EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Thumbnail Galleries for Procedural Models

S. Lienhard[2,1] M. Specht[1] B. Neubert[2] M. Pauly[2] P. Müller[1]

[1]Esri R&D Center Zurich, Switzerland [2]LGG, EPFL, Switzerland

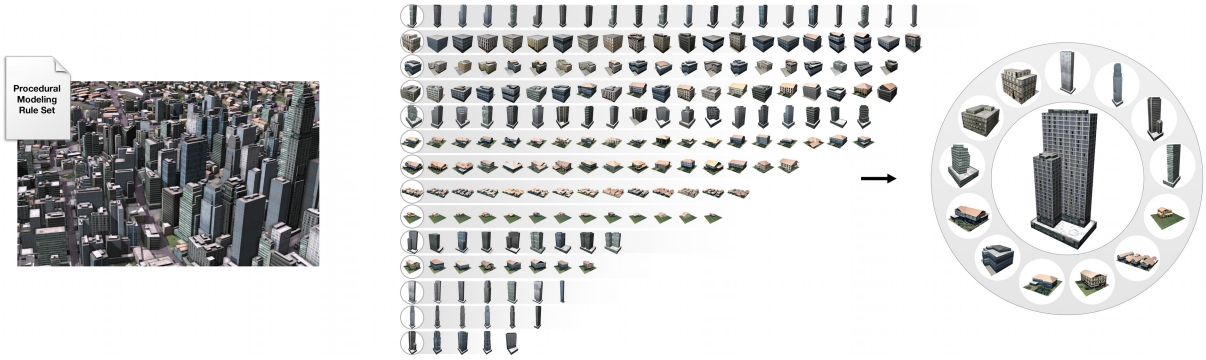


Figure 1: *In procedural modeling, a single rule set can produce a wide variety of 3D models (left). This paper presents a thumbnail gallery generation system which automatically samples a rule set, clusters the resulting models into distinct groups (middle), and selects a representative image for each group to visualize the diversity of the rule set (right).*

Abstract

Procedural modeling allows for the generation of innumerable variations of models from a parameterized, conditional or stochastic rule set. Due to the abstractness, complexity and stochastic nature of rule sets, it is often very difficult to have an understanding of the diversity of models that a given rule set defines. We address this problem by presenting a novel system to automatically generate, cluster, rank, and select a series of representative thumbnail images out of a rule set. We introduce a set of 'view attributes' that can be used to measure the suitability of an image to represent a model, and allow for comparison of different models derived from the same rule set. To find the best thumbnails, we exploit these view attributes on images of models obtained by stochastically sampling the parameter space of the rule set. The resulting thumbnail gallery gives a representative visual impression of the procedural modeling potential of the rule set. Performance is discussed by means of a number of distinct examples and compared to state-of-the-art approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems I.5.3 [Pattern Recognition]: Clustering—Similarity measures

1. Introduction

Rendering and display capabilities are leading to an increasing demand for high quality 3D models. Manually creating a large variety of detailed models is very tedious. Procedural modeling methods help to reduce the manual effort required to define a model, while at the same time providing an efficient way to describe and store a model. Moreover, once a procedural description (i.e., a rule set) of a model is obtained, one can easily generate variations of the model by just manipulating a few rule parameters. The diversity of models that can be represented procedurally is large and ranges from plants and furniture, to buildings and up to whole city layouts.

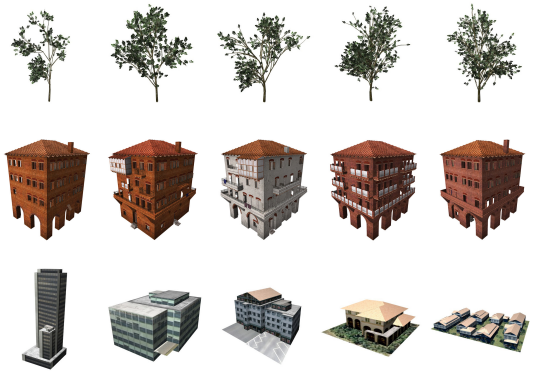


Figure 2: *Each row shows five potential outcomes of one single rule set. A rule set can be stochastic (often with unpredictable design emergence as shown in top row) or parametric (often with numerous, hard-to-use parameter sets) and produces geometry of arbitrary detail. A rule author is not restricted to a specific content scope e.g. a rule set can encode variations* ***within*** *a specific building style (middle row) or variations* ***of*** *different building styles (bottom row). As a consequence, the modeling potential of a rule set is difficult to predict, grasp and represent.*

However, one inherent problem of procedurally defined models is that without any additional information, the induced changes of a single parameter are hard to predict and might even have a global scope. In particular, small variations of one parameter might completely change the appearance of a model, while changes of another parameter might only affect details. Furthermore, because parameters are often not independent and rules can be of stochastic nature and contain conditional decisions, it is hard to oversee the vast variety of different models that can be produced from one rule set. Figure 2 visualizes this by means of three different rule sets. For each rule set, five models were generated by varying the rule set's parameters. Note how the differences in the Roman house in the second row are relatively subtle while the buildings in the bottom row are completely different both in types and numbers.

In order to overcome this problem, we propose a system that automatically generates and arranges a series of thumbnail images into a gallery as illustrated in Figure 1. This so-called thumbnail gallery captures the variety of potential designs to communicate the expressive power of a given rule set.

To define this set of representative images, we first generate a number of model exemplars by stochastically sampling the rule set's parameter space. We introduce a set of view attributes that can be used both for finding an optimal view (intra-model variation) and discriminate between different models (inter-model distance). These view attributes can be computed efficiently in image space using programmable graphics hardware. Amongst different views of the same model we chose the most representative view that maximizes a scoring function based on these view attributes. An adapted weighting scheme of our attributes allows to cluster the representative images of different models. The clusters are ranked and their centers are used to finally depict the model potential of the given rule set in a gallery.

The major contributions of this paper are:

- A set of normalized view attributes that allow for both best view finding and inter-model comparison.
- A system for automatically generating representative images from a procedural modeling rule set into a thumbnail gallery.

Compared to previous work, our view attributes permit finding better best-views. The gallery generation system exploits the above-mentioned view attributes and opens up the door for a number of more user-friendly procedural modeling applications.

2. Related Work

We discuss the related work corresponding to the intermediate steps: procedural modeling, best view selection, image clustering, and model retrieval.

Procedural Modeling. L-systems were proposed for plant and city modeling [PL90, PM01]. More recently, procedural modeling was used e.g. for buildings [MWH*06], furniture [GS09] or floor plans [MSK10]. Procedural modeling was also adapted by the industry, the most prominent examples are Houdini [Inc13] and CityEngine [ESR12]. For our examples we used the latter, but any procedural modeling system could have been applied.

Best View Selection. Some viewpoints reveal and preserve more details of a given model than others. A large body of related work deals with finding the best viewpoint under certain assumptions. Vázquez et al. [VFSH01] define the viewpoint entropy of a given view direction based on the relative area of the model's projected polygonal faces, and they successfully apply this quality measure for molecule visualization [VFSL02]. Mesh saliency, introduced in 2005 by Lee et al. [LVJ05] has the goal to include perception-based metrics in the evaluation of the goodness of a view. A Gaussian-weighted average of mean curvature across different scales is used as a surface feature descriptor and as a measure of local importance, and assigned to each vertex of the mesh. Gooch et al. [GRMS01] introduce a number of heuristics that aim at capturing esthetic considerations for scene composition. Podolak et al. [PSG*06] select views that minimize symmetry to avoid visual redundancy in the depicted objects. Secord et al. [SLF*11] and Dutagaci et al. [DCG10] evaluate a combination of different view attributes in a user study. While these methods define the quality of a viewpoint with

respect to a single model, Laga [Lag10] focuses on finding views that maximally discriminate between different models under the assumption that models belonging to the same class of shapes share the same salient features. Unlike these methods, we exploit view-dependent information to compare different models, and we use the semantic information contained in procedural models to develop new image-based features.

Image Retrieval. Clustering images for thumbnail generation requires a metric that quantifies the similarities of different depictions of the rule set. Defining such a metric is a challenging problem, and methods in this area can be grouped based on the features they use to do so. Low level features, such as color [DMK*01, MRF06] or texture and patterns [LP96, HD03] are readily available, while high level features need to be carefully extracted from images (see [LZLM07] for a detailed overview). In contrast to these methods we are able to employ additional semantic information about our 3D objects given the procedural definition.

3D Model Retrieval. Our approach shares the objective to identify similar objects within a group of models with 3D retrieval techniques (see [BKS*05] for a detailed overview). Here a set of features is computed across a database of models that is then used to implement a distance metric and allows for efficient nearest neighbor queries. In contrast to our approach the features used for this purpose are mostly model driven and an important goal is to be view independent, while we actively strive for discriminating features to identify most representative views.

3. View Attributes

A *view attribute* is a scalar value that quantifies a certain property of a 3D model seen from a given viewpoint and view direction. Examples are the visible surface, silhouette length, or contrast in the observed image (exact definitions follow below).

In general we use such view attributes to achieve two different goals:

1. We compare different views of a single model by ranking their view attributes to find the 'best' or most appealing view. To this end, a weighted linear combination of view attributes is used to assign a 'goodness' score to each viewpoint (see Section 4.1 for details) that we optimize for.
2. As a new contribution, we use the same view attributes to compare and distinguish different models seen from a fixed viewpoint. This inter-model comparison is used to cluster different variations of our procedural models into distinct groups.

Note that our approach is inherently different from classical 3D shape descriptors used for object retrieval. We show that 2D view attributes that work well for best view finding can also be used to compare different derivations of a procedural rule set seen from the same viewpoint. This inter-model comparison approach is also faster than comparing based on 3D descriptors.

The problem with most existing view attributes is that they are not suited for inter-model comparison. For example, surface visibility, the ratio of the model's visible to its total surface [PB96], is a relative value and has no information about the real size of objects.

Our scoring function is based on a combination of adapted existing and several new view attributes that are specifically designed to work well for inter-model comparison. In this section we present these view attributes grouped by the different aspects they capture: geometry based, esthetic, and semantic view attributes.

To calculate the view attributes for a given viewpoint, the model is rendered only once on the GPU and such information as normals, luminance, and color-coded information about terminals (used for view attributes a_2, a_7, and a_8 defined below) is stored. All proposed view attributes are computed using solely the stored 2D data and no further analysis of the polygonal geometry is necessary, which allows for very fast evaluation for a number of different viewpoints. We only look at the geometry in a preprocessing step that stores face and terminal area sizes in lookup tables and the color mapping (color value to face or terminal index) for our color-coded buffers. This information is reused every time we need to compute the view attributes from a new perspective. In our application we want to see the full model as large as possible. To this end, the camera is always placed at the distance where the model's minimal bounding sphere fits tightly into the frustum. We normalize all view attributes to lie in $[0,1]$.

3.1. Geometric View Attributes

a_1 **Pixel Count** is the ratio of projected area n (in pixels) of the model on the screen to the overall image size [PB96]:

$$a_1 = \frac{4}{\pi} \frac{n}{\text{width} \times \text{height}}.$$

The idea is that the larger the projected area, the more you see of the object.

a_2 **Surface.** The ratio of visible to total surface area seen from a specific viewpoint is called surface visibility [PB96]. Maximizing this view attribute minimizes the amount of occluded surface area. For our application, we define the surface view attribute as the logarithm of base $\mathcal{M} = 10^6$ of the visible surface area A in m^2:

$$a_2 = \log_{\mathcal{M}}(A+1).$$

Due to the logarithm there is a higher resolution for lower values of model sizes and a decreasingly lower resolution as the models get bigger. The choice of $\mathcal{M}$ leads to attribute

values in $[0,1]$ (unless we encounter models with visible surface larger than $10^6 m^2$, e.g., a fictional super skyscraper). Table 1 shows how a_2 correlates to object size. We increase A by one to be able to assess also small objects.

Example	max vis. surf.	a_2
Furniture	$\approx 1m^2$	0.05
Vehicle	$\approx 10m^2$	0.17
Residential building	$\approx 200m^2$	0.38
Apartment building	$\approx 1000m^2$	0.5
Office building	$\approx 10000m^2$	0.67
High-rise	$\approx 60000m^2$	0.8
City block	$\approx 300000m^2$	0.91

Table 1: *Surface view attribute a_2 for objects of different size seen from one point.*

a_3 **Silhouette length.** The longer the object's silhouette is in the rendered image, the more interesting details such as protrusions and concavities should be visible. To bring this value into a reasonable range, we define the view attribute as:

$$a_3 = \log_{16}\left(\frac{s}{l}\right),$$

where s is the silhouette length in pixels, and l is the side length of the largest square that could possibly be rendered (largest square that still fits into the projection of the objectâĂŹs bounding sphere). We choose 16 as base for the logarithm for the following reason: A standard rectangular object results in an outline of $s \approx 4l$. For this case, we want $a_3 \approx 0.5$ which results in base 16. Furthermore, tests showed that the outline of length $s \approx 16l$ ($a_3 \approx 1$) is an adequate limit for shapes of high complexity with many concavities.

3.2. Esthetic View Attributes

While the esthetic properties of an image are highly subjective, there exist compositional heuristics, e.g., the widely known *rule of thirds* [Smi97].

a_4 **Contrast.** Higher contrast stands for a larger dynamic range and a visual appealing image. We use root means square contrast [Pel90] of the rendered image, because it can be computed efficiently:

$$a_4 = 2\sqrt{\frac{1}{n}\sum_{i=0}^{n}\left(I_i - \bar{I}\right)^2},$$

where n is the number of pixels, I_i is the luminance at pixel i, and $\bar{I}$ is the average luminance of all n pixels. We only consider the pixels that have been rasterized and neglect the uniformly colored background.

Since the contrast depends on the lighting, we use the same lighting conditions for all models. We use a standard three point lighting technique, where key, fill, and back light are in fixed positions with respect to the camera and the scene center (see, e.g., [Bir00]).

a_5 **Normal ratio.** One artistic composition rule states that the projections of front/side/top of an object should have relative areas of 4/2/1 in an image [EB92, Arn54]. Gooch et al. [GRMS01] orient the object's bounding box so that the projections of its three visible sides fulfill that ratio (front and side dimensions can be exchanged). Bounding boxes are only a rough approximation of the true geometry and we analyze image space normals instead. They are grouped into the three categories: left (counting towards the front), right (counting towards the side), and up. While it is possible to compose these three variables into a scalar that quantifies the deviation from the desired ratio, our experience has shown that the same value doesn't perform well for inter-model comparison. Therefore, we decided to merely distinguish left from right pointing normals and defined the normal ratio as:

$$a_5' = \frac{l}{l+r}, \qquad a_5 = 1 - \left|1 - \frac{3}{2}a_5'\right|,$$

where l and r are the number of pixels with normals pointing towards the left or the right respectively. a_5 is used for best view selection and a_5' for inter-model comparison. The optimal ratio $\frac{l}{r} = \frac{4}{2}$ leads to $a_5' = \frac{2}{3}$ which maximizes $a_5 = 1$.

a_6 **Form.** Most appealing are renderings for which the object covers a greater part of the image. Degenerate objects, i.e., with very thin renderings are less favorable. The form view attribute is a function of the ratio of the height and the width of the 2D axis aligned bounding box (AABB) of the rendering:

$$a_6' = \frac{1}{2}\left(\log_l\left(\frac{h}{w}\right) + 1\right), \qquad a_6 = 1 - \left|\log_l\left(\frac{h}{w}\right)\right|,$$

where h and w are the height and the width of the AABB. The base of the logarithm, l, is the side length of the largest possible square (same l as in a_3).

For inter-model comparison we use a_6', which allows to differentiate between flat, square, and tall thin AABBs. a_6 is used for best view selection, a square is best while horizontally and vertically thin AABBs are equally unwanted. This view attribute is similar to a_1 for best view finding but it is useful for distinguishing thin tall from wide flat objects.

3.3. Semantic View Attributes

The procedurally generated models we are working with are annotated with extra information, e.g., the derivation tree or correspondences between geometry and terminal symbols. The following semantic view attributes exploit this extra information.

a_7 **Visible terminal types.** Every model is composed of terminal symbols that represent the model's geometry. Seeing many terminals does not necessarily mean that one obtains a lot of information, because many terminal shapes might be of the same type. What matters most is the number of different terminal symbols that are visible. Most of our examples

contain between 2 and 100 terminals (e.g., the Petronas Towers in Figure 6 only uses two terminal types called *glass* and *metal*). We define:

$$a_7 = \log_{100} N_t,$$

where N_t is the number of visible terminal symbols in the image. The logarithm maps that count onto $[0,1]$ (assuming that there are not more than 100 terminals types). Changes of N_t have more importance for situations with few visible terminal types.

a_8 **Terminal entropy.** Viewpoint entropy [VFSH01,Váz03] is an adaptation of Shannon entropy [Sha48]. It uses as probability distribution the relative area of the projected faces over the sphere: $p_i = \frac{f_i}{A_{\text{tot}}}$, with f_i the projected area of face i and A_{tot} the total projected area of all visible faces (both in solid angles). Our view attribute measures the entropy of visible terminal types. Terminal types provide a more semantic entity of information than polygonal faces:

$$a_8 = -\sum_{i=0}^{N_t} \frac{t_i}{A_{\text{tot}}} \log_{100} \frac{t_i}{A_{\text{tot}}},$$

where t_i is the projected area of terminal type i, and A_{tot} is the total projected area of the object. Both variables are again in solid angle. N_t is again the number of visible terminal types. We use the same base for the logarithm as a_7, with the same reasoning.

4. Thumbnail Gallery Generation

In this section we introduce a system for the automatic creation of a thumbnail gallery out of a rule set. The system is outlined in Algorithm 1 and consists of the following steps:

1. The default model is procedurally generated in CityEngine (CE) using the default values of the rule set's parameters. This model is then used to calculate the best viewpoint for the rule set (see Subsection 4.1).
2. The rule set's parameter space is stochastically sampled to generate model variations. Each model is rendered from the previously found viewpoint to obtain its thumbnail image and view attributes (Subsection 4.2).
3. The resulting list of view attributes is clustered into distinct groups, and for each group a center is selected. The thumbnail images associated with these centers define the final thumbnail gallery (Subsection 4.3).
4. For a comprehensible visualization, the selected representatives are sorted radially around the default model in a reduced 2D space. The latter is obtained by applying a PCA [Jol86] on the view attributes (Subsection 4.4).

4.1. Best View Selection

Once the default model has been created, we search for its best view by sampling a number of viewpoints on a sphere placed around the model. The camera's view direction always points towards the center of the sphere. We discard viewpoints that lie below the base plane of the model or look down too steeply. We found that good viewpoints were located between 0° and 45° above ground [BTBV96]. The model is rendered from all sample cameras and the view attributes are stored.

To increase stability, the view attributes are normalized to the $[0,1]$ range, i.e., for a given view attribute, the lowest sampled value will be mapped to 0, the highest value to 1, and in-between values are linearly interpolated. To rank the viewpoints, a score is defined as a linear combination of the normalized view attributes. We empirically determined the weights listed in the second column of Table 2. More sophisticated schemes for automatically choosing weights, e.g., through semi-supervised learning, are an interesting area of future work [KHS10]. For existing view attributes we started with values that Secord et al. [SLF*11] concluded from a user study.

Algorithm 1 System Overview

```
# compute best view for default model
defaultModel ← CE.generate(ruleset, ruleset.defaultParams)
bestview ← computeBestview(defaultModel, bestviewWeights)

# sample in rule set's parameter space
<thumbnail_0, viewAttrs_0> ← render(defaultModel, bestview)
for i = 1 to nSamples do
    params ← stochasticSample(ruleset.paramRanges)
    model ← CE.generate(ruleset, params)
    <thumbnail_i, viewAttrs_i> ← render(model, bestview)

# cluster view attributes and select thumbnails
clusters ← calcClusters(viewAttrs, clusterWeights, nClusters)
for each cluster in clusters do
    if 0 ∉ cluster then  # discard cluster with default model
        find viewAttrs_{i∈cluster} closest to center of cluster
        add index i to selection

# sort selected thumbnails and create gallery
viewAttrs2D ← PCA(viewAttrs, 2)
selection.sortRadial(viewAttrs2D_{∀i∈selection}, viewAttrs2D_0)
gallery ← thumbnail_{0∪∀j∈selection}
```

View Attribute	bestviewWeights	clusterWeights
Pixel count (a_1)	15%	10%
Surface (a_2)	20%	25%
Silhouette (a_3)	5%	30%
Contrast (a_4)	2%	2%
Normal ratio (a_5 or a'_5)	10%	5%
Form (a_6 or a'_6)	3%	3%
Visible terminal types (a_7)	25%	5%
Terminal entropy (a_8)	20%	20%

Table 2: *The weights for the view attributes for best view selection and for clustering. Note that normal ratio and form use slightly different view attribute definitions for best view selection and clustering.*

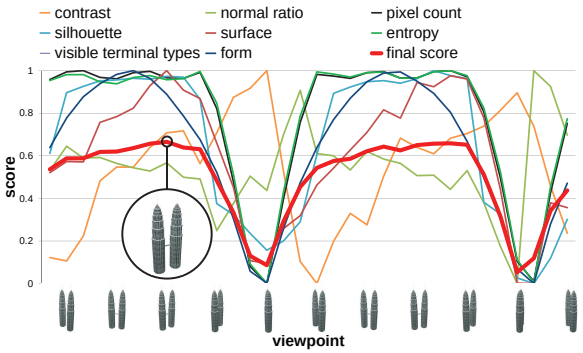
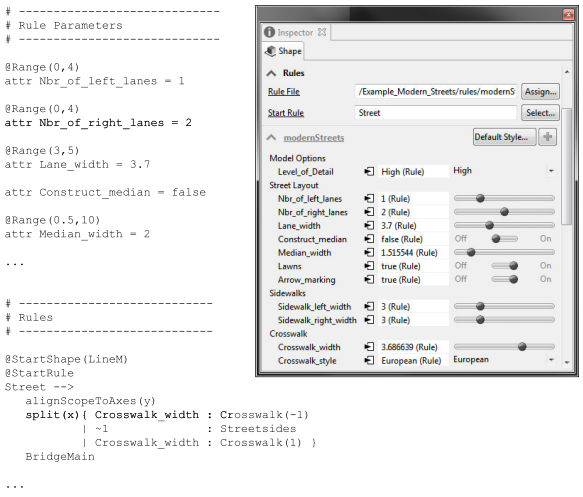


Figure 3: *Different viewpoints versus normalized view attributes for the Petronas Towers. The final score (red) is a linear combination with the weights in Table 2. Its peak using the corresponding thumbnail is highlighted.*



```
# ----------------------------
# Rule Parameters
# ----------------------------

@Range(0,4)
attr Nbr_of_left_lanes = 1

@Range(0,4)
attr Nbr_of_right_lanes = 2

@Range(3,5)
attr Lane_width = 3.7

attr Construct_median = false

@Range(0.5,10)
attr Median_width = 2

...


# ----------------------------
# Rules
# ----------------------------

@StartShape(LineM)
@StartRule
Street -->
   alignScopeToAxes(y)
   split(x){ Crosswalk_width : Crosswalk(-1)
           | ~1              : Streetsides
           | Crosswalk_width : Crosswalk(1) }
   BridgeMain

...
```


Figure 4: *Left: CGA rule set excerpt with parameter ranges and start rule. Right: User interface to control the many parameters of the rule set.*

The graph in Figure 3 plots different viewpoints versus the normalized view attributes and the final score. There are 32 viewpoints sampled on a ring $\frac{\pi}{8}$ above the horizon. Note that the overall score is minimized when one tower is hidden behind the other one.

4.2. Stochastic Sampling of Rule Parameters

A rule set typically comes with several parameters to steer the generation of the procedural 3D model. In CityEngine, rule sets are encoded using the shape grammar CGA [MWH*06] and continuous or discrete parameters can be defined as shown on the left in Figure 4. Note that every parameter needs to be initialized with a default value. Authors can also define the range of meaningful values for parameters using `@Range` annotations. In CityEngine, these ranges are used for constructing the sliders in the UI (Figure 4 on the right) and for us they limit the space within which we generate samples.

A rule set contains a start rule which is applied on an initial shape which is annotated with `@StartRule` in CGA. But the initial shape is undefined and could be of arbitrary form, dimension, or geometry. Thus, it could have a high impact on the generation, e.g., a conditional rule might determine to build a high-rise building instead of a small house based on the size of the initial shape. As a consequence, we extend CGA with the `@StartShape` annotation to set one or many predefined initial shapes such as `Point`, `Line`, `Rect` or `Cube`. In the example on the left of Figure 4, the author uses `@StartShape(LineM)` to define the default initial shape as a line (the letters `S`, `M` and `L` denote dimensions 10, 30 and 100 meters).

As described in Algorithm 1, we now stochastically sample the rule set within its given parameter ranges and initial shapes (by using CityEngine's Python interface). For each resulting model we render the thumbnail image and view attributes from the best viewpoint found in Subsection 4.1. The reasons why we use only the best view of the default model even though it might not be the best view when jointly considering all models are twofold: (1) performance, i.e., calculating best views of more samples requires additional processing time, and (2) since all initial shapes are similarly aligned, changing the viewpoint for each sample only worsens the visual understanding of model differences. As an alternative we also experimented with calculating the best view of every sample and taking the one viewpoint that had the most support (every model sample would give one vote to its best viewpoint). The results are similar but there is the drawback of having to compute the view attributes for all viewpoints for all samples.

4.3. Clustering View Attributes

As a next step, the list of view attributes is clustered into a given number of groups. Therefore we applied a slightly modified version of the k-means++ clustering algorithm [AV07]. K-means++ chooses only the first initial clustering seed completely at random and applies a distance-based probability heuristic to determine the remaining initial cluster seeds. However, to make the clustering even more stable, we set the first initial seed to the view attributes of the default model. This is justified by the fact that most rule authors intuitively place the default model near the center of the design space.

The cluster distance function is a weighted linear combination of the view attribute deltas. Since our view attributes are typically well distributed in the [0, 1] range, the weights reflect the importance of the corresponding view attributes

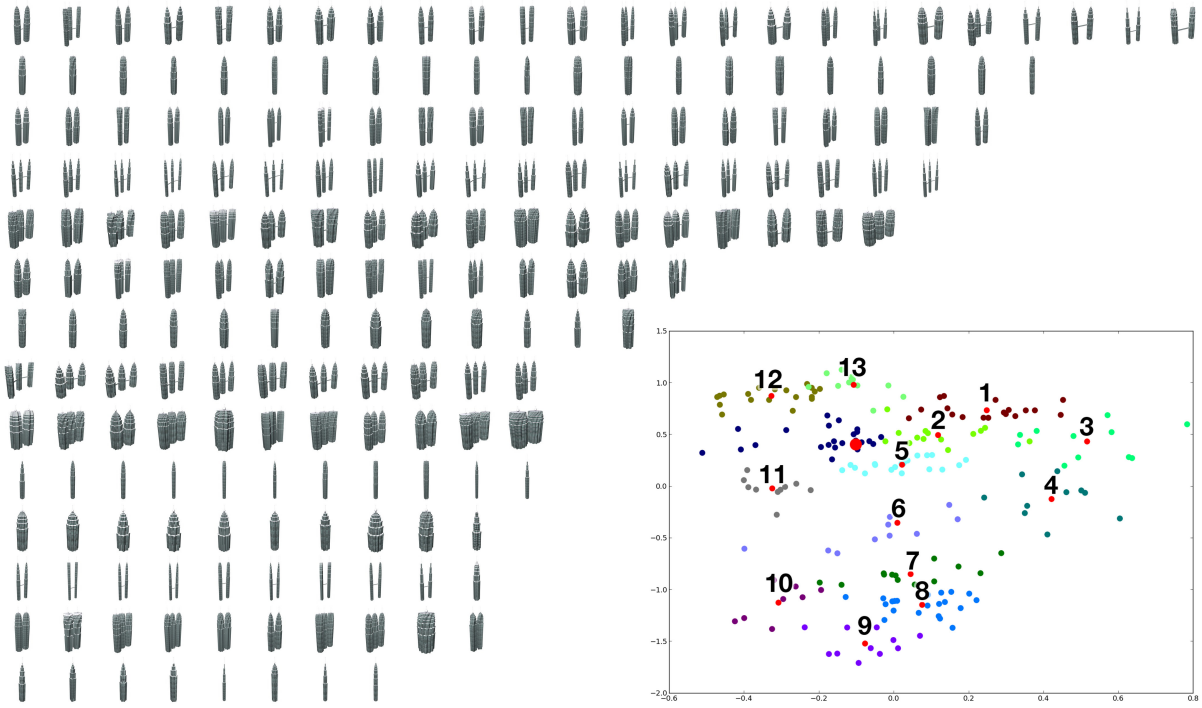


Figure 5: *Stochastic sampling of the Petronas Towers rule set. Left: Clustering results where each row of thumbnails represents one group with similar models. The cluster which received the most samples is depicted in the top row. The distance to each cluster center increases from left to right, i.e., the model on the left is the one nearest to the center of its cluster. Right: Representation of the clustering results using the first two principal components. The samples nearest to their cluster are marked red and the large one is the default model. The numbers illustrate the radial sorting around the default model and correspond to the numbers in Figure 6.*

for clustering. Accordingly, they are actually a user setting which decides what types of models should be grouped together. The weights we use were determined empirically, they are listed in Table 2 on the right. They provide visually appealing and representative clusters for arbitrary rule sets. A cluster result is shown on the left Figure 5.

4.4. Thumbnail Gallery Creation

To illustrate a cluster, we select the sample nearest to its center. To arrange the selected thumbnail images in a visually comprehensible way, we applied UI principles of Design Galleries [MAB*97]. There, the current design is in the middle and suggested variations are arranged around it in a manner that correlates to the editing distances. In our case, the default model is the center and the selected thumbnails of the other clusters are arranged around it as shown in Figure 6.

The leftmost thumbnail column in Figure 5 shows that the cluster size is not a well-suited sorting criteria for the radial arrangement. The user wants to visually compare similar thumbnails and is less interested in the cluster sizes. Therefore, we run PCA on the view attributes to project the selected samples from their original eight-dimensional space into a two-dimensional subspace spanned by the first two principal components. Within this space, the representatives are sorted radially around the default model. This is illustrated in Figure 5 on the right.

This arrangement is also the reason why we typically generate 14 clusters. If we display the selected samples at third the size of the default model, we can fit exactly 13 thumbnails on the circle. Nonetheless, the number of clusters can be chosen by the user.

5. Results

5.1. Best View

To evaluate our best view method, we conducted a preliminary user study with 39 participants. The test data set contained 21 buildings, each rendered from the optimal perspective according to our view attribute combination and according to the ‘linear-5’ combination suggested by Secord et

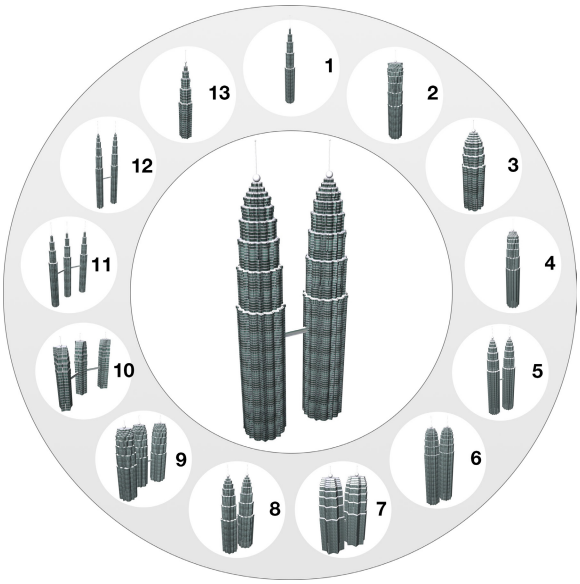


Figure 6: *Resulting thumbnail gallery of the Petronas Towers rule set. The thumbnail in the middle shows the default model and those around it depict other representatives. The ordering is the same as in Figure 5.*

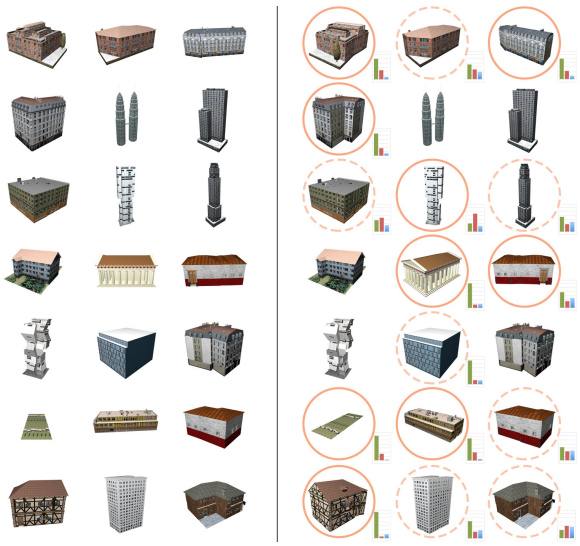


Figure 7: *Secord et al.'s [SLF*11] view attribute combination (left) versus ours (right). Circles denote differences between both methods, dashed circles stand for minor differences only. The bar charts show the user preference counts: left is ours, middle is Secord et al., right is no preference.*

al. [SLF*11]. For our test set, both systems provide the same thumbnails for 5 of the exemplars (different view directions indicated as circles in Figure 7). For the remaining models the subjects preferred our view direction in 351 cases, 161 views according to Secord et al., and had no preference in 112 cases. A significance test rejects H_0 (H_0: no preference, H_A: preference for the view direction obtained from our view attributes) clearly within the $p < 0.001$ (one-tailed) confidence interval. Focusing on the models for which the two viewing directions considerably changed (more than 10 degrees, see Fig. 7 solid circles) the subjects preferred our suggestions even more clearly (in 225 cases compared to 84 cases for the alternative, with 42 occasions of no preference).

5.2. Clustering

Clustering behavior varies for different rule sets. Generally, we observed that the clusters stabilize when at least 200 samples are used. K-means++ proved to be the most natural way to initialize clustering for our domain as we want the default model as the global center. We also experimented with other clustering methods with worse results: hierarchical clustering [FLP*51] with dendrograms, spectral clustering, and mean shift clustering [FH75].

Figure 8 shows thumbnail galleries for three different rule sets: Philadelphia, Paris, and procedural streets. The running times for those rule sets are listed in Table 3. We used OpenGL and rendered the thumbnails at a resolution of 500x500pixels on an Intel Core2 Duo 2.8 GHz Laptop with a Nvidia Quadro FX3700M graphics card. Clustering was done with NumPy and took about 0.5s for each rule set. The times show that model generation is the most expensive part of the system. The attribute computation varies between rule sets (Philadelphia takes 4 times longer than Paris). We believe that the reason for this is the increased number lookup-table reads (the number depends on the number of terminals and faces in the model) in former example.

Rule Set	Model generate	Best View	Attr. calc.
Petronas	334.00s	0.43s	64.34s
Philadelphia	279.60s	0.61s	123.93s
Paris	38.45s	0.67s	31.88s
Proc. streets	113.27s	0.35s	111.08s

Table 3: *Running times for our system for different rule sets using 200 model samples. The best view selection algorithm considered 32 different viewpoints.*

6. Discussion and Conclusion

Thumbnail galleries provide a new visual tool for procedural modeling that can significantly simplify rule selection during the content creation process. Our experiments demonstrate that the thumbnails automatically generated by our method yield good visual representations of the model diversity of a rule set.

Figure 8: *Thumbnail galleries for three rule sets. Left: Philadelphia, center: Paris-style building, right: procedural street.*

Our system also has some limitations which open areas for future research. We currently fail to distinguish models that have a very similar overall shape that differs only on lower-scale structural details. We believe that view attributes encoding the silhouette (e.g., Fourier descriptors [ZR72]) or the shape of the 2D rendering (e.g., Zernike moments [KH90]) could remedy those shortcomings. Another problem are rule parameters that change minor details, e.g., a building's window width or the leaf shape of a tree. Those changes are hard to spot from a perspective that features the full model. A beneficial feature of the system would be to detect those shapes and present close-up images. One could further investigate new color and texture based view attributes as our system ignores this information almost entirely (color changes influence the luminance images and have a minor effect on the contrast view attribute).

User-guided exploration of the procedural design space similar to Design Galleries [MAB*97] is also worth exploring: the user would repeatedly select a model while the system automatically provides new suggestions of models close to the selected one. A related question is if it's possible to reverse-engineer the influence of rule parameters given the view attribute.

Our system currently just samples parameters which have an optional range annotation. User-guided refinement could be used to detect sensible ranges for non-annotated parameters. Also our sampling strategy could be improved with an adaptive approach. Rather than sampling all parameters with the same probability we could detect the ones which lead to large variations of the view attributes and sample them more densely.

Conclusion. We present a system that finds the best view of a procedural model and that generates a thumbnail gallery depicting the design possibilities of a rule set. The best view selection algorithm works well for arbitrary mesh topologies and outperforms existing methods. Clustering in conjunction with our novel view attributes is a first step towards visualizing the immense variety of models encoded in a procedural rule set. To the best of our knowledge, we are the first to present such a system.

Acknowledgements

The research leading to these results has received funding from Esri and from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 257453: COSYM. We thank Matthias Buehler for many insightful discussions and Andreas Ulmer for the last minute support. We also thank the anonymous reviewers for their valuable comments.

References

[Arn54] ARNHEIM R.: *Art and Visual Perception: A Psychology of the Creative Eye*. University of California Press, 1954. 4

[AV07] ARTHUR D., VASSILVITSKII S.: k-means++: The Advantages of Careful Seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms* (2007), SODA '07, pp. 1027–1035. 6

[Bir00] BIRN J.: *Digital Lighting and Rendering*. New Riders Publishing, Indianapolis, Indiana, 2000. 4

[BKS*05] BUSTOS B., KEIM D. A., SAUPE D., SCHRECK T., VRANIĆ D. V.: Feature-based Similarity Search in 3D Object Databases. *ACM Computing Surveys 37* (2005). 3

[BTBV96] BLANZ V., TARR M. J., BÜLTHOFF H. H., VETTER T.: What Object Attributes Determine Canonical Views? *Perception 28*, 5 (1996), 575–599. 5

[DCG10] DUTAGACI H., CHEUNG C. P., GODIL A.: A Benchmark for Best View Selection of 3D Objects. In *Proceedings of the ACM Workshop on 3D Object Retrieval* (2010), 3DOR '10, pp. 45–50. 2

[DMK*01] DENG Y., MANJUNATH B. S., KENNEY C., MOORE M. S., SHIN H.: An Efficient Color Representation for Image

Retrieval. *IEEE Transactions on Image Processing 10*, 1 (2001), 140–147. 3

[EB92] EDELMAN S., BÜLTHOFF H. H.: Orientation Dependence in the recognition of Familiar and Novel Views of Three-dimensional Objects. *Vision Research 32* (1992), 2385–2400. 4

[ESR12] ESRI: CityEngine, 2012. URL: http://www.esri.com/software/cityengine. 2

[FH75] FUKUNAGA K., HOSTETLER L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 1 (1975), 32–40. 8

[FLP*51] FLOREK K., LUKASZEWICZ J., PERKAL J., STEINHAUS H., ZUBRZYCKI S.: Sur la liaison et la division des points d'un ensemble fin. *Colloquium Mathematicae (Vol. 2, No. 3-4)* (1951). 8

[GRMS01] GOOCH B., REINHARD E., MOULDING C., SHIRLEY P.: Artistic Composition for Image Creation. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (2001), pp. 83–88. 2, 4

[GS09] GERMER T., SCHWARZ M.: Procedural Arrangement of Furniture for Real-Time Walkthroughs. *Computer Graphics Forum 28*, 8 (2009), 2068–2078. 2

[HD03] HUANG P. W., DAI S. K.: Image Retrieval by Texture Similarity. *Pattern Recognition 36*, 3 (2003), 665–679. 3

[Inc13] INC. S. E. S.: Houdini, 2013. URL: http://www.sidefx.com. 2

[Jol86] JOLLIFFE I. T.: *Principal Component Analysis*. Springer Verlag, 1986. 5

[KH90] KHOTANZAD A., HONG Y. H.: Invariant Image Recognition by Zernike Moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 5 (1990), 489–497. 9

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3D Mesh Segmentation and Labeling. *ACM Trans. Graph. 29*, 4 (2010), 102:1–102:12. 5

[Lag10] LAGA H.: Semantics-driven Approach for Automatic Selection of Best Views of 3D Shapes. In *Proceedings of the 3rd Eurographics Workshop on 3D Object Retrieval* (2010), Eurographics 3DOR '10, pp. 15–22. 3

[LP96] LIU F., PICARD R. W.: Periodicity, Directionality, and Randomness: Wold features for Image Modeling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 7 (1996), 722–733. 3

[LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. *ACM Trans. Graph. 24*, 3 (2005), 659–666. 2

[LZLM07] LIU Y., ZHANG D., LU G., MA W.-Y.: A Survey of Content-based Image Retrieval With High-level Semantics. *Pattern Recognition 40*, 1 (2007), 262–282. 3

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proceedings of SIGGRAPH 97, Annual Conference Series* (1997), pp. 389–400. 7, 9

[MRF06] MALINGA B., RAICU D., FURST J.: Local vs. Global Histogram-Based Color Image Clustering. In *CTI Research Symposium* (2006). 3

[MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated Residential Building Layouts. In *ACM SIGGRAPH Asia 2010 papers* (2010), SIGGRAPH Asia '10, pp. 181:1–181:12. 2

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural Modeling of Buildings. *ACM Trans. Graph. 25*, 3 (2006), 614–623. 2, 6

[PB96] PLEMENOS D., BENAYADA M.: Intelligent Display in Scene Modeling. New Techniques to Automatically Compute Good Views. In *Proceedings of GraphiCon* (1996). 3

[Pel90] PELI E.: Contrast in Complex Images. *Journal of the Optical Society of America A 7*, 10 (1990), 2032–2040. 4

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer Verlag, 1990. 2

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural Modeling of Cities. In *Proceedings of SIGGRAPH 01, Annual Conference Series* (2001), pp. 301–308. 2

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A Planar-Reflective Symmetry Transform for 3D Shapes. *ACM Trans. Graph. 25*, 3 (2006). 2

[Sha48] SHANNON C. E.: A mathematical theory of communication. *Bell system technical journal 27* (1948). 5

[SLF*11] SECORD A., LU J., FINKELSTEIN A., SINGH M., NEALEN A.: Perceptual Models of Viewpoint Preference. *ACM Trans. Graph. 30*, 5 (2011). 2, 5, 8

[Smi97] SMITH J. T.: *Remarks on Rural Scenery*. Nathaniel Smith et al., 1797. 4

[Váz03] VÁZQUEZ P.-P.: *On the Selection of Good Views and its Application to Computer Graphics*. PhD thesis, Technical University of Catalonia, 2003. 5

[VFSH01] VÁZQUEZ P.-P., FEIXAS M., SBERT M., HEIDRICH W.: Viewpoint Selection using Viewpoint Entropy. In *Proceedings of the Vision Modeling and Visualization Conference* (2001), VMV '01, pp. 273–280. 2, 5

[VFSL02] VÁZQUEZ P.-P., FEIXAS M., SBERT M., LLOBET A.: Viewpoint Entropy: a New Tool for Obtaining Good Views of Molecules. In *Proceedings of the Symposium on Data Visualisation* (2002), VISSYM '02, pp. 183–188. 2

[ZR72] ZAHN C. T., ROSKIES R. Z.: Fourier Descriptors for Plane Closed Curves. *IEEE Transactions on Computers 21*, 3 (1972), 269–281. 9