This paper presents a novel approach for the automatic creation of vegetation scenarios in real or virtual 3D cities in order to simplify the complex design process and time-consuming modeling tasks in urban landscape planning. We introduce shape grammars as a practical tool for the rule-based generation of urban open spaces. The automatically generated designs can be used for pre-visualization, master planning, guided design variation and digital content creation in general (e.g. for the entertainment industry). In a first step, we extend the CGA shape grammar by Müller et al. (2006) with urban planning operations. In a second step, we employ the possibilities of shape grammars to encode design patterns (Alexander et al., 1977). Therefore, we propose several examples of design patterns allowing for an intuitive high-level placement of objects common in urban open spaces (e.g. plants). Furthermore, arbitrary interactions between distinct instances of the vegetation and the urban environment can be encoded. With the resulting system, the designer can efficiently vegetate landscape and city parks, alleys, gardens, patios and even single buildings by applying the corresponding shape grammar rules. Our results demonstrate the procedural design process on two practical example scenarios, each one covering a different scale and different contexts of planning. The first example illustrates a derivation of the Garden of Versailles and the second example describes the usage of high-level rule sets to generate a suburbia model.

**Keywords**: City modeling; design methodology; generative design; simulation; virtual environments.
1. Introduction

Since the traditional occupational field of architects melted into the domain of digital content creation their creative capabilities and expert knowledge are also demanded from entertainment industries. Besides this fact today’s challenging master planning projects require architects to merge digital content creation and former analogue design methods. These are time-consuming tasks, which are especially hard to handle on large-scale landscape and urban planning projects with high demands on the two key features reliability and quality of plans. It can take several man-years to model detailed 3D land- and cityscapes. Nextgen applications that are already used in other industries can remove conventional empiric planning methods in architecture and landscape design. They will also raise dramatically the design quality through (1) the added predictability of how the sketched landscape design will look like, (2) the added possibility to simulate certain measures for city planning like enhanced airflow through skillful placed vegetation, and (3) the given opportunity to redesign drafted models within seconds what could have been earlier required months of work.

There exist plenty of context-sensitive design decisions and simple repetitive digital content creation tasks. Furthermore, they depend on well-defined planning constrains. These tasks should be automated where applicable since there is no creative expert knowledge necessary to process them. We propose the use of a shape grammar in combination of procedural methods, which is adapted to the needs of architects, landscape and urban planners. Therefore, this paper introduces a novel approach for a grammar-based planning of open spaces in urban environments.

1.1 Related work

A landmark in the formal theory of architecture was the introduction of shape grammars by Stiny (1975). These shape grammars can be used to analyze and describe designs for a wide range of architecture, such as Palladian villas (Stiny and Mitchell, 1978), Mughul gardens (Stiny and Mitchell, 1980), Frank Lloyd Wright’s prairie houses (Koning and Eizenberg, 1981), or Alvaro Siza’s houses at Malagueira (Duarte, 2002). However, the application of Stiny’s original shape grammars as a practical modeling tool is unclear, because they are hardly amenable to computer implementation. Mayall (2005) presented an interesting implementation of a shape grammar specified for landscape design. The system produces urban scenes by generating parcels with simple 3D buildings and by distributing vegetation objects. A more recent approach has been presented by Müller et al. (2006). They introduce a novel attributed shape grammar called CGA Shape which is suited for applications in computer graphics. With this framework, a wide range of architectural designs can be encoded and detailed 3D models can be generated automatically.

Shape grammar rules that create certain architectural configurations can be grouped into collections within libraries. Such libraries exist in literary form and are described as patterns. The use of patterns in architecture is founded on Alexander et al. (1977). For example, Crowe and Mitchell (1988) practically applied Alexander’s patterns to describe specific landscape designs. The corresponding key components of patterns in landscape design have been specified and illustrated in detail by Bell (1999). Condensed as illustrated patterns, Turner (1996) gives a comprehensive overview of landscape attributes in an urban context. The formal relationship between landscape design and urban planning is described by Alexander (2002).

1.2 Overview

This paper is organized as follows: Section 2 starts with an overview of the CGA Shape grammar, and afterwards our novel extensions needed for the encoding of urban open spaces are introduced. Section 3 deals with the design of vegetation patterns and their implementation by using use our grammar. In section 4, we describe two design studies as
examples. Discussion about advantages and encountered difficulties can be found in Section 5. Finally we give conclusions and future work in Section 6.

2. A shape grammar for landscape architecture

2.1 CGA shape

In the following, we briefly introduce the main concepts of CGA Shape but refer the reader to (Müller et al. 2006) for a more comprehensive description. The CGA Shape framework consists of the shape definition, the production process, the rule notation with shape operations, and an element repository.

**Shape definition:** A shape consists of geometry, a symbol, and attributes. The most important attributes are the position $P$, three orthogonal vectors $X$, $Y$, and $Z$, describing a local coordinate system, and a size vector $S$.

**Production process:** The production process can start with an arbitrary configuration of shapes, called the *initial shapes*, and proceeds as follows: (1) Select an active shape with symbol $A$ in the set, (2) choose a rule with $A$ on the left hand side to compute a successor for $A$ resulting in a new set of shapes $B$, (3) mark the shape $A$ as inactive and add the shapes $B$ to the configuration and continue with step (1).

**Rules:** The CGA Shape production rules are defined in the following form:

```
id:   predecessor : condition    →    successor : prob
```

where $id$ is a unique identifier for the rule, predecessor is a symbol identifying a shape that is to be replaced with successor, and condition is an optional guard (logical expression) that has to evaluate to true in order for the rule to be applied. For example, it can be secured via condition that the rule is only applied if the predecessor shape is not occluded by another shape. To make designs stochastic, the rule is selected with probability $prob$ (optional). Several different types of *shape operations* can be applied to modify the successor shape, e.g. transformations like translate or scale. The insert command $I(objId)$ adds, according to the size of the current shape, an instance of a geometry object with identifier $objId$ (from the element repository). The *subdivision split* can subdivide objects along an axis. For example Subdiv(y,7,2,5){ A | B | A } splits the current scope along the Y-axis in three parts with sizes 7, 2, and 5 and adds three shapes $A$, $B$, and $A$ accordingly. To enable size-independent rules, we also include the letter $r$ to denote relative values, e.g. Subdiv(y,1r,2,4r) creates three parts: the middle part has absolute size 2 and the remaining two parts split the remaining space in a 1:4 proportion. We also employ a *repeat split*, e.g. Repeat(x,2){ A } places as many $A$ shapes of approximate size 2 as possible along the X-axis of the current shape. The *component split* can divide a shape into its geometric components like 2D faces or 1D edges, e.g. Comp(e){ B } splits the predecessor shape into edge shapes with symbol $B$.

**Element repository:** The library of 3D models consists mainly of basic primitives, elementary architectural objects (e.g. ionic capitals) and plant models. The elements are hierarchically organized in categories and types and each element has a unique identifier, shader attributes and optional metadata.

2.2 CGA shape extensions for landscape design

With the CGA Shape framework a big diversity of hierarchical landscape designs can be encoded mainly by using the subdivision, repeat and component split. Nonetheless, for an intuitive application in landscape design we have to extend the CGA Shape grammar with two novel shape operations.

The first extension deals with the huge variety of different plants in the element repository. For example, each tree type is classified into three different ages (young, medium and adult age). And for each of these categories, different appearances of the trees are stored. As a consequence, we extend the insertion operation $I(objId)$. Instead of accessing the objects only via $objId$, we use $I(category,type,instance)$ to place the tree models. For example, $I(birch,adult,rand(7))$ selects randomly one
of the first 7 instances of adult birches and inserts this object in the 3D model.

The second extension deals with the distribution of instances. We experienced several situations where it is not possible to randomly distribute small shapes (e.g. trees) within a bigger shape (e.g. parcel). Hence we introduce a new shape operation \( \text{Distr}(\text{density}, \text{deviation}) \) which places point shapes in the current shape. If the parameter \( \text{deviation} \) is 0 the points are homogenously distributed, otherwise the points are Gaussian distributed (both according to the \text{density} value).

2.3 The CityEngine system
The CGA Shape grammar with extensions is implemented in a tool called CityEngine (Parish & Müller 2001, Müller et al. 2006). It can process urban environments of any size i.e. ranging from a single lot up to a whole city. This input data is represented in a GIS format and consists of different regions like streets, parcels, building footprints, patios etc. Furthermore, these regions contain metadata information specified by the user in the GUI of the CityEngine. Depending on the metadata attributes, the regions trigger the selection and application of corresponding shape grammar rules. Therefore, the region itself (usually a polygon) is fed as initial shape into the grammar engine which derives it into an elaborate design by applying the selected rules. The resulting model can then be previewed in the OpenGL viewer of the CityEngine or photo-realistically visualized in a 3D application like Autodesk’s Maya.

3. Patterns
3.1 Enhancing traditional design cycles with shape grammars
We can classify the term design process in design disciplines as the definition of a design problem and its solving. Therefore the designer evolves the problem within a cyclical process into a solution, which has many facets and is bound on the designer’s intuition. This cyclical process often starts with a general understanding of a complex issue and is usually accompanied by the (re-)definition of its sub-issues. Within these iterative cycles the designer develops a better understanding of the problem’s key components and their interrelationships. The results are solutions based on rational, logical and also intuitive flash of insights. The implementation of these solutions involves the development and the realization of the underlying design idea, and its integration into physical and cultural contexts. The design of urban open spaces comprises especially the making of places. If we see places as a certain set of designs, which can be decomposed into subordinated problems and their associated solutions, then we can take into account that some of the subordinated problems are reoccurring problems with reoccurring general solutions. Subordinated solutions are grouped as sets within major solutions, which are in our example spatial arrangements of plant objects in an urban context. In this view major solutions contain the important design features like the spatial grouping, the alignment and the distribution of the vegetation species and the references to their geometric objects. Alexander et al. (1977) describes such sets of objects and features as design patterns. Our vegetation grammar is capable to interpret patterns as described above. Furthermore patterns can also have hierarchical linkages between them and stay in this way human readable. Since a designer is more or less aware of practically using patterns in his design, he should easily be able to integrate an automated pattern driven generation approach.

3.2 Patterns for urban open spaces
Landscape planning in urban environments operates on distinct major patterns. To give landscape planners access to a manageable view, we have simplified these patterns into the following regions: avenue zones, block zones, lot zones, patio zones, and also vegetation regions (figure 1).

Avenue zones are straightforward created along streets. They span over several blocks if necessary and can be split by crossing streets. They are vegetation-
only zones within the generation process so they do not take part in the creation of buildings nor interact with their footprints. Block zones are vegetation zones enclosed by streets. They are identical in the form to the CE block polygon, but do not include buildings or building footprints, since they are drawn of the pipeline before the creation of housing structures. Block zones are supposed to be containers for parks or small urban forests. Lot zones come into play when a combination of buildings and vegetation is desired. They are allocated to the regions list after the generation of buildings and thus hold information about their footprints, providing means to interact with its housing environment. Vegetation lot zones additionally get building type attributes from the City Engine framework. Attributes like commercial, industrial and residential can be used later for a further selection of suitable sub-patterns.

A lot zone without a street edge is characterized as a patio zone. It serves to fill up the empty space in the center of blocks, enclosed by buildings. Although patio zones share the possibility of lot zones to include footprint information, their primary purpose is to contain vegetation without housing. Areas that span several blocks or cover big parts of the city and contain elements of vegetation are classified as vegetation regions. Examples of vegetation regions include forests within city areas, big parks or fields.

3.3 Arbitrary linkage of patterns

In our given case patterns consists out of sets of attributes, which contain all necessary textual information to describe urban open spaces. From a system’s view the descriptions of each pattern – the grammar rules – can be stored within a corresponding pattern text file. Each pattern text file can reference well defined to subordinated patterns. This permits the easy exchange of certain features like climate zone, detail level of the achieved models for example to provide high resolution polygon models for visualization needs or low resolution primitives for the use in simulation purposes like airflow analysis.

Since the description of patterns starts at high level major pattern, which contain a more human readable description, and ends at low level grammar descriptions our system offers the possibilities to keep things simple for the non-grammar-expert and to keep things open for users that want to get deep into the features of the grammar.
4. Examples

4.1 Garden of Versailles
The garden of Versailles, finished around 1700 by order of the monarch Louis XIV is considered the prime example of the French Baroque Style in landscape architecture. Its repeating, geometric appearance proves to be an ideal object for a description using rules and patterns of our Grammar.

The rules set of the “Versailles-pattern” for this example starts with an occlusion rule, which assures that the open space of the center area does not get filled with items of the following productions, e.g. trees of the crossing avenues. Next, the region is subsequently split into sub-regions, each containing a center avenue and left and right grass zones. In the Versailles example, three recursive split steps subdivide the major region into eight sub-regions (figure 2).

Once a sub-region reaches the maximal recursion count, the general pattern grassArea sets the final style by applying a grass ground cover and distributing hedge elements on the edges. By passing the recursion depth to the treeAvenue pattern, avenue parameters like street width, tree density and tree type are adjusted automatically to the importance of the avenue.

4.2 Suburbia: Beverly Hills
Our second example shows a suburban environment, inspired by Beverly Hills. Different garden patterns in combination with stochastic rules ensure diversity in detail views, whereas major pattern descriptions provide the typical homogeneous appearance of the area when observed from a bigger distance (figure 3).

Context-sensitive grammar rules allow more precise details in lot and garden patterns: Regions are aware of bordering streets thanks to edge attributes and distribute sidewalks and avenue trees accordingly; footprints of buildings are accessible in vegetation rules and assure correct dimensioning and placing of surrounding elements like forecourts, fences and trees; special markers on lot and building (entrance, garagefront, pool) allow logical connections between house and lot features, e.g. doorways from street to garage.

Due to the modular pattern-based description of lot types, forecourt forms, border styles and fence appearances and a stochastic mixing

Figure 2 (left to right)
Rendered view of Versailles, original layout of the plan, modified layout with 6 recursive splits, original layout placed on different regions
of these elements and its controlling parameters an almost infinite visual variety of building lots is generated.

5. Conclusion and future work

Automatic and semi-automatic approaches can drastically reduce and ease complex and time consuming design and modeling tasks in landscape planning. Therefore we presented a stochastic shape grammar for Landscape Architecture as a novel approach for the automatic creation of vegetation scenarios in real or virtual 3D cities.

We showed that landscape architects could integrate this approach into their existing design pipelines quite easily for the planning of elaborated greenway and open space systems.

The rule-based distribution and placement of vegetation and landscape objects in urban environments enables the planner to apply further simulation steps and is also open for iterative (re-)design purposes. The complexity of procedural modeling became manageable with the help of reusable landscape patterns. We adopted our grammar to a promising shape grammar standard the CGA shape grammar by Müller et al. (2006). Our approach can be used for previsualization, master planning, guided design variation and for digital content creation purposes of the entertainment industries. Ongoing projects deal with the pattern-based procedural design of the Science City ETH's open spaces in Zurich, Switzerland, and also with the challenging planning of the Singapore-ETH Centre.
References


