Using shape grammars as a rule based approach in urban planning: A report on practice

Author(s):
Schirmer, Patrick; Kawagishi, Noboru

Publication Date:
2011

Permanent Link:
https://doi.org/10.3929/ethz-a-006648665

Rights / License:
In Copyright - Non-Commercial Use Permitted
Using shape grammars as a rule based approach in urban planning - a report on practice

Patrick Schirmer¹, Noboru Kawagishi²

¹Institute of Transport Planning and Systems, ETH Zurich, Switzerland
²KCAP Architects&Planners

¹http://www.ivt.ethz.ch, ²http://www.kcap.eu

¹patrick.schirmer@ivt.baug.ethz.ch, ²n.kawagishi@kcap.eu

Abstract. With the development of user friendly software, using procedural shape grammars has become productive for urban planning projects. Little about the experience of their use by architects and urban planning agencies has been reported yet. This paper will thus discuss the experience gained with the use of shape grammars in the projects of KCAP. We will show how the different scales of urban planning and urban design can be handled and how design concepts can be integrated into the procedural "pipeline" using the software “CityEngine”. We will also present an approach of "typological testing" that allows to test various design concepts for their possible developments. This work is the base for current research at ETH, integrating geometric aspects into behavioural simulation processes of urban simulation.

Keywords. Shape grammars; Urban planning; Urban Simulation; Urban Typologies.

Rule-based urban design

Urban planning projects are often large scale projects whose realization process will take years and decades. This is why planners understand urban planning as the creation of frameworks allowing for adaptation to multiple kinds of changes over time, such as infrastructural, economical, political, cultural and demographic changes by implementing flexible phasing strategies. Even if masterplans and results of competitions often will show a "final vision" through nice visualisations, legal regulations for future construction will reduce those to a set of rules forming a development framework with the already mentioned flexibility.

Such a framework might define major network-links, a zoning scheme and a set of typological examples, but will avoid to fix these in a plan showing a "final result".

The procedural approach of shape grammars is a great tool to test and visualize various versions of development within these frameworks, using suitable random processes. It thus forms a new possibility of testing framework definitions on their possible results.

The use of shape grammars for urban planning

Parish and Müller (Parish & Müller 2001) present an approach to use shape grammars for the procedural modelling of urban landscapes. This work has been an essential step for the use of shape grammars in urban planning. In later work, Müller et al. extend these grammars to the creation of facades and building structures (Müller et al. 2007; Müller et al. 2006). Although these studies show possible advantages of using shape grammars for urban planning, the lack of a user friendly software did not allow the use in practice.
Recently suitable software has become available, e.g. the software “CityEngine” provided by the company Procedural\(^1\). After initial experiences with the testing version of this software, KCAP became the first registered client of Procedural in 2008 and focussed on testing its use for urban planning and urban design. Although the use of shapegrammars in the field of urban planning is still very unusual, several other agencies are beginning to use CityEngine nowadays as well.

**Working with shapegrammars**

The workflow in the CityEngine can be described as a loop of subdivision processes. Beginning with a street-network, blocks are being created, those will get subdivided to form lots. Further subdivision will define the footprint of buildings. After their extrusion, the volumes can be split into their sides and further subdivisions will allow to create façades out of those.

The big advantage of the CityEngine for planning practice, is the option of interrupting this process at any time allowing for manual correction outside of the software.\(^2\) Additionally it is possible to implement constraints and designs by "attribute-maps". These are rasterfiles, e.g. representing a zoning scheme, of which the mapped values are used to parametrize the rules.

Extending the libraries (consisting of previous works at KCAP) through including individual rules of projects is an ongoing process that allows to create a higher complexity and variety in the results. But even with the current prototypes various case-studies at KCAP have shown the efficiency of such an approach for urban planning projects. These experiences can be grouped in three categories: visualisation, random subdivisions, typological testing.

**Visualisation**

Visualisation in high detail is important for large scale urban design projects and any urban planning projects in advanced phase of planning. These projects include many details on specific constrains, owners, given infrastructure, etc. As a result, the position, shape and height of buildings are already precisely defined and thus reduces the use of procedural shape grammar rules.

Using CityEngine as a visualisation tool means to create and apply facades rules that create volumes with detailed façades based on building footprints and height information. By importing CAD-drawings or shapefiles representing those footprints and including height information, a volumetric impression of the design-study with detailed façade structures can be created within a few minutes. This 3d-model can than be exported and rendered in a specific rendering program [figure 1]. This approach allows to create street

---

1 "Procedural Inc. has been founded by Pascal Mueller, Simon Schubiger, Dominik Tarolli and Prof. Luc Van Gool" (source: [www.procedural.com](http://www.procedural.com)) as a spin-off of the ETH-Zurich in 2007 to focus on the development of a user friendly software integrating the results of their reseach. In 2008 the first version of the software CityEngine was published, which was the first commercial software, that allows the procedural design of buildings on a large scale and guide these through maps that can be imported.

2 From version 2010.3 on CityEngine includes an interaction mode that allows to correct geometries interactively within the workspace. As described in the introduction this paper will focus on the simpler version 2009.3 to be comparable to other shape grammar approaches.
view impressions at early design states previously often done with collages on photos of physical models.

The number of 3D-objects in a scene gets large very fast when facades are created in high detail, which can have a significant effect on computational speed if done for several buildings, a district or a whole city. Several types of Level of Detail (LOD) where integrated into the shape grammar rules for this reason [table 1]. This allows to represent buildings with a lower relevance, e.g. neighbouring buildings of a project, in a lower detail. The different LODs are based on each other and can be created by setting different break-points within a facade rule.

![Figure 1](image1.png)

*Figure 1*
Examples of Streetviews in LOD 3 created with shape-grammar rules based on imported footprints and height information. The procedurally generated objects have been exported out of CityEngine and rendered in FormZ.

<table>
<thead>
<tr>
<th>LOD</th>
<th>Scale of use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Extrusion of the parcel based on constraints for height or density</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(general overview of site)</td>
</tr>
<tr>
<td>1</td>
<td>&gt; 1:1000</td>
<td>Simple volume of buildings representing urban typologies;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(optionally with mapped facade structures)</td>
</tr>
<tr>
<td>2a</td>
<td>1:1000 - 1:2000</td>
<td>Volumes including essential variations in height and depth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(e.g. add-ons, cantilevers)</td>
</tr>
<tr>
<td>2b</td>
<td>-</td>
<td>Same volume as LOD2a as floor plains.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(used as basis for calculation of floorspace)</td>
</tr>
<tr>
<td>3</td>
<td>1:500 - 1:1000</td>
<td>Volume including 3D-details on building structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(e.g. pylon, setback, roof, balcony, windows)</td>
</tr>
<tr>
<td>4</td>
<td>&lt;1:500</td>
<td>Model with highly detailed facade elements and accessories</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(e.g. handrail, window frame, joints, canvas blind)</td>
</tr>
</tbody>
</table>

*Table 1*
Level of Details (LOD) for Buildings as defined at KCAP
**Typological Testing**

A feature of great interest, is the use of shape grammars to test various volumetric combinations on a site. By defining the building typology and its height or the density of a plot, volumes are created that respect these definitions and the total floorspace is reported (figure 2). These tests are essential to get a grip on densities and typologies on a site and are normally based on physical models, e.g. with a foam-cutter, which is very time-consuming work.

CityEngine2009.2 does not include predefined building typologies. Therefore the definition and scripting of typologies was done through internal research.

Typologies were created for three kind of locations (referred to as "categories"): rural, suburban and urban. For each of those, several building typologies were scripted, representing a volumetric appearance (LOD 1) and the basis for floorspace-calculation (LOD2b). Facades that are fitting to those where created in a later phase.

The selection of the category to use can be controlled through attribute maps. Within these three types of location a random selection applies a specific typology per plot and creates new combinations of volumes with every new run of the grammar.

Shape grammars as implemented in CityEngine2009.3 can lead to very different results depending on the shape of a lot. The number of vertices and the length of an edge can have essential impact on the resulting volumes as explained in the chapter "the grammar". A possibility to avoid unrealistic types of buildings is the implementation of additional queries in the shape grammar scripts. Those will query the number of vertices, the size of the lot and also check for concave angles of a lot, before selecting a volumetric typology that fits properly.

---

**Figure 2**

Two versions of Urban Design and the resulting floorspace for the same site, that have been generated through the use of shape grammars based on attribute maps defining density and the mix of use.

---

3 Version 2010.2 now has basic shapes of typologies predefined.
Introducing a framework for evaluation of Urban Designs

The above mentioned work-packages were the first approach to use the CityEngine within our projects. Having tested those by creating several scripts did allow for combining these scripts into more complex shape grammar rules, thus creating an overall framework.

The aim of such a framework is to create a tool that will give a volumetric output to conceptional inputs such as a zoning map. This allows to test design-studies at an early stage on their volumetric impacts and to adapt the design based on these results. As described later-on this will also allow for combination with behavioural simulation-models as UrbanSim to create a visual output of the result.

Urban developments are guided by constraints that define zones with a certain mix of use and the density allowed per parcel. Urban planners often start a project by defining spatial variations in a similar way, e.g. in form of zoning maps, before developing a more detailed impression in form of masterplans with buildings, street-widths and private spaces.

Typological testing through shape grammars can therefore be based on these kind of zoning-maps and derive proper typologies for each plot based the zoning definitions. A possible option to include zoning maps into CityEngine's shape grammar rules is given by "attribute maps". These allow for integration of raster-files that are manually placed in a scene. The coding employs colour and brightness to define numeric values that will be integrated in the shape grammar rule as conditions. A blue zone might then represent the use "residential" and the brightness could represent the density permitted. Several attribute maps can be included, we normally use two maps within our framework- one for the density, the other one for the mix of use. Based on these a set of fitting typologies (referred to as “typological class”) is chosen, of which one will randomly be chosen through the shape grammar rule. Additional building parameters as the building-width, floor-height and building-height are set according to the use of a building.

As already mentioned each of the typologies implemented does only fit properly to a certain range of plot-sizes and plot geometries. The assignment of a typological class to a parcel was done manually for the typological testing. But this definition can done automatically within the framework, based on the users' attribute maps. Therefore additional queries are implemented to query the size and form of a geometry and react accordingly.

First query will be on the size of a parcel. The user can define minimum sizes of a site and maximum sizes. If a lot is smaller then the minimum size for construction, it will get coloured in green through the shape grammar rule, to mark it as an optional public space. If it is too big, the output will get coloured in red and can be subdivided by the user according to his design-idea. We also implemented the option of integrated subdivision. A plot that is too big will then get subdivided though a split along it's longest edge. This will be repeated until the plot is not too big any more.

When a plot's size is fit for construction, it will go through the process already described: A query will check the geometry of the parcel and define a set of fitting typologies, which we call “typological class”. One of the typologies inside the typological class will be chosen by a random process for each plot.

A later process will create the height of the building based on an attribute map. We implemented two approaches of controlling the height of buildings. The first will directly
define the height of a building. Buildings will then be extruded according to these definitions. This approach is necessary for urban planning practice as constraints on height, defining a maximum permitted are frequent. By creating an LOD2b output (see above), the floorspace of the buildings will get reported. Repeating this procedure allows for testing various typologies and gives an impression of floorspace that is possible. This is what used to be done through physical model-studies and is an essential step at the beginning of projects, to get a feeling for the site and its possible densities.

Figure 3
Structure of the urban design testing framework showing the various queries on the initial shapes and guidance through attribute maps.
The alternative approach starts by setting the floorspace and allows for flexible heights, which represents the investor-oriented approach. It generates individual height of buildings based on the definition of a floorarea ratio (FAR) of a parcel. The size of a parcel is read out and the possible amount of floorspace is calculated. By summing up the area of all footprints (of buildings) within a plot, the number of allowed stories can be calculated and will be used to procedurally create the buildings. Summing up the footprints can currently not be performed within the shape grammar language of the CityEngine as counting variables can not be included in the scripts. It was realized as a python script that is run within the CityEngine, but apart from the shape grammar rule. After having run the script that creates the building footprints, the python script is run and reports the number of stories allowed to the shape grammar rule. A second run will then create the buildings in the maximum height according to the FAR.

Grammar and software

Core of CityEngine is its procedural grammar scripting language, "Computer Generated Architecture" (cga). It allows to generate designs in a procedural way, i.e. by iterative refinement of created elements (Parish & Müller 2001). Basis for this process is an imported geometry (initial shape) and its bounding-box. These are being transformed by the user through the processes defined within the cga-script. The resulting geometry will go through further transformation routines, resulting in the refinement of the overall design.

Transforming procedures can be defined in a relational or an absolute way. To give an example, the subdivision of a plot can be set to a depth of 18m or to 1/3 of the current depth. These options are of obvious relevance when creating scripts for urban planning purpose. A building depth will generally have a maximum depth for natural illumination defined in absolute values, meanwhile its positioning on a plot might have to be described in relative forms.

Additionally the cga gives the opportunity to define approximate values. This allows to subdivide a building into several floors of approximately 3m height. The total number of floors will then depend on the overall height, they will have a height of around 3 meter so that they perfectly fit into the building.

It is possible to include attributes, variables, set queries, iterations and to hand over values to the next routine as well as to include attributes that rely on maps loaded into a scene. This allows for flexibility and integration of design, e.g. as definition of height or density per plot (see above).

The procedural approach of the CityEngine does not include the recognition of existing shapes or of shapes created within a previous procedure of the procedural shape grammar rule. This becomes essential when typologies of buildings have to react on each other, e.g. through a defined distance of buildings. Through hard-coding this relation within the shape grammar rule, typological appearance can be included, but they depend very much to the initial shapes as already mentioned⁴.

Typological rules that were programmed as part of this research consist of shapes that form certain spatial effects through the relation of buildings to each other. Simple

⁴ The description of this paper is based on the CityEngine2009.3. Later version of the CityEngine include basic typological shape-appearances as L and O. These are based on orientation and are not as sensitive to the initial shapes features as number of vertices or angle within polygons.
example would be a L-shape or an O-shape. In the shape grammar rule this has been realized by performing a "split" on one vertex of an initial shape and then change the position of the world-coordinate-system to the next vertex.

This approach allows to generate some complex typological forms that include building relations, but strongly depend on the initial shape. The picture below [figure 4] shows how one and the same rule creates different output on different initial shapes, which differ only with respect to the number of vertices of the initial geometry.

![Figure 4](image)

Results of two shape grammar scripts on various initial shapes, showing the effect of different number of vertices. The red edge is the initial edge, the circle shows the initial vertex that will be used within the script.

This example shows that it is not possible to write a shape grammar rule that creates the same output on every initial shape. A workaround to handle this problem, is to query the initial shapes feature within the cga-shape grammar rule and select a fitting transformation based on that query. This significantly reduced the number of available typologies for a plot, but still allowed a variance of typologies for most cases. It should be mentioned here that the version CityEngine2010.3 allows for integration of shapes based on geographical orientation. Having implemented this features in our framework avoided many of the above mentioned problems. With that feature it is possible now to use all typological shapes for all kind of initial shapes.

**Interaction and guidance**

CityEngine is a tool allowing the use of shape grammars for design within urban planning. As such it is meant to be used by non-computer-scientists and needs to handle interaction as easily as possible. This chapter will therefore highlight this aspect.

As a research product that evolved into a commercial product it was launched in an early phase of development. The very first version did include most of the cga-grammar features that are described by Parish and Müller (Parish & Müller 2001; Müller et al. 2006). Later versions allowed interactive postprocessing of the results of scripts. The current version (CityEngine2010.3) allows for interaction by drawing of new initial shapes, as well as dragging a vertex of imported shapes and scaling, rotating, moving or deleting of shapes and volumes.

Beside these modes of interaction the use of attributes is an essential possibility for the user to steer the results of the procedurally generated urban layout. This has always
been a key feature of the software. It is essential for the user to include attributes within his shape-grammar rules, that will allow these interactions during the execution of the procedural scripts. In our framework, attributes are used not only to define the building height and number of floors, but e.g. is also applied to the typology used, the optional implementation of subdivision processes (for plots that are too large), permission of add-ons on a building and to manually correct the depth of every side of a building.

In combination with the option to edit initial lots, this makes CityEngine a very interactive tool that finally allows its use within the design-process, through correction and adaptation of the procedurally created volumes to individual situations on a given site.

Current development and research

The work so far concentrated on the use of shape grammars, i.e. the software CityEngine, to create the vision of the designer using the software. Current research projects are aiming for representation of "natural behaviour" instead.\(^5\)

The evaluation of behaviour within urban systems is the aim of these projects. Wadell (2002) demonstrates how discrete choice models can be used to represent the behaviour of different actors within the urban system and their interaction to perform a simulation of urban developments.\(^6\) The open source software "UrbanSim" that he and his colleagues develop (P. Waddell 2002) has been used for various case studies in the United Stated and recently in Europe. The current version of UrbanSim uses buildings and parcels as spatial representation within its simulation process, implementing those as numeric values. As discussed earlier (Schirmer, 2010) this new approach allows for the implementation of detailed spatial geometries into the behavioural modelling process.

The experiences with shapegrammars reported here, show how the output of UrbanSim, could be used in form of attribute maps to create a volumetric representation of simulation results through the use of shapegrammars, i.e. the creation of possible future buildings on each parcel. The categories developed within the framework described here will be a first definition of typologies to integrate as attributes into the development options of the UrbanSim models. Within the project of SustainCity this is currently being tested.

Conclusion

In this paper we described the various uses of shape grammars as available with the software CityEngine in the urban design and urban planning projects of KCAP. As no other software for shape grammars has been tested for these projects, some of the experiences mentioned here might be specific to the software used, but it is expected that similar results could be obtained with other shape grammar software.

We showed that the different scales of urban design and urban planning can be handled through different approaches. For urban design the use as visualisation tool will be more appropriate, for urban planning projects typological testing might be the best

---

5 Example projects are SustainCity (www.SustainCity.org), FutureCityLaboratory (http://www.futurecities.ethz.ch) and Sustainable Urban Patterns (www.SUPat.ethz.ch) that are being realized with participation or cooperation of the IVT.

6 See also Wegener (Wegener 2004) for a review of the field.
approach. By combining shape grammar rules for visualisation, typological testing and enhancing them with logical queries on the geometry of a lot, a complex framework has been created that allows the creation of urban layouts based on the designer's overall strategy.

This shows that shape grammars can be a very effective tool at various scales of urban design and urban planning. Current research at the ETH concentrates on the description and analysis of behaviour of actors within the urban landscape. By extending this one with the options we gain through procedural modelling we expect that behaviour and the geometry will soon merge within the simulation process. This again will be of big advantage to support the planning processes of urban design and urban planning.

Acknowledgments

The authors would like to thank KCAP for the opportunity to publish the results of the internal research as presented here and thank Prof. K.W. Axhausen for his corrections and the inputs on references and urban simulation models.

References


