Creature grammar for creative modeling of 3D monsters

Xuekun Guo[a], Juncong Lin[b,*], Kai Xu[c,d], Xiaogang Jin[a,*]

[a]*State Key Lab of CAD&CG, Zhejiang University, China*
[b]*Software School of Xiamen University, China*
[c]*National University of Defense Technology, China*
[d]*Shenzhen VisuCA Key Lab/SIAT, China*

Abstract

Monsters and strange creatures are frequently demanded in 3D games and movies. Modeling such kind of objects calls for creativity and imagination. Especially in a scenario where a large number of monsters with various shapes and styles are required, the designing and modeling process becomes even more challenging. We present a system to assist artists in the creative design of a large collection of various 3D monsters. Starting with a small set of shapes manually selected from different categories, our system iteratively generates sets of monster models serving as the artist's reference and inspiration. The key component of our system is a so-called *creature grammar*, which is a shape grammar tailored for the generation of 3D monsters. Creature grammar governs the evolution from creatures with regular structures gradually into monsters with more and more abnormal structures through evolving the arrangement and number of shape parts, while preserving the semantics prescribed as prior knowledge. Experiments show that even starting with a small set of shapes from a few categories of common creatures (e.g., humanoids, bird-like creatures and quadrupeds), our system can produce a large set of unexpected monsters with both shape diversity and visual plausibility, thus providing great support for the user's creative design.

Keywords: Monster modeling, Creativity support, Creature grammar, Diversity

1. Introduction

Monsters are imaginary or legendary creatures with abnormal structures or physical deformities, which inspire horror or disgust. They are vastly found in comic books, movies and computer games. Some of them are so famous and fascinating that they keep the audiences or game players intrigued, for example, the *Fate Spinner* and *Lamassu* from *Might & Magic: Heroes*, and the *Centaur* from *Harry Potter films* (see Fig. 1). Modeling such kind of objects is a great challenge for artists, because of their bizarre shapes and structures, calling for creativity and imagination. Especially in a scenario where a large number of monsters with various shapes and styles are required, the designing and modeling process becomes even more challenging.

Although great strides have been made to assist 3D modeling, the support of creativity in 3D modeling process is still an emerging topic. As pointed out by Shneiderman et al. in [1], free exploration of alternatives is a key characteristic of creativity support tools. Thus research on computational creativity support seeks for techniques for generating customized examples that allow the user to explore the space of possibilities. For 3D modeling, a popular approach is to provide data-driven suggestions tailored to the user's current design. Such suggestions, either relevant parts [2, 3] or complete shapes [4, 5], are usually drawn from a set of pre-existing creations. Compared with part suggesting methods, shape suggesting methods not only can be used to inspire future creations, but also be used to amplify an existing shape database efficiently. Such kind of creativity support is especially useful for monster modeling since the shape and structure of monsters can be arbitrarily unexpected. However, existing methods for shape suggestions are more suitable to generate models belonging to the same category and with the same structure as input shapes, as they need pre-established correspondences among the existing creations. Monster models usually have characteristics from different categories, such as bird-like creatures, humanoids or quadrupeds. Besides, monsters are characteristic of various unexpected structures.

In this paper, we present an inspiration-oriented procedural approach to help artists create sets of various monsters from regular creatures quickly. The key component of our system is the creature grammar, which formally defines the way of evolving regular creatures into monsters with abnormal structures, while preserving the semantics of biological forms. The semantics consists of part adjacency relations, the number of parts from each part category, and symmetries. Given a small set of models from different categories (e.g., humanoids, bird-like creatures, and quadrupeds), our system adopts the creature grammar to iteratively synthesize a diverse set of monsters for artists for reference. The generated monsters become more and more complex in structure during the iteration, and thus they provide artists inspirations.

A key observation on monster modeling is that 3D artists

*Corresponding authors
Email addresses: jclin@xmu.edu.cn (Juncong Lin), jin@cad.zju.edu.cn (Xiaogang Jin)

commonly design monsters by firstly creating a regular creature and then modifying the shape structure to explore various possibilities to make the shape more and more abnormal. The operations which artists commonly apply can be classified into three types: (1) *increase*, duplicating or triplicating one part (e.g., triplicating the arm of a humanoid (Fig. 1(a)[1])); (2) *combine*, concatenating two torsos together (e.g., attaching the upper part of a human to the torso of a horse to create a Centaur-like creature (Fig. 1(b)[2])); (3) *insert*, adding a new type of part to the current shape (e.g., adding a wing to a quadruped (Fig. 1(c)[1])).

Mimicking the modeling process mentioned above, we propose the creature grammar, a tailored shape grammar for the generation of monsters. Creature grammar works with a set of shapes and generates new shapes by combining the configuration of two shapes selected from the current shape set, while preserving semantics of creatures. There are three types of rules in our creature grammar: (1) *selection rules*, which select shapes (rules) from a shape (rule) set; (2) *structure rules*, which alter the topological structure of the shape based on the three operations mentioned above; (3) *geometry rules*, which interpret the generated configuration into a geometry instance to get a monster shape. Based on the creature grammar, our system could automatically expand the structures of regular creatures, thus producing visually plausible monster shapes with abnormal structures.

2. Related work

Grammar-based procedural modeling. Grammar-based procedural techniques have been vastly adopted in computer graphics for various applications. Since introduced by Lindenmayer in [6] for a description of the cellular subdivision and endogenous information exchange between cells, L-systems have been applied and extended in many different directions such as plant modeling [7] and urban modeling [8, 9, 10]. As L-systems are linear, shape grammars [11] were then proposed to deal with connecting of 2D elements.

The concept of shape grammar was extended into split grammars for urban modeling [12]. Split grammars were further adapted to urban reconstructions from images and point clouds in [13]. The interactive editing of procedural rules in [14] exploits higher levels of user interactions and allows for an immediate visual feedback of the results. Finally, several attempts have been made to combine physically-based modeling and procedural systems [15]. The main difference between the creature grammar from previous ones is that the creature grammar works on configurations of shapes in a set (to generate new shape configurations through exchanging and expanding) while previous grammars deal with the expanding of an initial simple configuration (consisting of none or few basic primitives) into a complex one. To some extent, the creature grammar is tailored to design monsters (a special kind of freeform objects with abnormal structures) while previous grammars are designed to generate objects with repetitive primitives (trees or man-made objects).

For L-systems and shape grammars, one has to employ context sensitive rules [8] or place extra conditions [9, 12] into the grammar system to encode structural constraints. Monster modeling involves several structural constraints, which are encoded with some context sensitive rules in our shape grammar.

Data-driven object modeling. As an easy-to-use way for the 3D object creation, assembly-based modeling techniques become popular since the pioneer work of Funkhouser et al. [16]. Recent works focus on high-level tasks such as ensuring compatibility between replaceable parts [17] or determining the most suitable parts for composition [2, 3, 4, 5, 18]. In [4], a probabilistic model of component-based shape structure was trained on a set of examples. This learned model was used to synthesize new shapes by recombining various components from input shapes. In [5], the *fit and diverse* technique was developed to evolve a whole population of 3D models generation by generation. These methods focus more on generating shape variations sharing the same class and with the same *coarse-level structure* as input shapes. Different from them, our goal is to produce monsters by expanding the structures of regular creatures from different classes. In [19], parts were exchanged among shapes across different family classes by matching substructures with the same functionality. This technique was designed for man-made objects and emphasized on shuffling parts among different shape classes, while our method focuses more on synthesizing monsters from regular creatures by generating new structures.

Sketch-based modeling. There are a large volume of research on sketch-based modeling (see [20] for a comprehensive survey). In most sketch-based modeling systems, user-drawn 2D strokes are used for recovering the shapes of 3D models. The systems in [21, 22, 23, 24] create 3D shapes from user-input sketches via inflation. SmoothSketch [25] infers 3D shapes from complex sketches, which may have cups and T-junctions. Other systems [26, 27] take 2D strokes as handles to control the geometry of 3D models. The ILoveSketch system [28] allows professional product designers to create complicated concept 3D curves.

The models produced by these sketch-based modeling systems are generally with limited geometry details and relatively simple structures. Besides, the modeling process is completely controlled by users. Different from these approaches, our

[1]Photos are freely available on: http://might-and-magic.ubi.com/heroes-6/en-GB/game/creatures/index.aspx.

[2]The screenshot comes from film: *Harry Potter and the Order of the Phoenix.*

(a) *Fate Spinner* (b) *Centaur* (c) *Lamassu*

Figure 1: Three monster characters with abnormal structures from popular computer games or films.

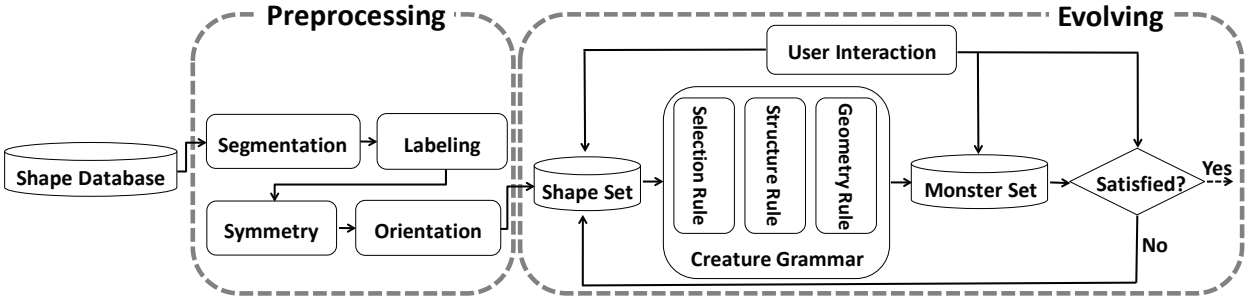


Figure 2: Pipeline of our inspiration-oriented 3D monster modeling framework. In the preprocessing stage, we classify regular objects into different categories, co-orient, semantically segment, and label objects. The symmetry and proximity relations between the parts per shape are pre-analyzed and stored. The evolving stage then iteratively generates sets of monsters starting from several regular objects chosen by the user.

method, which falls into the category of assembly-based modeling, automatically generates a diverse set of monster shapes with geometry details and various structures for users.

3. System overview

Fig. 2 illustrates the pipeline of our system. The input of our system is a set of pre-segmented and labeled regular creature shapes from different categories (we classified shapes into several categories such as bird-like creatures, humanoids, quadrupeds, etc.). They are normalized and co-oriented. The symmetry and proximity relations between the parts per shape are pre-analyzed and stored. The output is sets of monster models.

For each shape set, our system employs the creature grammar to generate a group of shapes from the current shape set with a subset selected to be presented to the user. User marks shapes according to their preference. The unaccepted shapes are moved to the background set. This process is repeated until the desired number of left shapes have been selected. The next shape set is produced by selecting a subset from the left shape set, current shape set and background shape set. The background shape set is produced by selecting a subset from the next shape set and the current background shape set. Fig. 3 shows the input shapes and some of the generated monster shapes. Notice that in order to preserve symmetry constraints, we apply the same rules to parts in the same symmetry group.

4. Monster generation with procedural evolution

4.1. Terminologies

Shape. The shape here is different from the one presented in [8, 29], which refers to some basic primitives (e.g., cube, sphere, cylinder, etc.) composing the final object (buildings for example). Here, shape refers to the geometry instance, the monster model.

Abstract shape is the 3D layout of a shape, a descriptive string. Each abstract shape can be considered as a configuration of meta-parts, which is defined below.

Meta-part is a logical entity of semantic significance [5, 30], e.g., a leg or an arm. A meta-part can be represented as:

$$\mathbf{M}\,(m_0, m_1, m_2, m_3, m_4)$$

where m_0 is the type of this meta-part, m_1 is its *id* in the abstract shape, m_2 is the *id* of the geometry instance corresponding to this meta-part, m_3 is its position, and m_4 is the torso super-part (which will be defined below) it attaches to. A different configuration of these attributes indicates a different meta-part. The consubstantial meta-parts form a meta-part family. We arrange meta-parts in super-part.

Super-part is a set of meta-parts, which are consubstantial and positioned in the same region. For example, there is a two-headed humanoid. Each head is a head meta-part. The two heads make a head super-part. A super-part can be represented as:

$$\mathbf{S}\,(s_0, s_1, s_2)$$

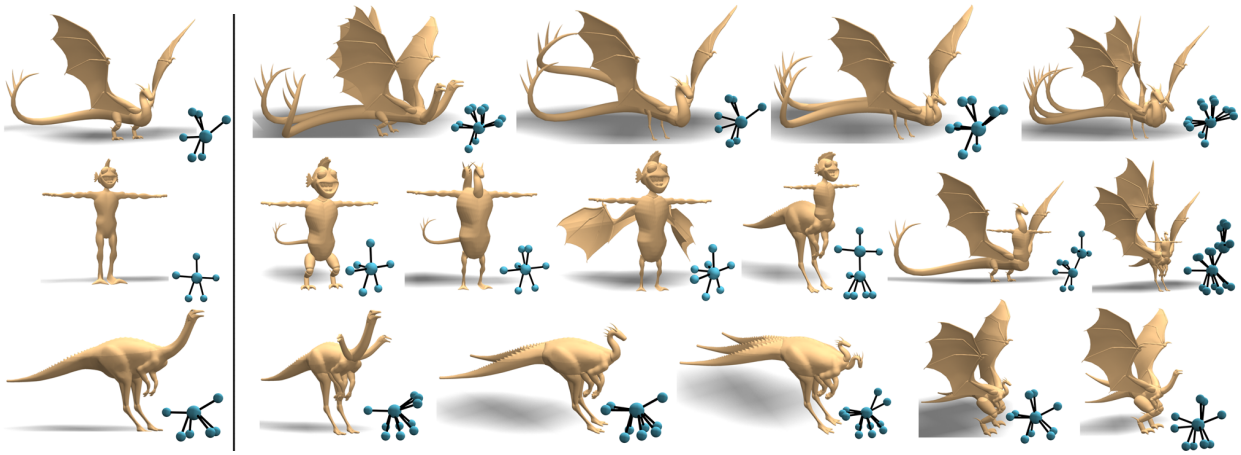
where s_0 is the type of this super-part, s_1 is the set of meta-parts belonging to this super-part, and s_2 is the arrangement mode of this super-part. A different configuration of these attributes indicates a different super-part. The consubstantial super-parts form a super-part family.

Allelic super-part represents the replaceability among super-parts. We define the allelic super-part according to functionality and position. Super-parts from the same family are allelic super-parts. *head* super-parts from bird-like creatures, humanoids and quadrupeds are allelic super-parts. Since wings locate in the front region of birds' torso and arms locate in the upper part of humanoids' torso, *wing* and *arm* super-part family are allelic super-parts. *lowerlimb* super-part from bird-like creatures, *hindleg* super-part from quadrupeds and *leg* super-part from humanoids are allelic super-parts.

Shape structure. The definition of shape structure is exactly the same as the one introduced in [30]. Shape structure consists of parts and relations. A part, with multiple parameters, is a *logical entity of semantic significance*, which defines the geometry of the part. A collection of parts capture the appearance of the whole shape geometry. Relations define how the parameters of parts are correlated.

Regular structure is the shape structure of common creatures (bird-like creatures, humanoids, and quadrupeds).

Abnormal structure is the shape structure, which is a composition or modification of regular structures.



(a) Input shapes (b) Generated shapes

Figure 3: Given several regular shapes from different categories (with regular structure), our system can iteratively generate sets of monsters with abnormal structure as revealed by the inset on the right bottom side of each model.

4.2. Production process

Different from previous grammars for shape modeling [8, 29], our creature grammar works with a set of shapes. During the evolving process, the grammar tries to generate new shapes by combining the configurations of every two shapes selected from the current shape set with the process: (1) select two shapes (*base shape* and *assist shape*) from the shape set by selection rules; (2) apply structure rules to *base abstract shape* and get the 3D layout of the *generated shape*; (3) apply geometry rules to the *generated abstract shape* to get its geometry instance. The process continues with step (1) until enough monster shapes are generated.

4.3. Production rules

4.3.1. Selection rules

We use the selection rule to either select a pair of shapes from the shape set or select a structure rule to apply. Our system supports three different selection strategies:

Rank selection selects the best shape (most appropriate rules) from the shape set (rule set) according to a certain metric of the shape (rule).

Uniform selection randomly selects a shape (rule) from the set.

Roulette wheel selection searches the members of the shape (rule) set using a weighted roulette wheel. Likelihood of selection is proportionable to the metric of the shape (rule).

Both strategies 1 and 3 require a metric defined on the shape (rule). For shape, we use the structure complexity as the metric, which is defined as the weighted sum of the edge number and the variation of the node valency on the structure graph of the shape (the insets of Fig. 3):

$$C_s = \omega n_e + \sum_i (v_i - \bar{v})^2$$

where n_e is the number of edges, v_i is the valency of node i, and $\bar{v}$ is the average valency of nodes from the structure graph of the current shape. In our experiments, we set $\omega = 0.95$.

For rules, we assign different constants according to their contributions to the structure and the structure complexity of the current shape pair. To be specific, for the increase rules, the metric is defined as:

$$C_r = \begin{cases} w_1 & C_s^1 + C_s^2 \le \delta \\ w_2 & C_s^1 + C_s^2 > \delta \end{cases}$$

where $w_1 = 0.5$, $w_2 = 0.1$, and $\delta = 63.33$. C_s^1 and C_s^2 are the structure complexity parameters of the base shape and the assist shape, respectively. For the combine and insert rules, we set $w_1 = 0.4$, $w_2 = 0.1$, and $\delta = 72.3$. For the replace rules, the metric is defined as:

$$C_r = w$$

where $w = 0.35$.

4.3.2. Structure rules

Structure rules are used to generate new shape configurations through exchanging or expanding configurations of two selected shapes. These rules are derived from the common operations adopted by 3D artists when designing monsters.

Notation: Structure rules have the following form:

$$\begin{aligned} &PRIORITY\ number \\ &id : predecessor : cond \\ &\quad \rightsquigarrow successor : prob \\ &\quad \rightsquigarrow predecessor : (1 - prob) \end{aligned}$$

where *id* is a unique identifier. *predecessor* is one or a list of symbol (s), representing the super-part (s) which will be replaced. The first branch replaces *predecessor* with *successor* with a probability of *prob*. This branch is called *function branch*. The second branch rewrites *predecessor* with its exact copy with a probability of $1 - prob$. This branch is called *identity branch*. *cond* is a Boolean expression, which must be true in order to apply the production. *number* is the priority of the rule defined by the 'PRIORITY' keyword.

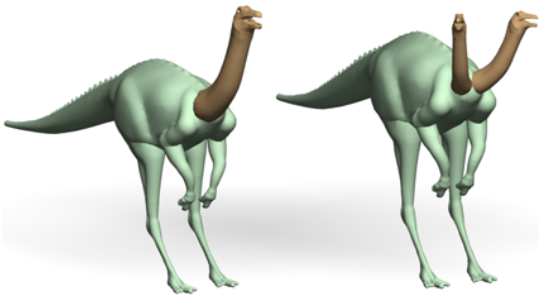


Figure 4: An example of the increase rule. The *function branch* adds a copy of the head meta-part in the head super-part and set arrangement mode for the super-part, resulting in a two-headed quadruped.

Increase rules have the function of adding a copy of meta-part in the super-part and assigning the arrangement mode for the new super-part. All the rules, belonging to this family, have similar ability with the only difference that they can be applied to different kinds of super-parts. This type of rules are context sensitive rules. An example of the increase rules is as follows:

$$
\begin{aligned}
0 :& \mathbf{S_H}\left(head_q, \left\{\left(head_q, 0, h_0, def, \mathbf{S_T}\right)\right\}, def\right) : cond \\
& \rightsquigarrow \mathbf{S_H}'\left(head_q, \right. \\
& \quad \left\{\left(head_q, 0, h_0, midl, \mathbf{S_T}\right); \left(head_q, 1, h_0, midr, \mathbf{S_T}\right)\right\}, \\
& \quad \left. midl\_midr\right) : p \\
& \rightsquigarrow \mathbf{S_H}\left(head_q, \left\{\left(head_q, 0, h_0, def, \mathbf{S_T}\right)\right\}, def\right) : 1 - p
\end{aligned}
$$

where *cond* is as follows:

$$\textbf{shape}.superpart.num <= N$$

and p is defined as follows:

$$p = \begin{cases} \alpha & g \leq 3 \\ \alpha e^{(\beta g)} & g > 3 \end{cases}$$

where g is the number of generation. The *function branch* replaces a one-headed super-part $\mathbf{S_H}$ by a two-headed super-part $\mathbf{S_H}'$ with a probability of p. The second head is the same as the first one. Its arrangement mode is *midl_midr*, which means that two heads locate in the upper region of the torso and are arranged in the left side (*midl*) and right side (*midr*), respectively (see Fig. 4). $head_q$ means that the type is head from quadrupeds. def indicates that the head locates in the default head position. $\mathbf{S_T}$ is the super-part, in which the torso meta-parts are included. h_0 is the *id* of the head geometry instance. *cond* means that the number of super-parts in the abstract shape should be less than or equal to N. N is set to 6, α is set to 0.4, and β is set to -0.1.

Insert rules have the function of adding a new super-part to a torso super-part. These rules can work on the torso super-part only. This type of rules are context sensitive rules. An example of the insert rules is as follows:

$$
\begin{aligned}
0 :& \mathbf{S_T}\left(torso_q, \ldots\right) : cond \\
& \rightsquigarrow \mathbf{S_T}\left(torso_q, \ldots\right)\mathbf{S_W}\left(wing, \ldots\right) : p \\
& \rightsquigarrow \mathbf{S_T}\left(torso_q, \ldots\right) : 1 - p
\end{aligned}
$$

Figure 5: An example of the insert rule. The *functionbranch* adds a wing super-part for a quadruped.

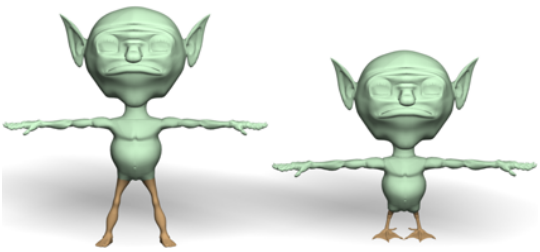


Figure 6: An example of the replace rule. The *function branch* replaces the leg meta-part of a humanoid with a lowerlimb meta-part from a bird-like creature.

where *cond* is as follows:

$$
\begin{aligned}
& no\ wing\ superpart\ attached\ to\ \mathbf{S_T} \\
& \&\& \textbf{Shape}.superpart.num > N
\end{aligned}
$$

and

$$p = e^{(\beta g)}\theta^{(t+1)}$$

where g is the number of generation, $t = \textbf{Shape}.metapart.num - N$. The *function branch* adds $\mathbf{S_W}$ to $\mathbf{S_T}$ with a probability of p (see Fig. 5). $torso_q$ indicates that the type of $\mathbf{S_T}$ is quadrupeds' torso. *wing* indicates that the type of $\mathbf{S_W}$ is wing from birds creatures. *cond* means that there should be no *wing* type super-part attached to $\mathbf{S_T}$ and the number of super-part in this abstract shape should be greater than N. N is set to 7, β is set to -0.05, and θ is set to 0.5.

Replace rules have the function of exchanging meta-part between allelic super-parts. All the rules, belonging to this family, have similar ability with the only difference that they can be applied to different kinds of super-parts. An example of replace rules is as follows:

$$
\begin{aligned}
0 :& \mathbf{S_L}\left(leg, \ldots\right) \\
& \rightsquigarrow \mathbf{S_M}\left(lowerlimb, \ldots\right) : p \\
& \rightsquigarrow \mathbf{S_L}\left(leg, \ldots\right) : 1 - p
\end{aligned}
$$

The *function branch* replaces $\mathbf{S_L}$ with $\mathbf{S_M}$ with a probability of p (see Fig. 6). *leg* indicates that the type of $\mathbf{S_L}$ is the humanoids' leg. *lowerlimb* indicates that the type of $\mathbf{S_M}$ is the birds' leg. p is set to 0.5.

Combine rules have the function of combining the upper or lower part of the current shape with a lower or upper part from another shape, resulting in a Centaur-like creature. An example

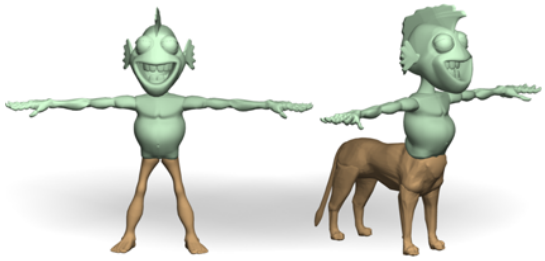


Figure 7: An example of the combine rule. The *function branch* combines the lower part of a humanoid with the lower part of a quadruped, resulting in a Centaur-like monster.

of the combine rule is as follows:

$$
\begin{aligned}
0 :& \mathbf{S_T}\left(torso_h, ...\right)\mathbf{S_L}\left(leg, ...\right) \\
&\rightsquigarrow \mathbf{S_T}\left(torso_h, ...\right)\mathbf{S_T}'\left(torso_q, ...\right)\mathbf{S_F}\left(foreleg, ...\right) \\
&\quad \mathbf{S_D}\left(hindleg, ...\right)\mathbf{S_I}\left(tail_q, ...\right) : p \\
&\rightsquigarrow \mathbf{S_T}\left(torso_h, ...\right)\mathbf{S_L}\left(leg, ...\right) : 1 - p
\end{aligned}
$$

where p is defined as follows:

$$p = \alpha e^{(\beta g)}$$

where g is the number of generation. The *function branch* replaces the lower part of the current shape $\mathbf{S_L}$ with the super-parts $\mathbf{S_T}'\mathbf{S_F}\mathbf{S_D}\mathbf{S_I}$ from another shape, resulting in a Centaur-like monster, with a probability of p (see Fig. 7). $torso_h$ indicates that the type of $\mathbf{S_T}$ is humanoids' torso. *hindleg* indicates that the type of $\mathbf{S_D}$ is quadrupeds' hind leg. $foreleg$ indicates that the type of $\mathbf{S_F}$ is quadrupeds' foreleg. $tail_q$ indicates that the type of $\mathbf{S_I}$ is quadrupeds' tail. α is set to 0.2, and β is set to -0.01.

4.3.3. Geometry rules

Geometry rules are used to adjust parts' scale and place them in an appropriate position and orientation. Similar to [4], we employ "slots" to specify where this part can be attached to other parts. Slots are extracted automatically using the method proposed by [4]. But for some slots, for example, the wing slot in the torso meta-part of humanoids, cannot be extracted using this technique. In this case, we label it manually. The "boundary" of a slot is the convex hull of all the vertices belonging to the slot.

Translate rule. $\mathbf{T}\left(\mathbf{S_T}\right)$ translates the current meta-part to a torso meta-part included in $\mathbf{S_T}$ by overlapping the corresponding slot center of the two meta-part geometry instances. The center of the slot is the average of the boundary vertices.

Scale rules. $\mathbf{S}\left(s\right)$ scales the current meta-part geometry instance by a scale factor s. $\mathbf{S}\left(\mathbf{S_T}\right)$ scales the current meta-part geometry instance to make it compatible with the torso meta-part geometry instance included in $\mathbf{S_T}$ in terms of slot length. To be specific, we first project the boundary vertices of the slot to a 2D plane [31]. Then we calculate the convex hull of the projected vertices. The slot length is the perimeter of this polygon.

Figure 8: The Centaur-like shape generated by geometry rules. The shape in the middle is the generated model. The shape in the top left corner is the base shape. The shape in the top right corner is the assist shape.

Rotate rules. $\mathbf{R}\left(\mathbf{S_T}\right)$ rotates the current meta-part geometry instance to make it compatible with the torso meta-part geometry instance included in $\mathbf{S_T}$ in terms of direction. $\mathbf{R}x\left(angle, \mathbf{S_T}\right)$, $\mathbf{R}y\left(angle, \mathbf{S_T}\right)$ and $\mathbf{R}z\left(angle, \mathbf{S_T}\right)$ rotate the current meta-part geometry instance in the local coordinate system of $\mathbf{S_T}$ around x, y, and z axis by *angle*, respectively. We use these three commands to fine-tune the direction between the current meta-part and the torso meta-part.

Instance rule. $\mathbf{I}\left(k_i\right)$ replaces the meta-part with its geometry instance k_i. '[' and ']' are used to push and pop geometry rules on a stack. Note that, we do not delete any meta-parts during the production process of geometry rules, but mark them as *terminated*, after they are interpreted into geometry instances. The process starts with torso meta-parts. Other parts are interpreted and attached to its torso meta-part in sequence. In order to simplify the transform process, all the meta-part geometry instances we manipulate come from its original shapes. The example below illustrates the composition of a Centaur-like monster (see Figure 8).

$$
\begin{aligned}
&A \rightsquigarrow \\
&\mathbf{I}\left(t_0\right)\left[\mathbf{T}\left(\mathbf{S_T}\right)\mathbf{S}\left(\mathbf{S_T}\right)\mathbf{I}\left(t_1\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}\right)\mathbf{S}\left(0.6\right)\mathbf{S}\left(\mathbf{S_T}\right)\mathbf{R}\left(\mathbf{S_T}\right)\mathbf{I}\left(h_0\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}\right)\mathbf{S}\left(0.6\right)\mathbf{S}\left(\mathbf{S_T}\right)\mathbf{R}y\left(180, \mathbf{S_T}\right)\mathbf{R}\left(\mathbf{S_T}\right)\mathbf{I}\left(h_0\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}\right)\mathbf{S}\left(\mathbf{S_T}\right)\mathbf{I}\left(a_0\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}\right)\mathbf{S}\left(\mathbf{S_T}\right)\mathbf{R}\left(\mathbf{S_T}\right)\mathbf{I}\left(w_1\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}'\right)\mathbf{S}\left(\mathbf{S_T}'\right)\mathbf{I}\left(f_1\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}'\right)\mathbf{S}\left(\mathbf{S_T}'\right)\mathbf{I}\left(w_1\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}'\right)\mathbf{S}\left(\mathbf{S_T}'\right)\mathbf{I}\left(hl_1\right)\right] \\
&\left[\mathbf{T}\left(\mathbf{S_T}'\right)\mathbf{S}\left(\mathbf{S_T}'\right)\mathbf{I}\left(ta_1\right)\right]
\end{aligned}
$$

where t_0, t_1, h_0, a_0, w_1, f_1, hl_1, and ta_1 are the geometry instances corresponding to torso super-part, head super-part, arm super-part, wing super-part, and tail super-part, respectively.

4.4. Monster generation

We use the creature grammar in an iterative manner to generate monsters. Beginning with an initial small set of regular

shapes selected by the user from different categories, the system firstly picks up two shapes with the shape selection rules, then structure rules are applied for several times to expand the structure of the base shape into a *valid* monster shape with abnormal structures. Each structure rule is accompanied with a geometry rule to adjust the meta-parts' scale and put them in an appropriate position and orientation. The selection-structure-geometry procedure is continuously repeated to generate a monster set from the initial shape set. We can finally evolve the initial set of regular shapes into generations of monsters by iteratively applying the whole process from the previous set of shapes (see Algorithm 1). A key criteria to evaluate the validity of a monster is whether it can stand upright. We take a similar strategy as [5] to check whether the projection of the center of mass is falling into the convex hull of the supporting points.

Algorithm 1: Monster Generation with Creature Grammar

Input: Regular shape set $\mathcal{R}$; Monster set $\mathcal{G}_0$; Background set $\mathcal{B}$; Desired number of shapes in each set N_s; Maximum number of shapes in each group N_g; Base shape b; Assist shape a; Generated monster shape m;

$\mathcal{G}_0 \longleftarrow \mathcal{R}$;
$\mathcal{B} \longleftarrow \mathcal{G}_0$;
$i = 0$;
while *the user is not satisfied* **do**
  $\mathcal{G}_{i+1} = \emptyset$;
  while $|\mathcal{G}_{i+1}| < N_s$ **do**
    for $i \leftarrow 1$ **to** N_g **do**
      $b \rightsquigarrow ShapeSelection(\mathcal{G}_i)$;
      $a \rightsquigarrow ShapeSelection(\mathcal{G}_i)$;
      $m \rightsquigarrow StructureRules(b, a)$;
      $m \rightsquigarrow GeometryRules(b, a)$;
      $\mathcal{D} = \mathcal{D} \cup \{m\}$;
    $\mathcal{D} \longleftarrow$ `SelectStandUpright` $(\mathcal{D})$;
    for $i \leftarrow 1$ **to** $|\mathcal{D}|$ **do**
      if $\mathcal{D}[i]$ *is disliked by the user* **then**
        $\mathcal{B} = \mathcal{B} \cup \{\mathcal{D}[i]\}$;
        delete $\mathcal{D}[i]$;
    $\mathcal{G}_{i+1} = \mathcal{G}_{i+1} \cup \mathcal{D}$;
  $\mathcal{G}_{i+1} \longleftarrow$ `DiversityControl` $(\mathcal{G}_{i+1}, \mathcal{G}_i, \mathcal{B})$;
  $\mathcal{B} \longleftarrow$ `DiversityControl` $(\mathcal{B}, \mathcal{G}_{i+1})$;
  $\mathcal{G}_i \longleftarrow \mathcal{G}_{i+1}$;
  $i = i + 1$;

Diversity control. It is very important to allow users explore various possibilities in a creative support system. For monster modeling, we would expect the system to generate a set of diverse monsters with various unexpected structures in each iteration. To this end, we introduce a shape structure-based diversity control mechanism. For a shape set, we extract all the meta-parts of different types and different positions. Each meta-part is an element of the shape structure vector. We first cluster shapes according to its shape structure vector. Inside each structure cluster, we use the diversity control mechanism proposed by [5] to sample shapes. The number of sampled shapes is in proportion to the number of shapes in the structure cluster. This shape structure-based diversity control mechanism is used to automatically select shapes from each group and keep a diverse set of shapes for the current set and background set.

5. Experimental results

We have implemented a prototype system with C++. Experiments have been performed on a desktop computer equipped with Intel® Core i7 clocked at 3.77 GHz, 8 GB of RAM and NVIDIA® Geforce GTX 660 GPU. Fig. 9 shows two monster design examples produced by our system. Fig. 10 shows the design result and the background sets. The preprocessing of each shape consists of segmentation, labeling, normalization, co-orientation and symmetry group detection. It costs 20 min on average. The creation of an offspring includes rule selection, structure evolution and shape synthesis. Generally, it takes 0.5 s. We use the openMT® technique to parallel the creation of shapes.

As mentioned before, diversity of the monster set in each generation of the evolution process is a critical metric for the system. We use a hierarchical diversity metric to quantitatively evaluate the structure diversity of the monster set. Given a set of shapes **S**, the *HDM* is defined as

$$HDM(\mathbf{S}) = N + \sum_{i=0}^{N} D_i$$

where N is the number of structure cluster among **S**, D_i is the normalized shape diversity in structure cluster i. We define D_i as

$$D_i = \frac{SD_i}{\max\left\{\sum_{i=0}^{N} SD_i\right\}}$$

where SD_i is the standard derivation of LFD for shapes in structure cluster i. Fig. 11 plots the diversity of the evolution sets generated by our structure-based diversity control mechanism (Structure-based) and diversity control mechanism (LFD-based) proposed by F&D [5]. It is easy to find that our diversity control mechanism can generate more diverse shape sets. The reason is that our mechanism emphasizes on shape structure while the mechanism proposed by F&D has the limitation in capturing the structure difference among shapes in a set. We compare our system with F&D in structure diversity. As is shown in Fig. 12, the diversity of the generated shape sets is nearly 8 times that of F&D. The reason is that F&D generates shape variations mainly through exchanging parts between shapes, while our system not only shuffles parts but also *reasonably* expands structures of the input set by the structure rules.

The probabilistic model introduced in [4] could also be used to synthesize sets of creature shapes. This technique is mainly designed for shapes belonging to the same class (e.g., quadrupeds, battleships, etc.) and aims to synthesize shapes by recombining parts among different input shapes. The resulting shapes are all with the same *coarse-level structures* as input

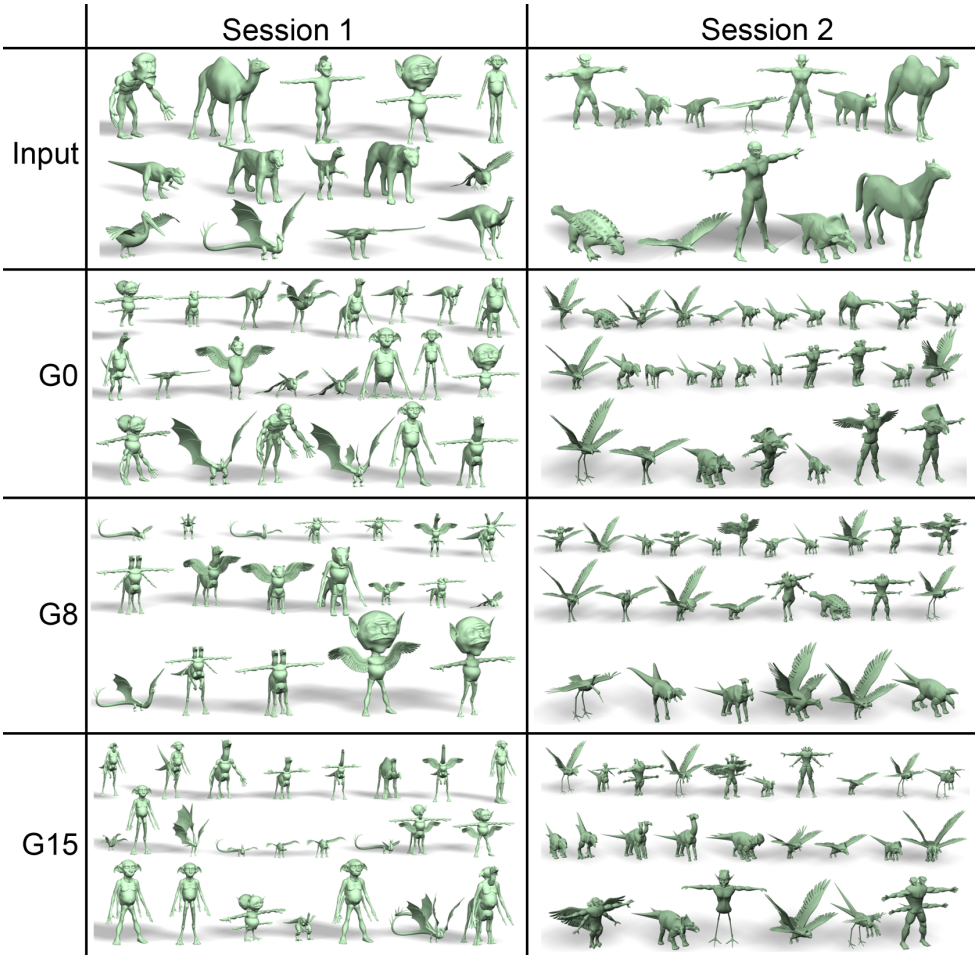


Figure 9: Some monster examples designed with our system. For each session, we show the input shapes and randomly selected shapes from three generations and background sets.

shapes. Starting with shapes of regular creatures, this technique can generate monster models as our system, if augmented by prior knowledge about monsters. However, our creature grammar is exactly a formal description of the prior knowledge about monsters.

5.1. Production rules

The increase rules contribute most to the shape diversity. Fig. 13 shows the different shape diversity plots with and without increase rules. Obviously, a lack of increase rules would lead to nearly a half decrease in shape diversity. Notice that other rules are not dispensable, since they can generate new structures and contribute to the shape diversity. For example, the combine rules have the ability to create Centaur-like monsters, the insert rules have the ability to add wings to humanoids and quadrupeds, and the replace rules can exchange parts. A subset of generated shapes are shown in Fig. 14. It is easy to find that a lack of increase rules will lead to shapes with only one meta-part super parts. On the contrary, our approach can produce a more diverse shape set by applying all the rules.

5.2. Expert review

We invited four 3D artists to evaluate our monster suggesting system. Three are with 4 years 3D modeling experience. The other one are with 7+ years 3D modeling experience. Firstly, the experts watched a 10 min tutorial about our system. Then, they were given 15 min to try our system to familiarize themselves with it. Next, each of them had 30 min to freely create monster models using our system. Lastly, each of the participants was asked to complete a questionnaire (see Fig. 15). Fig. 16 shows some shapes generated during expert review process. The feedback from the participants was quite positive. They all agreed that majority of the shapes generated by our system were monsters. Many shapes were really unexpected, especially in terms of the abnormal structures. They would definitely use a tool like this in the conceptual stage. One expert suggested that it would be better, if the system was augmented with the ability to analyze users' operations and evolve shapes according to their preference.

5.3. User study

To quantitatively evaluate the effectiveness of our technique in inspiring users during the modeling of monsters with abnor-

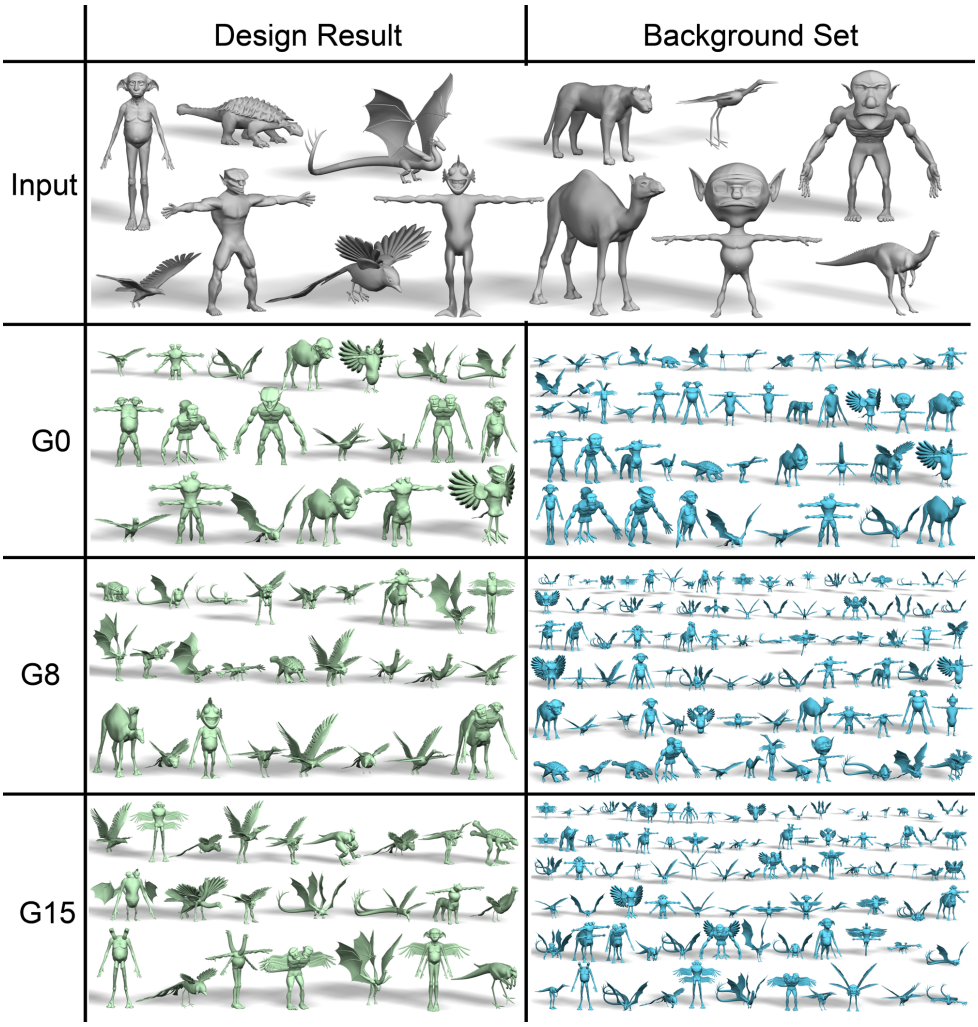


Figure 10: The monster example designed with our system. We show the input shapes (in gray color), design results randomly selected from three generations and background sets (in green color), and the background sets (in blue color).

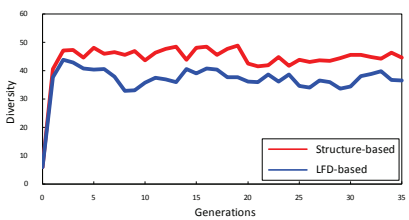


Figure 11: Plots of set diversity for our structure-based diversity control mechanism (red) and diversity control mechanism proposed by F&D (blue).

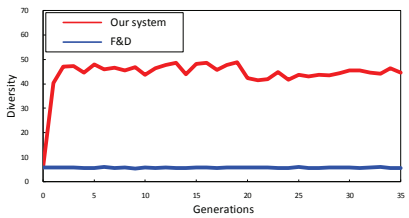


Figure 12: Plots of set diversity for ours (red) and F&D (blue).

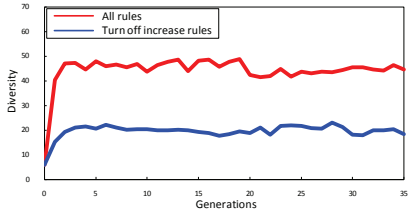


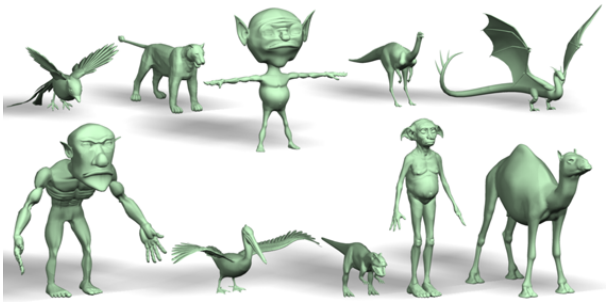
Figure 13: Plots of set diversity, when applying all the rules (red), turning off increase rules (blue).

mal structures, we compared our technique with F&D. We recruited twenty participants, 10 males and 10 females aged from 18 to 26. Four of them were art students. The other participants were all graduate students in the graphics lab in our university. We divided the participants into two groups, group *A* and group *B*, each including 5 males and 5 females. There were two art students in each group. The participants were asked to tackle the monster modeling task as follows:

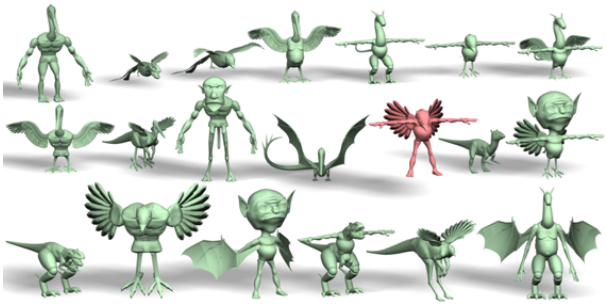
- You are a creature designer for the *Half-Life 3* computer

game in Valve Corporation®. In the game, Gordon Freeman will lead a Special Operations Group to enter a planet in Sirius galaxy surreptitiously. In the planet, there is completely different natural order compared with the earth, resulting in a fantastic world with fantastic creatures. Please use the two set evolution systems and mark the shapes which gave your inspiration.

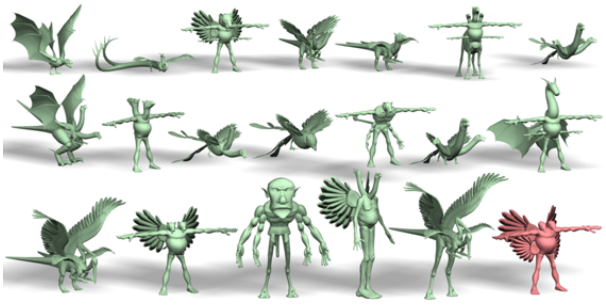
The experiment was designed as follows: the participant ran the two systems one after the other. In each generation, they were asked to rate the presented 18 shapes (9 shapes in each group) into 5 levels: 1 for banal, 5 for inspiring. Participants



(a) Input shapes



(b) Shapes generated by turning off increase rules



(c) Shapes generated by applying all rules

Figure 14: Shapes generated by applying all rules and turning off increase rules. (a) Input shapes. (b) Shapes generated by turning off increase rules. Notice that the pink shape is with one bird-like head and one pair of wing. (c) Shapes generated by applying all rules. Notice that the pink shape is with two bird-like heads and two pairs of wings. This shape is created by the application of the head increase rule and the wing increase rule.

from group *A* used our system first and F&D latter. Participants from group *B* used F&D first and ours latter. Before running each system, the participants watched a 10 min tutorial and then were given 15 min to familiarize themselves with the system.

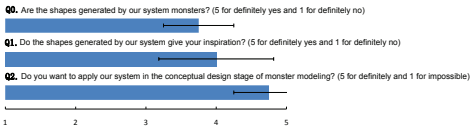


Figure 15: Questionnaire in the expert review. The black horizontal line on each bar is the standard deviation.

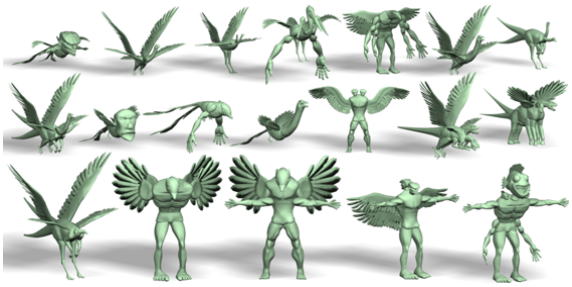


Figure 16: Shapes randomly selected from the shape set generated during the expert review.

We added up the number of shapes rated as 4 or 5 and called them *inspiring shapes*. The shapes rated as 1 or 2 were processed in the same way and called *banal shapes*. We averaged the percentage of inspiring shapes in each generation over all users (see Fig. 17). The percentage of banal shapes averaged over all participants was shown in Fig. 18. We plotted the sum grade of all shapes in each generation averaged over all participants (see Fig. 19). We presented three generations of shapes and their grades rated by a participant for the two systems (see Fig. 20). As shown in the figures above, the percentage of inspiring shapes generated by our system is almost 3 times that of F&D. The percentage of banal shapes generated by our system is nearly one third that of F&D. The total score of each generation is almost 2 times that of F&D. Statistics show that our system outperforms F&D in generating inspiring shapes, as well as presenting less banal shapes. The reason is that our system expands the structures of the input shape set by our creature grammar, while F&D could only generate shapes by shuffling and warping parts.

6. Conclusions

We have presented an inspiration-oriented system for creative modeling of 3D monsters. Given a set of pre-segmented and labeled shapes from several different categories, our system can iteratively generate sets of monsters allowing users to

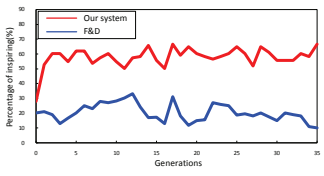


Figure 17: Plot of percentage of inspiring shapes, which are rated as 4 or 5.

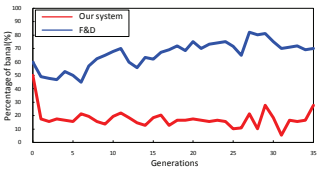


Figure 18: Plot of percentage of banal shapes, which are rated as 1 or 2.

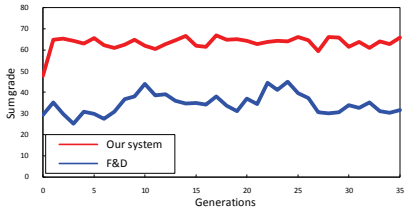
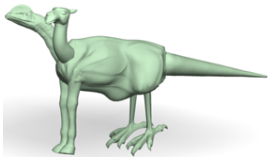


Figure 19: Plot of sum grade averaged over all participants in each generation.

explore various possibilities. The core of our system is the creature grammar, a tailored grammar, which we use to capture the way of evolving creatures with regular structures into monsters with abnormal structures, while preserving the semantics of creatures.



Limitations. Although the initial results are encouraging, there is still room to improve our current approach. First, our technique produces monster shapes mainly by generating new structures, which does not produce geometric variations or new non-existed parts. Second, our method does not take the compatibility of geometry style among assembled parts into account. Sometimes our system could generate failure shapes with a set of extremely incompatible parts (see the right figure). Finally, as the paper focuses on creative support of monster modeling, we synthesize shapes by simply positioning parts together. The boundary smoothness is not taken into account. Compositing parts into a manifold shape seamlessly [32, 33] will be explored in the future.

Future work. First, we would like to develop techniques, which can produce shape variations by semantically exaggerating the geometry or new parts by sketch-based methods. Second, we will enrich our creature grammar to make it generate more inspiring structures. Also of interest is to augment our creature grammar with the knowledge about geometric styles among shape parts. Third, it is interesting to investigate intuitive control methods of production process. Fourth, we will develop a formal definition of semantics of biological forms using the framework introduced in [34]. Finally, although our paper focuses on the modeling of monsters, we believe our grammar can be extended to the modeling of other kinds of objects, for example, trees, buildings and so on. We will investigate the possibility of the application of our grammar to other modeling domains in the future.

Acknowledgments. We would like to thank Xiaoqiang Zhu, Ming Wang, Linling Zhou, and the reviewers for their constructive comments. This work is supported in part by grants from the National Natural Science Foundation of China (61272298, 61328204, 61202142, and 61202333), the National Key Technology R&D Program Foundation of China (2013BAH44F00), and China Postdoctoral Science Foundation (2012M520392).

References

[1] B. Shneiderman, et.al, Creativity support tools: Report from a u.s. national science foundation sponsored workshop, International Journal of Human-Computer Interaction 20 (6) (2006) 61–77.

[2] S. Chaudhuri, E. Kalogerakis, L. Guibas, V. Koltun, Probabilistic reasoning for assembly-based 3d modeling, ACM Transactions on Graphics 30 (4) (2011) 35:1–35:10.

[3] S. Chaudhuri, V. Koltun, Data-driven suggestions for creativity support in 3d modeling, ACM Transactions on Graphics 29 (6) (2010) 183:1–183:10.

[4] E. Kalogerakis, S. Chaudhuri, D. Koller, V. Koltun, A probabilistic model for component-based shape synthesis, ACM Transactions on Graphics 31 (4) (2012) 55:1–55:11.

[5] K. Xu, H. Zhang, D. Cohen-Or, B. Chen, Fit and diverse: set evolution for inspiring 3d shape galleries, ACM Transactions on Graphics 31 (4) (2012) 57:1–57:10.

[6] A. Lindenmayer, Mathematical models for cellular interactions in development i. filaments with one-sided inputs, Journal of Theoretical Biology 18 (3) (1968) 280 – 299.

[7] P. Prusinkiewicz, A. Lindenmayer, The algorithmic beauty of plants, Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[8] P. Müller, P. Wonka, S. Haegler, A. Ulmer, L. Van Gool, Procedural modeling of buildings, ACM Transactions on Graphics 25 (3) (2006) 614–623.

[9] Y. I. H. Parish, P. Müller, Procedural modeling of cities, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, ACM, New York, NY, USA, 2001, pp. 301–308.

[10] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, V. Koltun, Metropolis procedural modeling, ACM Transactions on Graphics 30 (2) (2011) 11:1–11:14.

[11] G. Stiny, J. Gips, G. Stiny, J. Gips, Shape grammars and the generative specification of painting and sculpture, in: Segmentation of Buildings for 3DGeneralisation. In: Proceedings of the Workshop on generalisation and multiple representation , Leicester, 1971.

[12] P. Wonka, M. Wimmer, F. Sillion, W. Ribarsky, Instant architecture, ACM Transactions on Graphics 22 (3) (2003) 669–677.

[13] B. Hohmann, U. Krispel, S. Havemann, D. Fellner, Cityfit: High-quality urban reconstructions by fitting shape grammars to images and derived textured point clouds, in: Proceedings of the 3rd ISPRS Workshop, Citeseer, 2009.

[14] M. Lipp, P. Wonka, M. Wimmer, Interactive visual editing of grammars for procedural architecture, ACM Transactions on Graphics 27 (3) (2008) 102:1–102:10.

[15] E. Whiting, J. Ochsendorf, F. Durand, Procedural modeling of structurally-sound masonry buildings, ACM Transactions on Graphics 28 (5) (2009) 112:1–112:9.

[16] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by example, ACM Transactions on Graphics 23 (3) (2004) 652–663.

[17] V. Kreavoy, D. Julius, A. Sheffer, Model composition from interchangeable components, in: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, PG '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 129–138.

[18] S. Chaudhuri, E. Kalogerakis, S. Giguere, T. Funkhouser, Attribit: content creation with semantic attributes, in: Proceedings of the 26th annual ACM symposium on User interface software and technology, UIST '13, ACM, New York, NY, USA, 2013, pp. 193–202.

[19] Y. Zheng, D. Cohen-Or, N. J. Mitra, Smart variations: Functional substructures for part compatibility, Computer Graphics Forum (Eurographics) 32 (2pt2) (2013) 195–204.

[20] L. Olsen, F. F. Samavati, M. C. Sousa, J. A. Jorge, Sketch-based modeling: A survey, Computers & Graphics 33 (1) (2009) 85 – 103.

[21] T. Igarashi, S. Matsuoka, H. Tanaka, Teddy: A sketching interface for 3d freeform design, in: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999, pp. 409–416.

[22] Y.-J. Liu, C.-X. Ma, D.-L. Zhang, Easytoy: Plush toy design using editable sketching curves, IEEE Transactions on Visualization and Computer Graphics 31 (2) (2011) 49–57.

[23] L. Eggli, C. yao Hsu, B. D. Brderlin, G. Elber, Inferring 3d models from freehand sketches and constraints, Computer-Aided Design 29 (2) (1997) 101 – 112, solid Modelling.

[24] R. Schmidt, B. Wyvill, M. C. Sousa, J. A. Jorge, Shapeshop: Sketch-based solid modeling with blobtrees, in: ACM SIGGRAPH 2006

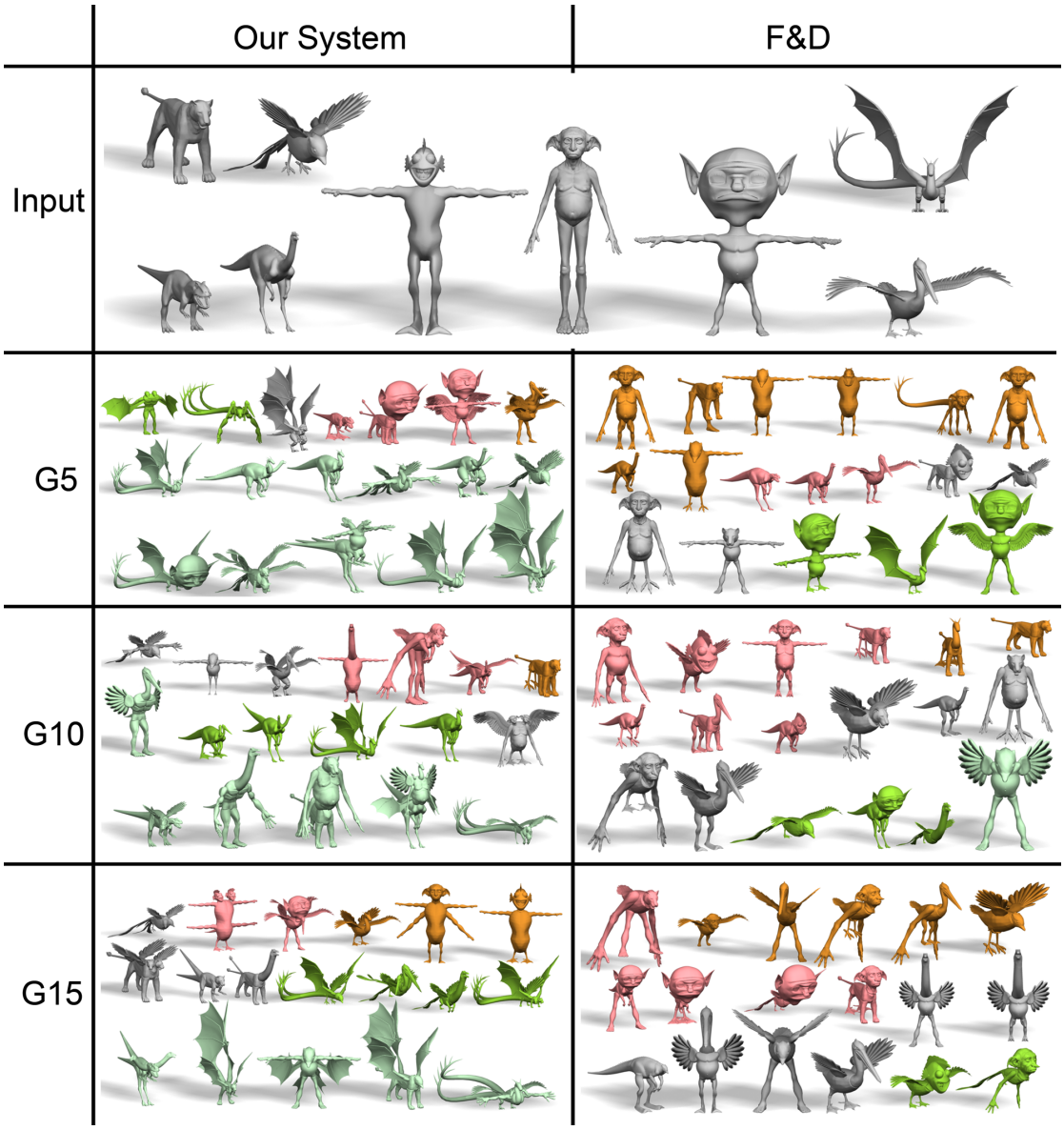


Figure 20: Three generations of shapes and their grades for the two systems rated by one of the participants. The shapes in green color are rated as 5. The shapes in yellow-green color are rated as 4. The shapes in white color are rated as 3. The shapes in pink color are rated as 2. The shapes in yellow color are rated as 1.

Courses, SIGGRAPH '06, ACM, New York, NY, USA, 2006.

[25] O. A. Karpenko, J. F. Hughes, Smoothsketch: 3d free-form shapes from complex sketches, ACM Transactions on Graphics 25 (3) (2006) 589–598.

[26] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Fibermesh: Designing freeform surfaces with 3d curves, ACM Transactions on Graphics 26 (3) (2007) 41.

[27] A. Nealen, O. Sorkine, M. Alexa, D. Cohen-Or, A sketch-based interface for detail-preserving mesh editing, ACM Transactions on Graphics 24 (3) (2005) 1142–1147.

[28] S.-H. Bae, R. Balakrishnan, K. Singh, Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models, in: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, UIST '08, ACM, New York, NY, USA, 2008, pp. 151–160.

[29] P. Karnick, S. Jeschke, D. Cline, A. Razdan, E. Wentz, P. Wonka, A shape grammar for developing glyph-based visualizations, Computer Graphics Forum 28 (8) (2009) 2176–2188.

[30] N. Mitra, M. Wand, H. R. Zhang, D. Cohen-Or, V. Kim, Q.-X. Huang, Structure-aware shape processing, in: SIGGRAPH Asia 2013 Courses, SA '13, ACM, New York, NY, USA, 2013, pp. 1:1–1:20.

[31] N. Qiu, R. Fan, L. You, X. Jin, An efficient and collision-free hole-filling algorithm for orthodontics, The Visual Computer 29 (6-8) (2013) 577–586.

[32] J. Lin, X. Jin, C. C. L. Wang, K.-C. Hui, Mesh composition on models with arbitrary boundary topology, IEEE Transactions on Visualization and Computer Graphics 14 (3) (2008) 653–665.

[33] R. Schmidt, K. Singh, Meshmixer: An interface for rapid mesh composition, in: ACM SIGGRAPH 2010 Talks, SIGGRAPH '10, ACM, New York, NY, USA, 2010, pp. 6:1–6:1.

[34] Y.-J. Liu, K.-L. Lai, G. Dai, M.-F. Yuen, A semantic feature model in concurrent engineering, IEEE Transactions on Automation Science and Engineering 7 (3) (2010) 659–665.