GEOWIN
A System for Creative Pattern Generation based on Rules

Joaquim A. M. dos Reis
ISCTE, Dept. Ciências e Tecnologias de Informação, Av. das Forças Armadas, 1600 Lisboa, Portugal

Keywords: Rule-Based Systems, Shape Grammars, Design, Creativity.

Abstract: GEOWIN, a computational system based on rules for generating visual patterns is being developed and this paper describes the work done so far. Rules are used to implement shape grammars to represent styles of visual composition. This work is a continuation of previously published work addressing the construction of a system employing shape grammar rules grouped to emulate styles of visual composition, in order to support the behavior of different artistic creative intelligent agents with different styles. One of the goals of this work is to have a multi-agent system that, by making use of the shape grammar formalism, will be able to support creative visual composition synthesis activities, with each intervening agent giving its creative contribution through a style of its own. Another goal is to use the system as a tool to realize, test and study creativity criteria and creative processes.

1 INTRODUCTION

In this paper, we show work done and in progress in implementing part of a system proposed in previously published papers. See (Reis, 2006a), (Reis, 2006b), (Reis, 2006c), (Reis, 2006d), (Reis, 2008a) and (Reis, 2008b), where we describe a multi-agent system in which different artistic creative intelligent agents, each with its own style, are able to involve in artistic visual composition activities, and where shape grammars are used to emulate styles of visual composition of each agent. We first expose some concepts centered around computational creativity, then we introduce shape grammars briefly, and finally we describe the work done and the present state of the system implementation.

By one side we are interested in having a tool to generate alternative creative visual compositions with specific styles, or mixture of styles (and, of course, possible applications for this kind of system could be visual composition generation with mixed styles, either free generation or controlled and goal oriented generation, *e.g.*, technical drawing, Web page layout design, output layout reconfiguration). On the other side, we are also interested in studying human creativity and computationally realize and test creativity criteria and creative processes in visual composition and design.

2 COMPUTATIONAL CREATIVITY

Creativity is an important issue we want to address and this is also because one of our goals is to help to understand creativity. Can we make the computer (a machine) to be creative or, at least, to emulate human creativity? This is seems to be the main goal in the area of computational creativity. But other questions are raised too. What is creativity? And how can we define it? And where does it com from? What is a creative idea, or a creative artifact? Is it just a new one, or a surprising new one, or a valuable new one? This subject seems to be two sided: one point of view is the study and understanding of human creativity, the other is to produce machine creativity and make computer to be, or at least to appear to be, creative.

We can view creativity through two perspectives: the combination-transformation perspective and the personal-human perspective (Boden, 2004). The former has to do with producing new ideas simply by combining old ones - “improbabilistic” creativity –, the latter consists in transforming completely the manner in which the new ideas are produced (more concretely the style used) - “impossibilistic” creativity. In the former case, a new idea comes from a new combination of familiar ideas (specifically those combinations that we value,

A. M. dos Reis J. (2009).
GEOWIN - A System for Creative Pattern Generation based on Rules.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 289-296
DOI: 10.5220/0002240402890296
Copyright © SciTePress

because they are highly improbable or because of some other kind of criterion). In the latter case a new idea comes from a radically new generation process. The personal-human perspective, has to do with whether a new idea is new within the personal context – personal, psychological, or P-creativity – or within the whole human history context – human, historic, or H-creativity. In the first case, an idea is P-creative if the person in whose mind it occurred never had that idea before. In the other case, not only the idea had never occurred before in the mind of the person but also no one had ever had that idea.

Two different additional questions, but practical ones, about computational creativity concern to the creativity criteria to evaluate new ideas and to the generative process to produce the new idea (Ritchie, 2001).

3 SHAPE GRAMMARS

How does a creative agent, human or artificial, build a composition step by step. That is to say, in each state of the composition, how does the agent choose, among the myriad of possible options, to proceed? The generative processes of the system we proposed in previously published papers, see (Reis, 2006a), (Reis, 2006b), (Reis, 2006c), (Reis, 2006d), (Reis, 2008a) and (Reis, 2008b), are based on the shape grammar formalism. Concerning the creativity criteria, we assume it will be provided by the user as she/he can intervene in the generative process.

Shape grammars were introduced in the 1970s by Stiny and Gips (Stiny, 1972). They are similar in principles to the grammars used in the area of Natural Language Understanding and Generation of Artificial Intelligence, with the difference of being based not on symbols, but on shapes (points, lines, two-dimensional and three-dimensional geometric shapes) as well as, by extension, also other parameters like dimensions, colors, *etc.*.

A vocabulary of basic shapes, an initial shape and a finite set of rules that specify how shapes can be generated from other, preexistent, shapes are the components of a shape grammar. These are an analogue, respectively, of the lexicon, of the initial symbol and of the grammar rules of a language in a natural language processing system. The rules of a shape grammar specify how, in a composition in progress, shapes existing in the composition can trigger the addition of new shapes. Each rule has a left side, pre-condition, or antecedent, and a right side, action, or consequent. The left side specifies the pattern for which the rule is applicable and the right side the respective pattern to add. Briefly, a rule is applicable if there is a similarity transformation (*i.e.*, an isometric or a scale) leading to a match of the shape of the left side of the rule with a shape existing in the composition. Rules are applied in a forward manner (from antecedent to consequent), like in the production/rule-based forward-chaining expert systems of Artificial Intelligence, which perform a kind of forward inference. When applied, a rule adds the shapes in its right side.

Shape grammars have been used in different design problems, in the context of synthesis (generation) and analysis (interpretation) of visual compositions and also as means to the description and the representation of styles, including for didactic purposes and also other specific applications, for instance in architectural drawings (Gips, 1999), (Tapia, 1999), (Knight, 2000), (Mitchell, 1990). A style is a way of someone doing something (Simon, 1971) and shows up when that someone chooses an alternative or a process for generating a solution. In the field of design, a style is a kind of design knowledge which is a characteristic of a product, or a set of products, of design, and is recognizable through the presence of some visual elements like shape, color, relative position, texture, dimension, orientation (Dondis, 1973), (Bonsiepe, 1983), (Wong, 1993) as well as certain ways of combining those elements. Visual compositions can be generated automatically according to specific styles. Each style can be implemented by a set of rules of a specific shape grammar. Additional information and theory about shape grammars can be found in (Shape Grammars, 2006), where lists of paper references, people and projects are included too. In (Knight, 2000) we can also find the history of applications of shape grammars in Architecture and Arts and a discussion about the roles of shape grammars in education and practice, as well as new and ongoing issues. In (Chau, et.al., 2004) we can find a survey of shape grammar implementations. Work on the algebras of shapes and issues of shape representation and recognition can be found in (Chase, 1996), (Krishnamurti, 1992) and (McCormack and Cagan, 2002). (Gross, 1996) and (Gross, 1991) are examples of work involving application to CAD and constraint based design. Other papers on creativity modeling, computational creativity and the relationship between art and technology can be found in (Gero and Maher, 1993) and (Candy and Edmonds, 2002).

4 GEOWIN

GEOWIN, a prototype program of the system described in the papers referred is being developed in the programming language Common Lisp (Steele, 1990). We choose this programming language because its exploratory and flexible object oriented programming model and for allowing the production of reusable and extensible software. GEOWIN presently includes the FC (Forward-Chaining), the GEO (GEOmetry) and the SG (Shape-Grammar) modules of Lisp code. Agents are not yet implemented.

```
(defun forward-chain ()
  (do ((newfacts nil) (results t))
      ((null results) newfacts)
    (setf results nil)
    (dolist (rule (kb-rules)
         (setf newfacts
            (append results newfacts)))
      (setf results
        (append (use-rule rule)
                results)))))
```

Figure 1: The `forward-chain` function.

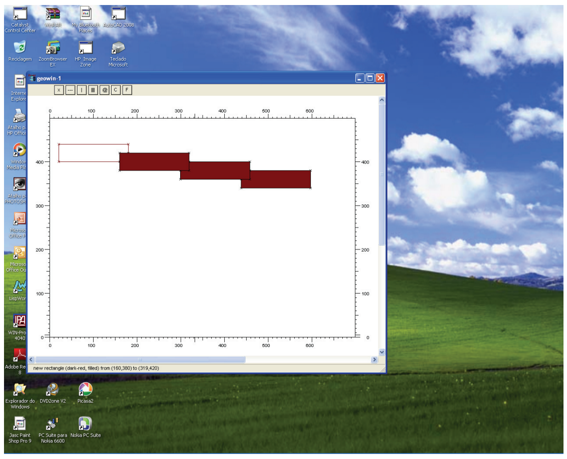
A forward-chaining inference engine to drive shape grammar rule application is implemented in the FC module, which includes a small language to express rules and facts. Rules are stored in the rule memory part and facts are stored in the working memory part of a knowledge base. In Figure 1. we show the code for the function `forward-chain`, the one that contains the main cycle of the FC module. The function `kb-rules` returns a list of applicable rules in the rule memory and the function `use-rule` is then used to apply each of those rules in each iteration of the `dolist` cycle, to generate new facts. This is repeated (by means of a `do` cycle) until no more new facts can be generated. As an alternative to `forward-chain` the function `use-rule` can also be used for selective and incremental rule application.

```
  (r1 "Horizontal rectangle rule."
 (and (rectangle ?x1 ?y1 ?x2 ?y2
                 ?color ?filledp)
      (is ?length (- ?x2 ?x1))
      (is ?height (- ?y2 ?y1))
      (rh-hmin ?hmin) (rh-hmax ?hmax)
      (rh-ratio-min ?rmin)
      (rh-ratio-max ?rmax)
      (<= ?hmin ?height ?hmax)
      (<= ?rmin
          (/ ?length ?height) ?rmax)
      (rh-delta ?delta)
      (is ?d (eval (floor
              (/ ?height ?delta))))
      (is ?x1a (- ?x2 ?d))
      (is ?x2a (+ ?x1a ?length))
      (is ?y1a (- ?y1 (- ?height ?d)))
      (is ?y2a (+ ?y1a ?height))
      (x-min ?xmin) (x-max ?xmax)
      (y-min ?ymin) (y-max ?ymax)
      (<= ?xmin ?x1a ?x2a ?xmax)
      (<= ?ymin ?y1a ?y2a ?ymax))
 ->
 (rectangle ?x1a ?y1a ?x2a ?y2a
                         dark-red t))
```

Figure 2: An example of a shape grammar forward-chaining rule, rule `r1`.

An example of a rule is shown in Figure 2.. The body of a rule has the antecedent and the consequent section, before and after the symbol `->`, respectively besides the rule identifier, `r1`, and a documentation string, as we can see. Facts are expressed as patterns which are represented by Lisp lists. When a rule is applied, each fact pattern, possibly containing rule variables, in the antecedent part of the rule is "pattern-matched" with facts in the working memory part of the knowledge base. A successful pattern-matching is expected to instantiate existing rule variables in one or more ways. When appearing in the consequent part a pattern means a new fact to be added to the working memory in case the pattern-matching process of the fact patterns in the antecedent part of the rule was successful (any variables in the fact pattern should have been then instatiated).

When applied once the rule `r1` shown identifies a rectangle fact and, apart from a number of constraint fact verifications (to exclude rectangles that are not "horizontal" or have proportions outside some intended limits and to avoid generation outside the limits of the drawing area) generates a new rectangle fact equal to the first one but lightly translated.



new rectangle (dark-red, filled) from (160,380) to (319,420)

Figure 3: Rule `r1` ("horizontal" rectangle) generation example. Generated shapes are solid, the start shape is the clear rectangle.

In Figure 3. we show the result of some successive aplications of rule `r1`, given an initial rectangle shape.

```
(r3 "Square rule."
  (and (rectangle ?x1 ?y1 ?x2 ?y2
                      ?color ?filledp)
       (is ?a (- ?x2 ?x1))
       (is ?height (- ?y2 ?y1))
       (= ?a ?height) (rq-amin ?amin)
       (>= ?a ?amin) (rq-delta ?delta)
       (is ?d (eval (floor
                      (/ ?a ?delta))))
       (is ?x1a (+ ?x1 ?d))
       (is ?x2a ?x2)
       (is ?y1a (- ?y1 ?a))
       (is ?y2a (- ?y2 (+ ?a ?d)))
       (x-min ?xmin) (x-max ?xmax)
       (y-min ?ymin) (y-max ?ymax)
       (<= ?xmin ?x1a ?x2a ?xmax)
       (<= ?ymin ?y1a ?y2a ?ymax))
  ->
  (rectangle ?x1a ?y1a ?x2a ?y2a
                           green t))
```

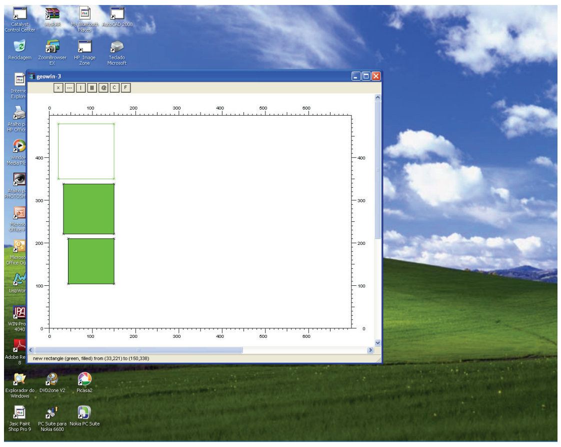
Figure 4. Another example of a shape grammar forward-chaining rule, rule `r3`.

In Figure 4. and Figure 5., we show additional rule examples, respectively rule `r3`, which, in the presence of a square, adds another one, scaled down and translated, and rule `r4`, which, in the presence of a two shape pattern (a "horizontal" rectangle and a square, lightly superinposed) adds a small circle.

```
    (r4  "Horizontal  rectangle,  square
and
     circle rule."
   (and (rectangle ?xr1 ?yr1 ?xr2 ?yr2
                         ?color1 ?fillp1)
        (is ?length (- ?xr2 ?xr1))
        (is ?height (- ?yr2 ?yr1))
        (rh-hmin ?hmin) (rh-hmax ?hmax)
        (rh-ratio-min ?rmin)
        (rh-ratio-max ?rmax)
        (<= ?hmin ?height ?hmax)
        (<= ?rmin (/ ?length ?height)
            ?rmax)
        (rectangle ?xq1 ?yq1 ?xq2 ?yq2
                          ?color2 ?fillp2)
        (is ?a (- ?xq2 ?xq1))
        (is ?aheight (- ?yq2 ?yq1))
        (= ?a ?aheight) (rq-amin ?amin)
        (>= ?a ?amin)
        (is ?delta (- ?yq2 ?yr1))
        (> ?height ?delta 0)
        (> ?xr2 ?xq1 ?xr1)
        (is ?r (eval (floor (/
         (- ?height (* 3 ?delta)) 2))))
        (> ?r 0)
        (is ?xc (+ ?xr1 ?delta ?r))
        (is ?yc (+ ?yr1 (* 2 ?delta)
                  ?r)))
   ->
   (circle ?xc ?yc ?r dark-blue t))
```

Figure 5. Another example of a shape grammar forward-chaining rule, rule `r4`.

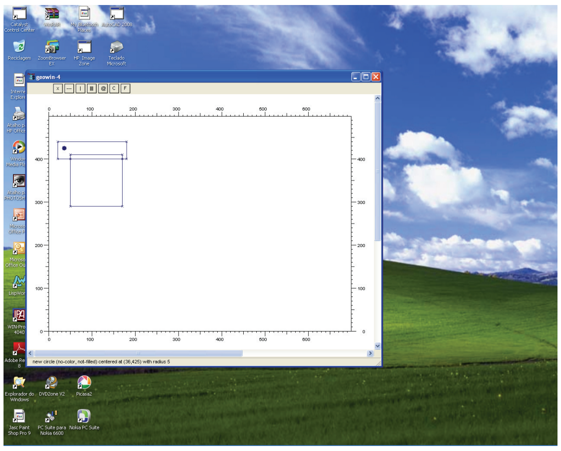
Rules `r3` and `r4` have their results exemplified in Figure 6. and Figure 7., respectively.



new rectangle (green, filled) from (33,221) to (150,338)

Figure 6: Rule `r3` (scaled square) generation example. Generated shapes are solid, the start shape is the clear rectangle.

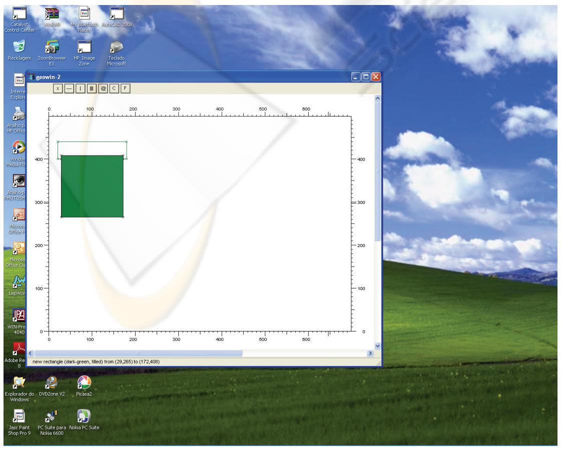
Additional examples of generation with rules for which, for reasons of space, we don't show code, are presented in Figure 8. and Figure 9.



new circle (no-color, not-filled) centered at (36,425) with radius 5

Figure 7: Rule `r4` ("horizontal" rectangle, square and circle) generation example. Generated shapes are solid, the start shape is the clear rectangle.

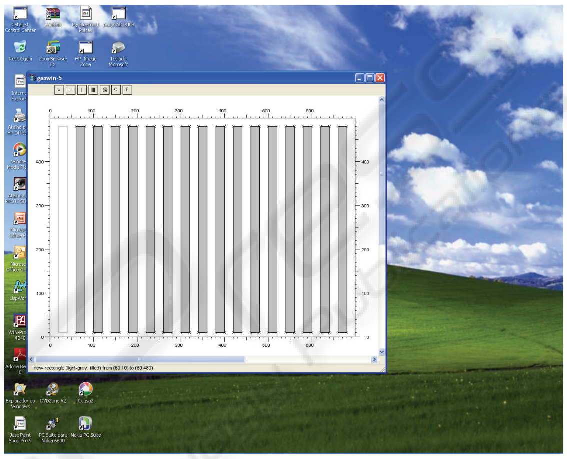
A fact is composed of the name of the predicate and the arguments, possibly variables, as in `(rectangle ?x1 ?y1 ?x2 ?y2 ?color ?filledp)`. Symbols starting with a `?` character are rule variables. Apart from facts, the rule language has special predefined operators, some of which can be seen in the example. These operators include the logical operators `and`, `or` and `not` (to logically combine facts in the antecedent part of the rule), arithmetic operators (`+`, `-`, `*` and `/`) and comparators (`=`, `/=`, `>`, `>=`, `<` and `<=`), the operator `is` for assignment (similar in nature to the `is` operator in the Prolog programming language) and an `eval` special operator to evaluate arbitrary Lisp expressions, possibly containing rule variables.



new rectangle (dark-green, filled) from (29,265) to (172,408)

Figure 8: Additional rule generation example.

Graphical shape representation and the graphical environment and its interactive and programmatic interfaces is implemented in the GEO module. Geometric shapes are implemented through objects of Common Lisp. The most general class is the `shape` class, and there are classes for points, line segments, rectangles and circles, all subclasses of `shape`. In Figure 10. we show segments of code including the definition of classes `shape` and `circle`.



new rectangle (light-gray, filled) from (60,10) to (80,480)

Figure 9: Another aditional rule generation example.

A user can create and introduce new points, segments, rectangles and circles that appear in a drawing area through the interactive interface. The programmatic interface allows the same through calls to the appropriate Lisp methods. In the near future the interactive interface will be expanded to allow also the creation new shape grammar rules interactively. In Figure 3., Figure 6. and Figure 14.., the window of the drawing area of the interface is shown.

```
(defclass shape ()
((color :reader color :initarg :color
                      :initform nil)))
         :
(defclass circle (shape)
  ((center :reader center
      :initarg :center :type point)
   (radius :reader radius
      :initarg :radius :type integer)
   (id :reader id
         :initform (circle-id-calc))
   (filledp :reader filledp
      :initarg :filledp :initform nil))
  (:documentation "Circle shape."))
```

Figure 10: Some class definitions of the GEO module.

An ontology of the geometric two-dimensional domain, some relations of which were implemented in the GEO module, was also defined. This ontology includes geometric points and relative position relations between geometric points along the x and the y axes, line segments and relative position relations between line segments along the x and the y axes, and rectangular shapes and relative position relations between rectangular shapes along the x and y axes on the plane.

```
(defmethod x-before ((s1 shape)
                     (s2 shape))
  (< (x-max s1) (x-min s2)))

(defmethod x-after ((s1 shape)
                    (s2 shape))
  (< (x-max s2) (x-min s1)))

(defmethod x-meets ((s1 shape)
                    (s2 shape))
  (= (x-max s1) (x-min s2)))
          :
(defmethod y-equals ((s1 shape)
                     (s2 shape))
  (and (= (y-min s1) (y-min s2))
       (= (y-max s1) (y-max s2))))
```

Figure 11: Some of the methods for relative position relations between shapes of the GEO ontology.

The ontology is described in the published papers, and is reflected in the GEO module in Lisp methods that implement the relations between shapes, as shown in example segments of code in Figure 11. (functions `x-max`, `x-min`, `y-max`, and `y-min` produce the extreme `x` and `y` coordinate values of a given shape).

Lastly, the SG module integrates the work done by the two others. Essentially, the functions contained in the SG module allow top level introduction of new shapes in a composition, introduction of new shape grammar rules and to start the generation of a composition. Part of the code for this functionality is shown in Figure 12.. Usually, a set of parameters are previously established, by means of facts added to the knowledge base, in order to allow some rules to be aplicable. Also, some shape grammar rules are added, as well as one or more start shapes. Finaly, a generation can be run. This is all accomplished through calls to SG functions.

The entire process is exemplified in Figure 13. for rule `r1` (shown in Figure 2.), including adding a rectangle as a start shape. Figure 3. shows the result of the corresponding generation with rule `r1`. A more elaborated generation example is shown in Figure 14., involving five shape grammar rules (including `r1`, `r3`, `r4` and the rules exemplified in Figure 8. and Figure9.) and two start shapes.

```
(defun sg-add-shape (fact &aux
                    newfact newshape)
  (and *geowin*
       (setf newfact
            (sg-add-fact fact))
       (setf newshape
             (sg-add-shape-aux fact)))
  (values newfact newshape))

(defun sg-add-rule (rule)
  (assert-rule rule))

(defun sg-gen (&aux newfacts newshapes
                   tmpshp)
  (and *geowin*
       (setf newfacts (forward-chain))
       (setf newshapes
         (mapcan
   #'(lambda (fact)
       (and (setf tmpshp
              (sg-add-shape-aux fact))
            (list tmpshp)))
   newfacts)))
  (values newfacts newshapes))
```

Figure 12: Some of the functions of the SG module.

5 FUTURE WORK

Generically, future work will improve and refine the GEWIN system, and will use it as a tool to realize and test creativity criteria and creative processes. In specific, we can point at some future work tasks needed to be done.

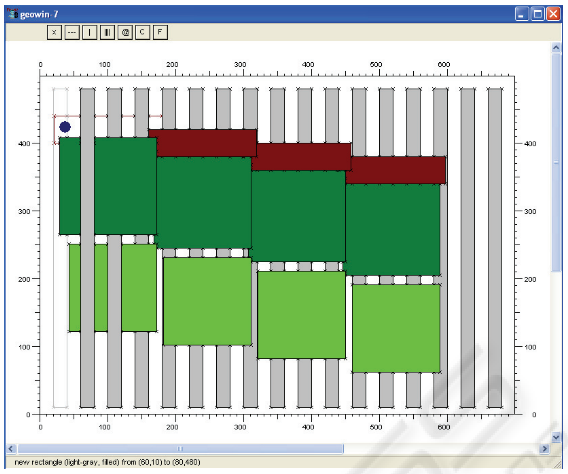
One of those tasks respects to improvement of the FC module. This can involve to improve and optimize the pattern matching process and the other forward-chaining functionalities, *e.g.*, by using the RETE algorithm, see (Forgy, 1982) or (Doorenbos, 1995), for instance.

The system will also evolve to a multi-agent one after we have implemented agents. Each agent will essentially be a forward-chaining subsystem with its own knowledge base (some adjustments may be necessary in the FC code for this). One of the most significant aspects to give special attention to are the control mechanisms for rule application, which can be viewed in the intra-agent and in the inter-agent perspectives. The latter involves the coordination of the activity of the agents in the multi-agent system.

```
(sg-ini)           ; initialize system
;; graphic area limits:
(sg-add-fact '(x-min 0))
(sg-add-fact '(x-max 700))
(sg-add-fact '(y-min 0))
(sg-add-fact '(y-max 500))
;; matching constraints:
(sg-add-fact '(rh-hmin 20))
(sg-add-fact '(rh-hmax 50))
(sg-add-fact '(rh-ratio-min 2))
(sg-add-fact '(rh-ratio-max 5))
;; generation parameters:
(sg-add-fact '(rh-delta 2))
;; shape grammar rule:
(sg-add-rule
 '(r1 "Horizontal rectangle rule."
    (and (rectangle ?x1 ?y1 ?x2 ?y2
                        ?color ?filledp)
         (is ?length (- ?x2 ?x1))
         (is ?height (- ?y2 ?y1))
         (rh-hmin ?hmin)
         (rh-hmax ?hmax)
         (rh-ratio-min ?rmin)
         (rh-ratio-max ?rmax)
         (<= ?hmin ?height ?hmax)
         (<= ?rmin (/ ?length ?height)
             ?rmax)
         (rh-delta ?delta)
         (is ?d (eval (floor
                 (/ ?height ?delta))))
         (is ?x1a (- ?x2 ?d))
         (is ?x2a (+ ?x1a ?length))
         (is ?y1a (- ?y1
                     (- ?height ?d)))
         (is ?y2a (+ ?y1a ?height))
         (x-min ?xmin) (x-max ?xmax)
         (y-min ?ymin) (y-max ?ymax)
         (<= ?xmin ?x1a ?x2a ?xmax)
         (<= ?ymin ?y1a ?y2a ?ymax))
    ->
    (rectangle ?x1a ?y1a ?x2a ?y2a
                           dark-red t)))
;; start shape: "horizontal" rectangle:
(sg-add-shape '(rectangle 21 400 180
                         440 dark-red nil))
(sg-gen)           ; start generation
```

Figure 13: Preparing and running a generation with rule `r1` using the programatic interface of the SG module.

Some work needs also to be done in the GEO module involving some computational geometry know-how. This will involve to expand the number of geometric shape types the system can represent and to have the possibility to edit and change preexistent shape instances in the drawing area, including by application of different geometric transformations.



new rectangle (light-gray, filled) from (60,10) to (80,480)

Figure 14: Example of a composition involving several rules. Generated shapes are solid, the start shapes are clear.

The geometric shape manipulation referred above, more specifically similarity transformations, will be useful particularly for the recognition mechanism of shape patterns in the antecedent part of shape grammar rules. This mechanism is presently limited to recognize a shape indicated almost in a literal manner in the rule, with limited provision to specify scale transformations (but at the cost of using some extra programming in the language of rules, as we could see by the example rules shown before).

Still another task to be done at the level of GEO is to program the interactive creation of new shape grammar rules using the graphical interface.

6 CONCLUSIONS

In this paper we have shown work done and in progress in developing a proposed computational system based on rules for generating patterns for visual composition. The system makes use of the shape grammar concept, *i.e.*, the rules can specify geometric shape manipulations to produce new shapes, given some initial one(s). Shape grammars can be used to emulate styles of visual composition which is an interesting idea that can be applied to exploratory design. In the future we plan to evolve the system to a multi-agent system to support visual composition synthesis activities, with each intervening agent giving its creative contribution through a style of its own, and use the system as a tool to our investigation in realizing and testing creativity criteria and creative processes.

ACKNOWLEDGEMENTS

Research presented in this paper was partially supported by FCT (the Portuguese Foundation for Science and Technology).

REFERENCES

Boden, M.A., 2004, The Creative Mind, Myths and Mechanisms (2nd.ed.), Routledge.

Bonsiepe, G., 1983, Teoria e Pratica del Disegno Industriale, Elementi per una Manualistica Critica, Giangiacomo Feltrinelli Editore, Milano, Italy.

Candy, Linda, and Edmonds, Ernest (eds), 2002, Explorations in Art and Technology, Springer-Verlag.

Chase, Scott C., 1996, Design Modeling with Shape Algebras and Formal Logic, ACADIA '96, Tucson, AZ, 31 October-3 November, 1996.

Chau, H. H., Chen, X., McKay, A., Pennington, Alan de, 2004, Evaluation of a 3D Shape Grammar Implementation, First International Conference on Design Computing and Cognition (DCC'04), J.S. Gero (ed), Kluwer Academic Publishers, 2004.

Dondis, D. A., 1973, A Primer of Visual Literacy, The Massachusetts Institute of Technology.

Doorenbos, R. B., 1995, Production Matching for Large Learning Systems, PhD Thesis, Carnegie Mellon University.

Forgy, C., 1982, RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19, pp 17-37.

Gero, John S., and Maher, Mary Lou (eds), 1993, Modeling Creativity and Knowledge-Based Creative Design, Lawrence Erlbaum Associates, Inc..

Gips, J., 1999, Computer Implementation of Shape Grammars, Workshop on Shape Computation, MIT, in http://www.shapegrammar.org/ implement.pdf.

Gross, M.D., 1996, Elements that Follow your Rules: Constraint Based CAD Layout, ACADIA ‘96 Tuscon, AZ.

Gross, M.D., 1991, Grids in Design and CAD, ACADIA '91, Los Angeles, CA pp. 33-43.

Knight, T., 2000, Shape Grammars in Education and Practice: History and Prospects, Dept. of Architecture, MIT, in http://www.mit.edu/~tknight/IJDC/.

Krishnamurti, R., 1992, The Maximal Representation of a Shape, Environment and Planning B: Planning and Design, volume 19, pages 267-288.

McCormack, J.P., Cagan, J., 2002, Supporting Designer's Hierarchies through Parametric Shape Recognition, Environment and Planning B: Planning and Design, volume 29, pages 913-931.

Mitchell, W. J., 1990, The Logic of Architecture, The MIT Press.

Reis, J., 2008a, Using Rules for Creativity in Visual Composition, Procs. of the SIGDOC 2008, Lisbon, Portugal, 22-24 Sept. 2008, 207-214.

Reis, J., 2008b, A Rule Language to Express Visual Pattern Generation, (*poster*), Proc. of the SIGDOC 2008, Lisbon, Portugal, 22-24 Sept 2008, 273-274.

Reis, J., 2006a, Agentes com Estilo próprio, Composição Visual Multi-Agente com Gramáticas de Forma, CISTI 2006, 1st Iberic Conference on Information Systems and Technologies, 21-23 June 2006, Ofir, Portugal, Maria Manuela Cunha, Álvaro Rocha (eds.), vol.II, 327-339 (in portuguese).

Reis, J., 2006b, Agents with Style – Multi-Agent Visual Composition with Shape Grammars, Procs. of the 3rd Joint Workshop on Computational Creativity, August 28-29, 2006, Riva del Garda, Italy, Simon Colton, Alison Pease (eds.), pp. 61-62.

Reis, J., 2006c, Agents, Grammars and Style: Multi-Agent Visual Composition with Shape Grammars, Procs. of the ICWI 2006 - IADIS International Conference WWW/Internet 2006, October 5-8, 2006, Murcia, Spain, Pedro Isaías, Miguel Baptista Nunes e Immaculada J. Martinez (eds.), Vol. II, pp. 278-282.

Reis, J., 2006d, An Approach to Multi-Agent Visual Composition with Mixed Styles, Procs. of the ICSOFT 2006, Int. Conf. on Software and Data Technologies, Sept, 11-14, 2006, Setúbal, Portugal, vol.1, 357-362.

Ritchie, G., 2001, Assessing Creativity, Research Report EDI-INF-RR-0039, University of Edinburgh.

Shape Grammars, 2006, http://www.shapegrammar.org/.

Simon, H.A., 1971, Style in Design, Proceedings of the Second Annual Environmental Design Research Association Conference, 1-10.

Steele, Guy L., 1990, Common Lisp, the Language, Digital Press/Butterworth-Heinemann.

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

Tapia, M., 1999, A Visual Implementation of a Shape Grammar System, Environment and Planning B: Planning and Design, vol.26, pp.59-73.

Wong, W., 1993, Principles of Form and Design, John Wiley & Sons, Inc..