Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

A Different Shape Grammar Approach for Automatic Design Generation

Filipe Santos and Joaquim Esmerado

***Abstract*—This paper proposes a different approach for shape grammars where designs are exclusively generated through computations of symbols. This option has the advantage of supporting declarative knowledge and thus facilitates shape reasoning capabilities. Our approach also detached procedural knowledge from shape knowledge using procedural notions that capture sequences, alternatives and tests that should be applied during design process. This decision provides good modular specifications. Moreover, we follow a non-deterministic procedural perspective in the characterization of design processes. Its implementation supports exploratory automatic design generation. Differences between this approach and the conventional approach are discussed and a case study is explored.**

***Keywords*—shape grammars, automatic design generation.**

I. Introduction

We are particularly interested in the automatic generation of designs based on shape grammars as a descriptive method for shapes [18]. We are currently developing computational tools for shape computing [14] and working on its application in the generation of urban and architecture designs [11] [12][13].

Shape grammars have successfully been used to generate a variety of designs [1], [4], [5], [16], [17]. However, shape knowledge within these specific design applications is represented in a procedural and ad hoc way and is therefore too rigid for generic automation.

We believe that a declarative knowledge-based approach would offer more flexibility to face new design situations and improve shape reasoning capabilities. We also believe that the shape grammar formalism should be extended with convenient abstractions for flexible design specification and generation.

In this paper we propose a different approach for shape grammars. Shapes are directly represented by symbols. These symbols support declarative knowledge representation and shape reasoning capabilities. A preliminary sketch of this approach has been presented in [13]. Herein we proceed forward by clarifying the approach, addressing concrete examples and discussing ways to support shape emergence.

An overview of the rest of the paper follows. We start by presenting the conventional shape grammar formalism. Subsequently, we present our approach based on symbols and supplemented with procedural primitives for describing design processes.

Filipe Santos and Joaquim Esmerado
Instituto Universitário de Lisboa (ISCTE-IUL), Portugal

Next, a case study on a constructive modular system with wooden modules exemplifies our approach. We conclude by mentioning our immediate research directions.

II. Shape Grammar

Shape grammar formalism was originally proposed by Stiny and Gips [18] for creating and understanding designs through computations with shapes, rather than through computations with text or symbols. For an historic and panoramic perspective on shape grammars see [2] or [3]. Stiny and Gips have proposed that the computation of shapes should be carried out in two steps: the recognition of a particular shape and its possible replacement by another shape.

A shape grammar consists of:

- a vocabulary of primitive shapes;

- shape rules of the form $A \rightarrow B$, where A and B are shapes;

- an initial shape.

Two shapes s and u may be combined and form a new shape $s + u$ (shapes in s or in u) or $s - u$ (shapes in s not in u). Given an appropriate vocabulary of shapes we may form an algebra where both operations are closed on the space of all possible shape combinations. Given a shape combination u, the recognition of a particular shape s in u can be supported by a sub-shape operation, $s<u$ denoting s is a sub-shape of u. Application of an Euclidean transformation (translation, rotation, reflection and scale) t to a shape A provides the production of a new shape $t(A)$.

Replacements of shapes are obtained through application of shape grammar rules. A shape grammar rule $A \rightarrow B$ applies to a shape s whenever there is an Euclidean transformation t such that $t(A)<s$. The result of the rule application is $s-t(A)+t(B)$, the shape obtained by replacing the sub-shape $t(A)$ of s by the shape $t(B)$.

Given a shape grammar, shapes may be generated/derived starting from the initial shape and sequentially applying shape rules to the obtained shapes, i.e., a sequence of shapes s_0, s_1, …, s_n, where s_0 is the initial shape and s_{i+1} is the shape obtained from s_i (i=0, …, n) by applying a rule of the shape grammar. Each possible generated shape forms the shape language defined by the shape grammar.

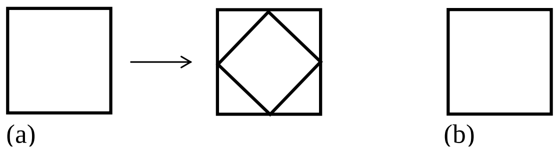


Figure 1. A shape grammar. (a) shape rule, (b) initial shape.

Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

Let us consider an adaptation of the following example from [16] of a shape grammar with lines as a vocabulary of shapes. The shape grammar of Figure 1 yields a language of inscribed squares as shown in Figure 2.

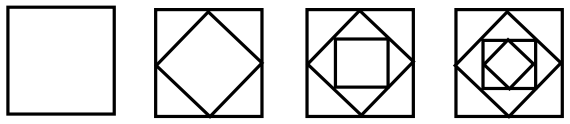


Figure 2. A language of inscribed squares.

Each shape of the generated language is obtained starting from the initial shape and repeatedly applying the shape grammar rule to the innermost square, each application using different Euclidean transformations combining rotation and scale applied to the square of left side of the shape rule.

Designing is the process of shape manipulation (2D or 3D) and materials information that is needed to guide the construction of an artifact. It is a iterative process of shape manipulation that ends when certain conditions are fulfilled. During this process, designers interpret the shapes obtained so far and their interpretation influences the progress of design. Designers frequently recognize emergent sub-shapes, i.e., shapes not explicitly introduced, thus providing new interpretations and new directions of design progress.

Within shape grammars emergence is a foundational feature mentioned and discussed by many researchers [10][19][20]. Emergent shapes may be seen as shapes that are not added by shape rule applications, i.e. any shape that is not a shape *t(B)* added by a previous application of a shape rule $A \rightarrow B$.

The previous language also illustrates these phenomena of emergent sub-shapes. Shape on Figure 3(a) emerges in four places of Figure 3(b)

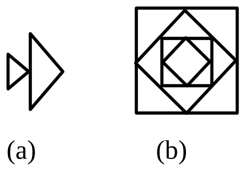


Figure 3. An emergent sub-shape within a shape.

The sub-shape operation is crucial for supporting shape grammar rules application to shapes and mainly depends on the primitive vocabulary of shapes. For 2D shapes, Stiny proposed a canonical representation of shapes [16] – the maximal lines form – based on lines. In this representation, a line is determined by a set of two distinct end points and a shape is a finite set of lines. The maximal line representation of a shape is the unique smallest set of lines that represent the shape.

Given a shape, the process for obtaining this maximal line representation consists in combining two collinear lines with the + shape operation only in the following four situations:

(1) two lines share an end point and the remaining end point of one line is coincident with the other line. The maximal line is represented by the shared end point and the remaining end point of the other line (Figure 4(a));

(2) both end points of a line is coincident with the other line. The maximal line is the second line (Figure 4(b));

(3) one end point of each line is coincident with the other line. The maximal line is represented by the remaining end points (Figure 4(c));

(4) the two lines share an end point and this point is coincident with the line formed by the two remaining unshared end points. The maximal line is represented by the remaining unshared end points (Figure 4(d)).

Given two shapes *s* and *u* represented by maximal line representation, *s* is a sub-shape of *u*, i.e. $s<u$, if and only if the end points of every maximal line of *s* are both coincident with a maximal line of *u*.

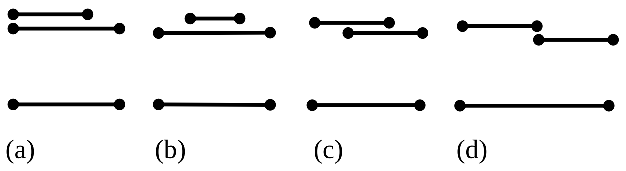


Figure 4. Situations of line combination of maximal lines.

Sub-shape detection (including emergent shapes) is supported by sub-shape operation when shapes are represented in maximal line form. The end points and cross points of the lines in the shapes represented in maximal line form are sufficient for the determination of an Euclidian transformation *t* such that $t(u)<s$. It is sufficient to find out a correspondence between 3 distinct non-collinear points p_1, p_2, p_3 of *u* with 3 distinct non-collinear points $t(p_1)$, $t(p_2)$, $t(p_3)$ of *s*. The remaining points *p* of *u* should also correspond to points *t(p)* of *s*.

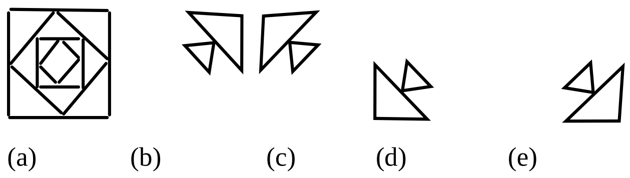


Figure 5. Maximal line representation and four sub-shapes.

The maximal line representation in Figure 5 (a) allows the detection of the emergent shape in Figure 3 (a) in 4 different places, since the Figures 5 (b) 5 (c) 5 (d) 5 (e) in Figure 5 (a) may be obtained from Figure 3 (a) by applying a rotation of 45^o, 135^o, 225^o and 315^o respectively, followed by a convenient translation.

Computing maximal line representation and finding Euclidian transformations for sub-shape operation is a time consuming process and appropriate computational forms of representation of shapes should be adopted if we want to avoid intractable algorithms [6][7][8][9].

The shape grammar formalism also employs labeled points as a way of controlling the application of shape rules during

Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

the design process. This control may avoid the application of particular Euclidian transformations or even completely block the application of one or more shape rules. Using labeled points, a sequential programming style can be used to describe the design process. Shape rules with labeled points on their right hand side must be used before other shape rules with the same labeled points on their left hand side. Moreover, the application of shape rules with labeled points on their left hand is blocked to shapes without the same labeled points. Using this strategy, shape grammars have been used to generate a variety of designs [1][4][5][16][17]. As a consequence, within these specific design applications shape knowledge and procedural knowledge are mixed together in an ad hoc way and is therefore too rigid for generic specification and automation.

We believe that shape knowledge and procedural knowledge should be detached from each other if we want to obtain a framework for flexible design generation and quick design specification. Detaching this knowledge would facilitate convenient abstraction and modularization for algorithmic development, one of most desirable properties in computer science. An approach for design generation should thus provide convenient abstractions for representing shape knowledge, procedural knowledge and also the definition of other relevant application concepts.

III. Our Alternative Approach

Like shape grammars our approach deals with shapes and shape rules. However, there is a fundamental difference: the emphasis is made on symbols, not on shapes.

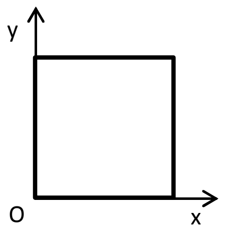


Figure 6. Picture identified by Square.

In our approach shapes are represented by identifiers associated with pictures. For instance, *Square* may be associated with the picture of Figure 6 defined in the respective coordinate system xOy.

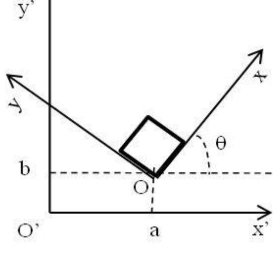


Figure 7. Positioning Square within a new coordinate system.

Positioning a shape defined in this way within a different coordinate system x'O'y' is represented by a pair (shape, transformation) specifying the transformation (translation, rotation, scale, etc.) required for positioning the coordinate system xOy associated to the defined shape into the new coordinate system x'O'y'. Note that each shape is represented by the identity transformation *Id* when the shape is defined and in this case the identity transformation may be omitted within the representation.

For instance, in Figure 7, Square is positioned into the coordinate system x'O'y' by a scale S(c,d) and rotation R(θ) followed by a translation T(a,b).

Using a matrix representation within an homogeneous coordinate system, the previous shape may be represented by (Square, T(a,b)R(θ)S(c,d)) where

$$T(a,b) = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \quad R(\theta) = \begin{bmatrix} \cos\theta & -\text{sen}\theta & 0 \\ \text{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(c,d) = \begin{bmatrix} c & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shape compositions may be represented by sets of shapes positioned in the same coordinate system. For instance, for a 1x1 unit Square, the 4 inscribed squares of Figure 3 (b) are represented by {Square, (Square, T(1/2, 0)R(45º)S(1/$\sqrt{2}$, 1/$\sqrt{2}$)), (Square,T(3/4,1/4)R(90º)S(1/2,1/2)), (Square, T(3/4, 1/2) R(135º) S(1/(2$\sqrt{2}$),1/(2$\sqrt{2}$)))}.

In this approach Shape Grammar Rules are represented by pairs of shape compositions. For instance, for a 1x1 unit Square, the shape grammar rule of Figure 1 (a) is represented by {Square} $\rightarrow$ {Square, (Square, T(1/2, 0)R(45º)S(1/$\sqrt{2}$,1/$\sqrt{2}$)}

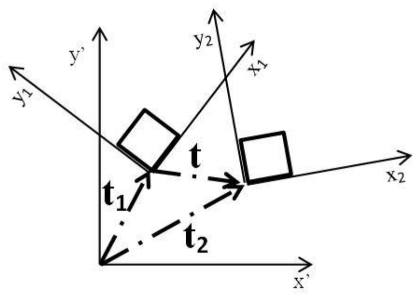


Figure 8. *t* transforms shape (Square,t_1) into shape (Square,t_2).

Using this representation and given two shape compositions A and B, sub-shape operation is defined by *A<B iff A⊆B*, i.e. all the shapes in A are also in *B*. And discovering an Euclidean transformation *t* such that *t(A)<s* for the application of a shape grammar rule *A→B* is simply finding a transformation able to transform every shape of A into shapes within *s*. Note that, using a matrix representation within an homogeneous coordinate system, an Euclidean transformation

Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

t able to transform shape (id,t_1) into shape (id,t_2) is easily obtained by $t_2t_1^{-1}$ (see e.g. Figure 8).

The following algorithm is used for obtaining such Euclidian transformation:

Algorithm: Euclidean transformation t such that *t(A)<s*

1. Find two shapes with the same identification *id* in both shape compositions *A* and *s*, i.e. $(id,t_1)\in A \wedge (id,t_2)\in s$;
2. $t = t_2t_1^{-1}$, i.e. t is the Euclidean transformation able to transform shape (id,t_1) into shape (id,t_2) (i.e. (id,t_2)= (id,t_1t));
3. Confirm that if $(id',t')\in A$ then $(id',t't)\in s$, i.e. t transforms every remaining shape in *A* into a shape of *s*.

Given the previous representation of 4 inscribed squares, the previous shape grammar rule may be applied with four different Euclidian transformations since the shape of it left side *Square* may be transformed in each of the four shapes of the right side of the rule using the transformations: *Id*, *T(1/2, 0) R(45º)S(1/ $\sqrt{2}$,1/ $\sqrt{2}$), T(3/4,1/4)R(90º)S(1/2,1/2)* and *T(3/4,1/2)R(135º)S(1/(2$\sqrt{2}$),1/(2$\sqrt{2}$)).*

Without any further mathematical machinery, the formalization that we have proposed so far does not allow the detection of emergent sub-shapes. However, we believe that the introduction of an equality operation on shape compositions allows our approach to surpass this limitation in the same fashion that the maximal line form does by presenting an alternative representation of the same shapes.

Let's consider again the previous shape of 4 inscribed squares of Figure 3(b). We already represent it using Square as a primitive shape. However, considering now the shapes Line and Triangle of Figure 9, the shape of 4 inscribed squares may be alternatively represented by 16 lines (Figure 10 (a)), as in the maximal line form, or even by 8 triangles (Figure 10 (b)).

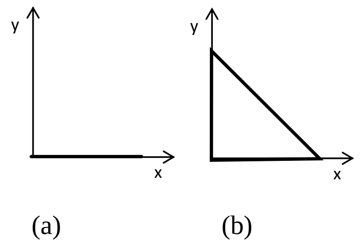


Figure 9. Pictures identified by Line and Triangle.

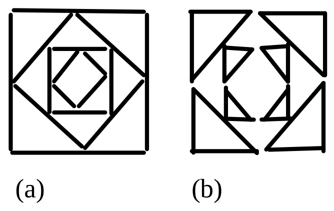


Figure 10. Different representations of Figure 3 (b) using Line and Triangle.

Considering a defined 1 unit long Line and a right Triangle with 1 unit long in both sides opposite to hypotenuse, the 4 inscribed squares may be alternatively represented by { Line, (Line, T(1,0)R(90º)), (Line, T(1,1)R(180º)), (Line, T(0,1) R(270º)), (Line, T(1/2,0)R(45º)S(1/$\sqrt{2}$,1)), (Line, T(1,1/2) R(135º)S(1/$\sqrt{2}$,1)), (Line,T(1/2,1)R(225º)S(1/$\sqrt{2}$,1)), (Line, T(0,1/2)S(1/$\sqrt{2}$,1)R(315º)), (Line, T(1/4,1/4)S(1/2,1)), (Line, T(3/4,1/4)R(90º)S(1/ $\sqrt{2}$,1)), (Line, T(3/4,3/4)R(180º) S(1/ $\sqrt{2}$,1)), (Line, T(1/4,3/4)R(270º)S(1/2,1)), (Line, T(1/2,1/4)R(45º)S(1/(2 $\sqrt{2}$),1)), (Line, T(3/4,1/2)R(135º) S(1/(2$\sqrt{2}$),1)), (Line, T(1/2, 3/4)R(225º)S(1/(2$\sqrt{2}$),1)), (Line, T(1/4,1/2)R(315º)S(1/(2 $\sqrt{2}$),1))} or by {(Triangle, S(1/2,1/2)), (Triangle, T(1,0)R(90º)S(1/2,1/2)), (Triangle, T(1,1)R(180º)S(1/2,1/2)), (Triangle,T(0,1)R(270º) S(1/2,1/2)), (Triangle, T(1/4,1/4)S(1/4,1/4)), (Triangle, T(3/4,1/4)R(90º) S(1/4,1/4)), (Triangle, T(3/4,3/4)R(180º)S(1/4,1/4)), (Triangle, T(1/4,3/4)R(270º)S(1/4,1/4))}.

Let's consider now the following shape composition Fish presented in Figure 11 represented by

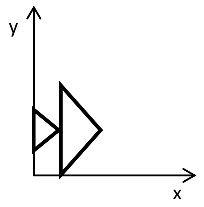


Figure 11. A shape composition identified by Fish.

{ (Triangle, T(1/(4$\sqrt{2}$),1/(2$\sqrt{2}$))R(135º)S(1/4,1/4)), (Triangle, T(3/(4$\sqrt{2}$),1/(2$\sqrt{2}$))R(135º)S(1/2,1/2))}

Given the previous Triangle representation of 4 inscribed squares, Fish may be transformed in each of the four corners shapes using the transformations:

1. *T(1/4,1/4)S(1/4,1/4)(T(1/(4$\sqrt{2}$),1/(2$\sqrt{2}$))S(1/4,1/4)))$^{-1}$*
2. *T(3/4,1/4)R(90º)S(1/4,1/4)(T(1/(4$\sqrt{2}$),1/(2$\sqrt{2}$))R(135º)S(1/4,1/4)))$^{-1}$*
3. *T(3/4,3/4)R(180º)S(1/4,1/4)(T(1/(4$\sqrt{2}$),1/(2$\sqrt{2}$))R(135º)S (1/4,1/4)))$^{-1}$*
4. *T(1/4,3/4)R(270º)S(1/4,1/4)(T(1/(4$\sqrt{2}$),1/(2$\sqrt{2}$))R(135º)S (1/4,1/4)))$^{-1}$.*

Although it needs further research, we believe that an equality operation on shape compositions based on a defined shape hierarchy allows the approach to deal with sub-shape operation applied to shapes with different representations. The main idea is to precise how a shape may be defined using other shapes. The definition may form a hierarchy that may be used to deduce other equalities and find a common representation for shapes. For instance, the following definitions may relate Square and Triangle with their composed lines and may be used to reduce shapes to lines:

{Square} = { Line, (Line, T(1,0)R(90º)), (Line, T(1,1)

R(180º)), (Line, T(0,1)R(270º)) }

{Triangle} = { Line, (Line, R(90º)), (Line, T(1,0) R(135º)S($\sqrt{2}$,1)) }.

There are various possibilities for defining sub-shape operation in this approach, e.g. given two shape compositions A and B:

A<B iff ∃A'=A ∃B'=B such that A'⊆B';

A<B iff ∀C∈A ∃D⊆B ∃D'=D such that {C}⊆D';

The first possibility proposes finding a common representation for shapes to which the sub-shape operation may apply, in the same fashion of maximal line representation proposed by Stiny, but not necessarily a canonical form in the bottom of the hierarchy of shapes. The second possibility proposes only a common representation for each shape in the shape composition A.

One main advantage of our approach is that it easily supports shape composition properties by using predicates φ whose evaluation depends on the relative position of shapes within a shape composition *s*, i.e. given a boolean valuation v: *Shape LF* → *boolean* on logical formulas *LF*, the property φ hold in s iff v(s,φ) = true. Examples are presented in the next section.

Instead of using labeled points for controlling the application of shape rules during the design process, our approach detached procedural knowledge from shape knowledge using procedural notions that capture sequences, alternatives and tests that should be applied to the initial shape during design process. The following BNF rules represent our language syntax for defining design processes:

< Design Process > ::= < Shape Grammar Rule >

| < Alternative Design Process >

| < Sequential Design Process >

| < Test >

< Shape Grammar Rule > ::=

< Shape Composition > → < Shape Composition >

< Alternative Design Process > ::=

(< Design Process > or < Design Process>)

< Sequential Design Process > ::=

< Design Process >;< Sequential Design Process > | ε

< Test > ::= Verify(< Boolean Condition >)

The meaning of the previous procedural notions is easily sketched by the following function *exec:Proc Shape* → 2^{Shape} that characterizes the changes of a shape composition s∈*Shape* by the application of the design processes *A*→*B*, (*p*$_1$ *or p*$_2$), (*p*$_1$*; p*$_2$), *Verify*(*φ*) (for *Transf* the set of Euclidean transformations and a boolean valuation v: *Shape LF* → *boolean* on logical formulas *LF*):

$$\text{exec}(A\rightarrow B,s) = \{s-t(A)+t(B) \mid \exists t\in \mathit{Transf}\ t(A)<s\}$$

$$\text{exec}((p_1 \text{ or } p_2),s) = \text{exec}(p_1,s) \cup \text{exec}(p_2,s)$$

$$\text{exec}((p_1 ; p_2),s) = \bigcup_{u\in \text{exec}(p_1,s)} \text{exec}(p_2,u)$$

$$\text{exec}(\text{Verify}(\varphi),s) = \{s \mid v(s, \varphi) = \text{true}\}$$

A shape composition *s* may be generated by *p* starting from the initial shape composition s_0 iff s∈exec(p,s$_0$).

Note that a design process applies repeatedly shape rules to the intermediate shapes obtained so far according to the order establish sequentially in the process. The shape grammar rules and the alternative compositions of the process offer the possibility to generate a shape composition among different alternatives. This means that we follow a non-deterministic perspective in the characterization of design processes. However, operationally designs in exec(p,s$_0$) may be produced by forward chaining using some operational preference in the choice of the alternatives. Each time a test process *Verify* fails or a shape grammar rule fails to apply, the system backwards trying to build a different solution.

IV. A Case Study

Our approach uses as a case study a work in shape grammars applied to a constructive modular system with wooden modules for building houses in a flexible way [15]. The proposed shape grammar uses plans for a variety of wooden modules for walls, some with windows and some others with doors. Walls may be connected by a fix set of connection beams within a set of different predefined areas or basements. Figure 12 gives an idea of the obtained houses.

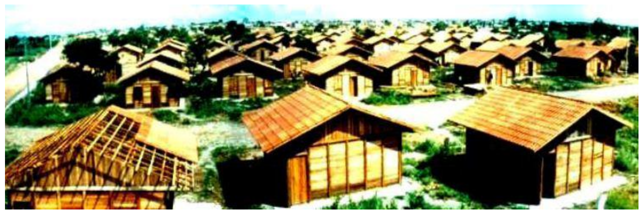


Figure 12. Houses built with wooden modules.

The proposed grammar offers a relative degree of freedom in the internal division of space. The wooden modules may be combined in different ways provided that some restrictions are satisfied. For instance, the kitchen should be placed near a circulation area and the living room but not near the bedroom.

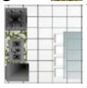
The design process follows a specific order. First a basement is selected, second the living areas are placed, next walls are chosen and finally connections beams are conveniently placed between walls. This imposes a procedural order for the application of the shape rules.

Let us illustrate the application of our approach emphasising the representation of the design process until the placement of living areas. Here we use an = to associate symbols/identifiers with their visual representations. We also use identifiers for shape compositions, rule processes and shape rules.

Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

Let us consider the following basic shapes:

BasicSquare = , CirculationArea = , Outside = ,

BathRoom = , BedRoom = , Kitchen = ,

LivingRoom = .

Shapes BasicSquare, CirculationArea and Outside have an area of 1.5X1.5. Shapes BedRoom and Kitchen have 3.2x3.2. Shape LivingRoom have 4.9x3.2 and shape BathRoom have1.5x3.2.

Using these basic shapes, we may define the following basements of Figure 13 with different dimensions and configurations using the following shape compositions, where shapes BasicSquare are spaced by 0.2 units:

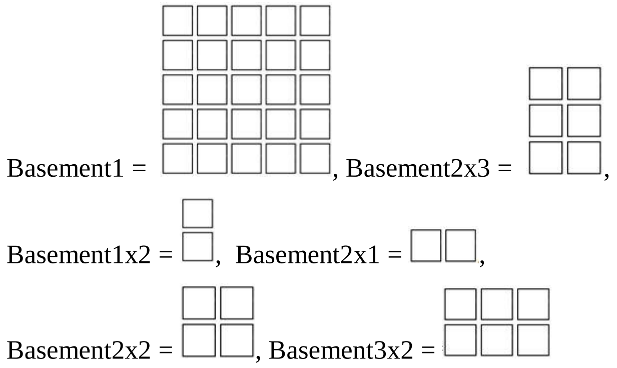


Figure 13. Different basements for modular houses.

Basement1 = { BasicSquare, (BasicSquare, T(1.7,0)), (BasicSquare, T(3.4,0)), (BasicSquare, T(5.1,0)), (BasicSquare, T(6.8,0)), (BasicSquare, T(0,1.7)), (BasicSquare, T(1.7,1.7)), (BasicSquare, T(3.4,1.7)), (BasicSquare, T(5.1,1.7)), (BasicSquare, T(6.8,1.7)), (BasicSquare, T(0,3.4)), (BasicSquare, T(1.7,3.4)), (BasicSquare, T(3.4, 3.4)), (BasicSquare, T(5.1,3.4)), (BasicSquare, T(6.8,3.4)), (BasicSquare, T(0,5.1)), (BasicSquare, T(1.7,5.1)), (BasicSquare, T(3.4,5.1)), (BasicSquare, T(5.1,5.1)), (BasicSquare, T(6.8,5.1)), (BasicSquare, T(0,6.8)), (BasicSquare, T(1.7,6.8)), (BasicSquare, T(3.4,6.8)), (BasicSquare, T(5.1,6.8)), (BasicSquare, T(6.8,6.8)) }

Basement1x2 = { BasicSquare, (BasicSquare, T(0,1.7))}

Basement2x1 = { BasicSquare, (BasicSquare, T(1.7,0))}

Basement2x2 = { BasicSquare, (BasicSquare, T(1.7,0)), (BasicSquare, T(0,1.7)), (BasicSquare, T(1.7,1.7))}

Basement3x2 = { BasicSquare, (BasicSquare, T(1.7,0)), (BasicSquare, T(3.4,0)), (BasicSquare, T(0,1.7)), (BasicSquare, T(1.7,1.7)), (BasicSquare, T(3.4,1.7))}

Basement2x3 = { BasicSquare, (BasicSquare, T(1.7,0)), (BasicSquare, T(0,1.7)), (BasicSquare, T(1.7,1.7)), (BasicSquare, T(0,3.4)), (BasicSquare, T(1.7,3.4))}

Based on the previous shape compositions we may now characterize the following shape grammar rules of Figure 14 used for generating space occupation of living areas:

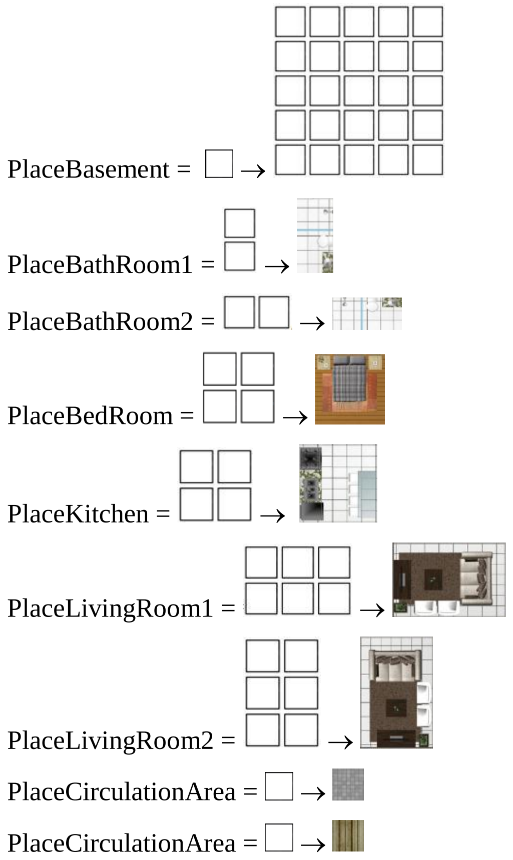


Figure 14. Shape grammar rules for modular houses.

PlaceBasement = {BasicSquare} $\rightarrow$ Basement1

PlaceBathRoom1 = Basement1x2 $\rightarrow$ {BathRoom}

PlaceBathRoom2 = Basement2x1 $\rightarrow$ {(BathRoom, T(3.2,0)R(90º))}

PlaceBedRoom = Basement2x2 $\rightarrow$ {BedRoom}

PlaceKitchen = Basement2x2 $\rightarrow$ {Kitchen}

PlaceLivingRoom1 = Basement3x2 $\rightarrow$ {LivingRoom}

PlaceLivingRoom2 = Basement2x3 $\rightarrow$ {(LivingRoom, T(3.2,0)R(90º))}

PlaceCirculationArea = {BasicSquare} $\rightarrow$ {CirculationArea}

PlaceOutside = {BasicSquare} $\rightarrow$ {Outside}

The shape compositions Basement1x2 and Basement2x1 are necessary to ensure facing up or down the non-squared shape BathRoom. The same happens with shape LivingRoom with respect with shape compositions Basement3x2 and Basement2x3. This is due to the fact that shapes BasicSquare, (BasicSquare,T(1.5,0)R(90º)), (BasicSquare, T(1.5,1.5) R(180º)) and (BasicSquare, T(0,1.5)R(270º)) have all the same visual representation. But without explicitly turn them equals, the only solution is considering different shape compositions

Proc. of the Second Intl. Conf. on Advances In Computing, Communication and Information Technology- CCIT 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-051-4 doi: 10.15224/ 978-1-63248-051-4-40

for substitution.

Let us now represent the design process for the placement of living areas:

ModularHouse = PlaceBasement; PlaceLivingAreas

PlaceLivingAreas = PlaceLivingRoom; PlaceKitchen; PlaceBedRoom; CirculationAreaProcess; OutsideProcess; Verify(LivingAreaRestrictions)

PlaceBathRoom = PlaceBathRoom1 or PlaceBathRoom2

PlaceLivingRoom = PlaceLivingRoom1 or PlaceLivingRoom2

CirculationAreaProcess = Verify(EverythingElseConnected) or(Verify(¬EverythingElseConnected); PlaceCirculationArea; CirculationAreaProcess)

OutsideProcess = Verify(exists(BasicSquare)) or (Verify(¬ exists(BasicSquare)); PlaceOutside;OutsideProces)

These processes uses testes on predicates LivingAreaRestrictions and EverythingElseConnected defined by the following equivalences (where ∧, ¬ and ↔ represents respectively the propositional conectives of conjunction, negation and equivalence):

LivingAreaRestrictions ↔ (next(Kitchen, LivingRoom) ∧ next(Kitchen, CirculationArea) ∧ ¬next(BedRoom,Kitchen) ∧ next(Outside, LivingRoom))

EverythingElseConnected ↔ (connected(LivingRoom, BedRoom) ∧ connected(LivingRoom, BathRoom))

The predicate next(Id1,Id2) is satisfied by a shape composition s iff there are shapes (Id1,t1) and (Id2,t2) in s occupying contiguous areas spaces. This property is easily defined knowing the extreme points of each area occupied by shapes (Id1,t1) and (Id2,t2). The predicate exists(Id) is satisfied by a shape composition s iff there is a shape (Id,t) in s. The predicate connected(Id1,Id2) is satisfied by a shape composition s iff there are shapes (Id1,t1) and (Id2,t2) in s occupying contiguous areas spaces or connected by contiguous CirculationArea shapes.

Almost all processes are self-explanatory. The design processes CirculationAreaProcess and OutsideProcess are just iterative processes using the previous design primitives, the former to ensure that all living areas are connected and the later to occupy the remaining spaces by outside space. Note that a conventional *while(φ)c* loop may be represented in the following way: Procid = Verify(¬φ) or (Verify(φ); c; Procid); and the usual alternative *if(φ) c else d* may be represented by (Verify(φ); c) or (Verify(¬(φ)); d).

The ModularHouse design process allows the following evolution of shapes compositions in Figure 15 satisfying internal division of space restrictions defined in design process identified by PlaceLivingAreas.

V. Conclusion

We believe that the declarative knowledge-based approach herein sketched supports shape reasoning capabilities and is thus more convenient for automatic design generation. This approach also detached procedural knowledge from shape knowledge facilitating a convenient abstraction and modularization for algorithmic development.

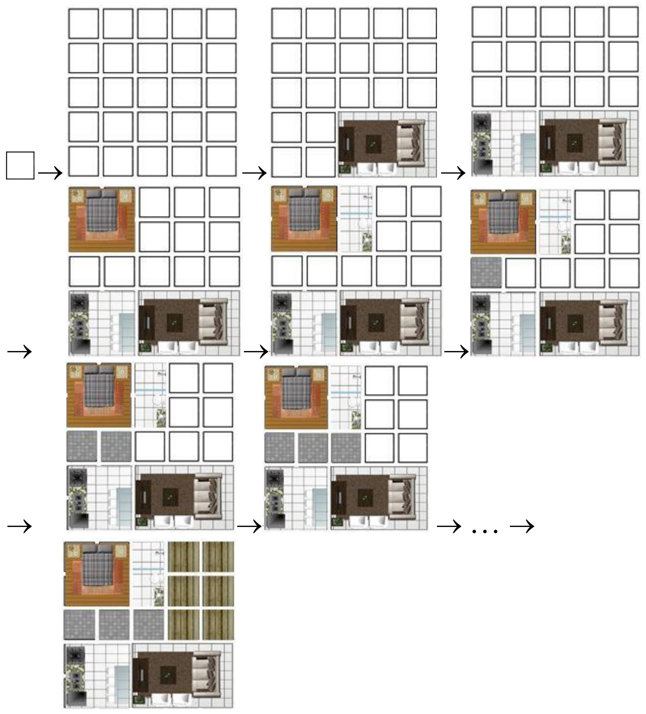
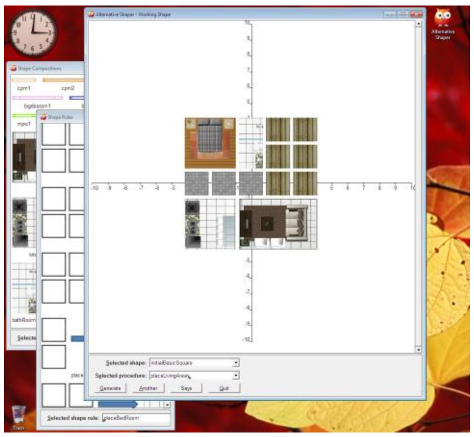


Figure 15. A design generation of house space.

The procedural part is essentially non-deterministic, either by shape rule application, either by the application of alternative processes. This option is strongly influenced by our research goals on the generation of urban and architecture designs. But we also consider the necessity of procedural determinism in cases where we want to block freedom in solution or partial solution generation. We are also aware that other procedural primitives would be necessary. We would consider different primitives as long we explore different case studies.



Selected shape: initialBasicSquare
Selected procedure: placeLivingAreas
Selected shape rule: placeBedRoom

Figure 16. Alternative Shaper Workbench.

A workbench supporting our approach has been developed for exploring diverse case studies (see Figure 16). This first tool was implemented in SWI-prolog in order to explore a depth first search strategy of the design solution space. We are aware that, due to the exponential nature of the search space, an intelligent guided search should be adopted in order to obtain tractable algorithms.

There are other issues that need further research, namely:

1. a complete development of a case study in order to find out approach limitations and propose other convenient abstractions;

2. study an appropriate algebra for shape manipulation using our approach and allowing a combination with the traditional shape grammar approach;

3. Define properly an equality operation on shapes and an adequate sub-shape operation;

4. study appropriate computational forms of representation of shapes and implementation of processes for avoiding intractable algorithms.

References

[1] M. Agarwal and J. Cagan, "A blend of different tastes: the language of coffeemakers", in Environment and Planning B, 25, 1998, pp. 205-226.

[2] S.C. Chase, Shape grammar implementations: the last 36 years (Shape grammar implementation: from theory to useable software), presentation in Design Computing and Cognition workshop, Stuttgart, July2010. http://www2.mech-eng.leeds.ac.uk/users/men6am/documents/DCC2010grammarsworkshop-Chase-revised.pdf.

[3] H.H. Chau, "Evaluation of a 3D Shape Grammar Implementation", in Design Computation and Cognition'04, J.S. Gero, Eds., 2004, pp. 357-376.

[4] J.P. Duarte, "A Discursive Grammar for Customizing Mass Housing: the case of Siza's houses at Malagueira", in Automation in Construction, 14(2), Elsevier Science, 2005, pp. 265-275.

[5] J. Heisserman, "Generative Geometric Design", in IEEE Computer Graphics and Applications, 14, 1994, pp. 37-45.

[6] R. Krishnamurti, "The arithmetic of shapes", in Environment and Planning B, 7, 1980, pp. 463-484.

[7] R. Krishnamurti, "The construction of shapes", in Environment and Planning B, 8, 1981, pp. 5-40.

[8] R. Krishnamurti, "The maximal representation of a shape", in Environment and Planning B, 19, 1992, pp. 267-288.

[9] R. Krishnamurti and R. Stouffs, "Spatial change: continuity, reversibility, and emergent shapes", in Environment and Planning B, 24, 1997, pp. 359-384.

[10] W. Mitchell, "A Computational View of Disign Creativity", in Modelling Creativity and Knowledge-Based Design, J. Gero and M. Maher, Eds., Lawrence Erlbaum Associates, 1993, pp. 25-42.

[11] A. Paio, J. Reis, F. Santos, P. Lopes, S. Eloy and V. Rato, "Emerg.cities4all: Towards a shape grammar bases computational system tool for generating a sustainable and integrated urban design", in Proceedings Conference eCAADe2011 respecting Fragile Places, eCAADe (Education and Research in Computer Aided Architectural Design in Europe), 2011, pp. 133-139.

[12] A. Paio, S. Eloy, J. Reis, F. Santos, V. Rato and P. Lopes, "Emerg.cities4all: Towards a sustainable and integrated urban design", in Proceedings 24th World Congress of Architecture, UIA2011, 2011.

[13] F. Santos, J. Reis, "A Language for Automatic Design Generation", in Proc. International Conference on Information Systems and Design of Communication, ISDOC 2013, 2013, pp. 64-68.

[14] F. Santos, J. Reis, P. Lopes, A. Paio, S. Eloy and V. Rato, "A Multi-Agent Expert System Shell for Shape Grammars", in Proc. 17th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, CAADRIA 2012, 2012,409-414.

[15] R. Sousa and A. Santos, Uma gramática para um sistema construtivo com painéis de madeira, Technical Report, ISCTE-IUL, 2012.

[16] G. Stiny, "Introduction to shape and shape grammars", in Environment and Planning B, 7(3), 1980, pp. 343-351.

[17] G. Stiny, "Kindergarten grammars: designing with Froebel's building gifts", in Environment and Planning B, 7, 1980, pp. 409-462.

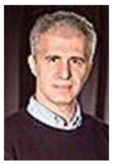
[18] G. Stiny and J. Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture", in Information Processing, 71, C. V. Freiman, Eds., Amsterdam: North- Holland, 1972, pp. 1460-1465.

[19] G. Stiny, "Shape Rules: closure, continuity and emergence", in Environment and Planning B, 21, 1994, pp. S49-S78.

[20] K. Terry, "Computing with Emergence", in Environment and Planning B: Planning and Design, 30, 2003, pp. 125-155.

About Author (s):

Filipe Santos has a PhD in Computer Science and Assistant Professor at Lisbon University Institute. Research topics: Institutional and Legal Knowledge Representation; Design Generation in Architecture.



Joaquim Esmerado has a PhD in Computer Science and Assistant Professor at Lisbon University Institute. Research topics: Computer Graphics.