achieve this with cutaways, they state it is important that occluding parts should be only removed partially. Thereby the viewer is able to mentally reconstruct the geometry which has been cut away. For this reason cutting conventions evolved among illustrators because for similar structured objects a certain shape or alignment of the cutaway is beneficial.

The first method that achieved interactive frame rates with adaptive cutaways for comprehensible rendering of dynamic polygonal scenes was presented by Burns & Finkelstein [BF08]. Even for a changing point of view the cutaway is adapted each frame and therefore view-dependent. Burns & Finkelstein used an image-based technique and computation that is able to handle complex scenes and models.

Recently a novel method was proposed for rendering cutaway illustrations of mesoscopic biological models [MMS⁺16]. Biological models consist of many instances of the same objects but a rather small number of different types. First the model is clipped to generate a cutaway view and a hierarchical list of bars to inform the user about the instance visibility distribution of each type. Further the user can interactively control these bars to fine-tune the final result. For a special effect, where many instances of a biological type completely cover other interior types, they proposed a technique to remove occluding instances close to the center of the whole object. They dub it the *aperture effect*. Therefore a 2D distance transform of the occluding instances is computed to mask their distance to the edge of the whole object and is further stored in a texture. In the fragment shader of occluding instances the resulting distance field is looked up, and discards the fragments according to a user-defined aperture coefficient.

Design principles for cutaway visualization of geological models presented by Lidal et al. [LHV12] are of interest for terrain cutaways. Five design principles for computer-generated cutaway visualizations are presented.

2.3 Billboarding

Billboarding originally emerged as a technique in order to increase frame-rates of real-time 3D graphics systems. In earlier stages graphics systems and thereby computer games struggled to improve in visual quality and keep up frame rates at the same time. Billboarding was originally a term used to describe a part of the technique behind dynamically generated impostors [CSG95]. Alongside with other frame-rate improving techniques like culling on non-visible objects and using several levels of details the use of impostors proved to be useful in that regard. The idea behind dynamically generated impostors was to replace distant polygon rendered objects with textures. These textures display the replaced object, that has previously been rendered into a texture buffer. Since the viewing angle on distant objects changes relatively minor it is practically impossible to see any difference. Reusing the same texture from the buffer many times for similar objects can save many calculations and offers to render more complex scenes.

Although this thesis will not use textures rendered onto billboards we are interested in the billboarding technique for other reasons. Billboarding has more benefits than only improving frame-rates, for instance the underlying data structure can be kept very simple. For an efficient visualization of particle trajectories Schirski and his team proposed a method that is based on billboarding [SKH⁺04]. These so called *Virtual Tubelets* use cylindrical billboards to simulate movement of particles through the flow domain. Basically the cylindrical billboards are quadrilaterals, which are always aligned towards the viewer. To finally get the illusion of self-illuminated tubes the *Virtual Tubelets* are textured appropriately. In addition to the illumination a coloration is applied to visualize for example the velocity of the particles trajectory. This simple billboarding approach had visual inconsistencies that had to be solved. At the beginning and the end of the trajectory rounded edges were missing and therefore the trajectories seemed to point away from the viewer. This issue was resolved by adding an additional quadrilateral at both ends. Textured appropriately and positioned orthogonally to each end of the trajectory it creates the illusion of a well-capped tube.

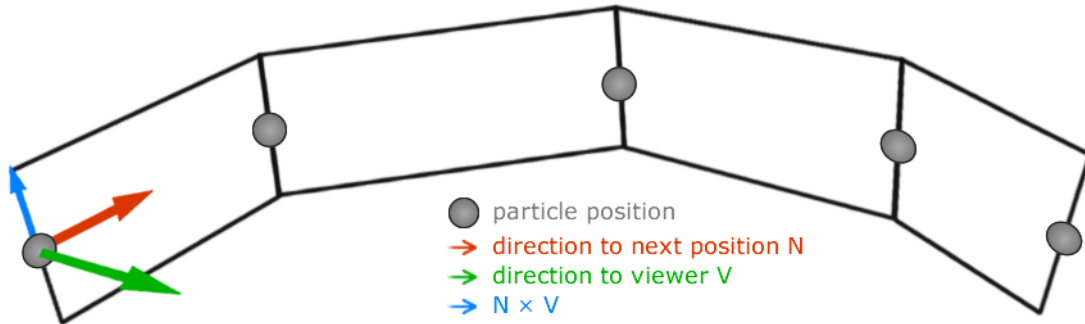


Figure 2.1: Redrawing of the Anatomy of a *Virtual Tubelet* by Schirski et al.

For each particle position P , as depicted in Figure 2.1, two vertices are forwarded to the graphics system. The upper and lower vertex, P_0 and P_1 , related to a particle position, is determined by the following equation.

$$P_i = P + r(-1)^i(\hat{V} \times \hat{N}), \quad i \in \{0, 1\} \quad (2.1)$$

Where r is the desired radius, \hat{V} is the vector to the viewer and \hat{N} is the vector from the current to the next particle position. Further the direction to the viewer \hat{V} is updated every frame which lets the billboards always face the viewer. Advantages of this technique are clearly the low amount of vertices that are necessary to render tube-like structures. Because there is no underlying geometry involved when using this approach, most tubelet parameters can be changed without having to recreate any polygonal representation. Parameters like width, length, color et cetera can be changed interactively.

A recent example of modern use of impostors can be found by Muzic et al. [MAPV15] where they introduce cellVIEW, a tool for illustrative and multi-scale rendering of large biomolecular datasets. Atoms of a cell are visualized as spheres. Since thousands of them have to be rendered, sphere meshes would result in bad frame rates. In their system individual atoms of a cell are rendered via 2D depth impostors due to a much lower vertex count.

Methodology

This chapter is about the methods and techniques which were used to realize our approach. To visualize water flow inside the sewage system three different objects have to be cut-open, the polygon meshed terrain, billboarded pipes and the shafts. Shafts are constructed by multiple polygon meshed cylinders. To go in visual order, the first section is about how the terrain was cut-open and our image-space technique to apply an artificial thickness to the cut. Afterwards this chapter will explain the technique used to expose water flow inside of the sewage pipes. As stated before, the approach to visualize them is to develop an efficient implementation with billboards which are cut open analytically in image-space. Water levels of the sewage shaft have to be exposed as well. Therefore the last section of this chapter will explain how this cutaway is performed.

3.1 Terrain Cutaway



Figure 3.1: Cut-open terrain to reveal the sewer system.

To expose the subsurface sewage system the terrain was cut open by discarding occluding fragments. Therefore a two-dimensional representation (without the height values) of the sewer network is drawn in black lines onto a white texture with a simple line drawing algorithm. Thereby black is encoded as 8-bit color value 0 and white as 255 on a grayscale texture. The same texture size is utilized which is used to color the terrain and aligned the 2D line representation to perfectly match with the positions of the sewer network in world coordinates. Additionally a one-dimensional distance transform under the L_1 distance is applied to this texture. The algorithm is applied horizontally and vertically [FH12]. The resulting distance field is depicted in Figure 3.2. It encodes the distance to the closest pipe in grayscale, whereas the pipe itself has a floating point value of 0, and beyond a certain distance to any pipe the value is 255. As you may notice there are artifacts due to the use of L_1 distance which cause the zigzag effect (jittering, sawtooth-like) on the cutaway. Thus, a rough cutaway is created which is considered an optional rule for computer generated cutaways for technical illustrations by Diepstraten et al. as mentioned in Table 2.1d. The rule is applicable for simple a cutout geometry to generate a higher level of abstraction [DWE03].

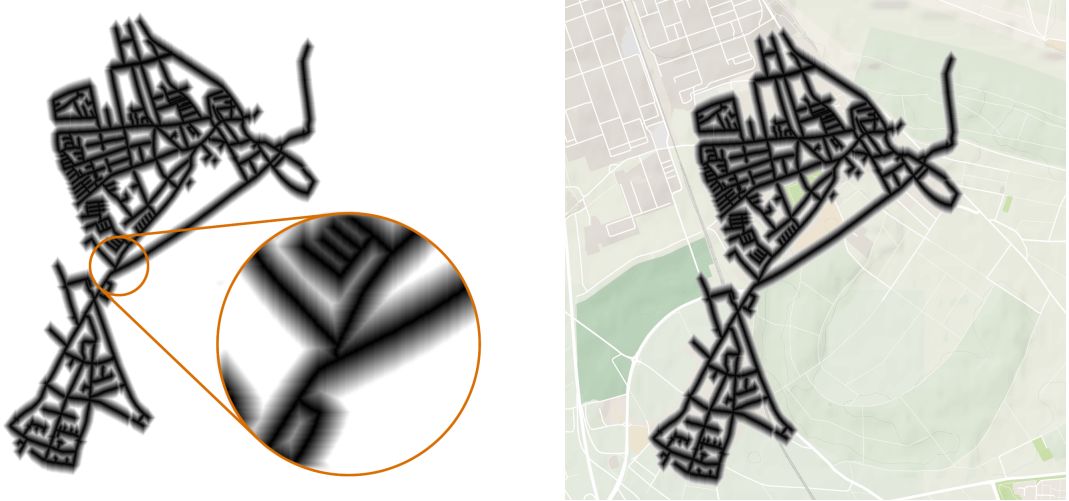


Figure 3.2: 2D distance field representation of the sewer network on white background (left), and aligned with the terrain texture (right).

Further the generated texture is forwarded to the graphics system and used as a distance field. For each fragment the euclidean distance to the camera d_c is evaluated. Furthermore a lookup on the distance field texture is performed, by the fragments texture coordinates $P_{uv}(u, v)$, to obtain the distance value $D_p(P_{uv})$ to the closest pipe. Obtaining $D_p(P_{uv})$ in the fragment program via a texture lookup will result in a floating point value between 0 and 1. Broadly speaking d_c and $D_p(P_{uv})$ are used to decide if the fragment is to discard or the terrain is drawn. The artificial thickness of the terrain has to be factored in as well. Before a fragment is actually discarded it is evaluated if it is part of the artificial thickness. Therefore the two horizontal coordinates of the view direction vector $\hat{\mathbf{v}}_{3D}(x, y, z)$, the

direction the camera is looking at, are used to make a two-dimensional peek in $\hat{\mathbf{v}}_{2D}(x, z)$ and perform another lookup on the distance field texture. In our system, y is the height coordinate and is ignored for this reason. The peek will provide information, if current fragment is close to not discarded terrain, and the artificial thickness has to be drawn instead of actually discarding this fragment. To interactively control the width of the terrain opening during runtime, a steering parameter w is used. Another steering parameter s is used to control the size of the opening. Basically with w , s and d_c a threshold t is computed for each fragment to control the width of the opening. The further away the camera is to a fragment, the lower the threshold t will be and therefore harder to pass. With $D_p(P_{uv})$ staying below the threshold t , the routine will result in either discarding the fragment or drawing the artificial thickness.

$$t = \frac{w(s^2 - d_c)}{s} \quad (3.1)$$

This equation causes the smooth closing and opening of the terrain as the camera is moving across. By default the parameters are set as $w = 0.015$ and $s = 10$. The following function f_1 is used by each fragment of the terrain to determine the final result. Where ϵ is a value that is at least big enough, to perform the peek lookup on the distance field, and obtain the next texel in viewing direction $\hat{\mathbf{v}}_{2D}$. More specifically the value of ϵ impacts the width of artificial thickness.

$$f_1(P_{uv}, t, \hat{\mathbf{v}}_{2D}) = \begin{cases} \text{draw terrain} & D_p(P_{uv}) \geq t \\ f_2(P_{uv} + \hat{\mathbf{v}}_{2D} \cdot \epsilon, t) & D_p(P_{uv}) < t \end{cases} \quad (3.2)$$

$$f_2(P_{uv}, t) = \begin{cases} \text{draw artificial thickness} & D_p(P_{uv}) > t \\ \text{discard fragment} & \text{otherwise} \end{cases} \quad (3.3)$$

Opinions are different regarding the zigzag effect and it is considered if it should rather be cleared out to get a smoother cut instead. This could be achieved by implementing the squared Euclidean distance instead of the L_1 distance for the distance transformation, which will not have such artifacts and therefore not produce the zigzag effect. However, our impression is that this zigzag effect is in fact appealing and thereby more of an optional feature than an artifact. This topic is discussed further in Section 6.3. The proposed method does not support illuminated shading of the cut surface. However, this could be achieved by computing a similar texture where each pixel encodes a vector as RGB value which points in the direction of the closest pipe. Thereby this vectors could be used as normal in the fragment shader and a illumination model can be calculated to shade the cut surface.

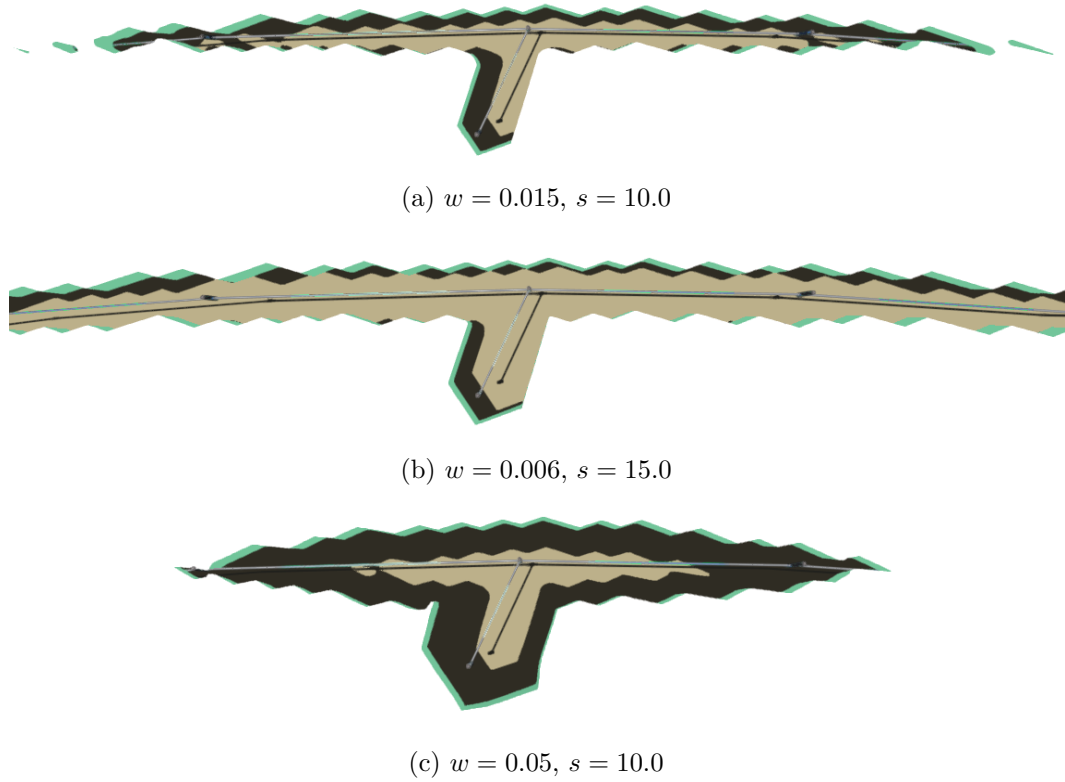


Figure 3.3: Terrain cutaways with constant distance to camera d_c and varying steering parameters w and s (w = width, s = size).

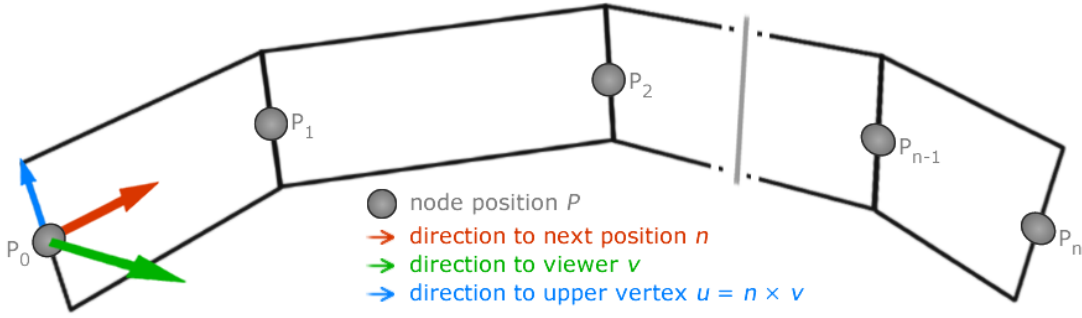
3.2 Drawing the Pipe



Figure 3.4: *Billboarded Pipe*

Each sewage pipe is defined by a list of three-dimensional nodes, as described more precise in Section 1.2. (The first node defines the starting position in world space coordinates and each successive node sets the spatial direction.) With the given data structure, the best solution to generate a visual representation of the sewage pipes is to use billboards. Basically this is done similar to the approach by Schirski and his team [SKH⁺04], which

has been examined in Section 2.3. Of course there are some discrepancies to the *Virtual Tubelets* by Schirski, since we are not rendering trajectories. With rendering trajectories the amount of used nodes for particles are varying in time, which is not the case when rendering static sewage pipes. Another difference is that Schirski et al. added additional quadrilaterals at both ends of the trajectories to resolve visual inconsistencies. Since sewage pipes are in general connected to sewage shafts, this problem is not present. Both ends are plugged into a sewage shaft and thereby hidden by the shaft, hence these visual inconsistencies do not occur. Therefore it is not necessary to render caps for the endings of the sewage pipes. However, connecting sewage pipes and shafts causes the billboarded pipes to look flat at their intersection. This problem is resolved by correcting the depth value of the billboards to fake their spatial capacity and create a realistic impression of the intersections. Furthermore *Virtual Tubelets* only used standard OpenGL and for this case, the orientation of the billboards was computed on the CPU. In our approach computing the billboards vertices is carried out by the geometry shader [Mic].

Figure 3.5: Anatomy of *Billboarded Pipes*

Components of a billboarded pipe are depicted in Figure 3.5 and as demonstrated their node positions are utilized to arrange the quadrilaterals. The node positions are wrapped in a buffer and forwarded to the geometry shader. As next step the geometry shader calculates the required vertices of the quadrilaterals. To compute the direction to the quadrilaterals upper vertex of the first node P_0 the cross product of the normalized vectors \hat{n}_0 and \hat{v}_0 is used as set out in Figure 3.5. The described method can be used for the last node P_n of a billboarded pipe basically in the same way. For remaining nodes, P_1 to P_{n-1} , the calculation of the corresponding direction to the quadrilaterals upper vertices is more complex. The vector \vec{u}_i pointing in the direction of the upper quadrilaterals vertex for each related node position P_i respectively, is determined by the following equations.

$$\vec{u}_0 = \hat{n}_0 \times \hat{v}_0 \quad (3.4)$$

$$\vec{u}_i = \frac{\hat{n}_{i-1} + \hat{n}_i}{|\hat{n}_{i-1} + \hat{n}_i|} \times \hat{v}_i, \quad i \in \{1, \dots, n-1\} \quad (3.5)$$

$$\vec{u}_n = \hat{n}_{n-1} \times \hat{v}_n \quad (3.6)$$

The direction to the billboards lower vertex \vec{l}_i is computed by inverting \vec{u}_i for each node position P_i .

$$\vec{l}_i = \vec{u}_i \cdot (-1), \quad i \in \{0, \dots, n\} \quad (3.7)$$

Further the vectors \hat{u} and \hat{l} are multiplied by the radius r of the sewage pipe, which can be changed interactively at any time. Finally all vertex position of the quadrilaterals are computed by translating the node position P by the calculated vectors u and l , to get the corresponding upper and lower vertex positions. Arranging the billboards as demonstrated, causes them to face the camera by rotating around multiple axis, which are spanned by their consecutive node positions. Thereby they are almost exactly aligned towards the viewer even though the camera is moving. Since one particular quadrilateral cannot use the same vectors \hat{n} and \hat{v} to calculate all four of its vertices, which would cause a gap between consecutive quadrilaterals, they have to be slightly twisted to line up seamlessly. This slight twist is achieved by calculating an average vector of \hat{n}_{i-1} and \hat{n}_i in Equation 3.5.

After arranging the billboards, the next step is to shade them in a non-photorealistic style. Therefore the billboards are colored and illumination is calculated to create the impression of an actual tube-like structure. For this purpose the normal-directions on billboards vertices are required to compute the Blinn-Phong shading model, which are set by the geometry shader during their creation. However, the illumination needs to be calculated as if the flat billboard would be tubular. Therefore two vectors are set for each vertex and propagated to the fragment program. First the normal-direction of a billboards face is calculated and set for all four vertices of a quadrilateral. Since we do not want the slight twist to be shown in the illumination the average face normal F_a of the four vertices is determined by the following equation.

$$F_{a_0} = \frac{\hat{u}_0 + \hat{u}_1}{|\hat{u}_0 + \hat{u}_1|} \times \hat{n}_0 \quad (3.8)$$

$$F_{a_i} = \frac{\hat{u}_{i-1} + \hat{u}_i}{|\hat{u}_{i-1} + \hat{u}_i|} \times \hat{n}_i, \quad i \in \{1, \dots, n-1\} \quad (3.9)$$

$$F_{a_n} = \frac{\hat{u}_n + \hat{u}_1}{|\hat{u}_0 + \hat{u}_1|} \times \hat{n}_0 \quad (3.10)$$

Additionally for the upper two vertices of a quadrilateral $N_{u_i} = \hat{N}_{a_i} \times \hat{v}_i$ is set and $N_{l_i} = \hat{N}_{a_i} \times \hat{v}_i \cdot (-1)$ for the lower vertices with the view direction vector \vec{v} . Thereby the normal vectors \hat{N}_{u_i} and \hat{N}_{l_i} are automatically interpolated by the graphics system, according to the fragments position on the billboard, when retrieving this value \hat{N} in the fragment shader. Since this automatic interpolation is insufficient, additionally the face normal-direction \hat{F}_a of a quadrilateral is used. In this way a fragments final normal-direction \hat{N}_f used for illumination is determined by, again, computing the average vector of F_{a_n} and \hat{N} for each fragment as followed.

$$\hat{N}_f = \frac{\hat{N} + \hat{F}_a}{|\hat{N} + \hat{F}_a|} \quad (3.11)$$

The resulting normal vector perfectly matches to mimick the illumination of a tube-like structure and is further used for a standard Blinn-Phong shading model and specular highlights as well. In combination with the proposed non-photorealistic shading the billboarded pipes look very similar to polygon meshes but are quite advantageous in terms of low memory footprint and faster rendering time, which is demonstrated in Section 6.2. Further the downsides of billboards are discussed in Section 6.4.

3.3 Cutting the Pipe



Figure 3.6: Cut-Open Billboard Pipe

Featuring water flow as object of interest is the main purpose of this visualization. Consequently water flow inside of the sewage pipes has to be exposed to the viewer. In order to achieve this in combination with billboarded pipes a new technique had to be developed. Therefore a proxy geometry is used to prepare areas in the stencil buffer to define whether the view-aligned billboards have to be rendered and shaded for inside or outside illumination of the pipe. The illumination technique of the billboards is described at the end of the previous subchapter. Shading the billboards to display the inside of a sewage pipe is easily done by inverting the computed normal direction in Equation 3.11. Additionally the base color is changed to enhance the ability to visually distinguish inner and outer parts of the sewage pipe. For the same purpose the base color of the cut surface is different and in high contrast to any other used colors in order to stand out as depicted in Figure 3.6.

In general the idea is to set up a static proxy geometry, like demonstrated in Figure 3.7), before the view-aligned billboards are applied. This will achieve two things. First the green cut surface is rendered immediately. Second the inner area of the proxy geometry is used to set a certain value in the stencil buffer. Now this area is prepared to render view-aligned billboards that are shaded accordingly to display the inside of the pipe. At a later stage in the rendering pipeline, the prepared area is used and the inside of the pipe is drawn.

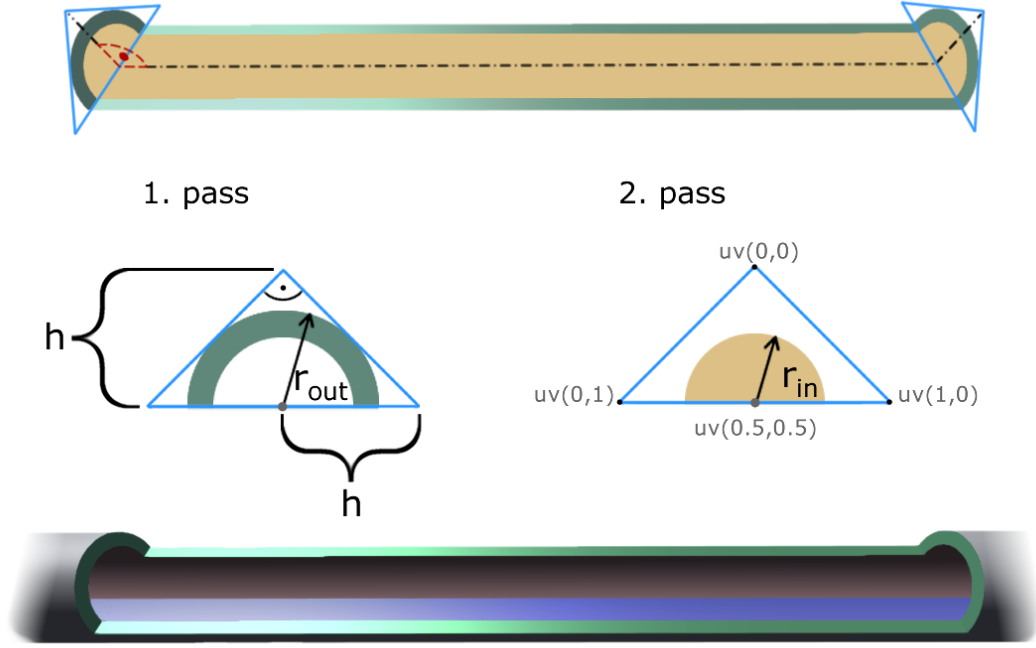


Figure 3.7: Proxy Geometry - Triangles

A value written to the stencil buffer by one particular shader pass is declared for the shader pass itself. It is executed for any drawn fragment and can therefore not be changed in the fragment shader. For this reason the described routine, and further processing of the proxy geometry, is carried out in two separate shader passes which are labeled in Figure 3.7 and 3.8 accordingly. Important to note is that in this Section whenever a cut surface is drawn, the stencil buffer is set to a certain value. This value will prevent the outside and inside of the pipes billboard to overdraw the cut surface.

Like demonstrated in Figure 3.7, the blue triangles are created by the geometry shader based on pipe node position like the quadrilaterals of the pipe. These triangles are arranged perpendicular to the slope of the related proxy geometry segment and not changing their alignment towards the viewer. The cut surface and the area to prepare the stencil buffer are determined analytically by the use of the texture coordinates and the distance to the center of the semicircle. Therefore the size of the triangles has to exactly provide accurate space for the sewage pipes outer radius r_{out} inside of them. Providing this space is achieved by constructing the triangle with $h = \sqrt{2} \cdot r_{out}$ and using h as demonstrated in Figure 3.7 when creating the vertex positions of the triangle in the geometry shader. Texture coordinates (u, v) are at the same time set as depicted and used to compute the Euclidean distance d to the texture coordinates $(0.5, 0.5)$ for each fragment.

$$d = \sqrt{(0.5 - u)^2 + (0.5 - v)^2} \quad u, v \in [0, 1] \quad (3.12)$$

Calculating d enables us to express the desired areas. The way the triangles are constructed, the outer radius r_{out} can now be described by $d_{out} = 0.5$. The inner radius r_{in} is expressed by $d_{in} = (\frac{r_{in}}{2r_{out}})$. If the distance d exceeds d_{out} , falls below d_{in} , or is in between, the fragment is handled by the following functions $T_1(d)$ and $T_2(d)$ for the first and second shader pass, respectively.

$$T_1(d) = \begin{cases} \text{draw cut surface} & d_{out} > d > d_{in} \\ \text{discard fragment} & \text{otherwise} \end{cases} \quad d \in [0, \sqrt{0.5}] \quad (3.13)$$

$$T_2(d) = \begin{cases} \text{prepare stencil buffer} & d < d_{in} \\ \text{discard fragment} & \text{otherwise} \end{cases} \quad d \in [0, \sqrt{0.5}] \quad (3.14)$$

Obviously the proxy geometry is made of more billboards than the two triangles. Additionally it is constructed with at least one quadrilateral in between these two triangles. In case the cut is done on a straight part of the pipe it comes down to only one quadrilateral. In case of a bended pipe the quadrilaterals of the proxy geometry are computed similar to the view-aligned billboarded pipes. The only difference is that instead of the view direction a vector pointing upwards, perpendicular to the slope of each segment, is used.

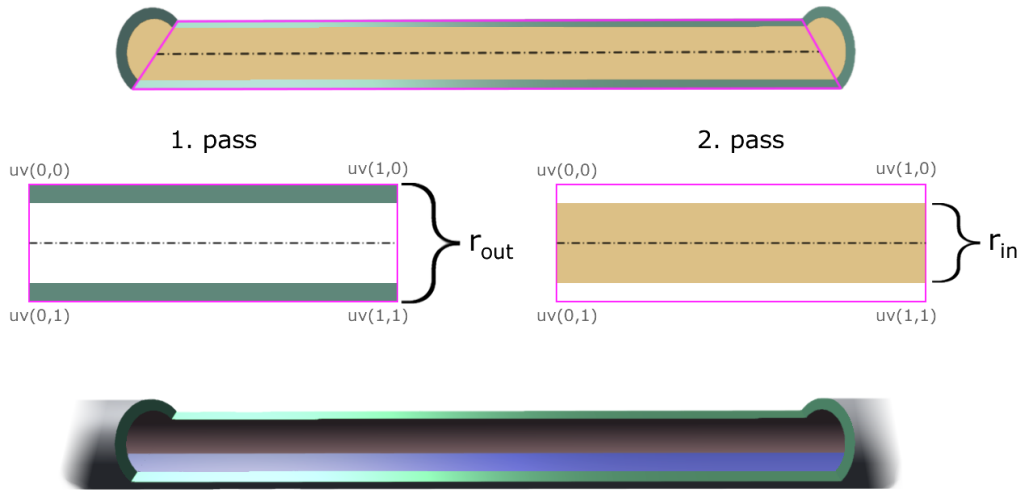


Figure 3.8: Proxy Geometry - Quadrilaterals

Almost the same procedure, as before with the triangles, is applied to this part of the proxy geometry. Apparently the pink quadrilateral is arranged to cut the pipe in half horizontally and shares vertex position with the triangles as depicted in Figure 3.8.

Differently to the triangles, this time the width of the quadrilaterals has the same value as the outer radius r_{out} of the pipe. Again the texture coordinates are used to analytically determine the two distinct areas and are therefore set as depicted in figure 3.8. Since the quadrilaterals are symmetric around their primary axis running along the length of the pipe, only the second value of the texture coordinates (u, v) is used to carry out this calculations. As before the outer and inner radius, r_{out} and r_{in} , function as a threshold to define the desired areas on the quadrilateral. Whereas the wall thickness t of the pipe is actually used and calculated by subtracting the inner radius r_{in} from the outer radius r_{out} . Since we need the wall thickness in texture space it is further divided by r_{out} , which results in $t = \frac{r_{out} - r_{in}}{r_{out}}$. Based on a fragments texture coordinate v it is determined whether to draw the cut surface or set a value in the stencil buffer to declare it as inside as followed by the functions $Q_1(t, v)$ and $Q_2(t, v)$ for the first and second shader pass, respectively.

$$Q_1(t, v) = \begin{cases} \text{prepare stencil buffer} & (1 - t) > v > t \\ \text{discard fragment} & \text{otherwise} \end{cases} \quad t, v \in [0, 1] \quad (3.15)$$

$$Q_2(t, v) = \begin{cases} \text{draw cut surface} & \text{if } v > (1 - t) \\ & \text{or } v < t \\ \text{discard fragment} & \text{otherwise} \end{cases} \quad t, v \in [0, 1] \quad (3.16)$$

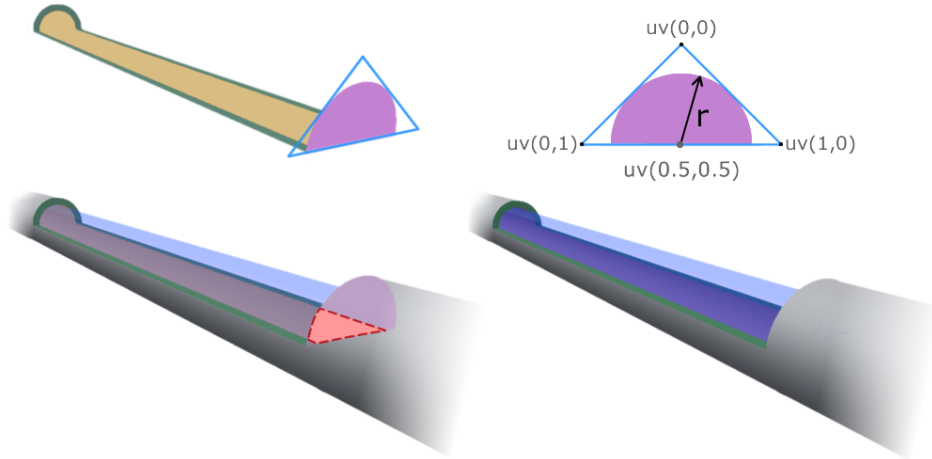


Figure 3.9: Proxy Geometry - Triangle Back-Faces

To prevent visual inconsistencies, the back-faces of the same triangles, as first mentioned in this Section, need to exchange the stencil buffer value when facing them. The visual incorrectness which would occur is demonstrated in Figure 3.9. For the violet area a new value is set in the stencil buffer, which prevents the inside view-aligned billboards to be rendered there, and ensures to display the outside of the sewage pipe instead. Otherwise the red-dotted area in Figure 3.9 would display the inside of the pipe, although it should not, given the way we are facing the cut.

Setting up the image-space with values in the stencil buffer to specify whether to render view-aligned billboards, that are shaded to display inside or outside of the sewage pipe, is thereby finished.

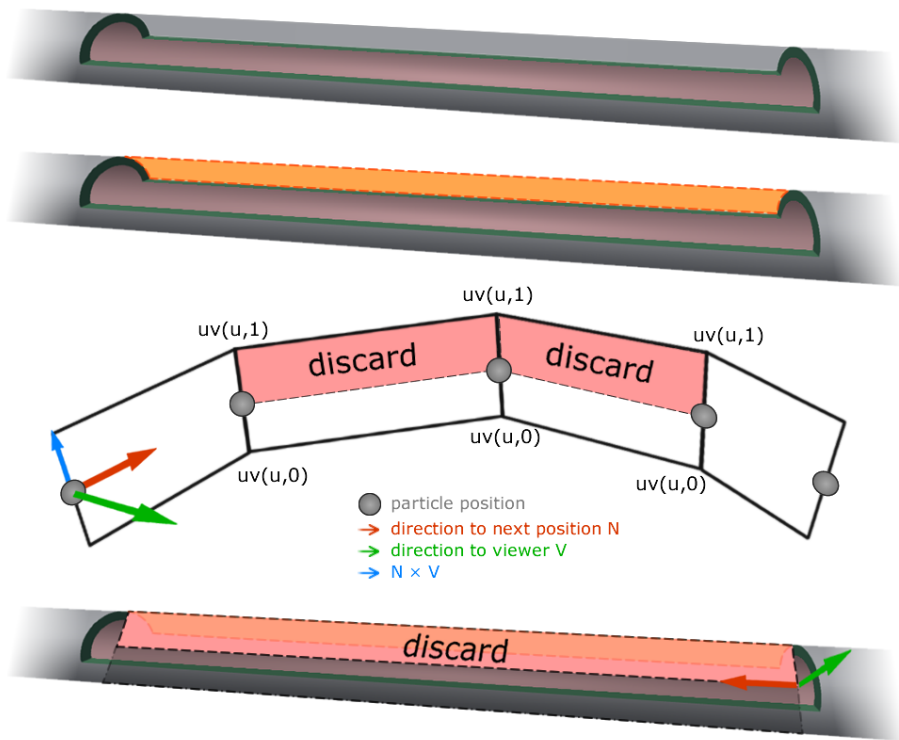


Figure 3.10: Billboarded Pipes - discarding unnecessary fragments on cut-open pipe segments.

Now the view-aligned billboards are drawn accordingly to the prepared areas in the stencil buffer. Currently the result, as observable in Figure 3.10, has an obvious flaw. Apparently the orange, red-dotted area was not prepared appropriately. The reason

for this is that it is easier and cheaper to simply discard this area right away on the view-aligned billboards than to use the stencil buffer for this as well. With the use of texture coordinates the upper half of billboards inside a cut-open segment are discarded in the fragment shader. Alternatively it is possible to prevent this problem in the geometry shader by not translating the upper vertices of the quadrilaterals in the cut-open pipe segment and therefore not even creating this area for the view-aligned billboards.

3.4 Cutting the Shaft

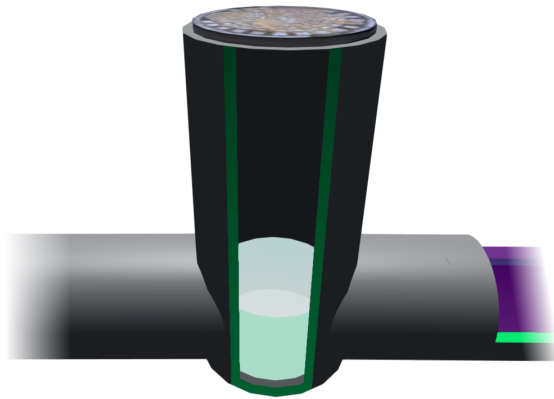


Figure 3.11: Polygon meshed and cut-open shaft.

Sewage shafts are constructed by multiple cylinder primitives and therefore made of polygon meshes. The shaft itself is constructed by an outer and an inner cylinder, where the inner one is slightly smaller than the outer one. For both, fragments are discarded when the dot product of the normal vector and view direction vector surpasses a certain coefficient. In this process the height coordinate of both vectors is ignored. This cutaway is thereby view-dependent and the opening will always face the viewer. The coefficient defines the opening width of the cutaway. Applying the cut surface color is achieved by using the same shader for back facing triangles of the outer cylinder and the front facing triangles of the inner cylinder.

Implementation

The proposed technique was tested and implemented in the Unity3D game engine. For the implementation only features of the personal edition were used. Unity can be downloaded at their website <https://unity3d.com/de/unity> and offers free licenses for students. Benefits of using Unity were the integrated rendering engine which dropped the workload for this thesis significantly. Unity has an easy to use editor which allows to test out several modifications during runtime of the application. Another aspect are several built in post processing image space effects of the Unity standard asset package.

Unity is basically put to use by the classes *GameObject* and *MonoBehavior*. As their naming suggests, a *GameObject* is created for every object in the game like a sewage shaft for example. Multiple instances of *MonoBehavior* can be attached to a *GameObject* to define how they are built up and how they behave in the rendering loop. For the implementation a *GameObject* dubbed *SewerReader* is used. The *MonoBehavior* attached to it has the job to parse the provided binary files and assign the data in a static class *SewerSystem*. Moreover, the *SewerReader* is used as a main coordinator of the rendering loop and resembles the *main* method as used in modern OpenGL rendering approaches. The class *SewerSystem* holds an instance of a class *Pipe* and *Shaft* for every sewage pipe and shaft that has been parsed, and stores the parsed properties. Every instance of *Shaft* creates a *GameObject* with multiple Unity-built-in cylinder meshes to construct it. At fixed time steps a method in the *Shaft* class is called to update the water depth according to the simulation data. Important to note is that the shafts cylinders, except the cylinders used for the water, should be declared as static. Thereby the Unity engine will handle them more efficiently.

For each group of surrounding objects like the terrain, buildings and trees a *GameObject* is created to hold the meshes of the corresponding OBJ files. Since they are static nothing further has to be done. The routine to automatically cut open the terrain is executed in its shader. The artificial surface which is exposed when the terrain is cut open, is a

flattened duplication of the terrain mesh translated downwards. The shadows on the artificial surface, which would be dropped by the terrain above, are computed by the same routine that is used to compute the cutaway of the terrain. Instead of discarding fragments it will color them appropriately to look like it is shadowed. In this process the shadows are displaced horizontally by the angle of the directional light to fit in the illumination model perfectly. The reason for this is that the terrain cutaway is done by discarding fragments, what apparently does not affect the built in shadow mapping in Unity. This would cause the artificial surface to be completely darkened by shadows even if the terrain above it is cut open.

Rendering the sewage pipes is done differently. A *ComputeBuffer* is created to store data of all pipes in the sewage system. For each node position of a pipe this *ComputeBuffer* stores a *struct* with its own node position, the next node position and the previous node position, if they exist. Further an identification number of the corresponding sewage pipe, its radius and a number to define the index of this node of the pipe is stored. Similarly two more *ComputeBuffer* are created to construct the triangles and quadrilaterals of the proxy geometry. Those three buffers are forwarded to the graphics system once and are reused every frame. Each frame the Unity method *Graphics.DrawProcedural* is called for each shader pass of the sewage pipe to draw them with the geometry shader fully procedurally. The geometry shader accesses the data in the *ComputeBuffer* and creates vertices as described in Section 3.2. Based on the same principles the construction of the proxy geometry and the water inside the sewage pipes is achieved. For the water two *ComputeBuffer* are updated each time step of the flood simulation and sent to the graphics system. They store the water depth and color of the current simulation time step, respectively. Finally the Unity standard asset anti-aliasing script is used to smooth the final result.

Demonstration

This chapter will showcase the proposed technique with the results of the final implementation. The demonstrated graphics will show the city model of Worringen, Cologne, Germany which we received from VRVis Research Center and is coupled with flood simulation timeline data. Additionally provided data encodes the depth of the water inside of the sewer system or the discharge velocity. Corresponding legends are shown in Figure 5.1. Each demonstrated graphic is complemented by a short verbal explanation describing individual aspects.

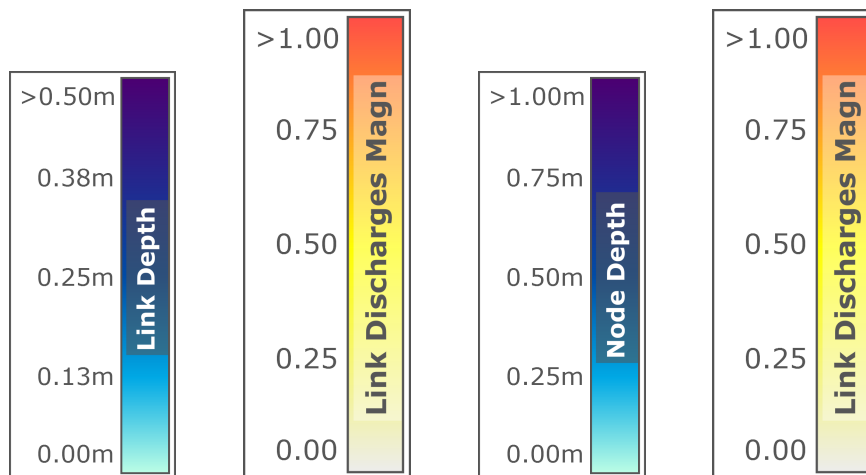


Figure 5.1: Legends of water colors visualizing depth of the water and discharge velocity.

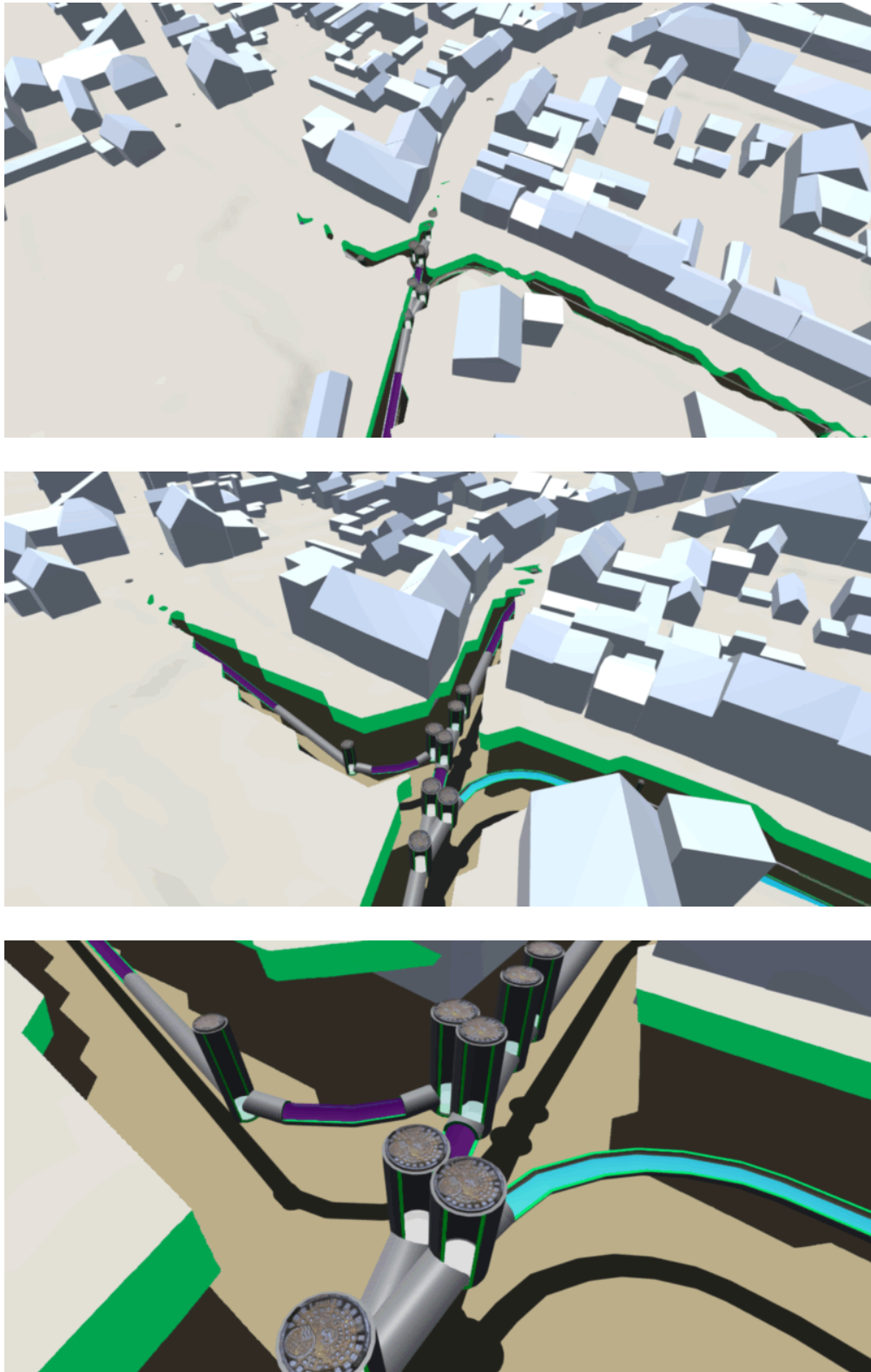


Figure 5.2: The first set of images demonstrates the terrain cutaway. As the camera is moving closer to the surface, the terrain is opening up more and the sewer network is exposed to the viewer. Below the actual terrain surface an artificial subsurface is positioned to catch shadows of the cut-open terrain and the sewer network.

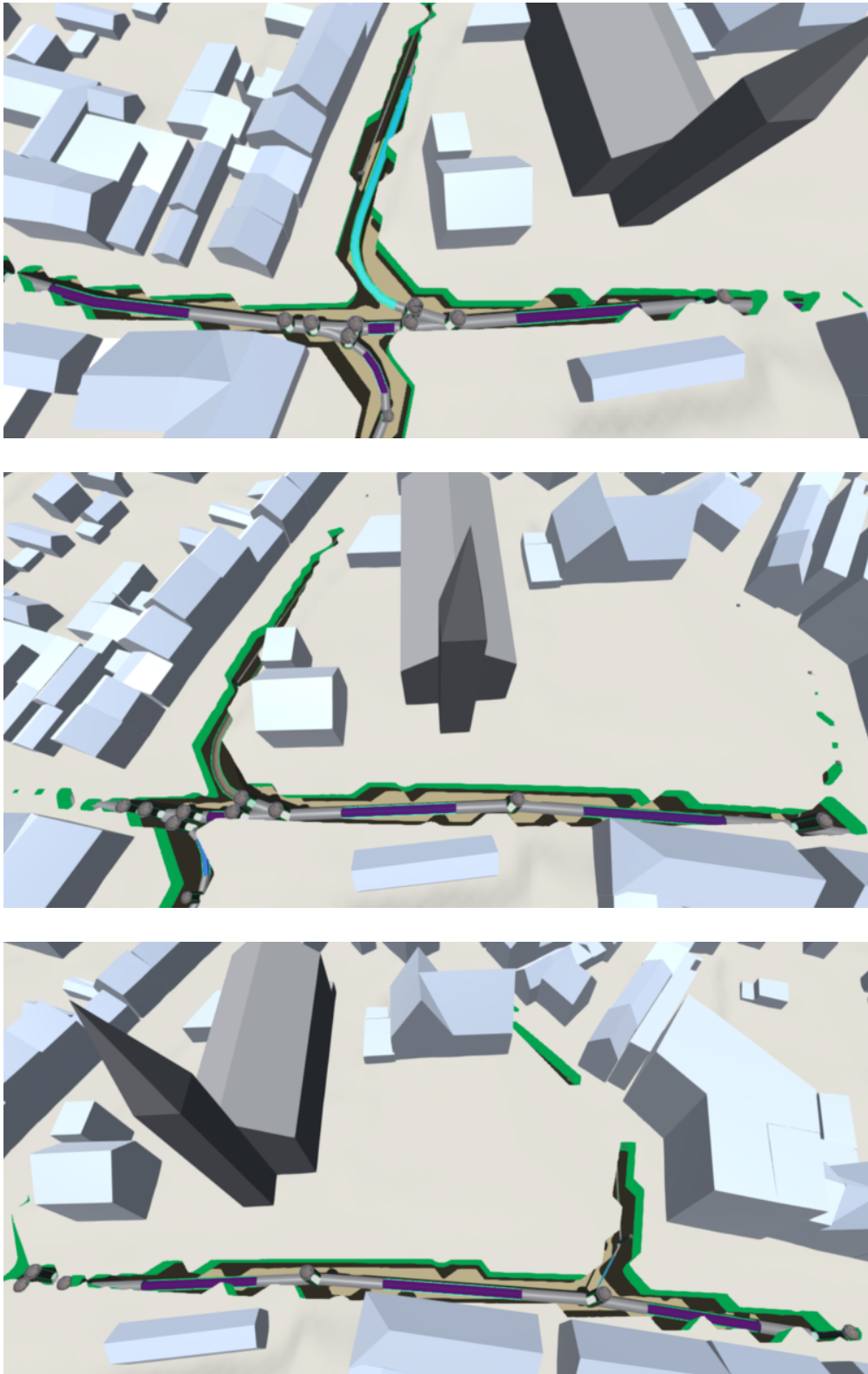


Figure 5.3: The second set of images demonstrates another feature of the terrain cutaway. As the camera is moving across the surface from left to right, the terrain cutaway is smoothly closing at the left end opening up at the right. Use the black church as a point of reference.

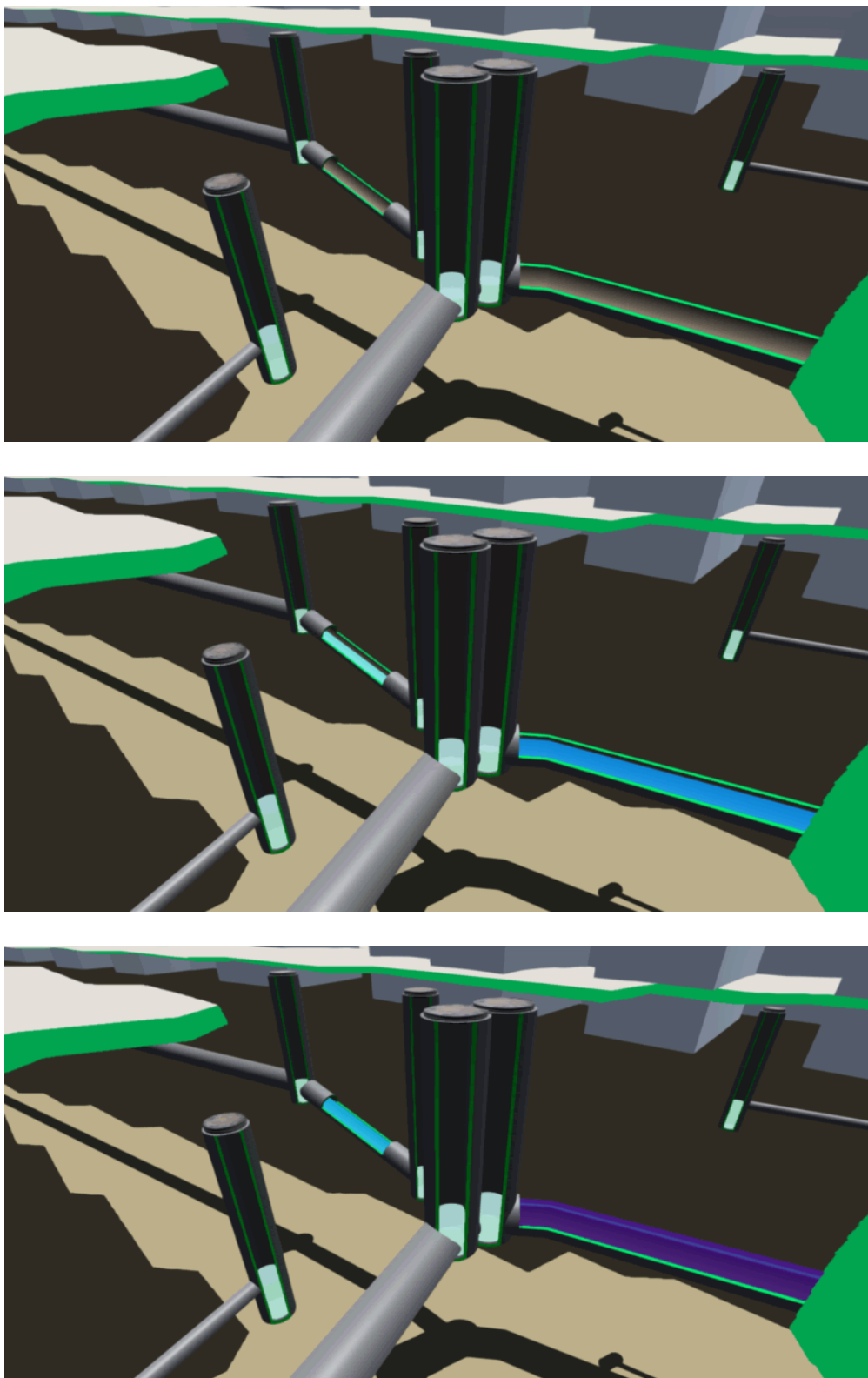


Figure 5.4: This set of images demonstrates the water filling up in the pipes as the flood simulation timeline progresses. Additionally it demonstrates the coloring of water depending on water depth. As the water depth increases it is colored accordingly. If you take a look on the corresponding legends in Figure 5.1 you might guess the absolute water depth.

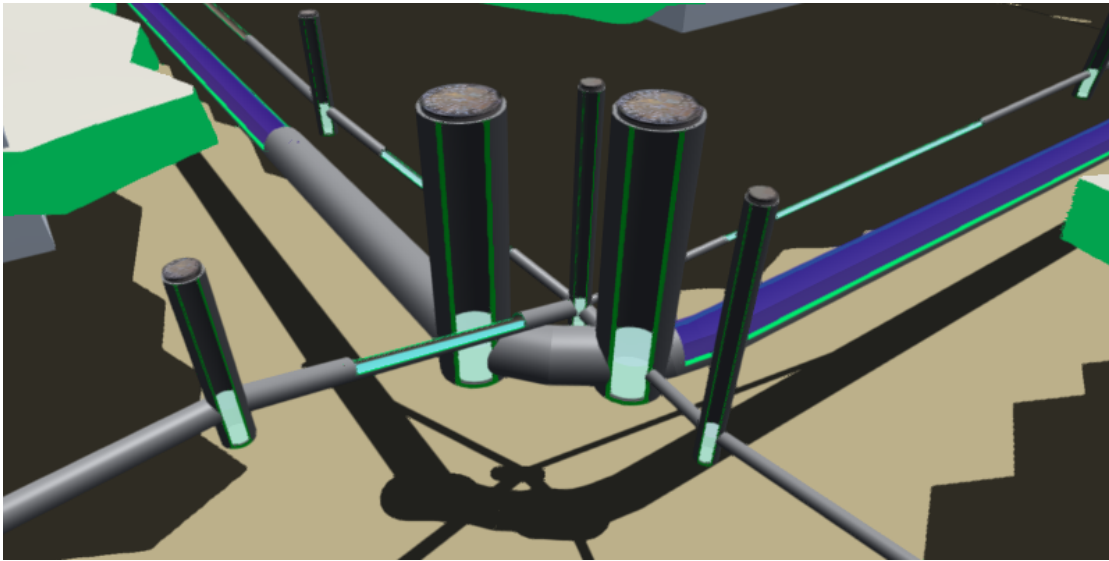


Figure 5.5: Another more complex scene of pipes and shafts with several different radii. The graphic shows the different sizes of pipes and shafts and how they are connected with each other. Demonstrates proper depth sorting.

Critical Reflection

This chapter aims to evaluate the proposed technique in quality and performance. The quality will be evaluated by comparing our results with several rules and design principles for cutaways that were reviewed in Section Occlusion Management Techniques 2.2. Further the performance is evaluated by the frames per second that each feature of the implementation achieves separately. Additionally the implementation was reviewed by visualization experts and their feedback will be summarized. The last section will critically reflect on open issues of the proposed technique.

6.1 Evaluation

To evaluate the quality of the proposed technique and the implementation it is reviewed if the set of rules by Diepstraten et al. [DWE03] is applicable to the billboarded cut-open sewage pipes.

Rule 2.1a suggests that inside and outside objects have to be distinguished from each other. This applies to our implementation since distinguishable colors are used for the inside and outside of the pipe. Rule 2.1b recommends that the cutout geometry is representable by the intersection of a few half spaces. Sewage pipes are cut open by an object-aligned box cut as the cutting convention by Li et al. [LRA⁺07] suggests. A box can apparently be described by a few half spaces. Object-aligned means that the part which is cut open and the cutout geometry is aligned on the same Cartesian coordinate system. In other words like Rule 2.1c describes: The cutout is located at or around the main axis of the outside object. Which is also true for the proposed technique. Rule 2.1e recommends to make the cut surface visible. Our implementation makes the cut surface visible by even emphasizing it with a high contrast color. For the cutaway of the sewage shafts the same rules as for the pipes are applicable. However, a different cutting convention by Li et al. is used. The convention is described as wedge cut, since it removes the shape of the wedge from the shafts cylinder.

Rule 2.1d suggest an optional jittering mechanism which was not applied to the sewage pipes or shafts but is used by the terrain cutaway. The cut surface of the terrain cutaway has been made visible as well and it is arguable that it is aligned with the sewer system. Thereby Rule 2.1c and Rule 2.1e are fulfilled. Certainly the cutout geometry used for the terrain can not be represented by a few half spaces.

6.2 Performance

The machine used for testing the implementation is a desktop computer with an AMD Phenom II X6 1090T processor running at 3200MHz stock frequency and 8GB of DDR3 memory. The video card is an AMD Radeon HD 6900 running at 880 MHz clock. The operating system is Microsoft Windows 10 x64. The whole dataset of the sewer system consists of 1190 sewage pipes with an average of 4.6 node positions per pipe. In addition it contains 1156 sewage shafts with exactly 2 nodes per shaft. The camera was positioned to render the whole sewer system. Running the implementation in Unity3D results in the following frame rates for each feature separately.

Feature	FPS	frame time	triangles	vertices	SetPass calls	Batches
Nothing	80	12ms	0	0	0	0
Pipes	78	13ms	39k	80k	12	0
Shafts	60	17ms	1.1M	1.2M	22	93
Vegetation	15	67ms	0.45M	0.83M	1	46
Buildings	30	31ms	0.7M	0.93M	2	209
Terrain	78	13ms	8.6M	6.6M	7	132

Batching in Unity3D means that static objects are combined into big meshes to reduce *SetPass* calls, which is basically a draw call. Another way to batch non-static objects is performed in case the meshes are small enough. Thereby similar vertices are grouped together on the CPU, sent back to the GPU and rendered in a single draw call[Uni17]. The table shows that the billboarded sewage pipes are rendered very efficiently as compared to the meshed shafts. If sewage pipes would be realized as polygonal meshes, they would have about twice as much vertices as the shafts show on the table. In addition to the fast rendering, the amount of vertices, and thereby the memory footprint on the GPU, is kept very low.

When viewing usual scenes with all features active, like the graphics show in the demonstration chapter, and moving the camera, the frame rates is capped to 60 frames per second. The application uses about 880MB of RAM.

6.3 Feedback from Domain Experts

Feedback from domain experts is always great to have. The architectural visualization experts Clemens Kraigher and Markus Stöger, who were introduced in Section 1.3

reviewed the Unity implementation and gave interesting feedback on the visual aspects of the software. Their impression is that the visualization of the sewer system is done very well. Spatial relationships of all structures are easy to comprehend. Furthermore the emphasizing of the cut surface colors makes it clear where something was cut away to reveal the interior. Therefore it is easy for the viewer to mentally reconstruct missing parts. Regarding the zigzag effect of the terrain cutaway, Kraigher mentioned that on a picture or a video he would prefer the zigzag effect to be cleared out. However, when moving the camera interactively he likes the effect of the terrain breaking up and to him it looks like a tectonic shift. In his opinion the biggest flaw of the implementation is the shading of the buildings and the terrain, which should be more detailed. In addition he mentioned that he would prefer the cut surface of the terrain not to be highlighted by a saturated color and rather choose a gray slightly darker than the terrain. From his perspective our approach of automatically cut open tube-like structures could be used for architectural visualizations only in very special cases.

6.4 Discussion of Open Issues

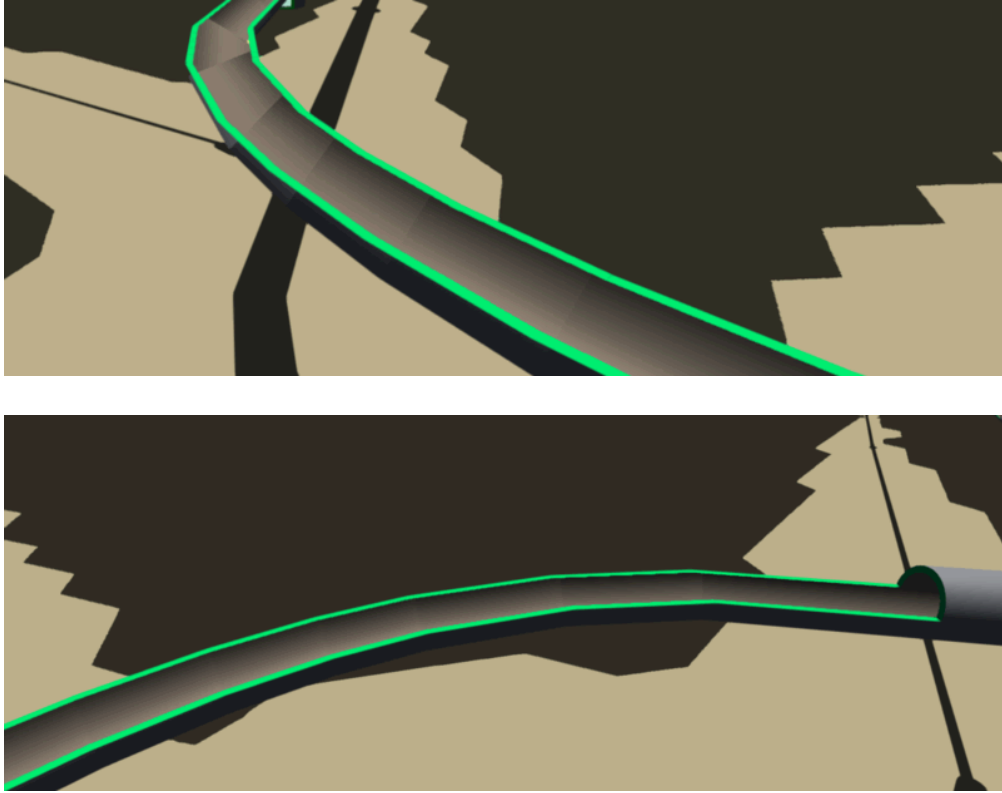


Figure 6.1: Comparison of a bended pipe viewed in two different angles. The first image shows the case where an unusual viewing angle causes the shading of the billboards to not match a tube-like structure.

Downsides of the billboarded sewage pipes are their inability to be used for sharply bended tubes, which would cause them to look flat and the impression of spatial capacity would be lost. Luckily sharp bends in sewage pipes would cause poor water flow rates. For this reason they are in general not bending in an angle which could impact the visual impression of the billboards negatively. However, if this happens a possible workaround is to replace too sharply bended parts of a billboarded sewage pipe with a polygon meshed conjunction. Still, as depicted in Figure 6.1, when viewing bended pipes in the implementation from a unusual angle, the billboards can cause the pipe to not look like a tube anymore. The impression of spatial capacity is thereby limited to the viewing angle. However, usually this viewing angle feels unnatural and does not achieve any special insights anyways.

Another issue is the intersection of cut open pipe segments. As shown in Figure 6.2 visual inconsistencies occur because the prepared ares in the stencil buffer are interfering with each other. To prevent this, such segments can be rendered without a cutaway. Alternatively a routine could be implemented to choose automatically the pipe segment for which it is more important to visualize its water flow and to be cut open. This kind of issue has only been found once in the whole data set.

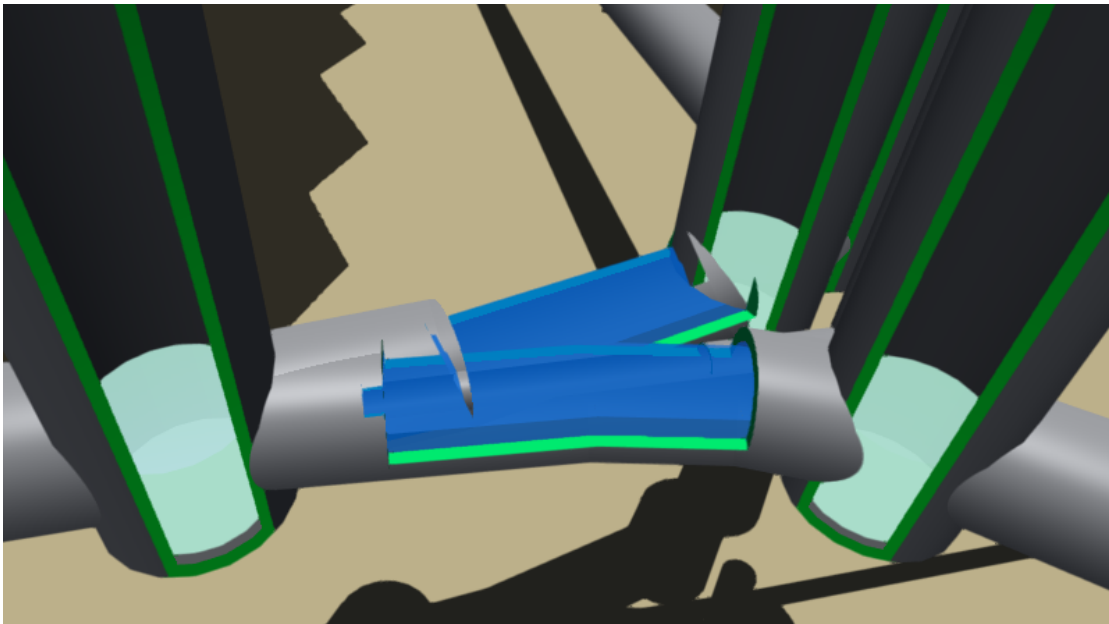


Figure 6.2: Three intersecting cut-open pipe segments causing visual inconsistencies.

Summary and Future Work

The thesis proposed a fast and efficient technique for billboarded and cut-open tube-like structures for real-time visualization rendering. The method is developed for, and demonstrated in the context of a flood simulation of a sewer system. Therefore the object of interest is the water inside of the sewage pipes. To perform a cutaway on a billboarded sewage pipe, a proxy geometry is used to prepare image-space areas via stencil buffer and to render the resulting cut surface immediately. According to the prepared areas the billboards are shaded to display either the outside or the inside of the sewage pipe. The shading is computed in the fragment program to simulate real cylindrical illumination. Both, billboards and the proxy geometry are constructed in the geometry shader, which only receives node positions of the pipe. The technique shows that visualizing cut-open tubular geometry can be done quite efficient with the use of modern GPU programming. Further the technique allows to interactively change the radius, wall thickness and coloration of the sewage pipes without expensive recalculations of meshed geometry due to an analytical approach. It seems that the use of billboards or impostors is gaining momentum in modern rendering tasks. As demonstrated by this thesis and others recently, the use of billboards is more and more frequently used to render detailed geometry, rather than only approximating it.

Future work includes to resolve the mentioned issues that occur when multiple cut-open segments of a sewage pipe are intersecting. In addition a routine to resolve visual inconsistencies at sharp bended pipes is considered to be useful. Future work also includes to add illuminated shading to the artificial thickness of the terrain cutaway. The proposed method has potential to be used for visualizing the interior of any tubular geometry. Applications are for example cut-open blood vessels, subway hoist ways and mining and traffic tunnels.

Bibliography

- [BF08] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics*, 27(5):1, 2008.
- [BGKG05] Stefan Bruckner, Sören Grimm, Armin Kanitsar, and M Eduard Gröller. Illustrative Context-Preserving Volume Rendering. *Proceedings of the Seventh Joint Eurographics / IEEE VGTC conference on Visualization*, 1:69–76, 2005.
- [BHW⁺07] Michael Burns, Martin Haidacher, Wolfgang Wein, Ivan Viola, and M Eduard Gröller. Feature Emphasis and Contextual Cutaways for Multimodal Medical Visualization. *Proceedings of the Eurographics/IEEE VGTC Symposium on Visualization*, pages 275–282, 2007.
- [Bir08] Jörn Birkmann. Globaler Umweltwandel, Naturgefahren, Vulnerabilität und Katastrophenresilienz. Notwendigkeit der Perspektivenerweiterung in der Raumplanung. *Raumforschung und Raumordnung (RuR)*, 1(January 2008):5–22, 2008.
- [CSG95] Kenneth Rohde Christiansen, Computing Science, and Rijksuniversiteit Groningen. The use of Imposters in Interactive 3D Graphics Systems. *Vide Science Technique Et Applications*, 1995.
- [DWE03] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive Cutaway Illustrations. In *Computer Graphics Forum*, volume 22, pages 523–532, 2003.
- [EEA11] EEA. Disasters in Europe: more frequent and causing more damage. <http://www.eea.europa.eu/highlights/natural-hazards-and-technological-accidents>, 2011. Accessed: 3-22-2017.
- [Eur07] European Parliament. Directive 2007/60/EC of the European Parliament and of the Council of 23 October 2007 on the assessment and management of flood risks. *Official Journal of the European Union*, L288:27–34, 2007.
- [Eur12] European Council. Multiannual Financial Framework (2014-2020)- Negotiating box. *15599/12*, pages 1–46, 2012.

- [Eur17] European Council. DECISION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the mobilisation of the European Union Solidarity Fund to provide assistance to the United Kingdom, Cyprus and Portugal. Brussels, 26.1.2017, COM(2017) 45 final, 2017.
- [FH12] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance Transforms of Sampled Functions. *Theory of Computing*, 8(1):415–428, 2012.
- [LHV12] Endre M Lidal, Helwig Hauser, and Ivan Viola. Design Principles for Cutaway Visualization of Geological Models. *Proc. Spring Conference on Computer Graphics (SCCG 2012)*, 1(212):53–60, 2012.
- [LRA⁺07] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics*, 26(3):31, 2007.
- [MAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. *Eurographics Workshop on Visual Computing for Biology and Medicine*, 2015.
- [Mic] Microsoft. DirectX Graphics and Gaming - Direct3D - HLSL - Geometry-Shader Object, online documentation. <https://msdn.microsoft.com/en-us/library/windows/desktop/bb509609.aspx>. Accessed: 3-22-2017.
- [MMS⁺16] M Le Muzic, P Mindek, J Sorger, L Autin, D S Goodsell, and I Viola. Visibility Equalizer Cutaway Visualization of Mesoscopic Biological Models. 35(3), 2016.
- [OLN] OLN - Office Le Nomade, Founders: Clemens Kraigher and Markus Stöger. High-end architectural and product visualization, brand of the IONOMO GmbH. <http://www.oln.at/>.
- [SKH⁺04] Marc Schirski, Torsten Kuhlen, Martin Hopp, Philipp Adomeit, Stefan Pischinger, and Christian Bischof. Efficient visualization of large amounts of particle trajectories in virtual environments using virtual tubelets. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry - VRCAI '04*, volume 1, pages 141–147, 2004.
- [SRCW13] Bernhard Sadransky, Hrvoje Ribicic, Robert Carnecky, and Jürgen Waser. Visdom Mobile: Decision Support On-site Using Visual Simulation Control. *Proceedings of the 29th Spring Conference on Computer Graphics*, pages 099:99—099:106, 2013.

- [Uni17] Unity Technologies. Unity3D Documentation: Draw call batching. <https://docs.unity3d.com/Manual/DrawCallBatching.html>, 2017. Accessed: 3-22-2017.
- [VKG04] I. Viola, A. Kanitsar, and M.E. Groller. Importance-driven volume rendering. *IEEE Visualization 2004*, pages 139–145, 2004.
- [WFRB11] Jürgen Waser, Raphael Fuchs, Hrvoje Ribičić, and Günter Blöschl. Visuelle Aktionsplanung im Hochwassermanagement mit Visdom. *Forum für Hydrologie und Wasserbewirtschaftung/FgHW*, 30.11:280-286, 2011.