Introduction

Shaper2D is a program that promotes the use of computers for learning about computational design. This program, created by a designer rather than a programmer, was developed to employ an intuitive, visual interface that encourages a „learning by designing“ approach to shape grammar education. Shaper2D is not the first application created for shape grammar exploration. Examining two previous computer implementations of shape grammars situates Shaper2D in its intended context. These software applications for exploring two-dimensional and three-dimensional shape grammars provide powerful tools for learning fundamental concepts, but their interfaces are not conducive to experimenting with shape grammars. While these programs demonstrate simple and accurate representations of the actual shape grammar computation, they do not provide an appropriate interface for designers.

A Programmer’s Approach to Programming

In order to understand the different approach taken with Shaper2D it is important to consider how a programmer develops a program. While the focus on an intuitive interface and relevant functionality may seem an obvious approach to designing an application, this is not always the case. Often the „worse is better“ philosophy, as described by Richard Gabriel (1999), is used in software development.

“…It is more important for the [program’s] implementation to be simple than the interface. Simplicity [of code] is the most important consideration in a design.”

Gabriel, 1999

While not all programmers fall prey to this idea, it is still a common approach. Essentially, „worse is
better“ programming sacrifices the interface and usability for simplicity of code design, that is:

- Simple code is more important than a usable interface, in other words, the programmer’s convenience is more important than the user’s convenience.
- The user is expected to learn to think about the problem the way the programmer thinks about the problem.
- The interface should reflect the internal structures of the program since they are the most fundamental description of the problem.

While these rules often result in maintainable and reliable programs, they do not take usability into consideration. This approach is borne out of convenience and often results in powerful but complicated programs, frequently with only a token interface as an acknowledgment for the end user. In other words, the outcome is a program with enormous functionality but which is difficult to use and has an unintuitive interface.

When a user has difficulty using a program, it is typical that they blame themselves rather than look to the inadequacies of the software they are using. This should not be the case.

A Designer’s Approach to Programming

Interface
From the outset, Shaper2D was designed with the end user in mind. Rather than programming an application and then tacking on an interface at the end, developing a dynamic, intuitive, and simple interface was the driving force behind the program design. Three rules were steadfastly adhered to:

- Never choose to sacrifice the usability of the program for the sake of making the code simpler.
- If a help manual is required for learning to use the program then the interface is flawed.
- Remember that the end user is a designer, not a programmer, so the program should be intuitive, visual, and responsive.

The Shaper2D interface, developed by a designer, attempts to mirror the way the user understands the problem without consideration for the ease of the programmer.

Functionality
Programmers are often reluctant to restrict what a program can do, regardless of whether simpler, less feature-heavy software would be more beneficial to the end user.

The deliberate restrictions placed on Shaper2D could be described as additional functions in their own right. Shaper2D was designed to force the user to operate the program in a certain way in order to learn basic shape grammars. Ensuring simplicity of use made the interface design more problematic and restricting the program’s functionality, such as forcing the user to manipulate the rule geometry and labels separately, complicated programming.

Previous Shape Grammar Programs

To frame the motivation behind the development of Shaper2D, it is helpful to consider two recent shape grammar programs, 3D Shaper (Wang 1999) and GEdit (Tapia 1999). These applications were developed to expedite the process of design using shape grammars and have been used on occasion for classroom teaching. However, their use in the design studio has been limited. Various reasons have been cited for this, including a completely unrestricted workspace that is only truly useful to more advanced uses of shape grammars, delayed feedback, and a designer-unfriendly interface. These are not criticisms of the software, as each application has some audience in current shape grammar pedagogy and research. However, the deficiencies listed above are those that Shaper2D seeks to address.
**3D Shaper**

*3D Shaper* was developed by Yufei Wang in 1999 to simulate the designs produced by manipulating wooden “Froebel” blocks. Compiled for the UNIX/SGI operating system, it is a very powerful program that enables the user to experiment with three-dimensional shape grammars in ways that are difficult to visualize mentally, and sometimes impossible to perform physically.

3D Shaper implements a static interface that requires the user to type in numerical parameters to determine the size and type of shapes, as well as the spatial relation between the shapes. The program then applies the rules and produces an Open Inventor file containing the three-dimensional design output. The user opens this file in an appropriate viewer for review. However, if a design change is needed then the user must return to the 3D Shaper window, input the necessary shape and/or rule revisions, and run the program again in order to generate a new Open Inventor file.

**GEdit**

The first computer implementation of shape grammars that included a viable user interface was the Apple Macintosh-based GEdit developed by Mark Tapia (1999). This interpreter is used for generating two-dimensional, non-parametric shape grammars and emphasizes that the “...drawing is the computation” (Tapia 1999). Tapia sought to minimize user distraction, both in terms of obscuring parts of the program and perverting the design flow, by limiting the use of drop-down menus and dialog boxes. He instead implemented an object-specific radial menu, which only appears when needed to ensure that the user’s focus is not removed from the design process. This is a program for shape grammar experts, since it is very open-ended and allows for the free exploration of almost any kind of non-parametric, two-dimensional shape grammar.

**What is different about Shaper2D?**

Previous programs developed to expedite the process of design using shape grammars lacked an accessible user interface, were operating system-specific, were too general, or did not account for the dynamic, interactive environment sought by a designer.

With 3D Shaper, the process of inputting numbers into a static interface and opening up a separate viewer in order to see the design does not tally with the designer’s habitual design process. While the parameters entered by the user fully describe the desired grammar, they do so in a way reflective of the internal computations of the software as opposed to the way the designer would manipulate shapes naturally. The interface was written for the ease of the programming, not for the benefit of the user. For many architects, conceptual designing is fluid and dynamic - 3D Shaper is an invaluable tool for experimenting with three-dimensional shape grammars, but the interface is not analogous to the design process and hence does not facilitate quick, experimental design.

GEdit, while introducing the notion of an elegant interface, is a generalized, unrestricted example of shape grammar software. It is suitable for advanced shape grammarians wishing to visualize a particular derivation, but is impractical for novice users who require a more constrained, structured environment or for users wishing to use shape grammars for real-world design problems.

Shaper2D is a visually dynamic shape grammar application for generating designs using very restricted kinds of shape grammars. Changes to the grammar are immediately reflected in the design. It was written to overcome the platform-specific limitations imposed by previous interpreters. The application has been run successfully under several major operating systems (Microsoft Windows, Mac OS X and Linux), and the applet runs under any web browser capable of running Java™2 (such as Mozilla, Netscape, Internet Explorer, and Opera).
The Development of Shaper2D

The interface is the key aspect of Shaper2D's development. It had been noted by several researchers in the field of shape grammars that while useful implementations of shape grammars have been developed, little attention had been focused on the interface (March, 1997; Knight, 1998; Tapia, 1999). Given the prospective audience of designers, this was a surprising oversight.

Visual seductiveness is very important to designers. So naturally, when developing Shaper2D, many graphic and interface design issues had to be considered. In particular, the interface needed to embrace the inherent visual uniqueness of shape grammars: the decision to exclude any need for manual typing-in of parameters or instructions was made at the program's inception.

The most comprehensible way of doing shape grammars is when the computation is done by hand - indeed, hand computation using tracing paper played an important role in the development of Shaper2D (McGill, 2001). Shaper2D strives to make the shape grammar process as understandable as possible through interactivity and dynamic response. However, because the actual computations are concealed the program could be described as a „black box“: the rule application process is hidden from the user. Physically flipping and rotating drawings on tracing paper, or manipulating three-dimensional wooden blocks, personalizes the act of performing the spatial transformations by exposing the designer to direct contact with the processes involved.

The dynamic interface endeavors to offset the inherent black box disadvantages by informing the designer of the effect of her actions immediately. The user can experiment quickly with different rules and iterations in order to compare the outcomes of different ideas and grammar applications.

How Shaper2D Works

Shaper2D was developed as both a Java applet (figure 1) and a stand-alone application (figures 2 and 3).

The Shaper2D user interface is composed of a menu bar, five panels - “Spatial Relation“ (1 and 2), „Rule“ (1 and 2), and „Design“ - and two toolbars for selecting different shapes. The default shapes are a...
Figure 2
The Shaper2D Basic Application (version 3.1).

Figure 3
The Shaper2D Advanced Application (version 3.1).
rectangle, square, equilateral triangle, and isosceles triangle. The „Spatial Relation 2“ and „Rule 2“ panels are disabled by default to emphasize the difference between using one rule and two rules. Whenever the user manipulates a shape—which can be done in both the „Spatial Relation“ and „Rule“ panels—the other active panels update in real-time to reflect the change. Similarly, when the position of a label is changed the „Design“ panel updates immediately.

The „Design“ panel updates dynamically according to changes made in the „Spatial Relation“ and „Rules“ panels. Additionally, to review the construction of a design, the user has the option to display the labels (figure 4)—with the most recent shape addition highlighted in red. The label display feature is particularly helpful when the user performs a walk-through by gradually incrementing the number of iterations displayed.

**Shaper2D Applet**
The advantage of the applet (www.mit.edu/~miri/shaper2d) is that it can be run directly from an Internet browser. This enhances portability by allowing users who may not be able to download and run the application to have access to the software. However, this version has limited functionality due to security restrictions imposed on applets.

**Shaper2D Application (Basic & Advanced)**
The Shaper2D application is a more feature-rich tool than the applet. The application allows the user to save and load rules and export a design in the DXF file format for import into another CAD application (such as AutoCAD), thus increasing the pedagogical and design value of the software. The user is also able to import a background image into the design window, in order to place a design in a given context. In turn, the user can change the line color of the design to improve visibility (figure 5).

Rather than overwhelm beginners with an all-inclusive, over-complex interface, two versions of the application were created, one for novices and the other for more experienced users. The decision to develop two versions was in response to Shaper2D being designed as a pedagogical tool. Thus, when

![Image](image-url)
Figure 5
Importing a background image and changing line color.

Figure 6
Shape Substitution.
the user is confident enough to progress to more complex shape grammar investigations she can change from the basic to the advanced interpreter. Files saved on each version can be used interchangeably.

**Shaper2D Application (Advanced)**

This version incorporates shape substitution (figure 6). This feature enables the user to import a different shape or design to be used in place of one of the four preexisting shapes. The substituted shape retains the symmetry of the reference shape it replaces, allowing the user to explore the meanings of different symmetry groups. More complex designs can be developed providing the user an opportunity to investigate how the different symmetries of the reference shape affect the rule application.

**Conclusion**

The development of Shaper2D responded to the need for an improved human-computer interface for exploring shape grammars. Shaper2D capitalizes on the advantages of computer power over hand computation, especially when it comes to more complex design generation.

While it has been widely agreed that hand computation is essential for a comprehensive understanding of shape grammar concepts, using the computer can facilitate designing with them as designs tend to get very complicated very quickly. Shaper2D makes the shape grammar process as transparent as possible through interactivity and dynamic response: designing becomes more comprehensible and designers are encouraged to persevere with an idea rather than give up through frustration or impatience.

Shaper2D was developed by a designer rather than a programmer, and strives to promote a more fluid approach to software design for designers. Previous shape grammar programs - designed by programmers - do not offer the user the opportunity to quickly explore many designs or follow the grammar process, so fail to offer many advantages over hand computation. They tie the user into the „plan, calculate, examine, redesign“ method of designing—in other words, an iterative and algorithmic approach—where each stage is seen as a distinct step. The user loses the ability to explore and experiment. Shaper2D encourages exploration by providing the user with instant feedback and a purely visual, designer-friendly interface.

**Future releases**

It is hoped that rather than having to run a separate application, the next version of Shaper2D will encompass both versions, with the basic starting by default and an option to switch to the advanced interface.

Future releases of the Shaper2D advanced application will include multiple rules, thus allowing the creation of more than two rules—creating a richer learning and design environment—and non-linear rule application, where the user can decide which rule to apply next in a derivation, assuming the rule application is legal.

If a 3D version of Shaper2D is developed, it will be an entirely different application. It is important to retain the emphasis of Shaper2D being a tool for learning basic, two-dimensional shape grammars.

**Acknowledgements**

Thanks to: Terry Knight, Oliver Dial, William Mitchell, Edith Ackermann, Mark Tapia, and everyone involved in testing Shaper2D.

**References**

Wang, Y.: 1999, 3D Architecture Form Synthesizer, SMArchS Thesis, Department of Architecture, Massachusetts Institute of Technology, Cambridge, MA.