Editing Shapes in a Prototype Two- and Three-dimensional Shape Grammar Environment

Andrew I-kang Li[1], Liang Chen[2], Yang Wang[3], Hau Hing Chau[4]
[1,2]School of Architecture, The Chinese University of Hong Kong,
[3]Department of Architecture, National University of Singapore,
[4]School of Mechanical Engineering, University of Leeds
[1]andrewli@alum.mit.edu, [2]chenliang@cuhk.edu.hk, [3]akiwangy@nus.edu.sg,
[4]H.H.Chau@leeds.ac.uk

Abstract: *Recently we developed a prototype general shape grammar system, called Grammar Environment (Li et al. 2009). It differs from other systems in that it aims to support designers who design with shape grammars. One task of such a system is to support users in editing shapes. The guidelines that we followed in developing Grammar Environment suggested that the shape editing system should both be integrated into the system and be powerful as a drawing tool. This seemed to be contradictory. We decided to make two shape editors: one stronger on integration, the other on drawing power.*

Keywords: *Shape grammars; shape grammar interpreter; shape grammar environment.*

Introduction

Recently we developed a prototype general system to support designers who design with grammars (Li et al. 2009). This system, called Grammar Environment, supports lines and labeled points in 2- and 3-space (i.e., U_{12}, U_{13}, V_{02}, V_{03}). In designing and constructing the system, we proposed and followed two general guidelines.

First, the system should be specific to the domain of design. That is, users should be able to see shapes and rules as graphical objects and to manipulate those graphical objects directly.

Second, the system should support the edit-run cycle. That is, users should be able to switch easily between editing and running the grammar. These guidelines may seem unsurprising, but they have not previously been articulated for shape grammar systems.

One of the tasks of a shape grammar system is to support users in editing shapes; how we did this in Grammar Environment is the subject of this paper. We begin by considering the shape editing capabilities of these three representative shape grammar systems (for a more complete list of existing systems, see Li et al. 2009):

1. GEdit by Tapia (1999). This is a general 2D system; i.e., it supports all shapes that are legal under the given set of basic elements (in this case, lines and labeled points). It has a generally graphical interface.
2. SGS by Chau et al. (2004) This is a general 3D system, but has only a partially graphical interface.
3. Shaper2D by McGill (2002). This is a limited 2D

	GEdit General	SGS General	Shaper2D Limited
Generality	++	++	+
Graphical display	++	+	++
Direct manipulation	+		++
Edit of shapes		+	+
Easy switching	+	+	++

Table1
The main characteristics of shape editing support in three representative existing shape grammar systems. "++" indicates complete support; "+", partial support; a blank, no support

system; shapes are restricted to a few types of triangles and rectangles with labeled points. It has a highly graphical interface.

From the general guidelines above, we derived more specific guidelines for shape editing systems as follows. First, rules should be displayed graphically, and their basic elements should be directly manipulable. Users should be able to draw; they should not have to deal with nongraphical representations such as sets of coordinates. With GEdit, users draw with a drawing program. With Shaper2D, users manipulate the triangles and rectangles directly. With SGS, users type coordinates into a text file. (This seems to be the case with most general systems, with the notable exception of GEdit.)

Second, the editing system should be general. That is, given a (finite) set of basic elements, usually lines and labeled points, users should be able to create any shape that is a finite set of those elements. GEdit and SGS are general; Shaper2D supports only triangles and rectangles with labeled points.

Third, shapes already created should be editable. Shaper2D supports editable shapes; indeed, it is perhaps the main point of the application. SGS supports editable text files. GEdit does not support editable shapes. Once created, shapes cannot be altered; they can only be created anew.

Finally, users should be able to switch easily between editing and running the grammar. When they modify shapes, they should be able to test the effect easily. With Shaper2D, users manipulate shapes directly with real-time feedback. With SGS, users can switch, albeit it between two applications (the system and a text editor). With GEdit, the question is moot, since users cannot edit shapes and rules.

The main characteristics of these three systems are summarized in table 1. GEdit is notable for its generality, graphic display, and direct manipulation, features which likely account for its perceived user-friendliness. However, once saved, shapes cannot be edited. SGS is general, but its editing is text-based, not graphical. Shaper2D is impressive in many ways and has much to teach us, even though it is not general.

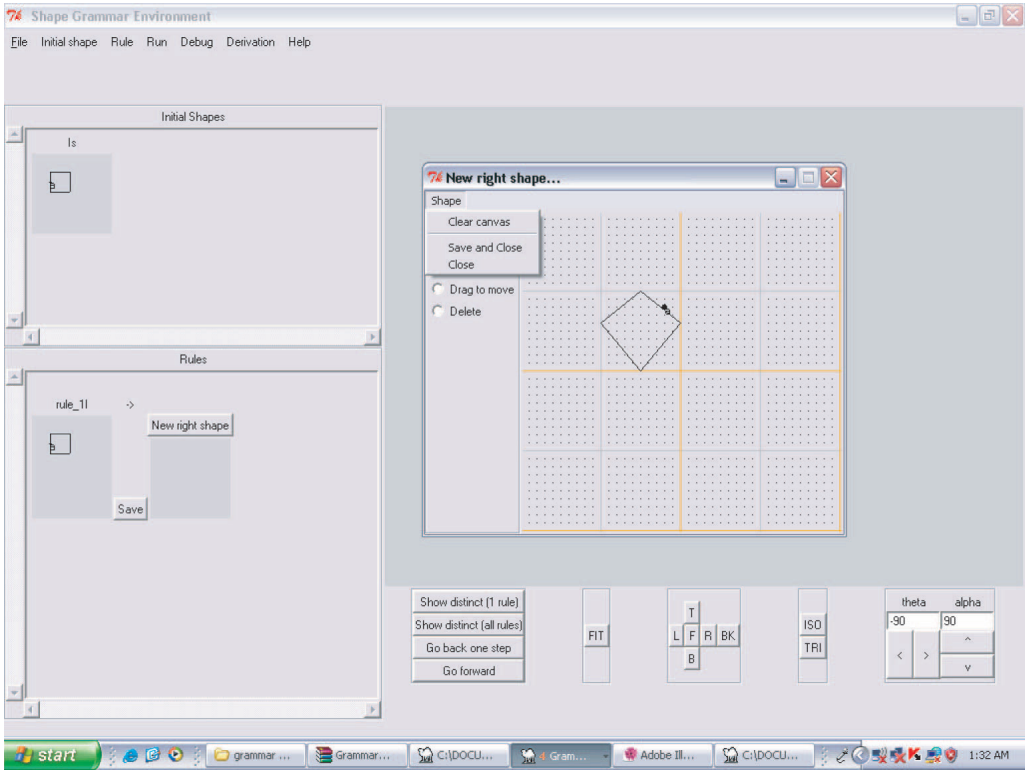
No existing shape editing system satisfies all our guidelines, and so no existing grammar system supports design work as we proposed recently. The solution would appear to be a single shape editing system that satisfies all the guidelines. However, in practical terms, this poses a dilemma.

On the one hand, ease of editing and direct manipulation suggest that the shape editor should support users in drawing as they normally do. But users are accustomed to sophisticated drawing applications that we have neither the time, the ability, nor the interest to replicate.

On the other hand, easy switching between editing and running suggests that the shape editor should be an integral part of the shape grammar system. But then, any shape editor that we could create would be too primitive for an experienced designer.

Since our goal is to make a prototype, we have simply made two shape editors. One is rather primitive as a drawing tool, but is built into Grammar Environment, making an integrated system for editing and running. This internal editor maximizes ease of switching at the cost of limiting drawing. The other editor is an Autocad applet. This external editor

Figure 1
The internal shape editor is in a pop-up window. The lists of rules and of initial shapes can be seen in the main window



makes it easy to draw precise and complex shapes, but at the cost of forcing users to switch between applications.

Internal editor

When users select a command to create or edit a shape, the internal shape editor appears in a pop-up window (figure 1). If the shape is new, the canvas is blank; if the shape already exists, it is displayed on the canvas. There are just a few commands: line, rectangle, labeled point, drag, and delete. The basic elements are lines and labeled points in 2-space, i.e., U_{12}, V_{02}. Users can create any 2D shape that is composed of these basic elements and that fits onto the canvas. When finished, they name the shape and close the window. The new or revised shape appears in the list of rules or initial shapes, and the system is ready for both running and continued editing.

External editor

As has been mentioned, the external shape editor is an applet in Autocad (versions 2007 and 2008). As a consequence, moving rules and initial shapes between Grammar Environment and the external editor takes a few more steps than with the internal editor. To move rules and initial shapes in this direction, users export them as files (with rul and is suffixes, respectively), switch to Autocad, launch the applet if necessary, and open one file at a time for editing.

To move in the other direction, there are two

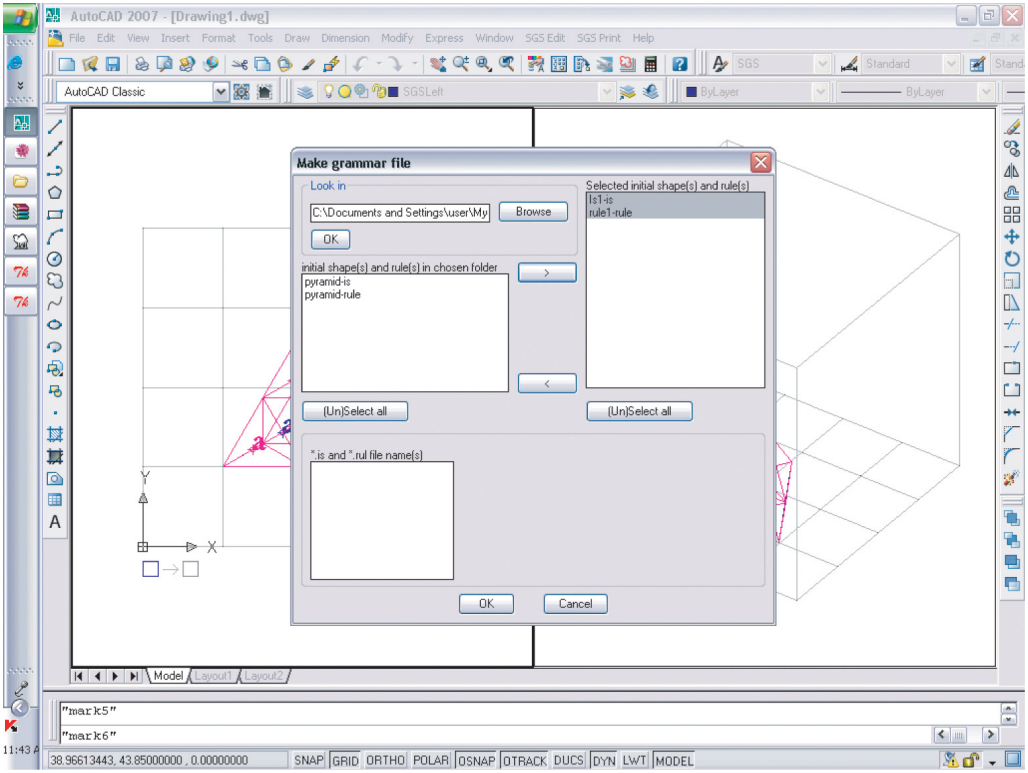


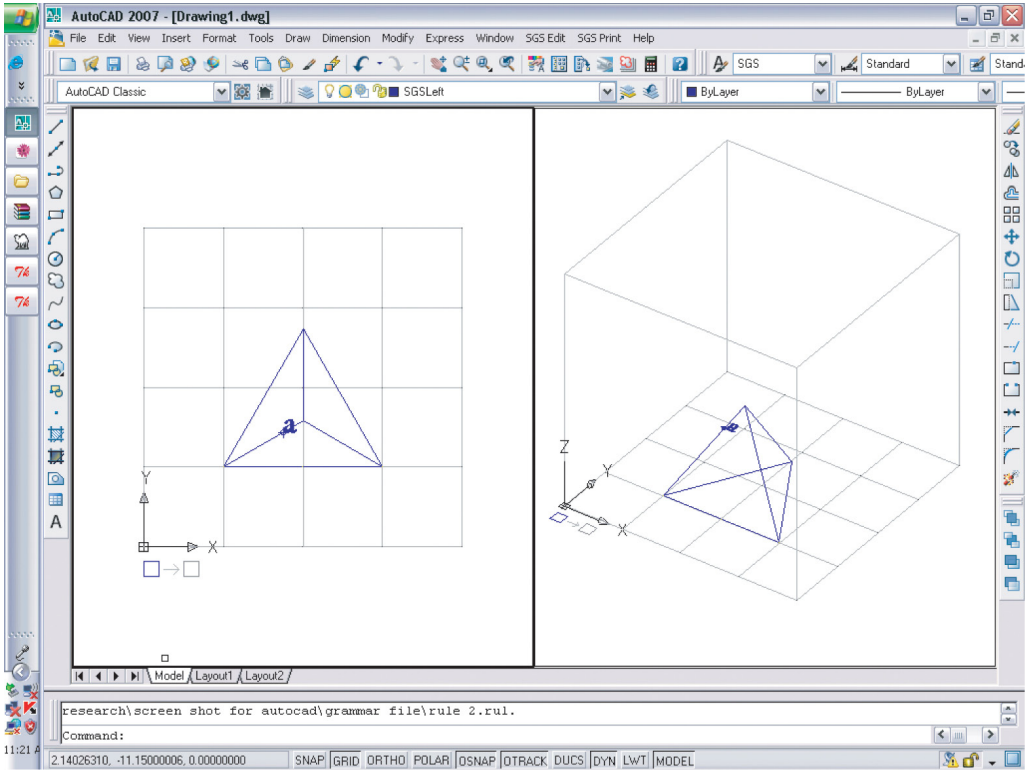
Figure 2
The dialog box of the external shape editor's dat file-maker routine, which allows users to assemble rules and initial shape files into a new grammar

pathways. One is to save the files, switch to Grammar Environment, open an existing grammar if necessary, and import them into that grammar. The other is to save the files, assemble those files into a new grammar (with a dat suffix) using the applet's dat file-maker routine (figure 2), switch to Grammar Environment, and open the new grammar. Rules and initial shapes created in the Autocad applet are identical to those created in the internal editor. However, 3D shapes cannot be edited in the internal editor.

The applet has two modes: edit and print. In edit mode, it displays a cube (figure 3) inside which users can draw and edit shapes, using as basic elements lines and labeled points in 3-space, i.e., U_{13}, V_{03}. Left and right shapes are drawn in different layers, somewhat similarly to GEdit.

In print mode, users can obtain several types of drawings (in dwg or dxf format) of their work. One shows the grammar: its rules and shapes, along with the names assigned by the users. A second shows the derivation of a final shape: the grammar, the shape after each rule application, and the names of the rules applied (figure 4). This is created from a derivation file (with a drv suffix) that users save in Grammar Environment. The third drawing, also created from the derivation file, shows the final shape. All these are useful as hard copies of users' work. The drawing of the final shape is especially useful as a link to downstream processes. For instance, it can be converted to a surface model and sent to rapid prototyping.

Figure 3
The external shape editor is an Autocad applet. Users draw the shapes inside the cube



Discussion

We have done informal testing with users of different ages (high school and up) and with different amounts of experience with shape grammars, with Autocad, and indeed with design. This testing was limited, but we were able to make two observations.

First, novice shape grammar users were comfortable with the internal shape editor. This is probably because their main task was learning how to edit and run shape grammars, and the integrated system helped clarify these tasks. On the other hand, they found the internal editor limiting, because its operations were too low-level and imprecise. For instance, they could not draw an equilateral triangle or easily locate a labeled point with respect to a line.

Second, users familiar with shape grammars usually started with the internal editor. However, they were usually also familiar with Autocad, so inevitably they became frustrated with the internal editor, and moved to the external editor. For these users, switching between Grammar Environment and Autocad seemed less of a problem than we expected. Apparently, they could keep the big operational picture in mind. In particular, they liked being able to draw with the power of Autocad, and to link to downstream processes.

From these observations we can draw some useful conclusions. First, users want to work graphically. The fact that novices found the integrated system generally congenial suggests that our guidelines are appropriate for them. Of course, those who

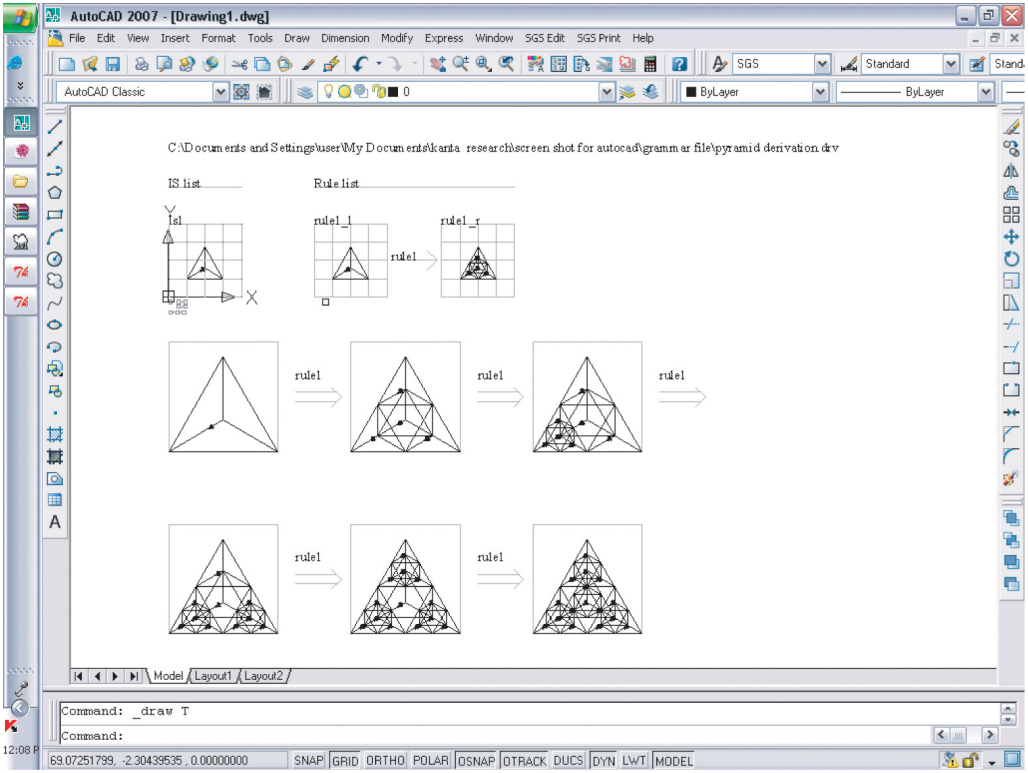


Figure 4
A derivation and grammar as constructed by the external shape editor

sometimes want to enter coordinates can do so in the external editor. But neither novices nor experienced users objected to direct manipulation of graphical objects.

Second, each editor has its own audience and purpose. In particular, the internal editor need not be seen as an underpowered version of the external editor. The internal editor is used by novices, who benefit from working inside a single application. The external editor, on the other hand, offers sophisticated drawing capabilities, but does not obscure the overall operational picture for experienced users.

In other words, users are not a monolithic group. Their domains, objects, and tasks vary with their experience and knowledge. The fact that novices could not easily draw what they wanted to draw suggests that we do not know enough about how they use shape grammars. In particular, instead of supporting direct manipulation of basic elements, we might reexamine the approach of Shaper2D, where users manipulate higher-level objects, i.e., triangles and rectangles. This might help us design a shape editor that is simple for novices. In general, we need to learn more about what types of users there are, and what they need.

Finally, the external shape editor worked better than we expected. We suspect that our general guidelines made the switching between applications less distracting than otherwise might have been the case. This makes the advantages that much more attractive: users can draw with all the power of a sophisticated drafting tool, they can import shapes

from upstream drawing processes, and they can export shapes to downstream processes.

Overall, it seems reasonable to say that our general approach is appropriate, but that some of our assumptions need to be revisited. In particular, we need to know more about what designers do when they work with shape grammars and shape grammar systems. Grammar Environment provides a platform for just this kind of inquiry, even as it supports designers doing design.

Grammar Environment, sample grammars, tutorial videos, and other supporting materials are available at http://www.cuhk.homeip.net/wikisgi/index.php/Main_Page.

Acknowledgements

This work was supported by a Competitive Earmarked Research Grant from the Hong Kong Research Grants Council, which we acknowledge with thanks. Nujaba Binte Kabir tested the systems thoroughly and provided the images.

References

Chau, H. H. Chen, X. J. McKay, A. and de Pennington, A.: 2004, Evaluation of a 3D shape grammar implementation, in J. S. Gero (ed), Design computing and cognition '04, Kluwer, Dordrecht, pp. 357–376.

Li, A. I. Chau, H. H. Chen, L. and Wang, Y.: 2009, A prototype system for developing two- and three-dimensional shape grammars, in T. W. Chang, E. Champion, S. F. Chien and S. C. Chiou (eds), CAADRIA 2009: proceedings of the 14th international conference on computer-aided architecture design research in Asia, Department of Digital Media Design, National Yunlin University of Science and Technology, Douliou, Taiwan, pp. 717–726.

McGill, M. C.: 2002, Shaper2D: visual software for learning shape grammars, in K. Koszewski and S. Wrona (eds), Design e-ducation: connecting the real and the virtual, Proceedings of the 20th Conference on Education in Computer Aided Architectural Design in Europe, eCAADe, Warsaw, pp. 148–151.

Tapia, M.: 1999, A visual implementation of a shape grammar system, Environment & planning B: planning & design, 26, pp. 59–73.