BUILT BY ALGORITHMS – STATE OF THE ART REPORT ON PROCEDURAL MODELING –

C. Schinko[a,b]*, U. Krispel[a,b], T. Ullrich[a,b], D. Fellner[b,c]

[a]Fraunhofer Austria Research GmbH, Visual Computing
[b]Institute of ComputerGraphics and KnowledgeVisualization (CGV), TU Graz, Austria
[c]GRIS, TU Darmstadt & Fraunhofer IGD, Darmstadt, Germany

Commission V, WG V/4

KEY WORDS: Generative Modeling, Procedural Modeling, Inverse Modeling, Modeling Applications, Shape Description, Language Design

ABSTRACT:

The idea of generative modeling is to allow the generation of highly complex objects based on a set of formal construction rules. Using these construction rules, a shape is described by a sequence of processing steps, rather than just by the result of all applied operations: Shape design becomes rule design. Due to its very general nature, this approach can be applied to any domain and to any shape representation that provides a set of generating functions. The aim of this report is to give an overview of the concepts and techniques of procedural and generative modeling as well as their applications with a special focus on Archaeology and Architecture.

1. INTRODUCTION

The task of generating highly complex objects based on a set of formal construction rules is called generative modeling (Krispel et al., 2014). In contrast to classical modeling, where the object is just the end result of applied operations, this modeling paradigm describes a shape by a sequence of processing steps. The result is a paradigm shift from shape design to rule design. This general approach can be applied to many domains.

1.1 Ruler and Compass

Geometry from the days of the ancient Greeks placed great emphasis on problems of constructing various geometric figures using only a ruler without markings (to draw lines) and a compass (to draw circles). Ruler-and-compass constructions are based on EUCLID's axioms (Heiberg, 2007) using points, lines and circles that have already been constructed. The resulting geometric primitives together with the ruler-and-compass constructions are the first algorithmic descriptions of generative models.

When a line is considered constructed when its two endpoints are located, all constructions possible with a compass and straightedge can be done with a compass alone. The reverse is also true, since JACOB STEINER showed that all constructions possible with straightedge and compass can be done using only a straightedge, as long as a fixed circle and its center have been drawn beforehand. Such a construction is known as a Steiner construction.

The long history of geometric constructions (Martin, 1998) is also reflected in the history of civil engineering and architecture (Mitchell, 1990). Gothic architecture, especially window tracery, exhibits a good example of these constructions. Their complexity is achieved by combining only a few basic geometric patterns. SVEN HAVEMANN and DIETER W. FELLNER show how constructions of prototypic Gothic windows can be formalized using generative modeling techniques (Havemann and Fellner, 2004). By combining modular construction rules it is possible that complex configurations can be obtained from elementary constructions. The different combinations of specific parametric features can be grouped together, leading to the concept of styles. A differentiation between basic shape and appearance allows, for example, the creation of ornamental decoration (Thaller et al., 2013a). This leads to an extremely compact representation for a whole class of shapes (Berndt et al., 2005a).

1.2 Civil Engineering

Because the generative modeling approach is very general, it can be applied to any domain and is not restricted to shape representations (Chakrabarti et al., 2011), (Compton and Mateas, 2006). The discipline of civil engineering focuses on computer-aided design, shape design, and 3D modeling.

Each design process which involves repetitive tasks is perfectly suited for a generative approach. Engineering processes can be classified in repetitive and creative processes. In contrast to creative processes, repetitive ones consist of nearly identical tasks and are therefore independent of creative decisions. This is a precondition for modeling them in a system of rules as demonstrated by GERALD FRANK (2012): Liebherr manufactures and sells an extensive range of products including different kinds of cranes. Each crane has to be partially or fully engineered tailored to the needs of the customer. Nevertheless, the design process of ascent assemblies is based on repetitive tasks that are described by a set of invariant rules. These rules have been modeled and stored by Liebherr. The integration into the existing CAD pipeline now allows a construction engineer to create ascent assemblies only by determine the defining parameters and filling out the corresponding input fields in a user interface. Using the procedural approach, the efforts of engineering ascent assemblies have been reduced to 10%.

1.3 Natural Patterns

In today's procedural modeling systems, scripting languages and grammars are often used to create a set of rules to achieve a description of an object or pattern. Early systems based on grammars were Lindenmayer systems, short L-systems, named after ARISTID LINDENMAYER (Prusinkiewicz and Lindenmayer,

*Corresponding author.

1990). They were successfully used for modeling plants (Deussen and Lintermann, 2005) or fractal structures (Mandelbrot, 1982). Given a set of string rewriting rules, complex strings are created by applying these rules to simpler strings. Starting with an initial string the predefined set of rules form a new, possibly larger string. In order to use L-systems to model geometry an interpretation of the generated strings is necessary.

The modeling power of L-systems was limited to creating fractals and plant-like branching structures. This limitation lead to the introduction of parametric L-systems. The idea is to associate numerical parameters with L-system symbols to address continuous phenomena which were not covered satisfactorily by L-systems alone.

In combination with additional 3D modeling techniques, Lindenmayer systems can be used to generate complex geometry. ROBERT F. TOBLER et al. (2002) introduce a combination of subdivision surfaces, fractal surfaces, and parametrized L-systems to create models of natural phenomena. Different combinations can be used at each level of resolution. Since the whole description of such multi-resolution models is procedural, their representation is very compact and can be exploited by level-of-detail renderers.

This trade-off between data storage and computation time can be found in various fields of computer graphics, e.g. the tessellation of curved surfaces specified by a few control points directly on the GPU. The result is low storage costs allowing the generation of complex models only when needed, while also reducing memory transfer overheads. Although L-systems are parallel rewriting systems, derivation through rewriting leads to very uneven workloads. Since the interpretation of an L-system is an inherently serial process, they are not straightforwardly applicable to parallel processing. In 2010, MARKUS LIPP et al. (2010) presented a solution to this algorithmic challenge.

2. LANGUAGES & GRAMMARS

Scripting languages have been designed for a special purpose, e.g., for client-side scripting in a web browser. Nowadays, scripting languages are used for many different applications. JavaScript, for example, is used to animate 2D and 3D graphics in VRML (Brutzman, 1998) and X3D (Behr et al., 2007) files. It checks user forms in PDF files (Breuel et al., 2011), controls game engines (Di Benedetto et al., 2010), configures applications, defines 3D shapes (Schinko et al., 2011a), and performs many more tasks. According to JOHN K. OUSTERHOUT (1998) scripting languages use a higher level of abstraction compared to system programming languages as they are often typeless and interpreted to emphasize the rapid application development purpose. System programming languages, on the other hand, are designed for creating algorithms and data structures based on low-level data types and memory operations. Consequently, graphics libraries (OpenGL Architecture, 1993), shaders (NVidia, n.d.) and scene graph systems (Reiners et al., 2002), (Voß et al., 2002) are usually written in C/C++ dialects (Eckel, 2003), whereas procedural modeling frameworks incorporate scripting languages such as Lua, JavaScript, etc.

2.1 Language Processing & Compiler Construction

For the evaluation of procedural descriptions typically techniques used for description of formal languages and compiler construction are used (Parr, 2010). There is a wide range of different concepts of languages to describe a shape including all kinds of linguistic concepts (Chomsky, 1956). The main categories to describe a shape are

- *rule-based:* using substitutions and substitution rules to generate complex structures out of simple starting structures (Özkar and Kotsopoulos, 2008), (Krecklau et al., 2010), (Müller et al., 2006c), (Snyder and Kajiya, 1992).
- *imperative and scripting-based:* using a scripting engine and techniques from predominant programming languages (Havemann, 2005), (Schinko et al., 2011a), (Krecklau and Kobbelt, 2011), or
- *GUI and dataflow-based:* using new graphical user interfaces (GUI) and intelligent GUIs to detect structures in modeling tasks, which can be mapped onto formal descriptions (Lipp et al., 2008), (Thaller et al., 2012).

The general principles of formal descriptions and compiler construction are the same in all cases – independent of ahead-of-time compilation, just-in-time compilation or interpretation (Schinko et al., 2012). In the first stage of the compilation process, the input source code is passed to lexer and parser. A first step here is to convert a sequence of characters into a sequence of tokens, which is done by special grammar rules forming the lexical analysis. Typically only a limited number of characters is allowed for an identifier: all characters `A-Z`, `a-z`, digits `0-9` and the underscore `_` are allowed with the condition that an identifier must not begin with a digit or an underscore. The lexer rules are embedded in another set of rules – the parser rules. They are evaluating the resulting sequence of tokens to determine their grammatical structure. The complete grammar is of hierarchical structure and consists of rules for analyzing all possible statements and expressions that can be formed in the language, thus forming the syntactic analysis.

For each available language construct a set of rules ensures syntactic correctness and incorporates mechanisms to report possible syntactic errors and warnings. These rules are also used to create the intermediate AST structure that is a representation of the input source code to be used for the next stage: semantic analysis. Once all statements and expressions of the input source code are collected in the AST, a tree walker checks their semantic relationships for errors and warnings. After performing all compile-time checks, a translator uses the AST to generate platform-specific files possibly involving other intermediate structures.

As mentioned in the Introduction, the first procedural modeling systems were L-systems. Later on, L-systems were used in combination with shape grammars to model cities. YOGI PARISH and PASCAL MÜLLER (2001) presented a system that generates a street map enriched with geometry for buildings using a number of image maps as input. The resulting framework called *CityEngine* is a modeling environment for the shape grammar *CGA Shape*. MARKUS LIPP et al. (2008) presented another modeling approach based on *CGA Shape* following the notation of PASCAL MÜLLER (2006). It enables more direct local control of the underlying grammar by introducing visual editing. Principles of semantic and geometric selection are combined as well as functionality to store local changes persistently over global modifications.

SVEN HAVEMANN (2005) takes a different approach to generative modeling. He proposes a stack based language called *Generative Modeling Language* (GML). The postfix notation of the language is very similar to that of *Adobe Postscript*. High-level shape operations are created by using low-level shape functionality. A number of applications are based on the GML platform because it is easily extensible and offers an integrated visualization engine. Current efforts in the context of the GML are devoted to directly creating interactive generative visualizations for the web.

Generative modeling inherits methodologies of 3D modeling and programming (Ullrich et al., 2008a), which leads to drawbacks in usability and productivity. The need to learn and use a programming language is a significant inhibition threshold especially for non-computer scientists. The choice of the scripting language has a huge influence on usability and effectiveness of procedural modeling. *Processing* is a good example of how an interactive, easy to use, yet powerful, development environment can open up new user groups. It has been initially created to serve as a software sketchbook and to teach students fundamentals of computer programming. It quickly developed into a tool that is used for creating visual arts (Reas et al., 2007).

Processing is a Java-like interpreter offering new graphics and utility functions together with some usability simplifications. The large community behind the tool produced libraries to facilitate computer vision, data visualization, music, networking, and electronics. The success of *Processing* is based on two factors: the simplicity of the programming language and the interactive experience. Instant feedback of the scripting environments allow the user to program via "trial and error".

2.2 Scripting Languages for Generative Modeling

There are many different programming paradigms in software development that are also used in the field of generative modeling, where some paradigms emerged to be useful for specific domains.

imperative: Many generative models are described using classical programming paradigms: A programming language is used to generate a specific object possibly using a library that utilizes some sort of geometry representation and operations to perform changes. Any modeling software that is scriptable by an imperative language or provides some sort of API falls into this category.

dataflow based: A generative description can be represented by a directed graph of the data flowing between operations. This graph representation also allows for a graphical representation; Visual Programming Languages (VPL) allow to create a program by linking and modifying visual elements. Many VPL's are based on the dataflow paradigm. Examples in the domain of generative modeling are the Grasshopper3D[1] plug-in for the Rhinoceros3D[2] modeling suite, or the work of GUSTOVA PATOW et al. (2012) built on top of the procedural modeler Houdini[3].

rule based systems: Another different representation for generative modeling are rule-based systems. These systems provide a declarative description of the construction behavior of a model by a set of rules. An example are L-Systems, as described in the Introduction. Furthermore, the seminal work of GEORGE STINY and JAMES GIPS (1971) introduced shape grammars, as a formal description of capturing the design of paintings and sculptures. Similar to formal grammars, shape grammars are based on rule replacement.

Shape Grammars
In the classical definition of GEORGE STINY and JAMES GIPS, a shape grammar is the 4-tuple $SG = (V_T, V_M, R, I)$, where V_T a set of shapes, $V_T{}^*$ denotes the set of the shapes of V_T with any scale or rotation. V_M is a finite set of *non-terminal* shapes (markers) such that $V_T{}^* \cap V_M = \emptyset$. R denotes the set of rules, which consists of pairs (u, v), such that $u = (s, m)$ consists of a shape $s \in V_T{}^*$ combined with a marker of $m \in V_M$, and v is a shape consisting of either

- $v = s$
- $v = (s, \tilde{m})$ with $\tilde{m} \in V_M$
- $v = (s \cup \tilde{s}, \tilde{m}$ with $\tilde{s} \in V_T{}^*$ and $\tilde{m} \in V_M$

Elements of the set $V_T{}^*$ that appear in rules of R are called *terminal shapes*. I is called the *initial shape*, and typically contains an $u \in (u, v) \in R$. The final shape is generated from the shape grammar by starting with the initial shape and applying matching rules from R: for an input shape and a rule (u, v) whose u matches a subset of the input, the resulting shape is another shape that consists of the input shape with the right side of the rule substituted in the matching subset of the input. The matching identifies a geometric transformation (scale, translation, rotation, mirror) such that u matches the subset of the input shape and applies it to the right side of the rule. The *language* defined by a shape grammar SG is the set of shapes that will be generated by SG that do not contain any elements of V_M.

Split Grammars
The work of PETER WONKA et al. (2003) applied the concepts of shape grammars to derive a system for generative modeling of architectural models. This system uses a combination of a spatial grammar system (split grammar) to control the spatial design and a control grammar, which distributes the design ideas spatially (e.g. set different attributes for the first floor of a building). Both of these grammars consist of rules with attributes that steer the derivation process. The grammar consists of two types of rules: *split* and *convert*. The *split* rule is a partition operation which replaces a shape by an arrangement of smaller shapes that fit in the boundary of the original shape. The *convert* rule replaces a shape by a different shape that also fits in the boundary of the original shape.

This system has further been extended by the work of PASCAL MÜLLER et al. (2006), which introduced a *component split* to extend the split paradigm to arbitrary 3d meshes, as well as occlusion queries and snap lines to model non-local influences of rules. For example, two wall segments that intersect each other should not produce windows such that the window of one wall coincides with the other wall, therefore occlusion queries are used to decide if a window should be placed or not.

The derivation of a split grammar, starting from an initial shape, yields a tree structure, which suggests that the derivation can be speed up by a parallel implementation, which has been shown by JEAN-EUDES MARVIE et al. (2012). Parallel generation is especially useful in an urban context, with scenes with high complexity and detail. The work of LARS KRECKLAU et al. (2013) used gpu accelerated generation in the context of generating and rendering high detailed building façades; the work of ZHENGZHENG KUANG et al. (2013) proposes a memory-efficient procedural representation of urban buildings for real-time visualization.

With more advanced shape grammar systems, the non-local influences are a problem because they introduce dependencies between arbitrary nodes of the derivation tree. Recent work by MARKUS STEINBERGER et al. (2014) shows how to overcome this problem in an GPU implementation. Furthermore, the same authors presented methods to interactively generate and render only the visible part of a procedural scene using procedural occlusion culling and level of detail (Steinberger et al., 2014b)

[1] http://www.grasshopper3d.com
[2] http://www.rhino3d.com
[3] http://www.sidefx.com

3. MODELING BY PROGRAMMING

3D objects consisting of organized structures and repetitive forms are well suited for procedural descriptions, e.g. by the combination of building blocks or by using shape grammars.

3.1 Building Blocks & Elementary Data Structures

Creating shapes with elementary data structures requires the definition of modeling operations. Depending on the underlying representation, certain modeling operations are difficult or impossible to implement. The selection of operations for these data structures are manifold and can be grouped as follows:

- *Instantiations* are operations for creating new shapes.
- *Binary Creations* are operations involving two shapes such as constructive solid geometry (CSG) operations.
- *Deformations* and *Manipulations* stand for all deforming and modifying operations like morphing or displacing.

Building blocks can also be regarded as modeling operations. When creating an algorithmic description of a shape, an important task is to identify inherent properties and repetitive forms. These properties must be accounted for in the structure of the description. Identified subparts or repetitive forms are best mapped to functions in order to be reusable. However, the true power of an algorithmic description becomes obvious when parameters are introduced for these functions. Even if only used to position a subpart at a different location. From that point on, the algorithmic description no longer stands for a single object, but for a whole object family.

3.2 Architectural Modeling with Procedural Extrusions

This method utilizes the paradigm of footprint extrusion to automatically derive geometry from a coarse description. Input to this system are polygons whose segments can be associated with an extrusion profile polygon. The system utilizes the weighted straight skeleton method (Aurenhammer, 2008) to calculate the resulting geometry. Examples can be seen in Figure 1.

The growing demand for new building models for virtual worlds, games, and movies, makes the easy and fast creation of modifiable models more and more important (Watson and Wonka, 2008). Nevertheless, 3D modeling of buildings can be a tedious task due to their sometimes complex geometry (Whiting et al., 2009). For historic buildings, especially the roofs can be challenging. JOHANNES EDELSBRUNNER et al. (2014) present a new method of combining simple building solids to form more complex buildings, and give an emphasis on the blending of roof faces. Their method can be integrated in common pipelines for procedural modeling of buildings and extends their expressiveness compared to existing methods.

3.3 Deformation Aware Shape Grammars

Generative models based on shape and split grammar systems often exhibit planar structures. This is the case because these systems are based on planar primitives and planar splits. There are many geometric tools available in modeling software to transform planar objects into curved ones, e.g. *free-form deformation* (Sederberg and Parry, 1986). Applying such a transformation as a post-processing step might yield undesirable results. For example, if a planar facade of a building is bent into a curved shape,

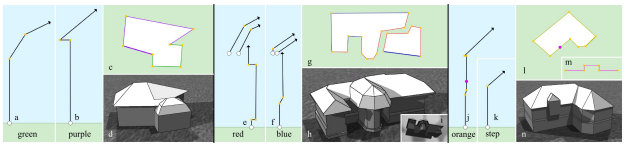

Figure 1: The work of TOM KELLY and PETER WONKA (2011) offers a framework to specify the geometry of a building by extrusion profiles. The segments of footprint polygons (*c*) are associated with extrusion profiles - the green segments are associated to the profile *a*, the purple segments to the profile *b*. The resulting geometry can be seen in *d*.

the windows inside the façade will have a curved surface as well. Another possibly unwanted property arises when an object is deformed by scaling: the windows on a façade would have different appearances.

RENÉ ZMUGG et al. (2013) introduced *deformation aware shape grammars*, which integrate deformation information into grammar rules. The system still uses established methods utilizing planar primitives and splits, however, measurements that determine the available space for rules are performed in deformed space. In this way, deformed splits can be carried out, the deformation can be baked at any point to allow for straight splits in deformed geometry. An example is shown in Figure 2.

3.4 Procedural Shape Modeling

The effectiveness of procedural shape modeling can be demonstrated with mass customization of consumer products (Berndt et al., 2012). A generative description composed of a few well-defined procedures can generate a large variety of shapes. Furthermore, it covers most of the design space defined by an existing collection of designs – in this case wedding rings.

The basic shape of most rings can be defined using a profile polygon, the angular step size defined by the number of supporting profiles to be placed around the ring's center, the radius, and a vertex transformation function. A ring's design variations are decomposed into a set of transformation functions. Each function transforms selected parts of the profile in a certain way. Effects can be combined by calling a sequence of different transformations. The creation of the basic shape is separated from optional steps to create engravings, change materials, or add gems. Engravings are implemented as per-vertex displacements (to maintain the option for 3D-printing) and can be applied on quadrilateral parts of the ring's mesh using half-edges to specify position and spatial extend.

Materials like gold, silver, and platinum are used for wedding rings. Their surfaces can be treated with various finishing tech-

Figure 2: Deformation aware shape grammars allow the integration of free-form deformation into a grammar-based system based on planar primitives and splits. An undeformed building with rooms (top image) is deformed using two different deformations (middle, bottom).

niques like polishing, brushing, or hammering. In order to account for these effects, a per-pixel shading model is used featuring anisotropic highlights. By using a cube map, visually appealing reflections are created and predefined surface finishes can be applied using normal mapping techniques. Procedural gem instances can also be placed on the ring.

The presented approch is used in a hardware accelerated server-side rendering framework (Schinko et al., 2014), which has been included in an online system called *REx* by *JohannKaiser*. It offers intuitive web interface for configuring and visualizing wedding rings.

This work demonstrates the efficiency of procedural shape modeling for the mass customization of wedding rings. The presented generative description is able to produce a large variety of wedding rings. Figure 3 shows a few results of the parametric toolkit.

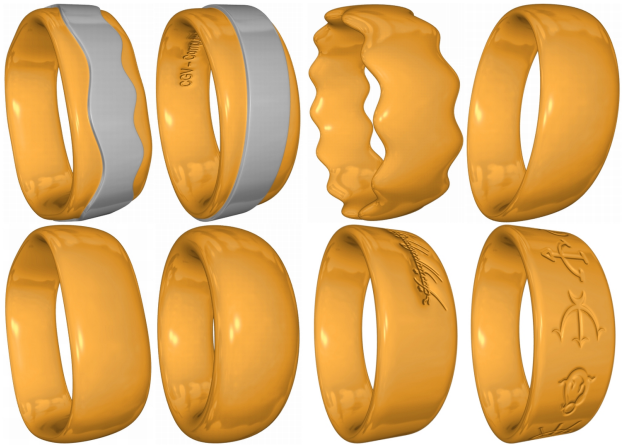

Figure 3: The presented generative description is able to produce a large variety of wedding rings. Features like engravings, recesses, different materials, unusal forms and gems can be created and customized.

3.5 Variance Analysis

The analysis and the visualization of differences of similar objects is important in many research areas: scan alignment, nominal/actual value comparison, and surface reconstruction to name a few. In computer graphics, for example, differences of surfaces are used to validate reconstruction and fitting results of laser scanned surfaces. Scanned representations are used for documentation as well as analysis of ancient objects revealing smallest changes and damages. Analyzing and documentation tasks are also important in the context of engineering and manufacturing to check the quality of productions.

CHRISTOPH SCHINKO et al. (2011) contribute a comparison of a reference / nominal surface with an actual, laser-scanned data set. The reference surface is a procedural model whose accuracy and systematics describe the semantic properties of an object, whereas the laser-scanned object is a real-world data set without any additional semantic information. The first step of the process is to register a generative model (including its free parameters) to a laser scan. Then, the difference between the generative model and the laser scan is stored in a texture, which can be applied to all instances of the same shape family.

A generative models represent an ideal object rather than a real one. The combination of noisy 3D data with an ideal description enhances the range of potential applications. This bridge between both the generative and the explicit geometry description is very important: it combines the accuracy and systematics of generative models with the realism and the irregularity of real-world data as pointed out by DAVID ARNOLD (2006). Once the procedural description is registered to a real-world artifact, we can use the fitted procedural model to modify a 3D shape. In this way we can design both low-level details and high-level shape parameters at the same time.

3.6 Semantic Modeling

In the context of digital libraries, semantic meta data plays an important role. It provides semantic information that is vital for digital library services: indexing, archival, and retrieval. Depending on the field of application, meta data can be classified according to the following criteria (Ullrich et al., 2010b):

Data Type The data type of the object can be of any elementary data structure (e.g. Polygons, NURBS, Subdivision Surfaces, ...).

Scale of Semantic Information This property describes, whether meta data is added for the entire data set or only for a sub part of the object.

Type of Semantic Information The type of meta data can be descriptive (describing the content), administrative (providing information regarding creation, storing, provenance, etc.) or structural (describing the hierarchical structure).

Type of creation The creation of the semantic information for an object can be done manually (by a domain expert) or automatically (e.g. using a generative description).

Data organization The two basic concepts of storing meta data are storing the information within the original object (e.g. EXIF data for images), or storing it separately (e.g. using a database).

Information comprehensiveness The comprehensiveness of the semantic information can be declared varying from low to high in any gradation.

Many concepts for encoding semantic information can be applied to 3D data, unfortunately only a few 3D data formats support semantic markup (Settgast, 2013):

Collada The XML-based Collada format allows storing meta data like title, author, revision etc. not only on a global scale but also for parts of the scene. This file format can be found in Google Warehouse where meta data is, for example, used for geo-referencing objects.

PDF 3D PDF 3D allows to store annotations separated from the 3D data even allowing annotating the annotations. An advantage is that the viewer application is widely spread and PDF documents are the quasi standard for textual documents.

Due to the persistent naming problem, a modification of the 3D model can break the integrity of the semantic information. Any change of the geometry can cause the referenced part of the model to no longer exist or being changed. There are a lot of examples for semantic modeling in various contexts (Boulch et al., 2013), (Haegeler et al., 2009), (Mendez et al., 2008), (Thaller et al., 2013b), (Van Gool et al., 2013), (Yong et al., 2012).

4. INVERSE MODELING

The full potential of generative techniques is revealed when the inverse problem is solved; i.e. what is the best generative description of one or several given instances of an object class? This problem can be interpreted in different ways. The simplest way is to create a generative model out of a given 3D object and to store it in a geometry definition file format. Obviously, this is not the desired result as the generative model can only represent a single object, not a family of objects.

4.1 Parsing shape grammars

Shape grammars can be used to describe the design space of a class of buildings / façades . An interesting question in this context is: given a set of rules and measurements of a building, typically photographs or range scans, which application of rules yields the measurements? Here, the applied rules can also be seen as *parse tree* of a given input.

The work of HAYKO RIEMENSCHNEIDER et al. (2012) utilizes shape grammars to enhance the results of a machine learning classifier that is pre-trained to classify pixels of an orthophoto of a façade into categories like windows, walls, doors and sky. The system applies techniques from formal language parsing to parse a two-dimensional split grammar consisting of horizontal and vertical splits, as well as repetition and symmetry operations. For the reduction of the search space, an irregular grid is derived from the classifications, and the parsing algorithm is applied to yield the most probable application of rules that yields a classification label per grid cell. These parse trees can easily be converted into procedural models.

FUZHANG WU et al. (2014) also address the problem of how to generate a meaningful split grammar explaining a given facade layout. Given a segmented facade image, the system uses an approximate dynamic programming framework to evaluate if a grammar is a meaningful description. However, the work does not contribute to the problem of facade image segmentation.

4.2 Model synthesis

PAUL MERELL and DINESH MANOCHA (2008) present an approach that given an object (i.e. a mesh) and constraints, derives a locally similar object. This method is related to texture synthesis. It computes a set of acceptable states, according to several types of constraints and constructs parallel planes that correspond to faces orientations of the input model. The intersections of these planes yield possible vertex positions in the output model. Acceptable states are assigned to a vertex while incompatible states are removed in its neighbourhood. The system terminates, if every vertex has been assigned a state.

4.3 Inverse procedural modeling of trees

The method proposed by ONDREJ STAVA et al. (2014) estimates the parameters of a stochastic tree model, given polygonal input tree models. This is done in such a way that the stochastic model produces trees similar to the input. The parameters are estimated using Markov Chain Monte Carlo (MCMC) optimization techniques. A statistical growth model consisting of 24 geometrical and environmental parameters is used. The authors propose a similarity measure between the statistical model and a given input mesh that consists of three parts: *shape distance*, measuring the overall shape discrepancy, *geometric distance*, reflecting the statistics of geometry of its branches, and *structural distance*, encoding the cost of transforming a graph representation of the statistical tree model into a graph representation of the input tree model. The MCMC method has also been applied by other methods to find parameters of a statistical generative model: (Talton et al., 2011), (Vanegas et al., 2012), (Yu et al., 2011).

4.4 Parameter Fitting and Shape Recognition

TORSTEN ULLRICH and DIETER W. FELLNER (2011) presented an approach that uses generative modeling techniques to describe a class of objects and to identify objects in real-world data e.g. laser scans. A point cloud P and a generative model M are the input data sets of the algorithm. It answers the questions

1. whether the point cloud can be described by the generative model and if so,
2. what are the input parameters x_0 such that $M(x_0)$ is a good description of P.

A hierarchical optimization routine based on fuzzy geometry and a differentiating compiler is used. The complete generative model description $M(x_1, \ldots, x_k)$ (including all possibly called subroutines) is differentiated with respect to the input parameters. This differentiating compiler offers the possibility to use gradient-based optimization routines in the first place. Without partial derivatives many numerical optimization routines cannot be used at all or in a limited way.

5. SEMANTIC ENRICHMENT

The increasing number of (3D) documents makes digital library services become more and more important. A digital library provides markup, indexing, and retrieval services based on available metadata. In a simple case, metadata is of the Dublin Core (1995) type: title, creator/author, time of creation, etc. This is insufficient for large collections of 3D objects, because of their versatility and rich structure.

Scanned models are used in raw data collections, for documentation archival, virtual reconstruction, historical data analysis, and for high-quality visualization for dissemination purposes (Settgast et al., 2007). Navigation and browsing through the geometric models should be possible on the semantic level - this requires higher-level semantic information. The need for semantic information becomes immediately clear in the context of electronic data exchange, storage and retrieval (Fellner, 2001), (Fellner et al., 2007). The problem of 3D semantic enrichment is closely related to the shape description problem (Maybury, 2012):

How to describe a shape and its structure on a higher, more abstract level?

The traditional way of classifying objects, pursued both in mathematics and, in a less formal manner, in dictionaries, is to define a class of objects by listing their distinctive properties. This approach is hardly realizable because of the fact that definitions cannot be self-contained. They depend on other definitions, which leads to circular dependencies that cannot be resolved automatically by strict reasoning, but rely on intuitive understanding at some point.

An alternative, non-recursive approach for describing shape uses examples. Each entry in a picture dictionary is illustrated with

a photo or a drawing. This approach is widely used, for example in biology for plant taxonomy. It avoids listing an exhaustive list of required properties for each entry. However, it requires some notion of similarity, simply because the decision whether object x belongs to class A or B requires measuring the closeness of x to the exemplars $a \in A$ resp. $b \in B$. This decision can be reached by a classifier using statistics and machine learning (Bishop, 2007), (Ulusoy and Bishop, 2005). A survey on content-based 3D object retrieval is provided by BENJAMIN BUSTOS et al. (2007). Statistical approaches clearly have their strength in discriminating object classes. However, feature-based object detection, e.g., of rectangular shapes, does not yield object parameters: width and height of a detected rectangle must typically be computed separately.

To describe a shape and its construction process, its inner structure must be known. Structural decomposition is well in line with human perception. In general, shapes are recognized and coded mentally in terms of relevant parts and their spatial configuration or structure (King and Wertheimer, 2005). One idea to operationalize this concept was proposed, among others, by MASAKI HILAGA et al. (2001), who introduce the Multiresolution Reeb Graph, to represent the skeletal and topological structure of a 3D shape at various levels of resolution. Structure recognition is a very active branch in the field of geometry processing. The detection of shape regularities (Pauly et al., 2008), self-similarities (Bokeloh et al., 2010) and symmetries (Mitra et al., 2006), (Mitra et al., 2007) is important to understand a 3D shape. To summarize, structural decomposition proceeds by postulating that a certain type of general regularity or structure exists in a class of shapes. This approach clearly comes to its limits when very specific structures are to be detected, i.e., complicated constructions with many parameter interdependencies.

A possibility to describe a shape is realized by the generative modeling paradigm (Özkar and Kotsopoulos, 2008), (Ullrich et al., 2010a). The key idea is to encode a shape with a sequence of shape-generating operations, and not just with a list of low-level geometric primitives. In its practical consequence, every shape needs to be represented by a program, i.e., encoded in some form of programming language, shape grammar (Müller et al., 2006c), modeling language (Havemann, 2005) or modeling script (Autodesk, 2007).

The implementation of the "definition by algorithm" approach is based on a scripting language (Ullrich and Fellner, 2011): Each class of objects is represented by one algorithm M. Furthermore, each described object is a set of high-level parameters x, which reproduces the object, if an interpreter evaluates $M(x)$. As this kind of modeling resembles programming rather than "designing", it is obvious to use software engineering techniques such as versioning and annotations. In this way, model M may contain a human-readable description of the object class it represents.

In contrast to other related techniques using fitting algorithms, such as "Creating Generative Models from Range Images" by RAVI RAMAMOORTHI and JAMES ARVO (1998), the approach by TORSTEN ULLRICH (2011) can classify data semantically. Although RAVI RAMAMOORTHI and JAMES ARVO also use generative models to fit point clouds, they modify the generative description during the fitting process. As a consequence the optimization can be performed locally with a computational complexity, which is significantly reduced. But starting with the same generative description to fit a spoon as well as a banana does not allow to generate or preserve semantic data.

An example illustrates this process. The generative model to describe a vase takes 13 parameters: $R(r_x, r_y, r_z)$ is the base reference point of the vase in 3D and $T(t_x, t_y, t_z)$ is its top-most point. The points R and T define an axis of rotational symmetry. The remaining seven parameters define the distances $d_0, \ldots, d_6$ of equally distributed Bézier vertices to the axis of rotation (see Figure 4). The resulting 2D Bézier curve defines a surface of revolution – the generative vase.

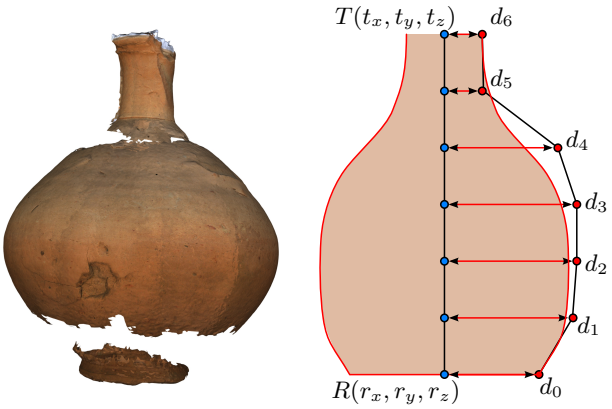

Figure 4: The vase on the left hand side is a digitized artifact of the "Museum Eggenberg" collection. It consists of 364 774 vertices and 727 898 triangles. The example of a procedural shape on the right hand side takes two points R and T in 3D and distance values, which define the control vertices of a Bézier curve.

6. ARCHAEOLOGY & ARCHITECTURE

The huge volume of cultural objects is a challenge even for the most ambitious plans for digitization campaigns (Arnold, 2014b). The fact that probably 90 percent of museum collections are in storage and not accessible to the public is almost demanding for digitization and public accessibility. However, the digitization alone is only part of a larger process that begins at a field excavation and does not end with the presentation in museum exhibitions. Secondary exploitation, database access and sustainable long-time archival of digitized artifacts is also part of the process (Havemann et al., 2006). A very important aspect is the choice of the 3D format used during this process (Niccolucci, 2002), (Niccolucci and D'Andrea, 2006). However, the availability of large quantities of cultural heritage data will enable new methods for analysis and new applications (Arnold, 2014a).

The presented modeling system by CHRISTOPH SCHINKO et al. (2010) is restricted to techniques to meet sustainability conditions. By using JavaScript, the inhibition threshold to use a programming language is reduced resulting in a beginner-friendly tool with a high degree of usability. RENÉ BERNDT et al. (2005) present a system for the production of three-dimensional interactive illustrations in the domain of medieval castles. A special focus is on creating generic modeling tools that increase the usability with a unified 3D user interface.

One of the advantages of procedural modeling techniques is the included expert knowledge within an object description (Ullrich and Fellner, 2011). Classification schemes used in architecture, archaeology and other domains can be mapped to procedures (Ullrich et al., 2008b). When a procedural object description is available, only type and instantiation parameters have to be identified in order to create an object (Ullrich et al., 2013) (see Figure 5).

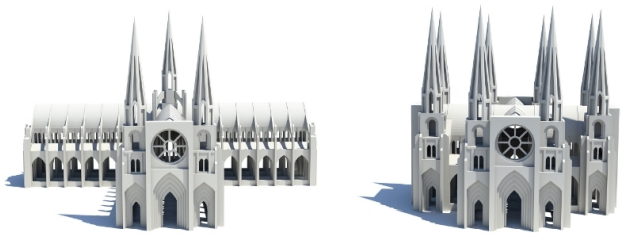

Figure 5: Gothic architecture is defined by strict rules with its characteristics. The generative description of Gothic cathedrals encodes these building blocks and the rules on how to combine them. These building blocks have been created by MICHAEL CURRY, http://www.thingiverse.com/thing:2030.

The usage of generative modeling techniques in architecture is not limited to buildings of the past (Müller et al., 2006a), (Müller et al., 2006b). Over the last few decades, architects have used a new class of design tools that support generative design. Generative modeling software extends the design abilities of architects and may even help to reduce costs by harnessing computing power in new ways. Computers, of course, have long been used to capture and implement the design ideas of architects by means of CAD and 3D modeling. Generative design actually helps architects design by using computers to extend human abilities (Hohmann et al., 2009).

In the context of urban modeling, procedural systems can be used to cover different levels of detail (Musialski et al., 2012b). On a coarse scale, the procedural paradigm is applicable to the generation of terrain using methods based on hydrology (Génevaux et al., 2013), as well as the generation of roads (Galin et al., 2010), entire city layouts (Lipp et al., 2011) and urban spaces (Vanegas et al., 2010). Within the scale of a building, layouts can be generated (Merrell et al., 2010), (Bao et al., 2013b) exhibiting different façades (Musialski et al., 2012a), (Bao et al., 2013a). Exterior Lighting can be designed even for buildings with complex constraints (Schwarz and Wonka, 2014). When it comes down to the interior of a building, furniture can be placed following interior design guidelines (Merrell et al., 2011).

7. OPEN RESEARCH QUESTIONS

According to DIETER W. FELLNER (2001, 2005) and SVEN HAVEMANN (2011) several research challenges have to be met: from the classification of shape representations via generic, stable, and detailed 3D markup to 3D query operations (Havemann and Fellner, 2007).

A particularly important problem occurs in the context of internal structure organization and interfaces. Within a composition of modeling functions, where each function is attached via its parameters to topological entities defined in previous states of the model, referenced entities must be named in a persistent way in order to be able to reevaluate the model in a consistent manner. In particular, when a reevaluation leads to topological modifications, references between entities used during the design process are frequently reevaluated in an erroneous way, giving results different from those expected. This problem is known as "persistent naming problem" (Marcheix and Pierra, 2002).

REFERENCES ON "THEORY OF MODELING"

Aurenhammer, F., 2008. Weighted skeletons and fixed-share decomposition. Computational Geometry 40(2), pp. 93 – 101.

Deussen, O. and Lintermann, B., 2005. Digital Design of Nature: Computer Generated Plants and Organics. Springer.

Galin, E., Peytavie, A., Marechal, N. and Guerin, E., 2010. Procedural Generation of Roads. Computer Graphics Forum 29, pp. 429–438.

Havemann, S., 2005. Generative Mesh Modeling. PhD-Thesis, Technische Universität Braunschweig, Germany 1, pp. 1–303.

Heiberg, J. (ed.), 2007. Euclid's Elements of Geometry. Fitzpatrick, Richard.

Krecklau, L., Pavic, D. and Kobbelt, L., 2010. Generalized Use of Non-Terminal Symbols for Procedural Modeling. Computer Graphics Forum 29, pp. 2291–2303.

Krispel, U., Schinko, C. and Ullrich, T., 2014. The Rules Behind – Tutorial on Generative Modeling. Proceedings of Symposium on Geometry Processing / Graduate School 12, pp. 2:1–2:49.

Lipp, M., Wonka, P. and Wimmer, M., 2010. Parallel Generation of Multiple L-Systems. Computers & Graphics 34, pp. 585–593.

Mandelbrot, B. B., 1982. The Fractal Geometry of Nature. W. H. Freeman and Co.

Marvie, J.-E., Buron, C., Gautron, P., Hirtzlin, P. and Sourimant, G., 2012. GPU Shape Grammars. Computer Graphics Forum 31, pp. 2087–2095.

Maybury, M. T. (ed.), 2012. Multimedia Information Extraction. John Wiley & Sons.

Merrell, P. and Manocha, D., 2008. Continuous Model Synthesis. ACM Transactions on Graphics 27, pp. 158:1–9.

Özkar, M. and Kotsopoulos, S., 2008. Introduction to shape grammars. International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2008 (course notes) 36, pp. 1–175.

Prusinkiewicz, P. and Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer-Verlag.

Schinko, C., Ullrich, T. and Fellner, D. W., 2012. Minimally Invasive Interpreter Construction – How to reuse a compiler to build an interpreter. Proceedings of the International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (Computation Tools) 3, pp. 38–44.

Schinko, C., Ullrich, T., Schiffer, T. and Fellner, D. W., 2011. Variance Analysis and Comparison in Computer-Aided Design. Proceedings of the International Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures XXXVIII-5/W16, pp. 3B21–25.

Schwarz, M. and Wonka, P., 2014. Procedural Design of Exterior Lighting for Buildings with Complex Constraints. ACM Transactions on Graphics 33, pp. 166:1–166:16.

Sederberg, T. W. and Parry, S. R., 1986. Free-form Deformation of Solid Geometric Models. Proceedings of the Conference on Computer Graphics and Interactive Techniques 13, pp. 151–160.

Settgast, V., 2013. Processing Semantically Enriched Content for Interactive 3D Visualizations. PhD-Thesis, Technische Universität Graz, Austria 1, pp. 1–233.

Snyder, J. M. and Kajiya, J. T., 1992. Generative modeling: a symbolic system for geometric modeling. Proceedings of 1992 ACM Siggraph 1, pp. 369–378.

Stava, O., Pirk, S., Kratt, J., Chen, B., Měch, R., Deussen, O. and Benes, B., 2014. Inverse Procedural Modelling of Trees. Computer Graphics Forum p. to appear.

Stiny, G. and Gips, J., 1971. Shape Grammars and the Generative Specification of Painting and Sculpture. Best computer papers of 1971 1, pp. 125–135.

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Mech, R. and Koltun, V., 2011. Metropolis Procedural Modeling. ACM Transactions on Graphics 30, pp. 11:1–14.

Thaller, W., Krispel, U., Havemann, S. and Fellner, D., 2012. Implicit Nested Repetition in Dataflow for Procedural Modeling. Proceedings of the International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (Computation Tools) 3, pp. 45–50.

Thaller, W., Krispel, U., Zmugg, R., Havemann, S. and Fellner, D. W., 2013. A Graph-Based Language for Direct Manipulation of Procedural Models. International Journal on Advances in Software 6, pp. 225–236.

Ullrich, T., 2011. Reconstructive Geometry. PhD-Thesis, Technische Universität Graz, Austria 1, pp. 1–322.

Ullrich, T. and Fellner, D. W., 2011. Generative Object Definition and Semantic Recognition. Proceedings of the Eurographics Workshop on 3D Object Retrieval 4, pp. 1–8.

Ullrich, T., Krispel, U. and Fellner, D. W., 2008a. Compilation of Procedural Models. Proceeding of the 13th International Conference on 3D Web Technology 13, pp. 75–81.

Ullrich, T., Schinko, C. and Fellner, D. W., 2010a. Procedural Modeling in Theory and Practice. Poster Proceedings of the 18th WSCG International Conference on Computer Graphics, Visualization and Computer Vision 18, pp. 5–8.

Ullrich, T., Schinko, C., Schiffer, T. and Fellner, D. W., 2013. Procedural Descriptions for Analyzing Digitized Artifacts. Applied Geomatics 5(3), pp. 185–192.

Ullrich, T., Settgast, V. and Berndt, R., 2010b. Semantic Enrichment for 3D Documents: Techniques and Open Problems. Publishing in the Networked World: Transforming the Nature of Communication, Proceedings of the International Conference on Electronic Publishing 14, pp. 374–384.

Ullrich, T., Settgast, V. and Fellner, D. W., 2008b. Semantic Fitting and Reconstruction. Journal on Computing and Cultural Heritage 1(2), pp. 1201–1220.

Ulusoy, I. and Bishop, C. W., 2005. Generative versus Discriminative Methods for Object Recognition. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2, pp. 258 – 265.

Voß, G., Behr, J., Reiners, D. and Roth, M., 2002. A multithread safe foundation for scene graphs and its extension to clusters. Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization 4, pp. 33–37.

Watson, B. and Wonka, P., 2008. Procedural Methods for Urban Modeling. IEEE Computer Graphics and Applications 28(3), pp. 16–17.

REFERENCES ON "ARCHITECTURE"

Bao, F., Schwarz, M. and Wonka, P., 2013a. Procedural Facade Variations from a Single Layout. ACM Transactions on Graphics 32, pp. 8:1–8:13.

Bao, F., Yan, D.-M., Mitra, N. J. and Wonka, P., 2013b. Generating and Exploring Good Building Layouts. ACM Transactions on Graphics 32, pp. 122:1–122:10.

Berndt, R., Fellner, D. W. and Havemann, S., 2005. Generative 3D Models: a Key to More Information within less Bandwidth at Higher Quality. Proceeding of the 10th International Conference on 3D Web Technology 1, pp. 111–121.

Edelsbrunner, J., Krispel, U., Havemann, S., Sourin, A. and Fellner, D. W., 2014. Constructive Roof Geometry. Proceedings of the International Conference on Cyberworlds 12, pp. 63–70.

Génevaux, J.-D., Galin, E., Guérin, E., Peytavie, A. and Beneš, B., 2013. Terrain Generation Using Procedural Models Based on Hydrology. ACM Transactions on Graphics 32, pp. 143:1–143:13.

Havemann, S. and Fellner, D. W., 2004. Generative Parametric Design of Gothic Window Tracery. Proceedings of the 5th International Symposium on Virtual Reality, Archeology, and Cultural Heritage 1, pp. 193–201.

Hohmann, B., Krispel, U., Havemann, S. and Fellner, D. W., 2009. Cityfit: High-Quality Urban Reconstructions by Fitting Shape Grammars to Images and Derived Textured Point Clouds. Proceedings of the ISPRS International Workshop 3D-ARCH 3, pp. 61–68.

Kelly, T. and Wonka, P., 2011. Interactive Architectural Modeling with Procedural Extrusions. ACM Transactions on Graphics 30, pp. 14:1–15.

Krecklau, L., Born, J. and Kobbelt, L., 2013. View-Dependent Realtime Rendering of Procedural Facades with High Geometric Detail. Comput. Graph. Forum 32(2), pp. 479–488.

Lipp, M., Scherzer, D., Wonka, P. and Wimmer, M., 2011. Interactive Modeling of City Layouts using Layers of Procedural Content. Computer Graphics Forum 30, pp. 345–354.

Lipp, M., Wonka, P. and Wimmer, M., 2008. Interactive Visual Editing of Grammars for Procedural Architecture. ACM Transactions on Graphics 27(3), pp. 1–10.

Merrell, P., Schkufza, E. and Koltun, V., 2010. Computer-generated residential building layouts. ACM Transactions on Graphics 29, pp. 181:1–11.

Merrell, P., Schkufza, E., Li, Z., Agrawala, M. and Koltun, V., 2011. Interactive Furniture Layout Using Interior Design Guidelines. ACM Transactions on Graphics 30, pp. 87:1–10.

Müller, P., Vereenooghe, T., Ulmer, A. and Van Gool, L., 2006a. Automatic Reconstruction of Roman Housing Architecture. Recording, Modeling and Visualization of Cultural Heritage 1, pp. 287–298.

Müller, P., Wonka, P., Haegler, S., Andreas, U. and Van Gool, L., 2006b. Procedural Modeling of Buildings. Proceedings of 2006 ACM Siggraph 25(3), pp. 614–623.

Musialski, P., Wimmer, M. and Wonka, P., 2012a. Interactive Coherence-Based Facade Modeling. Computer Graphics Forum 31, pp. 661–670.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., van Gool, L. and Purgathofer, W., 2012b. A Survey of Urban Reconstruction. Proceedings of EUROGRAPHICS, State of the Art Report (STAR) 31, pp. 1–28.

Parish, Y. and Müller, P., 2001. Procedural Modeling of Cities. Proceedings of the 28th annual conference on Computer graphics and interactive techniques 28, pp. 301–308.

Patow, G., 2012. User-Friendly Graph Editing for Procedural Modeling of Buildings. IEEE Computer Graphics and Applications 32, pp. 66–75.

Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D. W. and Bischof, H., 2012. Irregular lattices for complex shape grammar facade parsing. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 25, pp. 1640–1647.

Steinberger, M., Kenzel, M., Kainz, B., Müller, J., Peter, W. and Schmalstieg, D., 2014a. Parallel Generation of Architecture on the GPU. Computer Graphics Forum 33, pp. 73–82.

Steinberger, M., Kenzel, M., Kainz, B., Wonka, P. and Schmalstieg, D., 2014b. On-the-fly generation and rendering of infinite cities on the GPU. Comput. Graph. Forum 33(2), pp. 105–114.

Thaller, W., Zmugg, R., Krispel, U., Posch, M., Havemann, S. and Fellner Dieter, W., 2013. Creating Procedural Windowbuilding Blocks using the Generative Fact Labeling Method. Proceedings of the ISPRS International Workshop 3D-ARCH 5, pp. 235–242.

Tobler, R. F., Maierhofer, S. and Wilkie, A., 2002a. A Multiresolution Mesh Generation Approach for Procedural Definition of Complex Geometry. Proceedings of the Shape Modeling International 6, pp. 35 – 44.

Tobler, R. F., Maierhofer, S. and Wilkie, A., 2002b. Mesh-Based Parametrized L-Systems and Generalized Subdivision for Generating Complex Geometry. International Journal of Shape Modeling 8, pp. 173–191.

Van Gool, L., Martinovic, A. and Mathias, M., 2013. Towards Semantic City Models. Proceedings of Photogrammetric Week 1, pp. 217–232.

Vanegas, C. A., Aliaga, D. G., Wonka, P., Müller, P., Waddell, P. and Watson, B., 2010. Modelling the Appearance and Behaviour of Urban Spaces. Computer Graphics Forum 29, pp. 25–42.

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B. and Waddell, P., 2012. Inverse Design of Urban Procedural Models. ACM Transactions on Graphics 31, pp. 168:1–.

Whiting, E., Ochsendorf, J. and Durand, F., 2009. Procedural Modeling of Structurally-Sound Masonry Buildings. ACM Transactions on Graphics 28, pp. 112:1–9.

Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W., 2003. Instant Architecture. International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2003 22(3), pp. 669 – 677.

Wu, F., Yan, D.-M., Dong, W., Zhang, X. and Wonka, P., 2014. Inverse Procedural Modeling of Facade Layouts. ACM Transactions on Graphics 33, pp. 121:1–121:10.

Yong, L., Mingmin, Z., Yunliang, J. and Haiying, Z., 2012. Improving procedural modeling with semantics in digital architectural heritage. Computers & Graphics 36, pp. 178–184.

Zmugg, R., Thaller, W., Krispel, U., Edelsbrunner, J., Havemann, S. and Fellner, D. W., 2013. Procedural Architecture using Deformation-Aware Split Grammars. The Visual Computer 12, pp. 1–11.

REFERENCES ON "CULTURAL HERITAGE"

Arnold, D., 2014a. Computer Graphics and Cultural Heritage: Continuing Inspiration for Future Tools. Computer Graphics and Applications 34, pp. 70–79.

Arnold, D., 2014b. Computer Graphics and Cultural Heritage: From One-Way Inspiration to Symbiosis. Computer Graphics and Applications 34, pp. 76–86.

Berndt, R., Gerth, B., Havemann, S. and Fellner, D. W., 2005. 3D Modeling for Non-Expert Users with the Castle Construction Kit. Proceedings of the 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST) 6, pp. 1–9.

Haegeler, S., Müller, P. and Van Gool, L., 2009. ProceduralModeling for Digital Cultural Heritage. Journal on Image and Video Processing 9, pp. 1–11.

Havemann, S., Settgast, V., Krottmaier, H. and Fellner, D. W., 2006. On the Integration of 3D Models into Digital Cultural Heritage Libraries. Proceedings of the 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST) 1, pp. 161–169.

Müller, P., Vereenooghe, T., Wonka, P., Paap, I. and Van Gool, L., 2006. Procedural 3D Reconstruction of Puuc Buildings in Xkipche. Proceedings of Eurographics Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST) 1, pp. 139–146.

Schinko, C., Strobl, M., Ullrich, T. and Fellner, D. W., 2010. Modeling Procedural Knowledge – a generative modeler for cultural heritage. Proceedings of EUROMED 2010 - Lecture Notes on Computer Science 6436, pp. 153–165.

Settgast, V., Ullrich, T. and Fellner, D. W., 2007. Information Technology for Cultural Heritage. IEEE Potentials 26(4), pp. 38–43.

REFERENCES ON "CAD / ENGINEERING"

Berndt, R., Schinko, C., Krispel, U., Settgast, V., Havemann, S., Eggeling, E. and Fellner, D. W., 2012. Ring's Anatomy – Parametric Design of Wedding Rings. Proceedings International Conference on Creative Content Technologies 4, pp. 72–78.

Bokeloh, M., Wand, M. and Seidel, H.-P., 2010. A Connection between Partial Symmetry and Inverse Procedural Modeling. Proceedings of ACM SIGGRAPH 2010 29, pp. 104:1–104:10.

Boulch, A., Houllier, S., Marlet, R. and Tournaire, O., 2013. Semantizing Complex 3D Scenes using Constrained Attribute Grammars. Proceedings of Eurographics Symposium on Geometry Processing 32, pp. 33–42.

Frank, G. and Hillbrand, C., 2012. Automatic support of standardization processes in design models. Proceedings of the International Conference on Intelligent Engineering Systems (INES) 16, pp. 393–398.

Krecklau, L. and Kobbelt, L., 2011. Procedural Modeling of Interconnected Structures. Computer Graphics Forum 30, pp. 335–344.

Mendez, E., Schall, G., Havemann, S., Fellner, D. W., Schmalstieg, D. and Junghanns, S., 2008. Generating Semantic 3D Models of Underground Infrastructure. IEEE Computer Graphics and Applications 28, pp. 48–57.

Ramamoorthi, R. and Arvo, J., 1999. Creating Generative Models from Range Images. Proceedings of ACM Siggraph 1, pp. 195–204.

Schinko, C., Berndt, R., Eggeling, E. and Fellner, D., 2014. A Scalable Rendering Framework for Generative 3D Content. Proceedings of the International ACM Conference on 3D Web Technologies 19, pp. 81–87.

Yu, L.-F., Yeung, S.-K., Tang, C.-K., Terzopoulos, D., Chan, T. F. and Osher, S., 2011. Make it Home: Automatic Optimization of Furniture Arrangement. ACM Transactions on Graphics 30, pp. 86:1–11.

FURTHER REFERENCES

Arnold, D., 2006. Procedural methods for 3D reconstruction. Recording, Modeling and Visualization of Cultural Heritage 1, pp. 355–359.

Autodesk, 2007. Autodesk Maya API. White Paper 1, pp. 1–30.

Behr, J., Dähne, P., Jung, Y. and Webel, S., 2007. Beyond the Web Browser – X3D and Immersive VR. IEEE Virtual Reality Tutorial and Workshop Proceedings 28, pp. 5–9.

Bishop, C. M., 2007. Pattern Recognition and Machine Learning. Springer.

Breuel, F., Bernd, R., Ullrich, T., Eggeling, E. and Fellner, D. W., 2011. Mate in 3D – Publishing Interactive Content in PDF3D. Publishing in the Networked World: Transforming the Nature of Communication, Proceedings of the International Conference on Electronic Publishing 15, pp. 110–119.

Brutzman, D., 1998. The virtual reality modeling language and Java. Communications of the ACM 41(6), pp. 57 – 64.

Bustos, B., Keim, D., Saupe, D. and Schreck, T., 2007. Content-based 3D Object Retrieval. IEEE Computer Graphics and Applications 27(4), pp. 22–27.

Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Vargas-Hernandez, N. and Wood, K. L., 2011. Computer-Based Design Synthesis Research: An Overview. Journal of Computing and Information Science in Engineering 11, pp. 021003:1–10.

Chomsky, N., 1956. Three models for the description of language. IRE Transactions on Information Theory 2, pp. 113–124.

Compton, K. and Mateas, M., 2006. Procedural Level Design for Platform Games. Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference 2, pp. 109–111.

Di Benedetto, M., Ponchio, F., Ganovelli, F. and Scopigno, R., 2010. SpiderGL: a JavaScript 3D graphics library for next-generation WWW. Proceedings of the 15th International Conference on Web 3D Technology 15, pp. 165–174.

Eckel, B., 2003. Thinking in C++: Introduction to Standard C++, Practical Programming. Prentice Hall.

Fellner, D. W., 2001. Graphics Content in Digital Libraries: Old Problems, Recent Solutions, Future Demands. Journal of Universal Computer Science 7, pp. 400–409.

Fellner, D. W. and Havemann, S., 2005. Striving for an adequate vocabulary: Next generation metadata. Proceedings of the 29th Annual Conference of the German Classification Society 29, pp. 13 – 20.

Fellner, D. W., Saupe, D. and Krottmaier, H., 2007. 3D Documents. IEEE Computer Graphics and Applications 27(4), pp. 20–21.

Havemann, S. and Fellner, D. W., 2007. Seven Research Challenges of Generalized 3d Documents. IEEE Computer Graphics and Applications 3, pp. 70–76.

Havemann, S., Ullrich, T. and Fellner, D. W., 2012. The Meaning of Shape and some Techniques to Extract It. Multimedia Information Extraction 1, pp. 81–98.

Hilaga, M., Shinagawa, Y., Kohmura, T. and Kunii, T. L., 2001. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. Proceedings of the 28th annual conference on Computer graphics and interactive techniques 28, pp. 203–212.

Initiative, D. C. M., 1995. Dublin Core Metadata Initiative. http://dublincore.org/.

King, B. D. and Wertheimer, M., 2005. Max Wertheimer & Gestalt Theory. Transaction Publishers. ISBN 0-7658-0258-9.

Kuang, Z., Chan, B., Yu, Y. and Wang, W., 2013. A Compact Random-access Representation for Urban Modeling and Rendering. ACM Trans. Graph. 32(6), pp. 172:1–172:12.

Marcheix, D. and Pierra, G., 2002. A Survey of the Persistent Naming Problem. Proceedings of the ACM Symposium on Solid Modeling and Applications 7, pp. 13–22.

Martin, G. E., 1998. Geometric Constructions. Springer.

Mitchell, W. J., 1990. The Logic of Architecture: Design, Computation, and Cognition. MIT Press.

Mitra, N. J., Guibas, L. J. and Pauly, M., 2006. Partial and approximate symmetry detection for 3D geometry. ACM Transactions on Graphics 25, pp. 560 – 568.

Mitra, N. J., Guibas, L. J. and Pauly, M., 2007. Symmetrization. International Conference on Computer Graphics and Interactive Techniques 26, pp. 1–8.

Niccolucci, F., 2002. XML and the future of humanities computing. SPECIAL ISSUE: First European workshop on XML and knowledge management 10, pp. 43–47.

Niccolucci, F. and D'Andrea, A., 2006. An Ontology for 3D Cultural Objects. Proceedings of the 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST) 7, pp. 203–210.

NVidia, n.d. NVIDIA CUDA C Programming Guide.

OpenGL Architecture, R. B., 1993. OpenGL Reference Manual. Addison-Wesley Publishing Company.

Ousterhout, J. K., 1998. Scripting: Higher Level Programming for the 21st Century. IEEE Computer Magazine 31(3), pp. 23–30.

Parr, T., 2010. Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. Pragmatic Bookshelf.

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H. and Guibas, L. J., 2008. Discovering structural regularity in 3D geometry. ACM Transactions on Graphics 27, pp. #43, 1–11.

Reas, C., Fry, B. and Maeda, J., 2007. Processing: A Programming Handbook for Visual Designers and Artists. The MIT Press.

Reiners, D., Voss, G. and Behr, J., 2002. OpenSG: Basic concepts. Proceedings of OpenSG Symposium 2002 1, pp. 1–7.

Schinko, C., Strobl, M., Ullrich, T. and Fellner, D. W., 2011. Scripting Technology for Generative Modeling. International Journal On Advances in Software 4, pp. 308–326.