Design Motifs: A Grammar Based Approach

Joseph Mazeika, Jim Whitehead
University of California, Santa Cruz
{jmazeika, ejw}@soe.ucsc.edu

ABSTRACT
The notion of generating artifacts using a design motif has a long history in the tradition of generative systems, however no formal definition of design motif currently exists. We present a formal definition that unifies these previous approaches, while also proposing several novel systems that incorporate this definition, in a way that allows generators to switch the design motifs that they generate with.

Categories and Subject Descriptors
I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

Keywords
Design Motif, Procedural Content Generation, Grammars

1. INTRODUCTION
A *design motif* is defined as a set of visible features that fuzzily defines group membership for a set of objects. These can range in specificity from the very loose design motif that defines the set of all automobiles, and the highly specific design motif for Ford Mustangs. Design motifs are found in all kinds of objects, from the automotive examples, to art movements and laptop designs. The features of a design motif can be as simple as a color scheme, or as complex as changing the size and shape of a given feature in a design, such as making a fireplace in a building twice as tall, or even changing how the parts of a design are constructed. While individual motifs are simple to describe, no formal definition of design motif currently exists, much less one that could be embodied in a computational system. In order to correct this, we built a generator that incorporates this definition in a simple, but expressive, domain: specifically, the domain of Lego models.

Legos are an appealing domain for this research, because the company itself provides strong examples of design motifs in its products. Legos are typically marketed in kits - collections of pieces that can be assembled to form a particular model: such as a car, a building, or a spaceship. But, more importantly, Lego has various groups of models that the company presents, called themes. These themes range from generic genres to focused sub-themes to licensed themes that are based on other properties popular in the target demographic. For instance, we have instances of the Lego Space theme, and contained within it, themes such as Lego Mission to Mars or Lego Space Police. While all of these all pull from the same global set of bricks, Lego uses different design motifs to distinguish the various sub-themes from each other.

2. PRIOR WORK
Since the advent of the shape grammar [12], many attempts at generating designs that visually resemble hand-authored specifications have been made. These approaches range from work done by Pugliese and McCormack in generating artifacts with a particular brand identity [7, 9], to systems which generate buildings that invoke the principles of particular architects [1, 5, 13]. The individual systems are only able to generate one particular style of artifact - for instance, Koning and Eizenberg's prairie house grammar is only able to generate this one class of house, using the particular constraints given to these houses. However, they all aim to do a similar task, the generation of artifacts that share common and distinct visual elements, or design motif. Similar work has been done in the context of procedural filters [14], with where the authors construct a system for altering the appearance of a 3D scene by mapping it through a filter. However, this work only addresses the surface level appearance, and does not allow for changes in the form of the object.

In addition, Legos have been used previously in generated systems - mostly focused on physical realization of structures and generating physically sound objects [3, 4, 15]. Legos also have some useful properties that we can exploit in this domain: they are a modular system that incorporates both simple rectangular bricks and elaborate decorative pieces. This allows designs that feature unusual shapes and curved portions with very little additional effort. Additionally, while most of the work on shape grammars has been focused on two-dimensional space, there is precedent for using them in 3D [2, 8].

Figure 1: Two sample kits from the Space Police theme (Source: [10], [11])

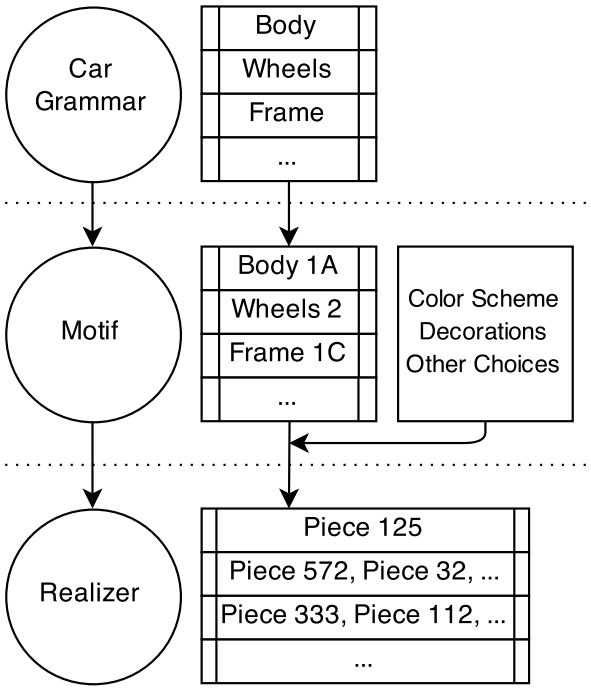

Figure 2: System Diagram

3. DESIGN MOTIF

Figure 1 shows two kits from the Space Police theme. The main vehicles in both of the images show common design principles. First and foremost, we have the superficial similarities. The designs of both spaceships feature the same color scheme: white bodies, with blue glass compartments and black accents. The weaponry on the ships is represented by translucent, light green cones, and blue and red translucent pieces that invoke the lights of a modern police vehicle.

More subtly, both space ships feature bilateral symmetry, with very unusual but streamlined appearance, and neither design directly resembles anything that exists in the world today, but that feel reasonable in a futuristic setting. They both feature differing but similar chunks that represent the thrusters in the ship, and other kits from the same theme also feature these same features.

This particular set of features is by no means either necessary or sufficient to encompass all design motifs, however it provides us we a good sample of the feature space for design motifs.

4. THE SYSTEM

Currently, a simple grammar that generates Lego car models has been constructed. This grammar incorporates a simple design motif consisting of a color scheme and a few simple decorative elements. Each of the decorative elements has the binary option of being present or not present, while the color scheme is a simple ordered list of the colors of bricks to be included in the design. The first color is the primary color in the model, the second is the secondary, and so on.

As shown in Figure 2, the grammar is a mutli-stage process, comprised of three main parts. The first is the default 'car' grammar which contains the information needed for the basic high-level structure - at this stage a car is comprised of wheels, a base, a compartment for the minifigure, a front and back wall, and a roof. After this initial expansion, each symbol is passed through the design motif, which determines how the pieces will be realized as Legos. This process output another set of symbols - called 'piece groups' - that are converted into the Lego bricks by a third layer.

In order to insure this level of modularity, the generator uses the design motif as a bridge between the other two layers. The design motif, in this case, is simply a set of functions that map the high-level symbols onto the low-level piece groups. The design motif handles all non-determinism in the generation process - any and all choices that happen in the final layer are completely deterministic, and serve only to prevent, for instance, issues of overlapping pieces and choices of filling the space appropriately. However, the functions contained within the design motif don't necessarily need to map to any of the symbols in the grammar; as long as all of the symbols in the top level grammar are accounted for, and as long as they are mapped onto valid symbols in

the Interpreter, design motifs and grammars can be mixed freely.

5. GENERATION EXAMPLE

We start with our baseline grammar:

[Base, Front Wheels, Back Wheels, Compartment, Front Wall, Back Wall, Roof]

These symbols are fed, one by one, through the design motif, so we start with the first symbol, *Base*. This gets expanded into the motif symbol, still called *Base* for simplicity, and also has the color scheme information attached. In this case, the colors we use are Blue, Dark Grey and Black. *Base* also sets up two pieces of information for the interpreter to use - the locations of the two *Wheel* pieces that are to be attached to it.

We then move to the next symbol, *Front Wheels*. Our design motif takes this and returns two symbols, one called *WheelType1* and one called *WheelHousing1*. There are multiple types of wheels and housings, and so we distinguish them at this level. We include the color scheme information, as well as metadata required by the *WheelHousing* symbol: namely, that this the *Front Wheel* set.

At this point, our set of symbols output from the Design motif are: *Base, WheelType1(Front), WheelHousing1(Front)*. We continue the process, expanding out each symbol as we go, until we've addressed every symbol in the original grammar. After this process, we have the following string of symbols:

[Base, WheelType1(Front), WheelHousing1(Front), WheelType1(Back), WheelHousing1(Back), Compartment, WindShieldSlope, Wall, Roof]

Notice that the WheelType1 and WheelHousing1 symbols are repeated, but with different metadata. From here, the symbols are entered into the final stage. Each symbol is sequentially read, and the 3D Lego model is output. For instance, we start with *Base*. This is made of Lego Piece 52036, placed at position (0,0,0) with no rotations, and colored Black. So, the interpreter simply outputs that line.

Next, we have *WheelType1(Front)*. This consists of 5 pieces—a plate with pegs for the wheels, the wheels themselves and the tires. Both the plate and the wheels are Dark Grey, but the tires are always Black in this symbol, regardless of the color scheme. All of these pieces are arranged together into a single unit, which is then translated into place based on the position set for it by in the *Base* expansion in the Design Motif.

All other expansions happen similarly, and in the end, we get a full car file—in this case, the bottom car in Figure 3.

6. EXAMPLE ARTIFACTS

Figure 3 and Figure 4 show three sample outputs of the grammar in its current state, using the following design motifs:

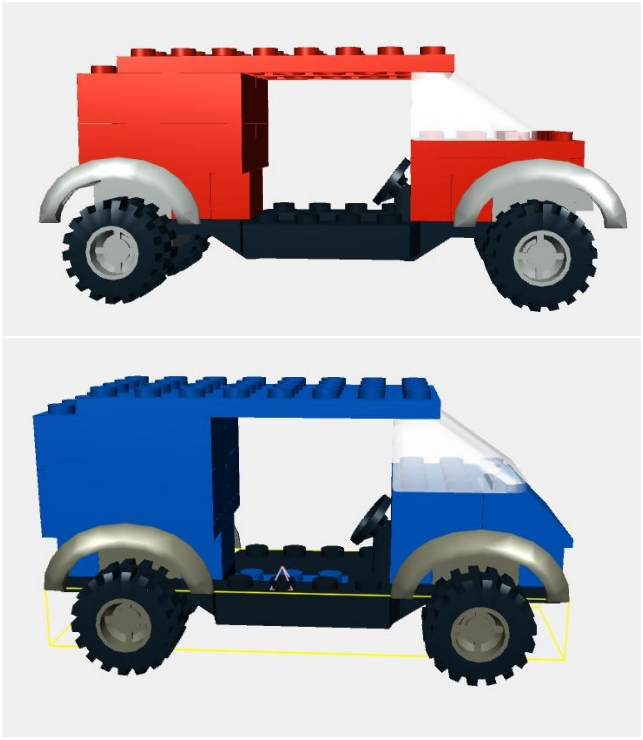

Figure 3: Sample output of the current generator.

Grammar Symbol	Red Motif	Blue Motif
Primary Color	Red	Blue
Secondary Color	Light Grey	Dark Grey
Tertiary Color	Black	Black
Base Type	Basic	Basic
Wheel Type	Basic	Basic
Wheel Position	Wide	Narrow
Compartment Height	Low	Raised
Decorative Piece 1	None	Sloped Front
Decorative Piece 2	None	None

The results are chosen not as an all-encompassing expression of the grammars capabilities, however, the particular instances are chosen to show a small range of changes that can be made within the realm of comparability. Changes featured in the designs include the color scheme (moving from red and light grey to blue and dark grey), the position of the wheels, the height of the cab, and the inclusion of the additional sloped piece in the blue car. In contrast, Figure 4 is the result of using the first design motif with a different grammar - one that used a narrower base and a windshield on both front and back.

These changes are simple on their own, but some of them have cascading effects. For instance, the placement of the wheels in turn determines the placement of the curved pieces, as they are always need to be centered over the wheels. Additionally, raising the minifigure compartment height raised the height of the roof by the same amount, because otherwise, the minifigure wouldn't have been able to fit.

Figure 4: A different car grammar, with the first design motif.

7. FUTURE WORK

First and foremost, the system needs to incorporate more expressivity, mainly in terms of the range of possible outputs. At the moment, there are only so many symbols that the initial symbols can be expanded into in the design motif stage, and this is the biggest limitation of the system so far. As more chunks get authored, the number of possible artifacts gets expanded.

The other direction that looks promising for this research would be to redesign the symbol as a constraint solver. By modeling "car" as a set of constraints on the entire possibility set of Lego pieces, and by modeling the design motifs as further constraints placed upon that system, we can exploit many of the important properties that we want in this system, while leaving the generator free to fill the constraints in any way that fits the constraints. This also reduces some of the burden of hand-authoring, and gives the system more expressivity, almost for free.

The next big question that this word leads to is related to the core concept of the design motif - specifically, where the boundaries between design motifs fall. Using results from psychological research on how humans categorize objects [6], user studies can be designed to compare similar objects to discover which features best separate similar motifs from each other, as well as the converse problem of discovering how dissimilar objects can be before they're grouped into two separate classes.

Finally, we hope to eventually expand into other domains, such as generated architecture (thanks to all the groundwork previously lain in generating designs based on specific architects) and generation of assets for games across a variety of genres and artistic styles. Currently, games require a larger number of assets than their artists could ever create. A system for generating objects that are consistent with a games given design motif would allow designers to reduce both the amount of hand-authored content, and the reuse of the content that they do create.

8. CONCLUSION

Generation using design motifs is an unexplored area of procedural content generation - while systems are in place that generate artifacts to match a particular design motif, there has been no work into expanding this into a more general case. This research aims to correct this gap, by starting in the domain of Lego models and then expanding into other domains using this groundwork. By incorporating the notion of design motifs in to grammars and grammar-like systems, artifacts that incorporate a notion of style can be generated in such a way that they can be checked against the given style - and similar ones - to ensure that the artifact is a good example of the motif used to generate it.

9. REFERENCES

[1] G. Çağdaş. A shape grammar: the language of traditional Turkish houses. *Environment and Planing B: Planning and Design*, 23(5):443–464, 1996.

[2] H. H. Chau, X. Chen, A. McKay, and A. de Pennington. Evaluation of a 3D shape grammar implementation. In *Design Computing and Cognition 04*, pages 357–376. Springer, 2004.

[3] A. Devert, N. Bredeche, and M. Schoenauer. Blindbuilder: A new encoding to evolve lego-like structures. In *Genetic Programming*, pages 61–72. Springer, 2006.

[4] P. Funes and J. Pollack. Computer evolution of buildable objects. *Evolutionary design by computers*, 1:387–403, 1999.

[5] H. Koning and J. Eizenberg. The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B*, 8(3):295–323, 1981.

[6] G. Lakoff. *Women, Fire, and Dangerous Things: What categories reveal about the mind*. Cambridge Univ Press, 1990.

[7] J. P. McCormack, J. Cagan, and C. M. Vogel. Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars. *Design studies*, 25(1):1–29, 2004.

[8] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. In *IACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006, 25(3)*, pages 614–623, 2006.

[9] M. J. Pugliese and J. Cagan. Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design*, 13(3):139–156, 2002.

[10] Image - 5974 box.jpg - Brickipedia, the LEGO Wiki. Accessed: 2015-02-19.

[11] Image - Undercover Cruser(Box).png - Brickipedia, the LEGO Wiki. Accessed: 2015-02-19.

[12] G. Stiny and J. Gips. Shape Grammars and the Generative Specification of Painting and Sculpture. In *IFIP Congress (2)*, pages 1460–1465, 1971.

[13] G. Stiny, W. J. Mitchell, et al. The palladian grammar. *Environment and planning B*, 5(1):5–18, 1978.

[14] T. Tutenel, R. van der Linden, M. Kraus, B. Bollen, and R. Bidarra. Procedural filters for customization of virtual worlds. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in*

Games, page 5. ACM, 2011.

[15] M. Waßmann and K. Weicker. Maximum flow networks for stability analysis of LEGO® Structures. In *Algorithms–ESA 2012*, pages 813–824. Springer, 2012.