

CARNEGIE MELLON UNIVERSITY

School of Architecture
College of Fine Arts

Thesis

Submitted in Partial Fulfillment of the requirements for the degree of
Master of Science in Computational Design

TITLE:

Design with data-driven shape rule recognition
through learning-based shape grammar interpreter

AUTHOR:

Yujeong Seo

ACCEPTED BY ADVISORY COMMITTEE:

_____
Professor Vernelle A. A. Noel Principal Advisor

May 10, 2024

DATE

Professor Daniel Cardoso Llach Committee member

May 10, 2024

DATE

Professor Jinmo Rhee Reader

05/08/2024

DATE

Abstract

Shape grammars started from the concept of generating infinite design alternatives with a limited number of shape rules through the shape itself. Shape grammar interpreter is a computational framework dealing with the language of shapes using shape grammar formalism, allowing users to transform one shape to another by applying shape rules. As a rule-based method, it has been applied to various design problems, research, and education. Recently, the Shape Machine has solved the long-standing sub-shape recognition problem in the shape grammar interpreter in the 2D space. It enables users to explore designs precisely by visually coding shape rules using vector data. This research offers a different perspective on coding the shape rules from existing shape grammar interpreters such as Shape Machine, which focuses on accurately recognizing and generating the shape. With the rapid growth of artificial intelligence (AI) research, there have been increasing attempts to solve design problems using data-driven methods like deep learning. This research proposes a learning-based shape grammar interpreter by introducing deep learning, allowing designers to select shape data and train interpreters to recognize shape rules. Designers understand and input the shape rules into the rule-based interpreter based on visual representation. On the other hand, the learning-based interpreter recognizes the shape rules by analyzing the relationships of elements selected by the designers from the data. The designers choose the relationship that will be analyzed instead of analyzing it themselves. This is determined by the designer's perspective on the design problem and may offer different shape rules. After constructing the new interpreter, to understand designers' experiences with it, a user study using simple geometric shape rules evaluated the learning-based interpreter by comparing it with the existing rule-based shape grammar process. This research demonstrates the potential of a learning-based interpreter to enrich the rule space through different design experiences and unexpected design results, supporting current shape grammar interpreters. However, it also shows the limitations of the design of the learning-based interpreter (challenging in data acquisition and processing, visualization, and understanding of rules) and experiment (limited data selection, symmetry issues in comparative experiments).

KEYWORDS

Shape grammars, Shape grammar interpreter, Deep learning, Data-driven design, Interaction

Acknowledgments

This thesis would not have been accomplished without the support of many people.

First of all, I would like to express my sincere gratitude to my primary advisor, Professor Vernelle A.A. Noel, for her continuous and unconditional support for my academic journey. I'm deeply indebted to her kind and detailed feedback, suggestions, patience, and encouragement. I will cherish the memory of our conversation every Thursday afternoon, after which I always felt encouraged and inspired.

Next, I could not have undertaken this journey without my secondary advisor, Professor Daniel Cardoso Llach, for giving me the guidance to expand the scope of the idea through critical thinking. His keen advice always leads to great inspiration and rigor in the thesis.

I would like to express my deepest appreciation to my tertiary advisor Jinmo Rhee, not only for his expert feedback on the research but also for his sincere advice on shaping the future direction.

Many thanks to my cohorts and peers in the Computational Design program, including Felicia Luo, Yanting Deng, David Troetschel, Hongfei Ji, Steve Hu, Zain Hao, Jingyang Liu, Emek Erdolu, Nik Kim, and Sohyun Jin, whose insights and perspectives have enriched my understanding and inspired new ideas. My time spent in the Code lab with them has been invaluable and enjoyable.

Special thanks to Niloofar Nikookar and Semina Yi for making this path full of joy and for their unwavering support.

Thank should also go to peers in SoA and SoD who participated the user study and gave insights. Finally, I would like to express my endless gratitude to my family for their unconditional support and encouragement throughout this entire journey. Their support has been the cornerstone of my study, and I owe them a debt of gratitude for their endless support and belief in me.

Table of Contents

Abstract	2
Acknowledgments.....	3
Table of Contents.....	4
1. Introduction	5
2. Background	8
2.1 Development of Shape Grammars: Theory to Application	8
2.2 Shape grammar interpreter	13
2.3 Data-driven design with deep learning	17
3. Methods	20
3.1 The learning-based shape grammar interpreter prototypes	20
3.2 Interpreter design for the user study	26
3.3 User Study	34
4. Results.....	43
4.1 Design outcomes.....	44
4.2 Results of the questionnaire.....	46
5. Discussion	51
5.1 Findings	51
5.2 Limitations.....	53
6. Conclusion	56
6.1 Summary.....	56
6.2 Future Works	58
Appendix.....	59
Reference	63
List of Figures and Tables.....	67

1. Introduction

Shape grammars started from the concept of generating infinite design alternatives with a limited number of shape rules based on the shapes themselves (Stiny and Gips, 1971). They are methods of dealing with design problems computationally with the shape rules, not by the other types of primitives, such as numbers or strings, but by shape itself exclusively (Hong, 2021). As one of the rule-based design systems, shape grammars have proposed a way of design with the aid of computers beyond simply transferring drawings from paper to computers (Stiny, 1982).

Shape grammars have been studied theoretically, technically, and practically. The framework of shape grammar computation consists of 1) receiving shape rules generated by the designer and the initial shape as input values, 2) recognizing the shape selected by the designer among the shape rules in the initial shape, and 3) replacing recognized shape with a new shape according to the shape rule. By repeating this process, designers can generate surprising design alternatives. Shape grammar interpreters are computational frameworks implementing shape grammar formalism, allowing users to transform one shape into another by applying shape rules (Krishnamurti, 1982). The critical issue in computationally implementing the interpreter lies in accurately computing the shape rules and recognizing selected shapes, namely sub-shapes, which has long been challenging. But the Shape Machine has recently solved this problem (Hong, 2021). Instead of developing shape grammar interpreters technically, some research has tried to approach real-life design problems by applying this concept or technology or by studying how designers can generate shape rules (Eloy et al. 2018). The application of shape grammars has been studied in various areas that use visual images, such as architecture, urban, and product design. 'Making grammars' expanded the applications to material and making practice, but it applied shape grammar conceptually without technical implementation (Knight and Stiny, 2015). In fact, shape grammars are mainly used in design education because they allow users to develop design ideas by applying explicit shape rules procedurally (Knight, 1999). There have been studies on the interaction between the designer and shape grammar interpreters concerning how designers use interpreters in real design situations, mainly focusing on the interface of the interpreter.

The core objective of the state-of-the-art interpreter, Shape Machine, was to compute shapes accurately. It enables users to explore designs precisely by visually coding shape rules using vector data (Hong, 2021). Subsequent research about use cases has demonstrated the potential of Shape Machine for various fields such as architecture, product design, and engineering. I aim to approach shape grammar interpreters from a different perspective than the existing shape grammar interpreters, such as Shape Machine, focusing on accurately recognizing and generating shapes. Exploring shapes through shape grammar interpreters requires inputting various shape rules for diversity in outcomes and enjoyment in the process. How to design shape rules is a design problem in the shape grammar approach (Knight, 1999). Actively searching for the shape rules based on designers' experience and expertise can be the design process. However, when it comes to detecting shape rules from a corpus requires labor and time, and there may be unintended missing rules. Moreover, some novice designers may not have many ideas for the shape rules and even find it challenging to detect them from diverse design cases. What if designers can input shape rules with data?

With the rapid growth of artificial intelligence (AI) research following the development of computer hardware in the last decade, there have been increasing attempts to solve design problems using data-driven or learning-based methods through deep learning in architecture (Rhee et al., 2023). Unlike the rule-based method, where designs are generated according to the rules defined by designers, the learning-based method provides a different approach to recognizing problems by extracting patterns from a large amount of data. The existing shape grammar interpreters, based on the rule-based method, are used with the shape rules generated from the designers' experience. (Hereafter referred to as rule-based interpreters.) I developed a learning-based interpreter that recognizes the shape rules based on the data selected by users. If the learning-based method is introduced to the shape grammar, designers can generate design alternatives with the shape rules the interpreter recognizes from the data. Designers understand and input the shape rules into the rule-based interpreter based on visual representation. On the other hand, the learning-based interpreter recognizes the shape rules by analyzing the relationships of visual elements selected by the designers from the data. The designers choose the relationship that will be analyzed instead of analyzing it themselves. This is determined by the designer's perspective on the design problem and may offer different shape rules.

There are many ways to manipulate data and generate training algorithms, but the new interpreter in the research aligned with the advantages of a shape grammar interpreter suitable for educational tools. Thus, I designed an interpreter that is easily accessible to designers who are not even familiar with the data. Then, through the user study to understand designers' experiences, I evaluated the new shape grammar interpreter that allows users to select data relationships to detect shape rules by comparing the rule-based interpreter. Designers typically design and input shape rules to the rule-based interpreter by using their design knowledge. However, this user study compared two interpreters by assuming the same scenario of identifying shape rules from data and inputting them into the interpreter. Specifically, I examined designers' thinking, usability, and emotional aspects. The research aims to implement the learning-based interpreter, to examine whether the learning-based interpreter enables designers to understand the shape rules in a different way from the rule-based interpreter and to evaluate its usability and emotional aspects.

Therefore, my research question is that

“How does a learning-based shape grammar interpreter compare with the rule-based shape grammar interpreter?”

The hypothesis is as follows.

- Designers will input shape rules by considering the relationship between data rather than visual properties identified by their design experience.
- Learning-based shape grammar interpreters may help to expand the rule space with unexpected shape rules, but it may be challenging to understand.

This study initially investigates the theoretical, technical, and practical development of shape grammar research, understanding what problems it has been interested in and how it has been trying to be solved, and delves into data-driven design using deep learning. The experiment for the research question begins with constructing the shape grammar interpreter using a learning-based method. Through a user study comparing the constructed learning-based interpreter with the rule-based interpreter, this research evaluates the new interpreter from users' perspectives, exploring its potential and limitations.

2. Background

2.1 Development of Shape Grammars: Theory to Application

Shape grammar started with making infinite design alternatives with a limited number of shape rules by inspiration from linguistic grammar. In computational design, shape grammar is a method of dealing with design problems computationally with the shape rules, not by the other types of primitives but exclusively by the shape itself (Hong, 2021). This rule-based model is one of the early generative design systems in which designers make a design system generating design alternatives rather than creating and comparing design alternatives by themselves. For about 50 years, shape grammar has been studied in various areas, including theoretical research centered on Stiny, technical research related to algorithms for implementation centered on Krishnamurti, and research on applying shape grammar to real-life design practice, research, and education. A brief look at the history of shape grammar centered on prominent researchers in each field is as follows.

2.1.1 Theoretical research

Shape grammar theory has been developed and centered on Stiny since it was introduced in 1971 by Stiny and Gips (Stiny and Gips, 1971). Starting with the theory, which used markers as restrictions of rule application, shape grammar was developed into a subshape-driven shape grammar theory that represented shapes specified by maximal lines (Stiny, 1975). The standard formalism of shape grammar, the most widely used, was constructed by Stiny (Stiny, 1980). Even though its shape was limited by two-dimensional straight lines, non-parametric and parametric shape grammar are defined together. Stiny has been the center of the shape grammar theory, proposing a set grammar that addresses shape as an element of a set (Stiny, 1982), extending the dimensions of recognized geometry (Stiny, 1991), and introducing algebras of shapes(U) augmented by symbols such as labels and weights in (Stiny, 1992). Knight has researched shape grammar theory with Stiny, especially related to expressive parts, and has been interested in applying shape grammar theory without using computer programs, as she mentioned (Knight, 1983). She has broadened the theoretical areas in shape grammar, proposing

theories such as 'Color grammar'(Knight, 1989) that proposes quality components in shape grammar related to the visual art field and 'Making grammars'(Knight, 2015), which deals with shape rules about physical material with temporal component and says that making is 'Doing and Sensing' instead of 'Seeing and doing.'

2.1.2 Technical research

In shape grammar research, technical research is integral so that the interpreter can be implemented as a computer program. This field has been developed centered on Krishnamurti. He developed algorithms for Boolean operation and Euclidean transformation that represent shapes and showed this by implementing the first general shape grammar interpreter, 'SGI' (Krishnamurti, 1980; *ibid.*, 1981; *ibid.*, 1982). After research about SGI, including editing shapes, many studies have been conducted with Stouffs (Krishnamurti, 1986). The main challenge in shape grammar interpreter research is how to make sub-shapes recognized accurately with any dimension of shapes or how to solve the complexity issues. Krishnamurti and Stouffs have researched computer implementation in shape grammar, such as sub-shape recognition in higher dimensions of shape algebras and weighted shapes (Krishnamurti and Stouffs, 2004; *ibid.*, 2018). Stouffs developed a general-purpose interpreter, SortAL GI, based on Rhino3D to prove the technical research (Stouffs, 2012). Yue contributed to technical research by integrating Stiny's shape grammar computation definition of non-parametric shape grammar and parametric shape grammar (Yue and Krishnamurti, 2013). Starting with a study on three-dimensional line sub-shape recognition, Earl has mainly conducted research with Jowers (Earl and Krishnamurti, 1992). They studied shape grammar implementation in curved shapes and developed 'QI' (Jower and Earl, 2011). Jowers et al. tried a different approach to the problem of sub-shape recognition, which is considered the trickiest, using a computer vision method other than the maximal line, but this also had limitations, and no follow-up research was conducted (Jowers et al., 2010). Recently, the two researchers extended the theory to Kinematic grammar in a mechanical area with Harrison (Harrison et al., 2019).

2.1.3 Application research

Some studies apply shape grammars to real-life design problems and find design solutions rather than a theoretical or technical approach (Eloy et al., 2018). Starting with Palladian grammars by Stiny and Mitchell, research in this area covers 1) design analysis, in which the research analyzes existing design to understand shape rules and generate design alternatives based on them, 2) design generation, in which designers create shape rules and generate design alternatives from these rules, and 3) design education, in which people generate shape rules and explore the design idea with the rules to understand the specific shape rules or form-making, particularly for pedagogical purposes (Haakonsen et al., 2023). The application of shape rules has been studied not only in architecture but also in various domains using visual parts such as urban and product design. While studying grammars for the customization of Siza's mass housing at Malagueira, Duarte proposed grammars in which designers could make design decisions by combining the shape, constraints, qualities, cost, and heuristics (Duarte, 2005), and it was developed into urban grammars and interpreter for urban design generation by Beirão (Beirão, 2009). Eloy has developed the research by applying shape grammars as a design tool for residential users in housing transformation situations (Eloy, 2018). As interest in materiality has increased, research to interpret shape grammars from visual and material aspects has increased. Along with Making grammars, which is proposed by Knight and Stiny, Gürsoy and Özkar conducted a study to introduce shape grammars in the process of developing designs while physically making alternatives (Gürsoy and Özkar, 2015). Noel has researched encoding and documenting a vanishing craft method of wire-bending to 'Bailey-Derek grammars' and using it for education (Noel, 2015). In the mechanical product design field, McKay, Chau, and Pennington have studied three-dimensional shape grammars implementation and the use of shape grammars in early concept design. (Chau et al., 2004)

2.1.4 Interaction research

Since research about shape grammar interpreters combines theory and technology, more interpreters have been implemented in the second half of the shape grammars research for nearly 50 years. Only a few studies about seamlessly usable interpreters for designers have been conducted. Tapia's 'GEdit' is the first interpreter to consider user interface, and users can use

interpreter programs by drawing shapes, not by writing coordinates (Tapia, 1999). Economou started research by using symmetry in shape grammar theory and has conducted research on implementing shape grammar interpreters based on theory and technology. While developing the interpreter, he also studied the user interface. After developing the shape grammar interpreter called 'GRAPH' which implements parametric shape grammars by using graph algorithms to recognize sub-shapes, Economou and Grasl conducted a user study with the GRAPH to understand the design workflow during the workshop (Economou and Grasl, 2013; *ibid.*, 2017). Recently, Economou and Hong implemented the 'Shape Machine', which advanced in two-dimensional maximal line sub-shape recognition and indeterminacy and user interface using drawings instead of writing codes (Hong, 2021). With Ligler, he studied how these interfaces would affect design (Ligler and Economou, 2019). Li has been interested in using shape grammars as an educational tool and the interaction between interpreter and user (Li, 2001). He made a program that allows users to generate and test shape grammars in Rhino3D by using Sortal AL GI as the backend and developing the front end (Li and Stouffs, 2021). Despite some studies on the interaction between designer and interpreter, they mainly focused on the interpreter's interface.



2.2 Shape grammar interpreter

2.2.1 Generation process in Shape grammar interpreter

The definition of shape grammar computation has evolved under the contribution of Stiny since its first appearance in 1971. The initial theory involved the application of shape rules based on markers. However, since Stiny introduced the sub-shape approach that shapes are specified with maximal line representation, this became the standard of shape grammar computation, leading to the development of interpreters accordingly (Yue and Krishnamurti, 2013).

As shown in Figure 2, a shape grammar interpreter generates design alternatives with three input values through a five-step process. First, the shape rules ($u \rightarrow v$) that the designer recognizes from the analysis of design problems or phenomena are encoded to the interpreter, the initial shape (W) is input, and the shape rule is selected by the designer. Then, the input shape's maximal representation is encoded. Since the maximal representation, calculated as a maximum shape including all the shapes and from which the sub-shapes are extracted, allows to decompose and fuse the shapes, designers can get emergent results from the interpreter (Stiny, 1980). Second, the interpreter finds all Euclidean transformations (f), that the shape of the selected rule becomes a sub-shape of the initial shape. When the sub-shapes are recognized, they ($f(u)$) are subtracted from the initial shape, and the right-hand-side shape ($f(v)$) is added by Boolean operation. Finally, emergent design alternatives are generated by iterating this process.

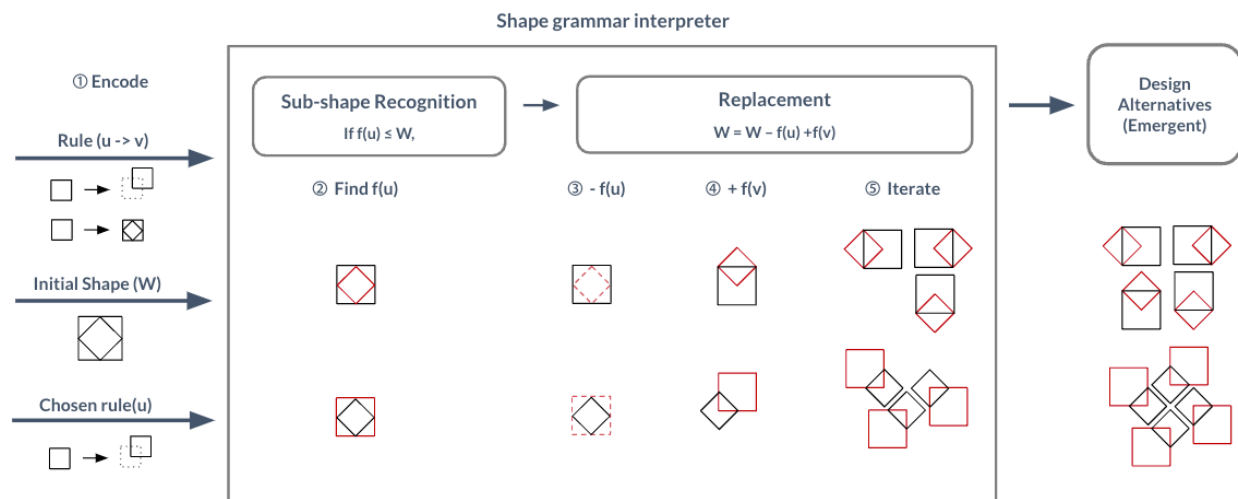


Figure 2 Principle of shape grammar interpreter computation

2.2.2 Related works on general shape grammar interpreters

Conceptually, shape grammars are intuitive compared with other rule-based systems using numbers or strings as mediums, such as fractals or L-systems, but implementing the shape grammar interpreters computationally is not easy. Research on general shape grammar interpreters applicable to general shape rules, rather than specific design domain interpreters, has focused on addressing technical issues.

SGI

SGI, the first general shape grammar interpreter introducing the sub-shape recognition method, was implemented by Krishnamurti (Eloy et al., 2018). This interpreter was developed as an educational tool and used its own language for command (Krishnamurti, 1982). While it is a limited model capable of recognizing only some basic rectilinear shapes in 2D as shape rules, it has significance as a fundamental model for subsequent interpreter development.

Shape Machine

As the latest general-purpose shape grammar interpreter, Shape Machine is a design automation program that allows users to explore designs by visually coding shape rules using vector data. Since SGI, especially in general-purpose applications, the major technical challenge of the research in shape grammar interpreters has been on accurately recognizing sub-shapes regardless of dimensions of shapes, whether in parametric or non-parametric interpreters (Grasl and Economou, 2013). Even if shapes seem similar, they may not function if they are mathematically different. As the dimension of the shape increases, complexity also increases. In addition, as finite machines, computers cannot perfectly compare and compute the geometries of two shapes. Therefore, the significance of Shape Machine lies in its resolution of the long-standing challenge among shape grammar interpreter researchers, which is sub-shape recognition through maximal representation. While its accuracy may decrease with complex shape or mixed shape types, it has resolved most issues in 2D space (Hong, 2021).

Shape Machine accurately recognizes the shape rules inputted by users computationally and applies them to the given vector image, allowing users to explore design easily and intuitively. Since its implementation, numerous use cases have been proposed. Similar to how

computers find and modify text, it can precisely detect and transform visual shapes, allowing users to experiment with visual forms intuitively. It has demonstrated potential applications in various fields using visual shapes, such as architectural drawings or product designs. Notably, it has an advantage as an educational tool by visualizing the design process step by step.

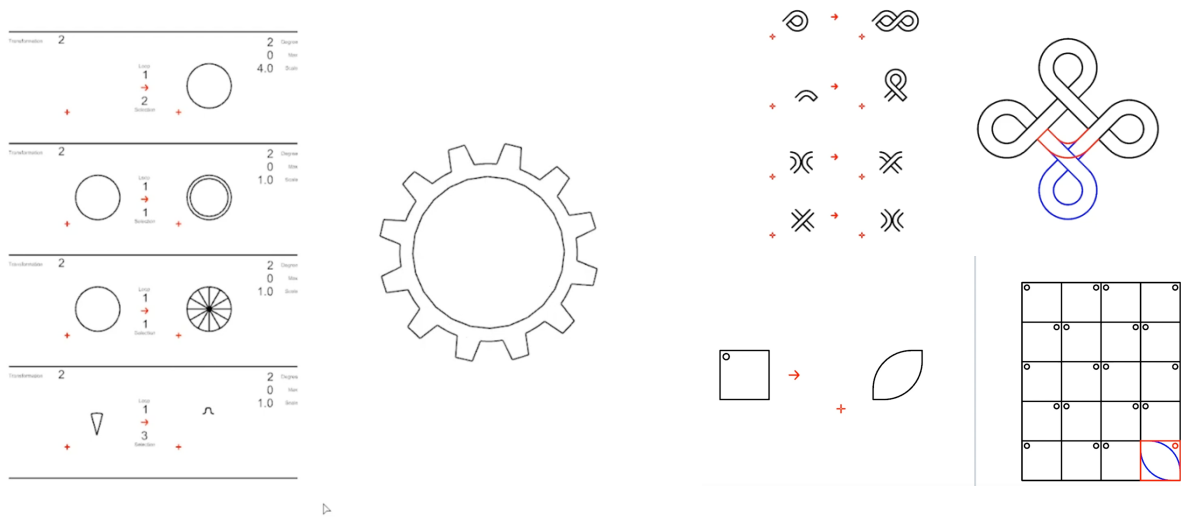


Figure 3 Examples of design application of Shape Machine (<https://shape.design.gatech.edu/index.html>)

Interpreter through Computer Vision

In 2010, Jowers et al. developed an interpreter that introduced the computer vision approach as a completely different method to solve the sub-shape detection problem for curved shapes (Jowers et al, 2010). Traditional interpreters analyze the linear features to detect sub-shapes from input images, making it difficult to distinguish visually identical but analytically different shapes and requiring many detection algorithms. However, input images in this interpreter are defined as a set of points. They tried sub-shape detection by computing the Hausdorff distance between the points of two images and finding the shape that minimized the distance. Since points are undecomposable elements, sub-shape detection can be reduced, and shapes are recognized as raster images rather than vectors. However, pixelated points lose geometric features from the original shapes. Computer vision methods inherently have lower accuracy than mathematical algorithms, which do not align with the primary goal of interpreter

research, which is to recognize accurate shapes. As a result, no further research has been conducted in this direction.

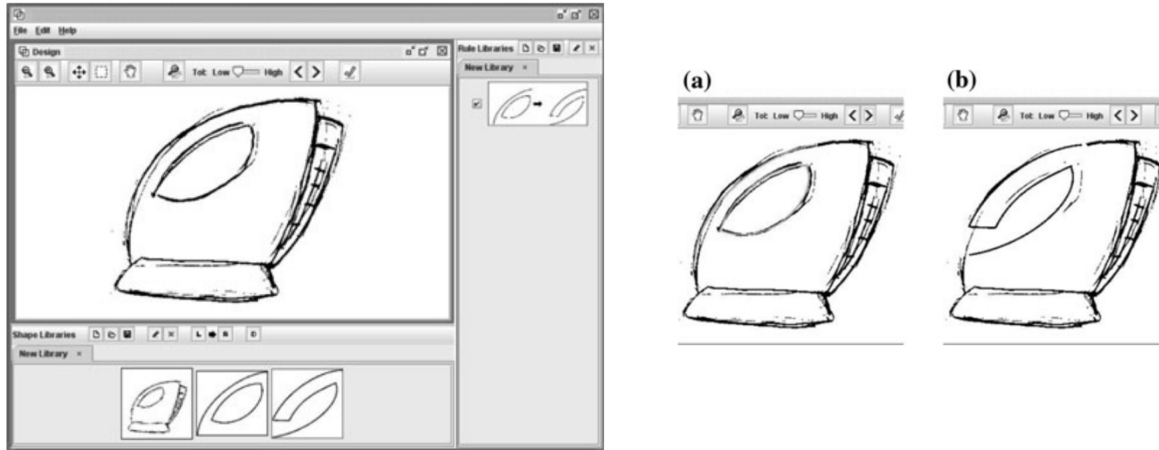


Figure 4 Interface of the interpreter (Left), Example of rule application (Right) (Jowers et al., 2010)

2.3 Data-driven design with deep learning

Generative design is the indirect design methodology with which designers design generators that synthesize the output design alternatives under input conditions and interact with the generator by controlling the input to generate design alternatives instead of design ideation with design media directly (Veloso, 2023).

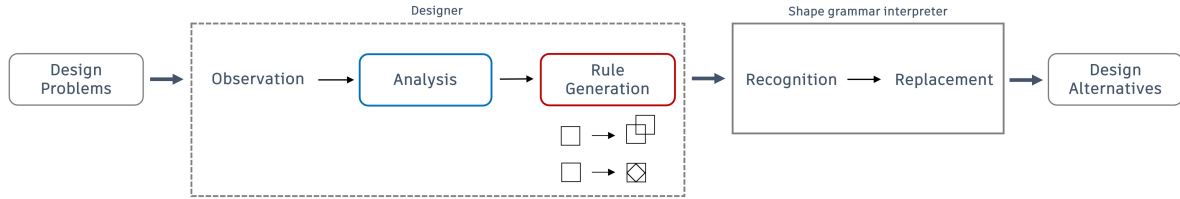
Rule-based design vs. Data-driven design

Various approaches of design logic can be applied to the generator, and one of them is the rule-based design, which is the methodology of shape grammar interpreter. Rule-based design is the deductive methodology in which design alternatives are generated from the explicit rules or constraints that the designer encodes (Chen, 2019). These rules can be understood as the designers' design methodologies or identities but rely on personal experience and expertise. This can be a limitation since there are only some rules that individuals can recognize and develop among numerous rules. In addition, as control and the emergence of design are a trade-off, if a generator considers the control a lot, design alternatives may not differ significantly from those generated by the designers despite the speed advantage (Veloso, 2023).

A data-driven design methodology approaches design problems inductively by inputting data inducing the rule into the generator instead of the explicit rule. This is the opposite of the rule-based method, in which the designer designs rules or design methods deductively and applies them to solve the problem (Veloso, 2023). A data-driven design system generates design alternatives with learning-based models that train generators with abundant data and recognize the features of the data, i.e., design logic. By utilizing computing power to process a lot of data quickly and extract features, designers' time and labor are reduced, and design phenomena or precedents are interpreted based on the pre-set logic (Hasey, 2022). Therefore, in this case,

designers need to design data and models for computers to extract features needed for decision-making (Cai, 2020).

Rule-based



Learning-based

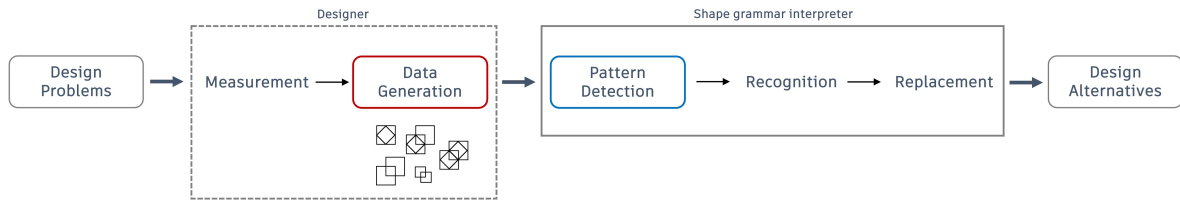


Figure 5 Comparison of the overall design process for rule-based shape grammar interpreter and learning-based shape grammar interpreter

Deep Learning

Artificial intelligence (AI) research has grown rapidly in the last decade based on big data and the development of computing power. Deep learning (DL), one of the machine learning (ML) methods, is a computer algorithm that imitates the way humans process information. By converting a large amount of data into tensors that the computer can recognize and passing through a multi-layer neural network consisting of nodes, designers can extract the features of data to predict and generate the desired information. Basically, since deep learning is based on probability, the more data the computer uses as a reference, the more likely it is to produce accurate results. Moreover, all types of data, such as images, voice, or text, that computers can recognize can be used for learning only if they are well pre-processed. Therefore, in the architecture domain, design research using deep learning models is increasing, and applied to generation and recognition is the recent trend (Rhee et al, 2023).

Data-driven design with deep learning

Deep learning allows designers to understand and express designs from various kinds of stored data beyond visual hints, which may lead them to unexpected directions or generate new results because patterns are not analyzed relying on designers' experience and knowledge. And it will require a redefinition of the relationship between designers and computers. Therefore, with the deep learning model, how to design input data, that is, which data is to be taken and how to pre-process it, for the computer to extract patterns from data and generate or predict results becomes an important issue. The designer's perspective of the problem is reflected in the design of the data. For example, Rhee et al. and Steinfeld approached the design problem differently in an urban context. Steinfeld used image data divided by the regions to synthesize them with the GAN model to capture the predominant visual properties of places (Steinfeld, 2019). On the other hand, Rhee et al. used GIS data of building footprints and parcels for building massing in the urban context (Rhee et al., 2020). According to the designer's intention or design ideation, the way data is designed and applied is quite different, and the results give different insights.

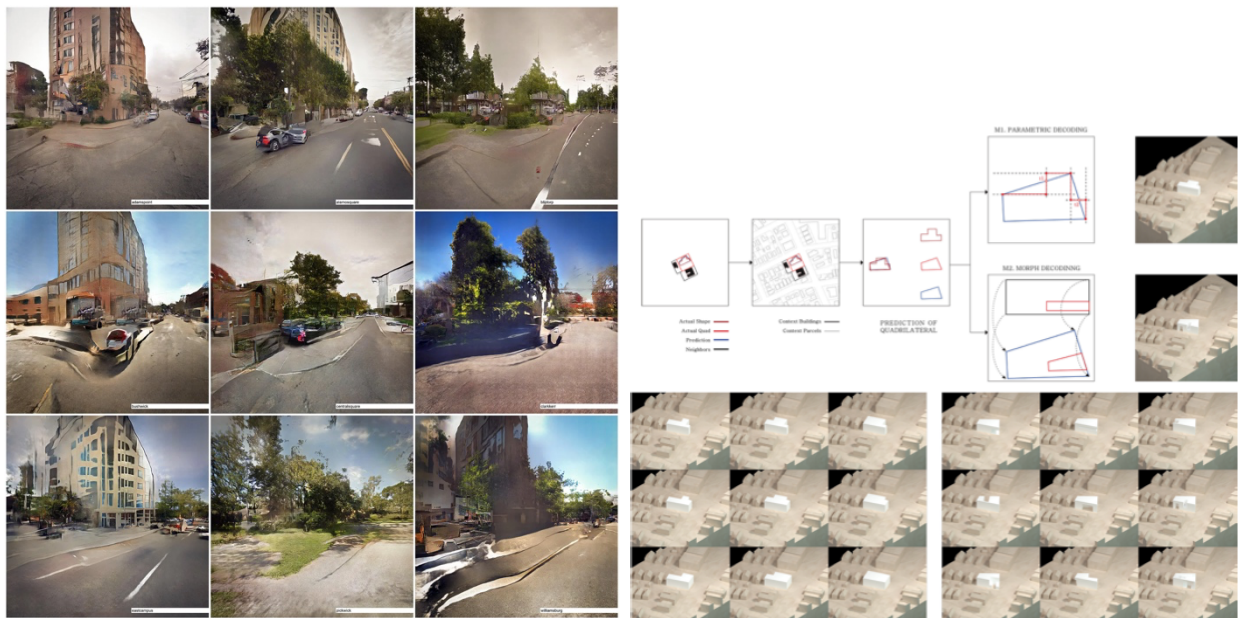


Figure 6 The designer's perspective of the problem is reflected in the design of the data and results. Predominant visual properties of places worked by Steinfeld (Left), building massing design using GIS data of building footprints and parcels worked by Rhee et al. (Right)

3. Methods

To examine the research question comparing the learning-based interpreter with the rule-based shape grammar interpreter, I designed the learning-based interpreter using deep learning algorithms. Then, the data and interface of an interpreter were designed for the user study. After constructing the interpreter, the user study was designed to evaluate it. The user study compares the rule-based interpreter, in which designers recognize shape rules in a given design case and input the rules, with the designed learning-based interpreter, in which designers manipulate and input data for the interpreter to extract the shape rules.¹

3.1 The learning-based shape grammar interpreter prototypes

First of all, I designed learning-based shape grammar interpreter prototypes. Unlike the rule-based shape grammar interpreter in which designers directly input shape rules, a learning-based interpreter requires a large amount of data implying rules and extracts patterns from them. So, I started with generating data. The amount of data has always been at the center of the discussion about the learning model. Small datasets result in bias and overfitting during training and do not show robust performance from newly input data. Thus, the deep learning model performs better as it is run with more data. After generating the datasets, since the visual data type is a raster image or vector, I generated interpreters that recognize rules from these two types of simple geometry data. After implementing the structure of the learning-based interpreter, I designed the interface to enable designers to use this tool.

¹ General purpose shape grammar interpreters, which recognize sub-shapes and generate design alternatives from maximal line representation, can only recognize the sub-shapes that accurately match the ‘before’ shape of the shape rules in the input image. For example, in a shape rule where a square transforms into a diamond, the interpreter cannot recognize the shape if the initial image is a rectangle. However, since design rules found in reality are not always precise and deep learning recognizes shape rules based on probabilistic predictions, a learning-based interpreter can recognize shapes even if they are not exactly the same, depending on the shape rules. For instance, in a shape rule where connecting the midpoint of each side of a square forms a diamond shape, any geometry with four sides is considered recognizable. This experiment evaluates the learning-based interpreter by comparing two design approaches based on shape rule recognition. Therefore, the comparison counterpart, the rule-based interpreter, is set to generate design patterns through hand drawing by roughly using the basic concept of the shape grammars, recognizing the ‘before’ shape in the initial shape and generating the ‘after’ shape, rather than directly use the existing rule-based interpreter with constraints on shape rules. This is also to conduct user study within a limited time.

3.1.1 Interpreter for image dataset with simple rule

The first prototype dealt with image data of a simple geometry rule. In the shape rule representation, the before shape, referred to as the left-hand-side (LHS) shape, is a square, and the after shape, referred to as the right-hand-side (RHS) shape, is a diamond shape, with each point being the center point of the original shape. I generated 2K image data pairs with Grasshopper script to make the model detect the shape rule. Each pair is randomly augmented, such as translation, rotation, and scale.

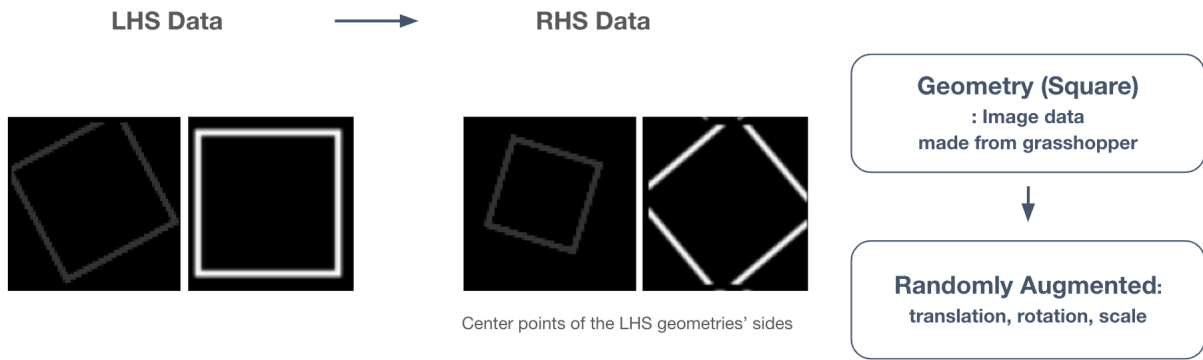


Figure 7 Data generation of the first experiment (2K)

In the research, conditional variational auto-encoder (CVAE) is used as a core structure to train the dataset and infer data patterns, which are said to be the shape rules (Kingma et al., 2014). Along with the GAN and diffusion model, VAE is one of the most popular generative models and is structured with an encoder and decoder. In the encoder part, the data features are recognized in the convolution layer and compressed in the pooling layer. While passing through the encoder, the features of the high-dimensional original image are compressed to the low-dimensional latent vector, Z , whose size determines how much to compress. Unlike auto-encoder (AE), which focuses on generating the same image as the original, VAE aims to generate similar but not identical images. Thus, based on the assumption that the feature vectors extracted to the latent space follow the Gaussian probability distribution, Z gets the distribution of the feature vectors, and by passing through the decoder, it expands to the original high-dimensional vector and generates images. CVAE has a model architecture that extracts the features and generates images better by appending conditional terms. In the experiment, LHS shapes are used as the

condition to generate the correct RHS shapes. When training and validating, LHS shape is used as a condition to recognize the shape rules in both the encoder and decoder. After training, an initial shape was input in the latent space as a condition. The interpreter model recognizes the transformed LHS shape($f(u)$) in feature vectors of the initial shape, and it can be regarded as shape recognition. Since training is conducted with data transformed, it is invariant to Euclidean transformation and recognizes $f(u)$. Then, the decoder generates a transformed RHS shape($f(v)$) according to $f(u)$. This is similar to the shape recognition in the original shape grammar interpreter. The prototype was trained with an AWS G4dn.xlarge instance powered by NVIDIA T4 GPU.

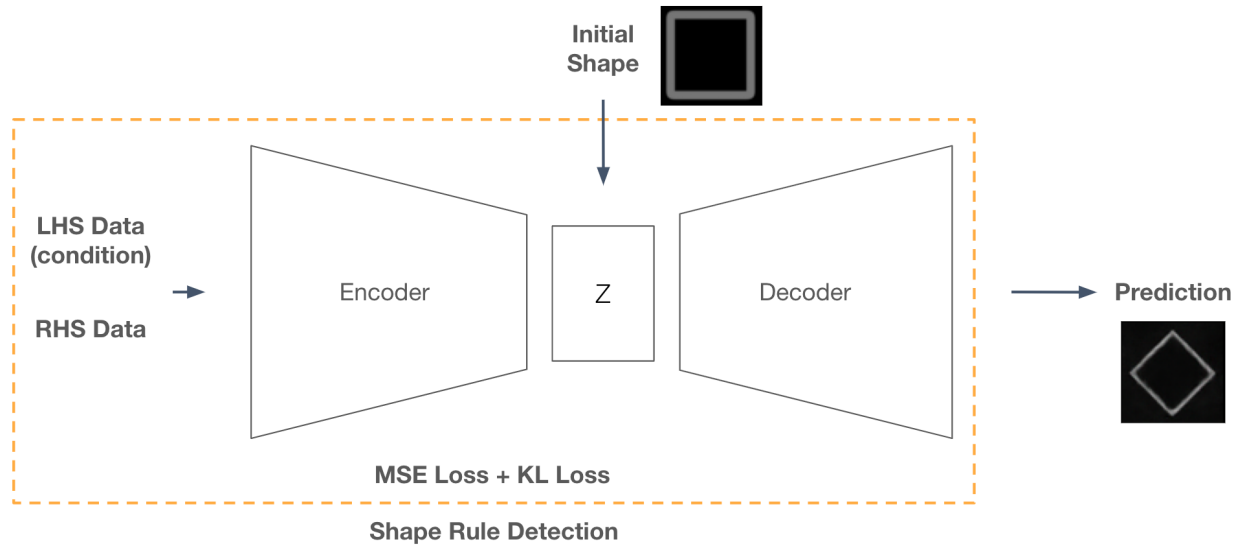


Figure 8 Model structure of the first prototype

In the first prototype, the resulting images were blurry and imprecise, and training took a lot of time. Even though images are the most common data type in the design field, this prototype was not a suitable method for user study requiring precise, exact geometry outcomes.

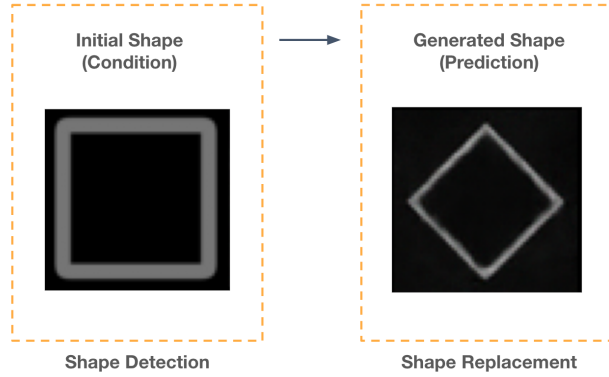


Figure 9 Result of the first experiment

3.1.2 Interpreter for vector dataset with simple rule

The second prototype dealt with point vector data using the same simple geometry rule. LHS shape is a parallelogram, and RHS shape is another parallelogram that each point is the center point of the original geometries' side. I generated 10K pair data of the geometries, which are converted to vector data based on the vertices x and y coordinates with Grasshopper python script. So, each geometry is converted to 8 vectors. I applied translation, rotation, scale, and shear for augmentation to make variations in the data.

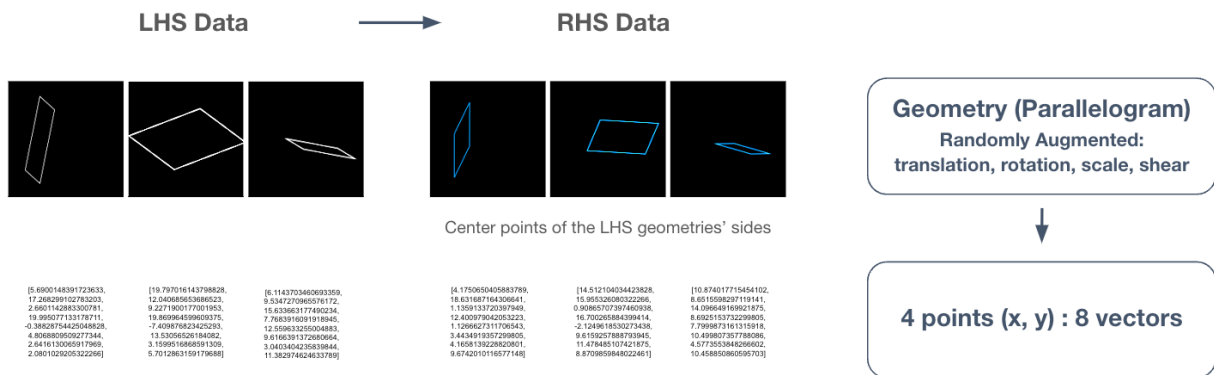


Figure 10 Data generation of the second prototype (10K)

This preprocessed dataset was trained by passing through the neural network, which is the basic structure of deep learning. The network comprises six linear layers with batch norm, activation function (GELU), and dropout. When data passes through the neurons of each layer forward and backward, the difference between prediction and the correct value gets smaller. To

calculate the loss stably and reconstruct the shape easily, I made the dataset have values between -1 to 1 instead of using normalization of the values. The prototype was trained with an AWS G4dn.xlarge instance powered by NVIDIA T4 GPU. After the training, shape rules were detected, and I input the initial shape converted to vector data. The interpreter recognized the data and generated the predicted vector data based on the rules. Then, the Grasshopper script took that vector data and turned it into the result shape.

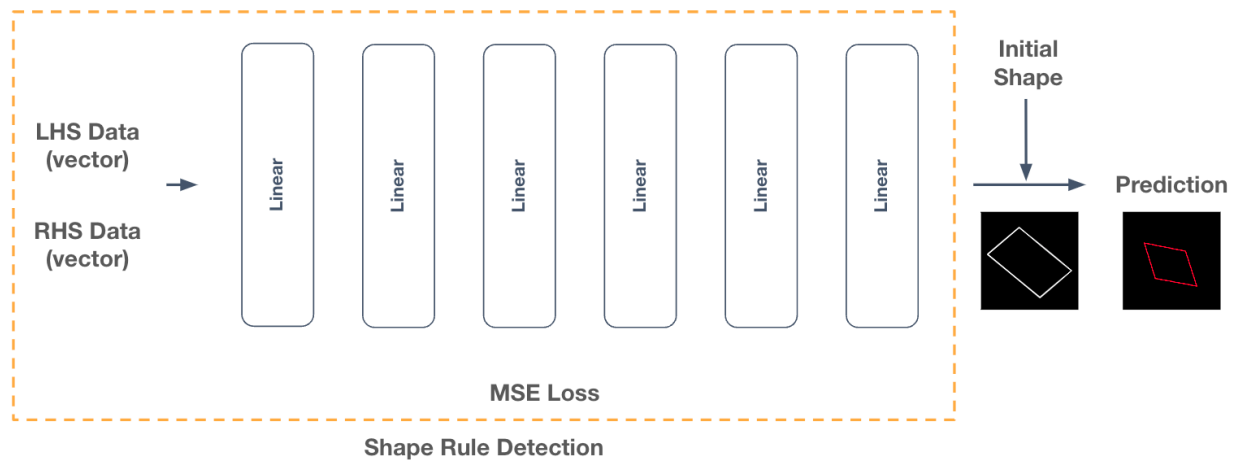


Figure 11 Model structure of the second and third prototypes.

It showed about 95% average accuracy, meaning it can predict the RHS shape with 95% accuracy. Since basic geometries do not require many vectors, the training speed was fast, and the results were accurate.

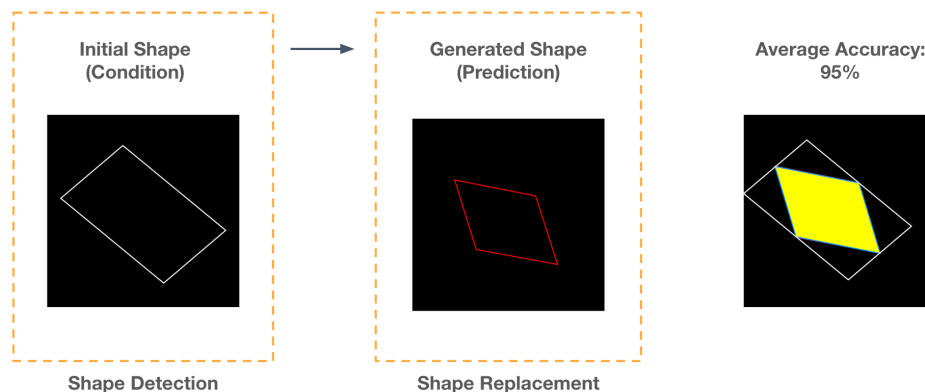


Figure 12 Result of the second prototype

3.1.3 Interpreter for vector dataset with complex rule

In real-life design problems, when a shape is changed by a shape rule, the possible outcome is usually one of several shapes instead of only one. For example, in the case of the grammar of Taiwanese traditional vernacular dwellings, the grammar of building dimensions such as height, depth, and width does not have a one-to-one relationship but has possible ranges following specific constraints (Chiou and Krishnamurti, 1995). When the shape rule is not precise, like in this case, recognizing shape rules is more challenging and time-consuming, but deep learning may be better suited to this task since it is basically a predictive tool based on probability.

Thus, the third prototype was designed to deal with a vector dataset with more complex rules. Unlike the previous prototype with the exact location of eight-point vectors as training data, in this prototype, LHS shapes were polygons with various numbers of vertices, and RHS shapes were scaled-down shapes in random ratios in certain scale constraints. To increase the accuracy of the advanced problem, I generated more data (90K) than for the previous prototype (10K) and augmented them. The model generation was similar, but manipulating the code was needed, mostly in part related to data processing. The initial shape was converted to vector data of the vertices' coordinates. The interpreter detected the coordinate data and generated the predicted vector data based on the rules. Then, the Grasshopper script took that vector data and turned it into the result shape. The prototype was trained with an AWS G4dn.xlarge instance powered by NVIDIA T4 GPU.

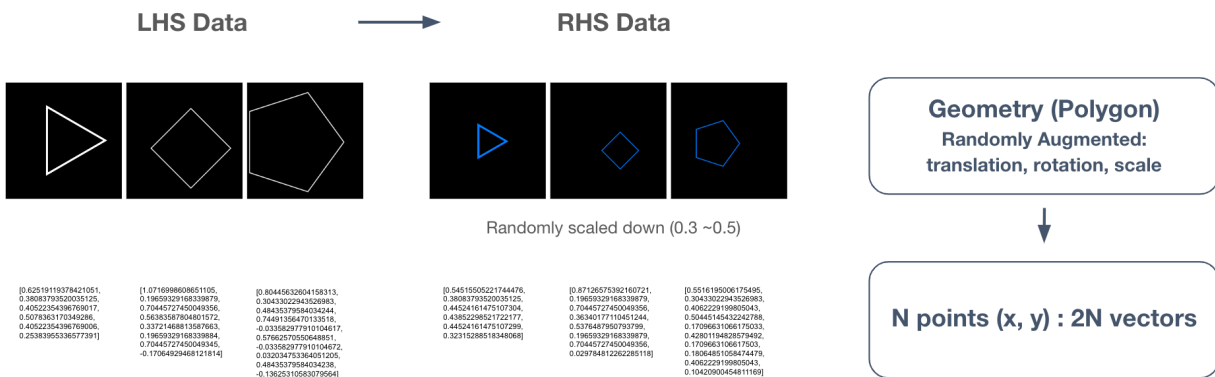


Figure 13 Data generation of the third prototype (90K)

It showed about 85% of average accuracy. Because the condition had constraints of possible data and each LHS shape was not the same number of vector data, it might be less apparent for the interpreter to recognize the shape rules than the previous condition.

Lastly, the second and third prototypes were used for the user study.

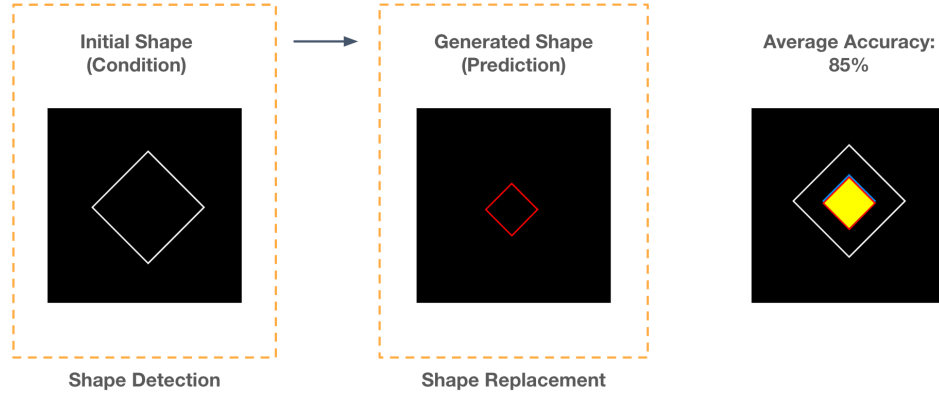


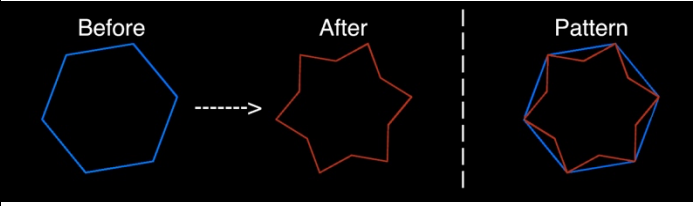
Figure 14 Result of the third prototype.

3.2 Interpreter design for the user study

3.2.1 Data design

To exclude each participant's domain knowledge or experience and to conduct the experiment in a limited time, the shape rules used in the study were designed with simple geometries that didn't require background knowledge to understand. Two different shape rules with different challenges were designed for the user study. As Table 1, the first rule turns the hexagon into a star-like shape by moving the center point of each side of the hexagon by a predetermined ratio in the direction of the center of the hexagon. The input shape and the output shape have a one-to-one correspondence. The challenge of this rule is that LHS shapes continue to change, even though they have the same topology. On the other hand, the second rule rotates the hexagon at a random angle within a specific range. Therefore, the output generated from the input under this rule is one of the virtually infinite possible shapes rather than one precise shape. As mentioned in the previous chapter, this case better reflects the real-life design problem. In addition, the scale also changes at a constant rate.

Table 1 Given rules and challenges.

Rule	Examples of shapes	Challenges
Rule1 ‘Sharpening Stars’ : midpoints for hexagon (inner points in star-like shape) move 10% towards the center’		‘Before’ shape continues to change.
Rule2 ‘Rotation at random angle (35° – 50°) + 5% scale down’		‘After’ shape is one of possible shapes. (not having one-to-one correspondence)

The shape data consists of the x and y coordinates of the vertices (#: 1, 3, 5, 7, 9, 11), the midpoint of each side (#: 2, 4, 6, 8, 10, 12), and the center point of the hexagon (#: 0). Each dataset consisted of 100K pairs of LHS and RHS shapes, which were generated by applying the shape rules to the LHS shapes.

3.2.2 Interface design

Based on the structure of a learning-based shape grammar interpreter implemented above, a user interface for the designers was designed. In this research, the learning-based interpreter was integrated with Rhino Grasshopper and Google Co-lab, consisting of three main parts: Data display, Shape rule recognition, and Shape generation.

While designers can view the dataset in various ways, the dataset with the simple shape rules for the user study was represented on the Rhino space since it was generated by the investigator using the Grasshopper script. Since the data is in list format, the interface was designed to represent the LHS shape (‘Before’ shape), RHS shape (‘After’ shape), and combined shape of LHS and RHS shapes (‘Pattern’ shape) of the selected index number using a slider on the Rhino space.

Learning-based Shape grammar interpreter

1. Samples of pair shapes

You can see the given pair shapes by choosing the number of pairs of shapes. They have the same shapes as the previous session. The blue-colored shapes are the ones before the shape rule is applied. The red-colored shapes are the ones after the shape rule is applied. In this session, you also have the coordinates data representing the specific points of each shape. They are labeled with numbers. 'Pattern' is made by combining 'Before' and 'After' shapes. (Note: There is no correlation between pairs. You only need to think about the shape rule by focusing on the change between 'Before' and 'After' of each pair of samples.)

Rule 1

Choose pair number

Pair number 00

Rule 2

Choose pair number

Pair number 013

Figure 15 User interface of the data display part (Grasshopper screen). Participants controlled the sliders to see shape data.

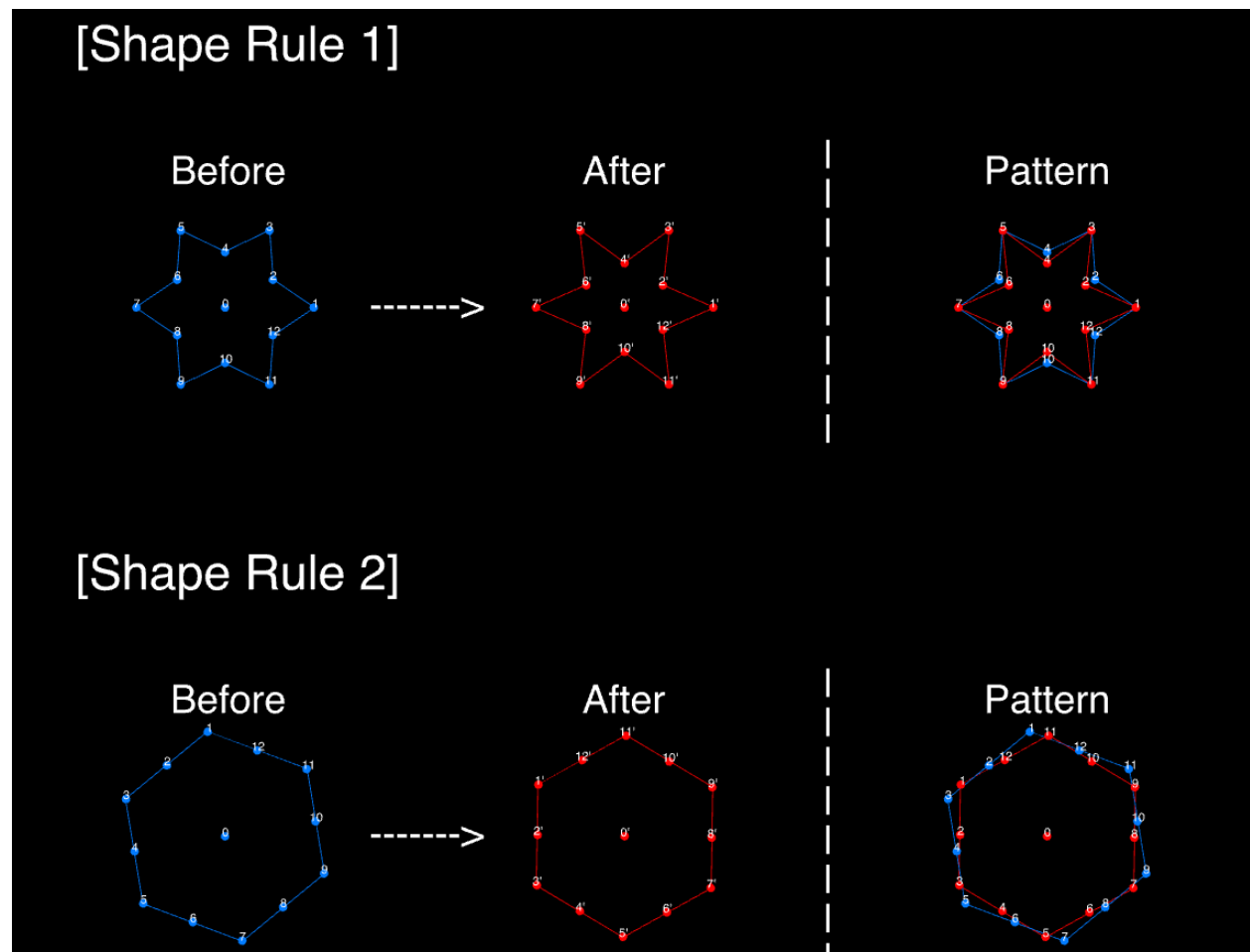


Figure 16 User interface of the data display part represents the pair shape samples (Rhino screen).

Although the three prototypes implemented before were designed, including data processing, user study should enable designers to select the data to train the interpreter and have the interpreter recognize shape rules based on the selected data. In particular, to make the interpreter accessible for designers unfamiliar with data manipulation, it was designed to handle data by selecting given shape elements, i.e., points, rather than actively manipulating the dataset. Selectable elements were represented on the shape with numerical labels.

In the second part, for the shape rule recognition, all points constituting the shapes are displayed as individual components with the numerical labels to be connected to the data collector component so that designers can collect the selected points' data. The collected point numbers are converted to a list and saved as a file by activating the toggle. The learning-based interpreter was developed in the Google Colab environment for convenient real-time training. The saved file was applied to the data preprocessing code to train the interpreter to recognize shape rules. Training was done with a Tesla T4 GPU, and training each shape rule took about 3 minutes running in 25 epochs. After completing the training, the interpreter saved the best model consisting of weights minimizing the loss value for each rule, enabling it to generate shapes.

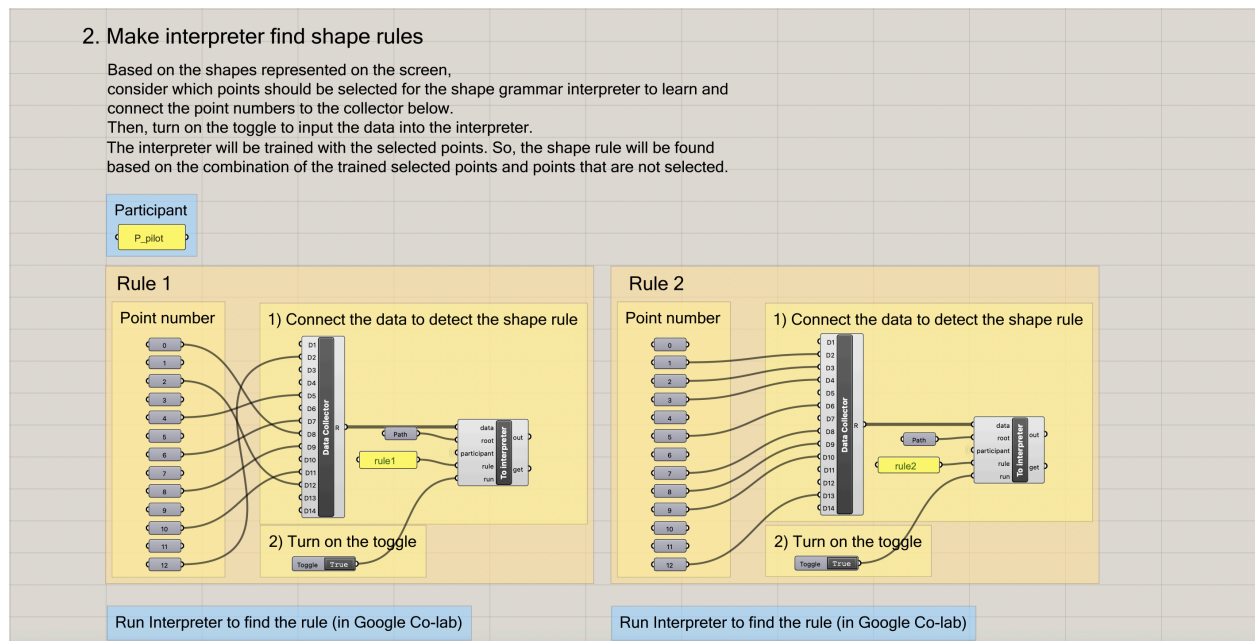


Figure 17 User interface of shape rule recognition part (Grasshopper screen). Participants connected the point number to the collector.

For the last part, in the Grasshopper space, as in Figure 18, the interface was designed for the designer to input the name of the shape rule and the input shape number, starting from 0. Then, toggling is activated, inputting the corresponding data to the generation part of the interpreter. As a result, vector data applying the shape rules was generated in the Google Co-lab environment. Since the data was stored in x and y coordinate values, toggling the ‘show’ toggle on the Grasshopper space loaded the coordinates of the selected point numbers.

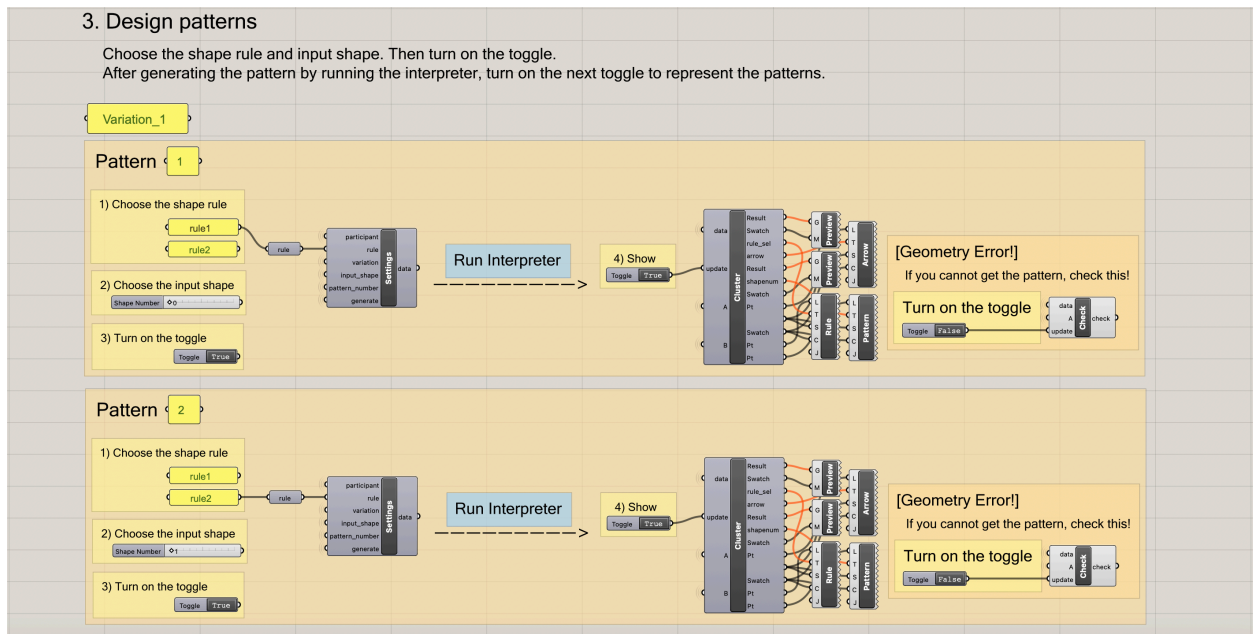


Figure 18 User interface of shape generation part (Grasshopper screen). Participants designed patterns through the interpreter by selecting rules and inputting the names of initial shapes.

At this point, the coordinates of the selected point numbers represented the locations with applied shape rules, while the coordinates of the unselected point numbers represented the locations of the input image. Connecting all points in sequential order displayed the shape on the Rhino screen.

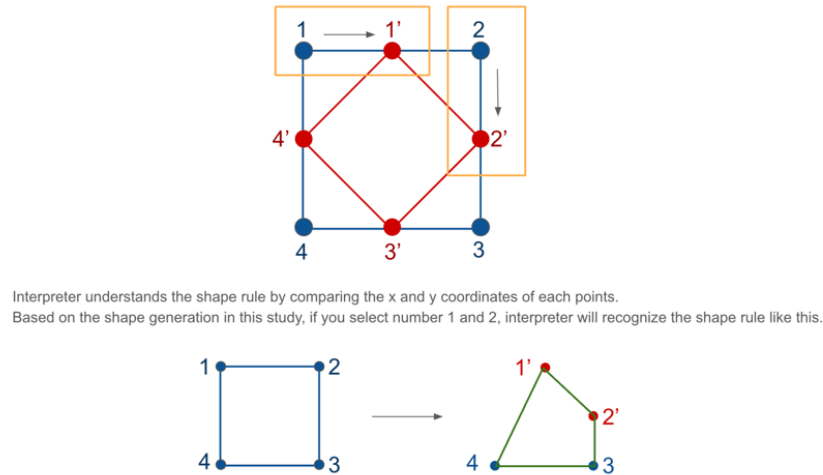


Figure 19 The logic of shape rule generation (Description used in this user study)

The shape generation process could be iterated by duplicating the code block and increasing its pattern number. In this interpreter, the generated shape was displayed as ‘shape,’ and the shape number gradually increased by 1 from the base shape, shape 0. As iterating the generation process, generated shapes could be accumulated into one shape called ‘pattern,’ the goal of the task, and the pattern number also increased by 1 from pattern 1. The patterns were displayed procedurally, representing which shape rules were applied to the shape.

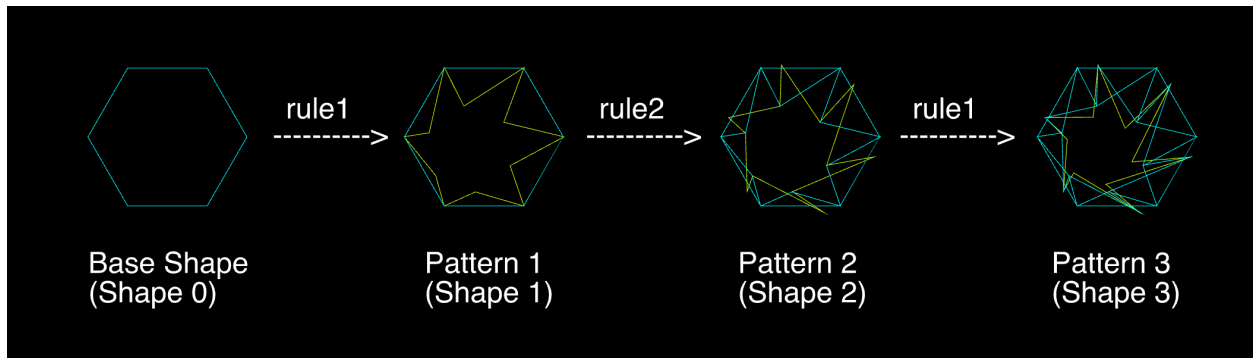


Figure 20 User interface of the shape generation part represents the patterns (Rhino screen)

If the participants select inappropriate points for finding the shape rules, patterns couldn't be represented on the screen. To represent the shape, the curves were offset, and the surface was made between two curves to be displayed. Therefore, when connecting the points to make a shape, if the points were in improper order, the shapes did not appear. In that case, the participants could check the detected shape rule by error-checking block.

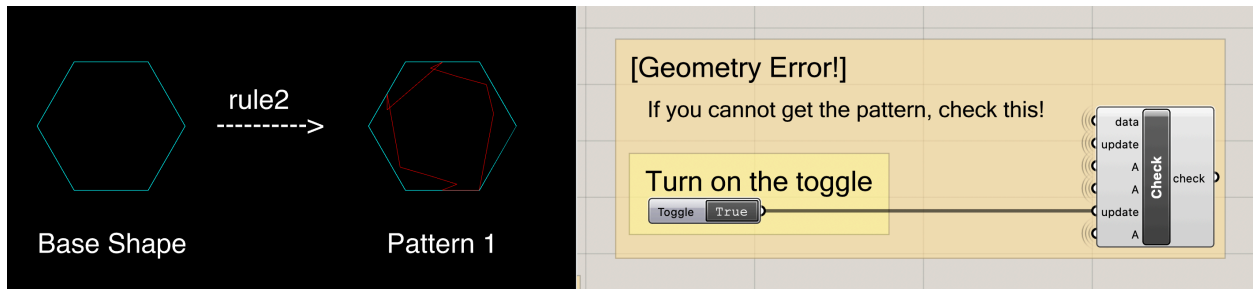


Figure 21 Error-checking block

Meanwhile, for the rule-based interpreter as a comparison counterpart, the user interface to display the given shape data in the Rhino Grasshopper space was designed to be almost the same as the data display part of the learning-based interpreter's interface. However, the shapes' points representation was not included since users did not need to handle the point data.

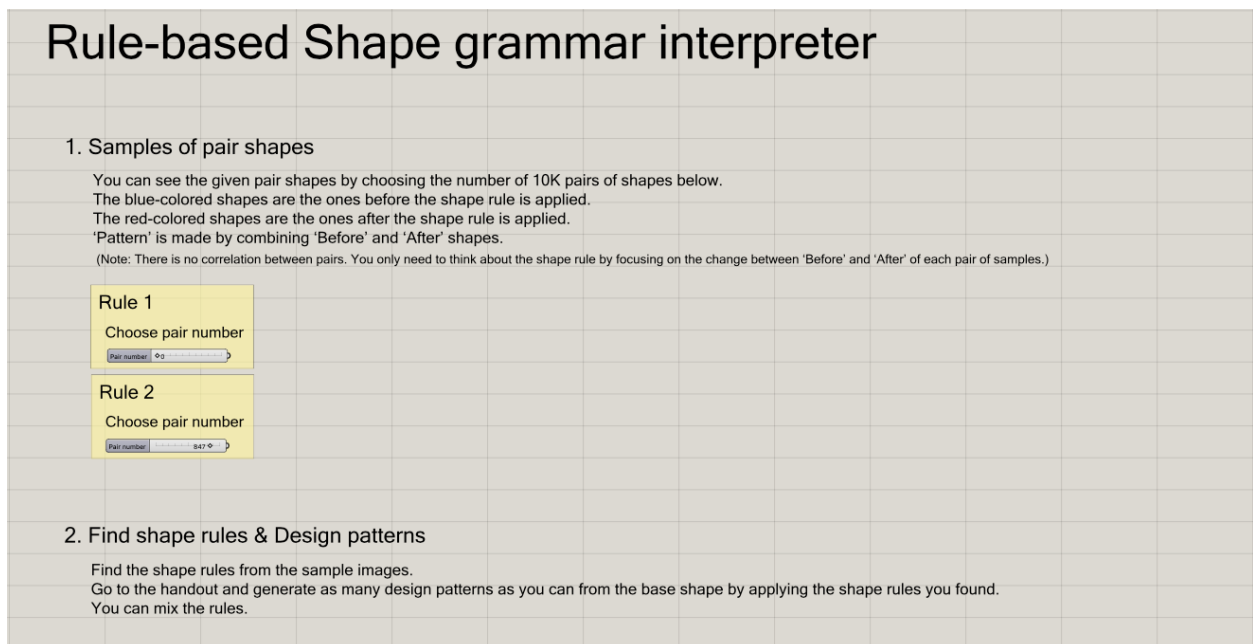
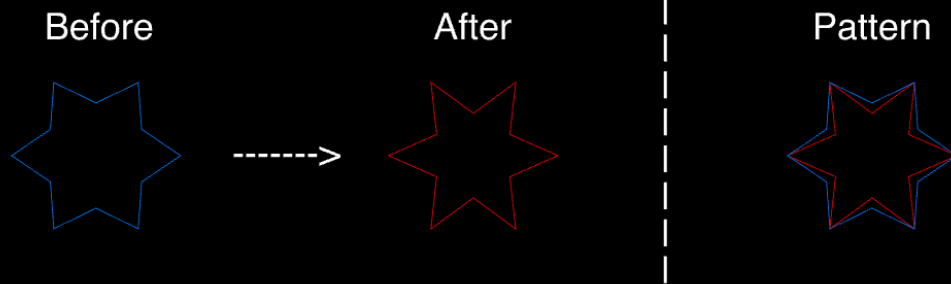


Figure 22 User interface of rule-based interpreter (Grasshopper screen). Participants controlled the slide bars to see samples of pair shapes

[Shape Rule 1]



[Shape Rule 2]



Figure 23 User interface of the rule-based interpreter represents the pair shape samples (Rhino screen).

3.3 User Study

3.3.1 Overview

Using the learning-based shape grammar interpreter generated in the experiment, I designed the user study to examine how a learning-based interpreter compares with a rule-based interpreter. This study compared two types of shape rule recognition and evaluated the learning-based interpreters.

The study consisted of two sessions; the first was about the rule-based shape grammar interpreter, and the second was about the learning-based shape grammar interpreter. Assuming that the participants were going to figure out patterns in nature or analyze specific design patterns, they were required to find the shape rules from the given samples in two different ways during each session and design as many patterns as possible by applying the shape rules they found. In this study, recognizing the shape rule was the design problem, and the design solution was expected to differ between the two sessions. After each session, participants evaluated each shape grammar interpreter and shared their design experience using them.

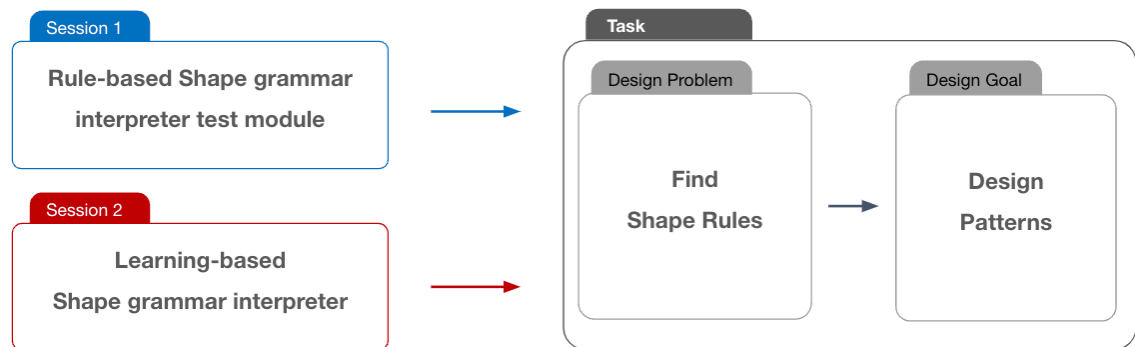


Figure 24 Task of the user study in two sessions

3.3.2 Procedure

Once the study started, an overview of the research was presented verbally, including the purpose, tasks, and data collection information. After the participants confirmed the research procedure and data collection, they completed a brief questionnaire about their experience with design and data (Figure 25). These questions were asked to ensure that the participants were satisfying the inclusive criteria of being graduate students in architecture or design departments with

design experience and to determine the correlation between previous experience and the design approaches in this user study.

The screenshot shows a questionnaire section titled "Personal Experience" in a blue header. Below the header, there is a light gray box containing the text: "The following data will be used as a reference to understand the individual experience." This is followed by three white boxes with rounded corners. The first box asks "Which degree program are you in?" and has three radio button options: "Undergraduate program", "Master's program", and "Doctoral program". The second box asks "What program are you in? (e.g. Computational Design in SoA)" and has a text input field with the placeholder "Your answer". The third box asks "How many years of design experience? (including in academia and work)" and has a text input field with the placeholder "Your answer". The fourth box asks "What experience do you have with AI and data ? And how many years?" and has a text input field with the placeholder "Your answer".

Figure 25 Section 1-2 of the questionnaire, asking about experience with design and data

The first session was about the rule-based interpreter. The participants received a handout and Rhino Grasshopper file for this session. I explained the concept of a rule-based interpreter, the given shape dataset, how to see the given shapes through the Rhino Grasshopper interface, and the tasks in the session. The tasks were to find the shape rule by themselves from the given samples of pair shapes and then to design patterns in 20 minutes. The participants could see a thousand pairs of sample images to identify each shape rule. Then, they were required to write down or draw the shape rules and design patterns by repeatedly applying the rules to the base shape (initial shape) on the provided handout. One hexagon was given as a base shape for the generation task, and the participants could mix two shape rules to design the pattern. How the

participants recognized the shape rules, how long it took to obtain them, and how they applied them to generate design alternatives were observed.

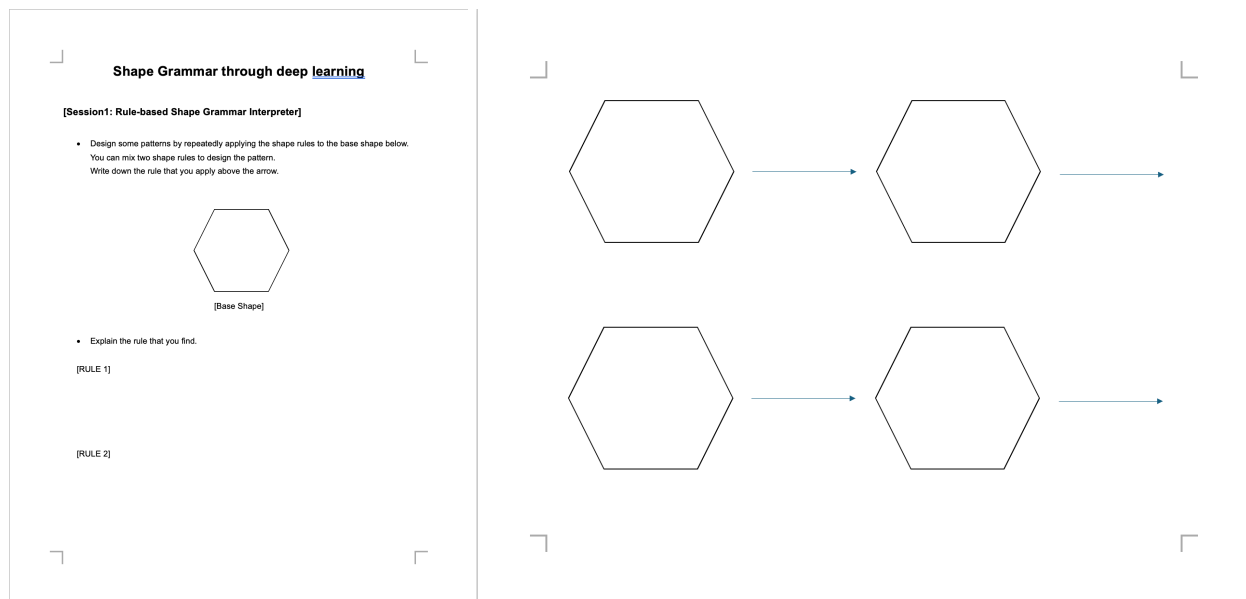


Figure 26 Handout design for session 1

After finishing the design, the participants were asked to complete a questionnaire consisting of six questions about their experience with the rule-based interpreter (Figure 27). The purpose and the hypothesis of each question are shown in Table 2.

Table 2 Design of the questionnaire

Question	Purpose (Hypothesis)
1. What did you consider when recognizing the shape rule?	To understand the solution of the recognizing the rule (Rule-based: It will be different depending on designers. / Learning-based: It will be similar between designers, but different from the rule-based one.)
2. Did you use background knowledge to recognize the shape rule? If it was used, what kind of background knowledge? How did you use it?	Background Knowledge
3. How challenging was it to recognize the shape rules?	Usability, Challenge

4. Are the resulting patterns similar to what you expected? (only for the learning-based interpreter)	Usability
5. How satisfied were you with the resulting designs?	Satisfaction
6. How did this way of finding the rules (rule detection) affect the design alternatives you generated?	To understand the designers' experiences generating the design results by using the shape rules (Designers will be able to generate results with relatively more flexible shape rules in rule-based interpreter, but results will be limited by designers' own knowledge.)
7. How would you rate your experience in this design process?	Enjoyability
[Wrap-up] How would you compare two shape rule interpreters in the aspects of the design approach and generation of the design alternatives?	To understand the core idea of the user study

Rule-based Shape Grammar Interpreter

Share your experience with a rule-based shape grammar interpreter after the first session.

1. What did you consider when recognizing the shape rule?
Your answer

2. Did you use background knowledge to recognize the shape rule? If it was used, what kind of background knowledge? How did you use it?
Your answer

3. How challenging was it to recognize the shape rules?

1

2

3

4

5

Very easy
☐
☐
☐
☐
☐
Very difficult

4. How satisfied were you with the resulting designs?

1

2

3

4

5

Not satisfied
☐
☐
☐
☐
☐
Very satisfied

5. How did this way of finding the rules (rule detection) affect the design alternatives you generated?
Your answer

6. How would you rate your experience in this design process?

1

2

3

4

5

feel bored
☐
☐
☐
☐
☐
enjoy

Figure 27 Section 3 of the questionnaire, asking about their indirect experience with the rule-based interpreter.

The second session was about the learning-based shape grammar interpreter. The participants received a handout and Rhino Grasshopper file for this session (Figure 28). In this session, the shape rules and the final goal of the task, designing the patterns, were the same as in the previous session, but they were required to find the rules using the learning-based interpreter.

I explained the concept of a learning-based shape grammar interpreter, the given shape dataset, how to choose the data, and how to generate patterns through the Rhino Grasshopper interface.

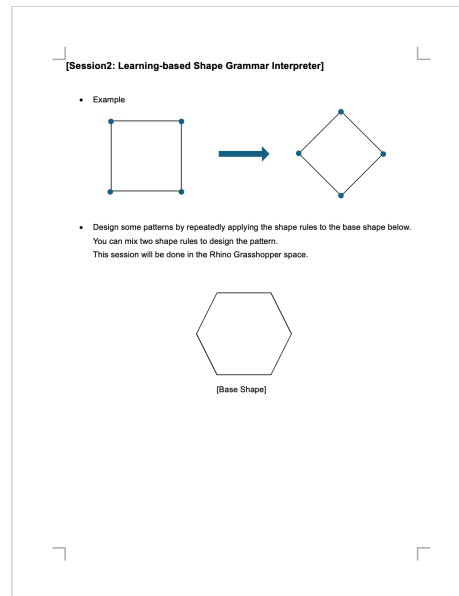


Figure 28 Handout design for session 2

Based on the shapes represented on the screen, the participants needed to connect the point numbers to the collector components, considering which points should be selected for the interpreter to learn the shape rules. As the participants turned on the toggle to input the data into the interpreter, it was trained in Google Co-lab. After training the interpreter for the shape rules, the participants were required to design patterns with the interpreter. The base shape for pattern generation was the same as the previous session. They chose the shape rule they wanted to apply and input the shape number. After they turned on the toggle to input the data into the interpreter, the code for generation in Google Co-lab was run. The participants could see the results by turning on the next toggle.

After finishing the design, the participants were asked to complete a questionnaire consisting of seven questions about their experience with the learning-based interpreter and a wrap-up question comparing the two interpreters (Figure 29). The purpose and the hypothesis of each question are shown in Table 2.

Learning-based Shape Grammar Interpreter

Share your experience with a learning-based shape grammar interpreter after the second session.

1. What did you consider when recognizing the shape rule? Why did you select those numbers?

Your answer

2. Did you use background knowledge to recognize the shape rule? If it was used, what kind of background knowledge? How did you use it?

Your answer

3. How challenging was it to recognize the shape rules?

Very easy

1

2

3

4

5

Very difficult

4. Are the resulting patterns similar to what you expected?

Not at all

1

2

3

4

5

Very much

5. How satisfied were you with the resulting patterns?

Not satisfied

1

2

3

4

5

Very satisfied

6. How did this way of finding the rules (rule detection) affect the design alternatives you generated?

Your answer

7. How would you rate your experience in this design process?

Feel bored

1

2

3

4

5

Enjoy

[Wrap-up] How would you compare two shape rule interpreters in the aspects of the design approach and generation of the design alternatives?

Your answer

Figure 29 Section 4 of the questionnaire, asking about their experience with the learning-based shape grammar interpreter.

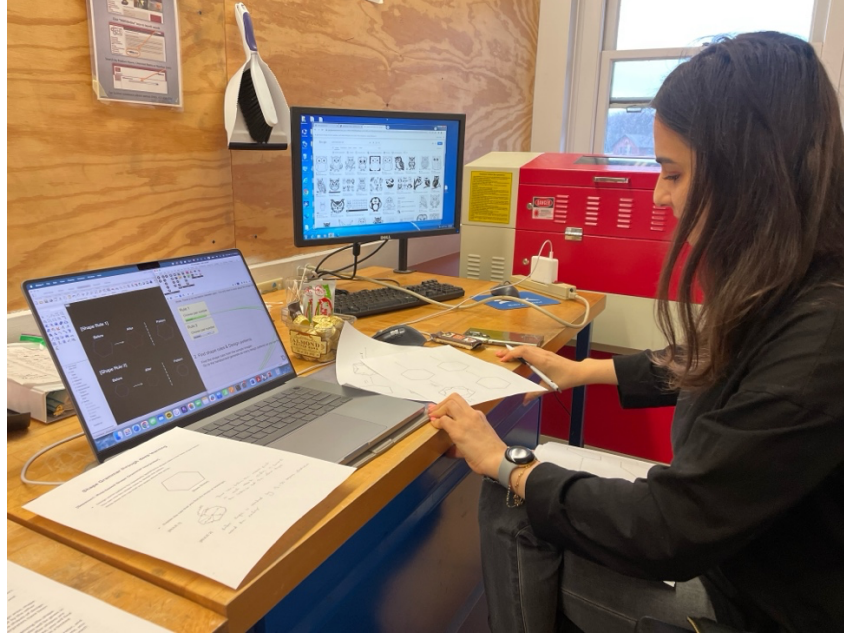


Figure 30 User study was conducted in person individually

The whole study took approximately 70 minutes for each participant, and it was conducted in person in a small, quiet room on the 4th floor in the Margaret Morrison building of Carnegie Mellon University (Figure 30, Table 3).

Table 3 Schedule of the study.

Order	Activity	Time
1	Introduction of overall study including the completion of the consent form and questionnaire	5 min
2	Introduction of the rule-based shape grammar interpreter	5 min
3	Design with the rule-based shape grammar interpreter	20 min
4	Questionnaire for the rule-based shape grammar interpreter	5 min
5	Introduction of the learning-based shape grammar interpreter	10 min
6	Design with the learning-based shape grammar interpreter	20 min
7	Questionnaire for the learning-based shape grammar interpreter	5 min

Targeted participants were primarily found at SoA (School of Architecture) or SoD (School of Design) in CMU, where most students have experience in design. I recruited participants by sending e-mails to students in both departments. This study was conducted with 13 participants.

Participants' behavior was observed by the researcher. Their design outcomes drawn on the paper and the Rhino Grasshopper interface were collected. In addition, participants' departments, design experience, and survey results from the study were collected through questionnaires.

4. Results

According to the survey questions, to understand participants' knowledge about design and data, they have experience in design ranging from 5 to 15 years and had different experiences with data or AI (Figure 31).

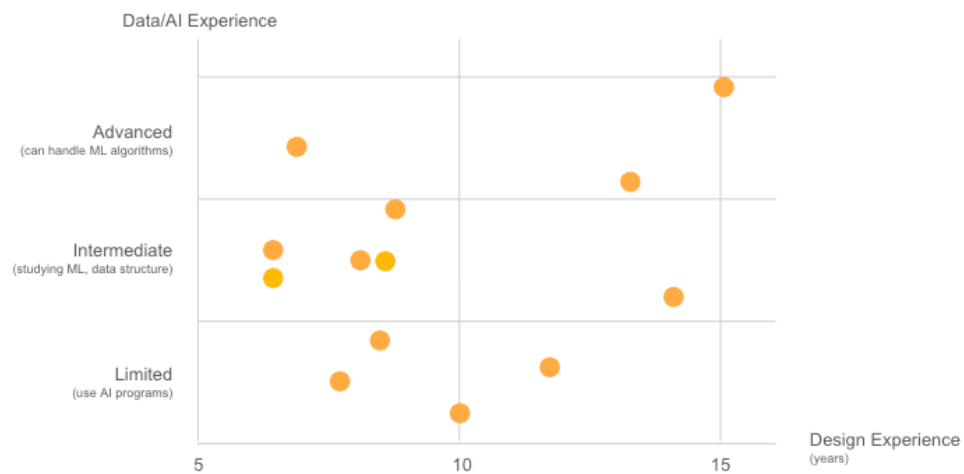


Figure 31 Participants' experience with design and data/AI

4.1 Design outcomes

I could categorize the participants' design patterns for each session according to their shape rules.

4.1.1 Session 1

For the first session, which was done with the rule-based interpreter, the participants tried to recognize the given shape rules. Even though they tried to find the correct shape rules, few people found the given rules. As shown in Figure 32, about half of the participants understood rule 1 using fractal logic rather than original logic. For rule 2, only a few participants caught the scale change, and most were not interested in or felt hard knowing the rotation angle. Some participants found it challenging to recognize the rules themselves. A few participants understood the shape rules by considering the shape between pairs even though they knew there was no correlation. The participants could generate various patterns with their recognized shape rules. However, many participants complained of difficulty in drawing complex patterns by hand.

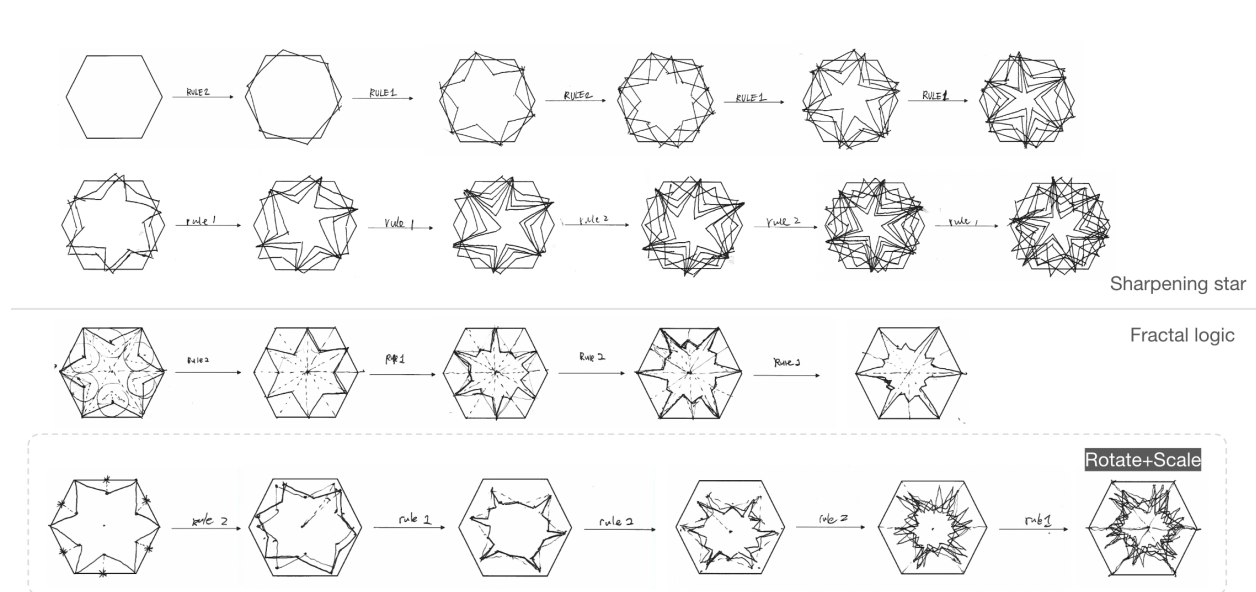


Figure 32 Design outcomes by some participants in Session 1

4.1.2 Session 2

For the second session, which was about the learning-based interpreter, the participants found it challenging to identify the given shape rule for rule 2 compared to rule 1. To discern the original shape rule, the participants should select at least a set of points: even-numbered points for rule 1 and all points except the center for rule 2. However, only two out of the entire group could get the given shape rules. Half of the participants selected the odd-numbered points instead of all points to identify the shape rules for rule 2, which led to an unexpected pattern. They chose them because they considered the data based on one of the generation logics of Rhino Grasshopper geometries rather than the logic of interpreters. Some participants who understood machine learning well intentionally selected other points as training data. The others who had no knowledge of data or AI played around with the point data to see what the interpreter generated. Due to the limited time for the user study, each participant could only train once for each rule, preventing them from trying various rules. Since the task of the experiment was to recognize the shape rule inherent in data, participants could generate different patterns if they deliberately selected different points. Nevertheless, many participants were eager to discover different shape rules compared to Session 1.

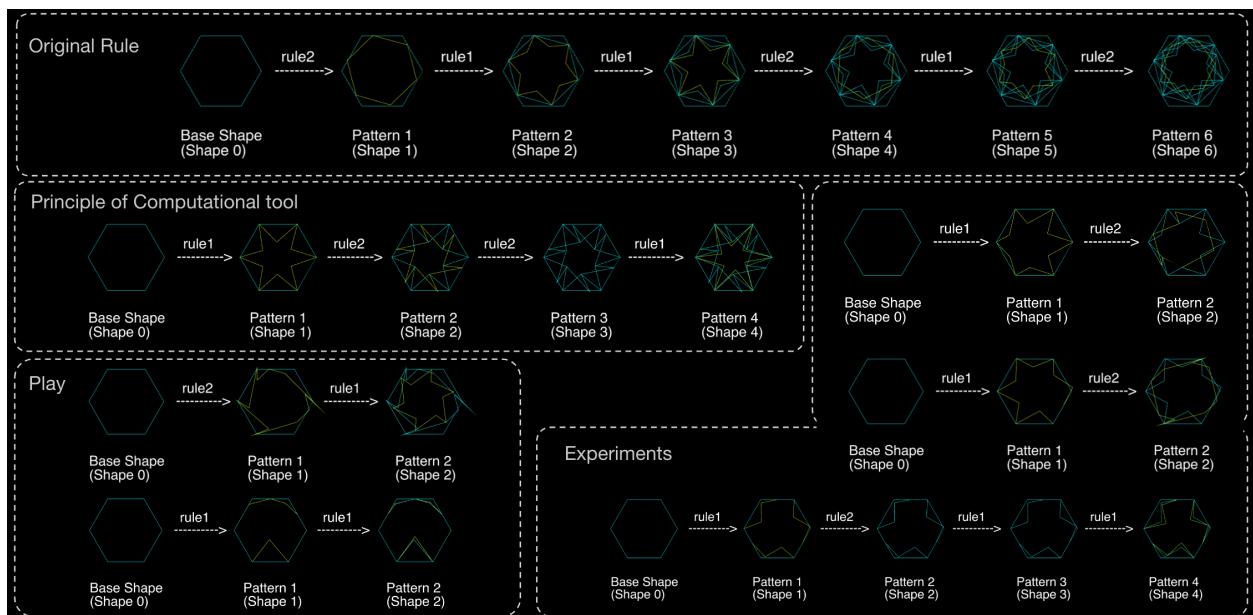


Figure 33 Design outcomes by some participants in Session 2

4.2 Results of the questionnaire

Based on the observations and the survey question, I compared the two interpreters in terms of the designer's thinking, usability, and emotion.

4.2.1 Designers' thinking

Through question 1, 'What did you consider when recognizing the shape rule? Why did you select those points?' the study examined how users understood shapes when the interpreter changed. With the rule-based interpreter, participants compared geometric properties such as vertex positions, number or length of edges, movement direction, or angles. It could be either comparing individual geometric elements or entire shapes. They also recognized shape rules by reminding familiar design logic, such as fractals. On the other hand, with the learning-based interpreter, participants found the shape rules by trying to consider how computers process. To find the shape rules of the given samples, they needed to select points and train the interpreters by comparing whether the point locations of 'before' and 'after' shapes were the same. However, more than half of the participants selected points based on the principles of geometry generation in Rhino Grasshopper. Despite being informed in advance about the interpreter's structure in which the shape is generated after training, connecting all points in turn, they did not select midpoints when training for rule 2. This was because typical rotation commands were done through vertices. Some participants randomly selected points to understand the logic from the results due to not knowing the data, even choosing phone numbers as if playing a game. Question 2 asked what background knowledge was used to better understand designers' thinking. Most participants mentioned that they used similar general design and geometric knowledge or shape grammar knowledge in Sessions 1 and 2. Only around 10% of participants said using machine learning knowledge.

In question 5, 'How did this way of finding the rules (rule detection) affect the design alternatives you generated?', and in the wrap-up question, 'How would you compare two shape rule interpreters in the aspects of the design approach and generation of the design alternatives?', participants commented the strengths and weaknesses of the two interpreters as shown in Table 4.

Table 4 Some comments from participants about two shape grammar interpreters

		Comments
Rule-based	Pros	“I can generate more complex shapes than what I initially thought using shape rules and base shapes” “This made it easier to understand and employ the rules effectively” “The rules guide me to find a controlled manner to create complexity” “More sense to me” “More direct and straightforward working for engineering problem(goal-oriented)” “Easy to adapt to and more digestible”
	Cons	“It is relatively limited to generative enough variations and fewer rounds of design since the rules are relatively restrictive.”
Learning-based	Pros	“It is helpful to understand actual logic of the shape rule which were not the same as the participants found” “Faster and easier way” “Do not need to analyze by myself” “Provide more design opportunities by building another data manipulating method” “Unexpected results make me wonder what computer defined the rules and how it found the rule. This would be helpful in creating a design that I didn't think of and would allow me to explore more.” “More DIY patterns are possible while thinking of choosing the data points”
	Cons	“Due to the lack of knowledge about how computer deal with the data, it was hard to find the shape rules that I expect.” “I need to try several tests to know how it works.” “Not knowing the outcome after setting up the rule.” “Training accuracy and computation time”

4.2.2 Usability

From question 3, 'How challenging was it to recognize the shape rules?', the usability of the two interpreters could be compared (Figure 34). Since simple geometries were provided as the samples, most participants found it similarly easy to recognize rules using both interpreters. However, finding rules using the learning-based interpreter was more manageable for those with a better understanding of data and AI. The time taken to recognize the shape rules varied for each person, ranging from 2 to 20 minutes in Session 1, with an average of around 7 minutes. In contrast, in Session 2, it typically took 1 to 5 minutes to select point data for the two shape rules together, with each rule requiring 3 to 4 minutes of training time, totaling about 10 to 15 minutes for detecting shape rules. However, most of the time was spent on the interpreter extracting the rules rather than the participant. In addition, since most participants could not predict the results of the learning-based interpreter, they were unable to control the generated shape rules.

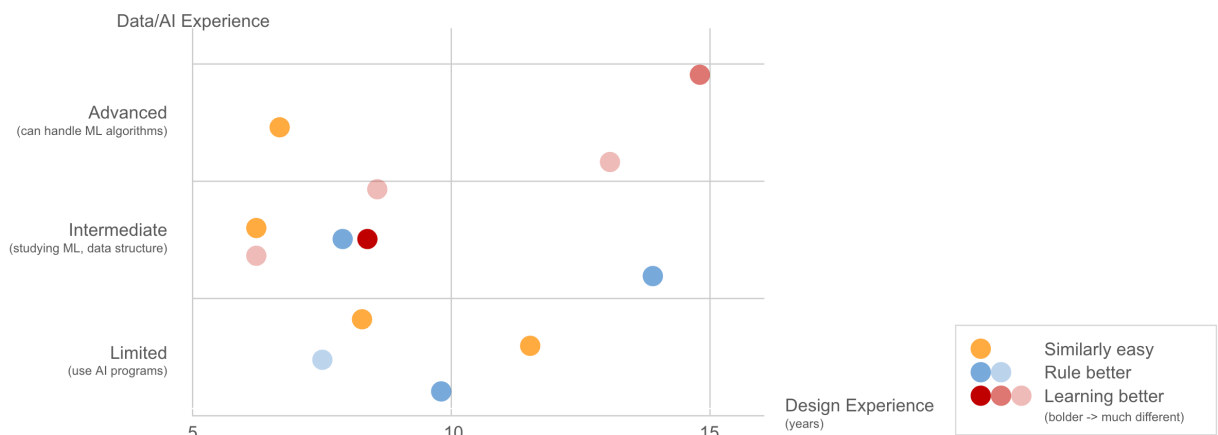


Figure 34 Evaluation about usability

4.2.3 Emotion

Firstly, through question 4, 'How satisfied were you with the resulting designs?', and observations, the satisfaction with the two interpreters was compared (Figure 35). About 60% of participants were satisfied with the learning-based interpreter as it produced unexpected patterns. However, when the learning-based interpreter failed to generate predictions due to errors during the training process or when it could not generate valid shape rules because of improper point

selections, participants were dissatisfied with the outcomes and stated that the rule-based interpreter was better. However, some of them mentioned that it would be understandable and useful if they could conduct experiments multiple times with the interpreter to generate shape rules. About 15% of participants did not feel satisfied with the design patterns generated by both interpreters due to the limited rules in shape grammar itself.

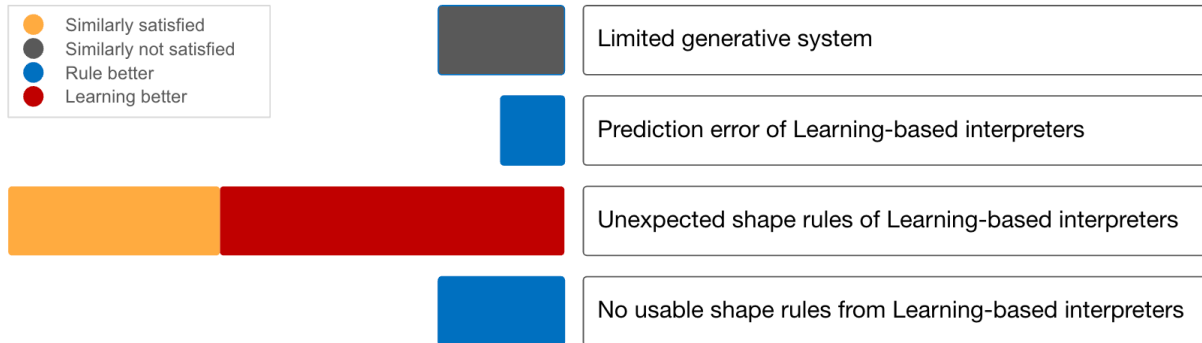


Figure 35 Analysis of participants' satisfactory

Through question 7, 'How would you rate your experience in this design process?', the enjoyability of the two shape grammar interpreters was evaluated (Figure 36). Excluding those who found shape grammar itself uninteresting, most participants enjoyed generating patterns using the shape grammar interpreter. However, when testing with the rule-based interpreter, some participants quit designing patterns during Session 1 because they could no longer design new patterns or became bored. While most participants found the learning-based interpreter enjoyable due to unexpected outcomes, some mentioned that the rule-based interpreter was more enjoyable when the outcomes of the learning-based interpreter were not satisfactory or when they did not understand the data.

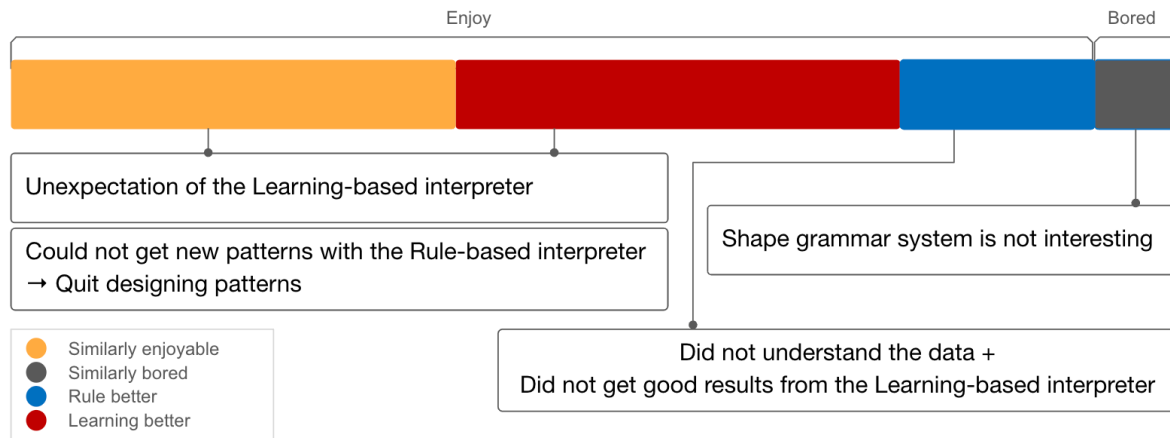


Figure 36 Analysis of participants' enjoyability

5. Discussion

5.1 Findings

Through the user study, I evaluated the learning-based shape grammar interpreter compared to the current rule-based shape grammar interpreter in terms of designers' thinking, usability, and emotions. I investigated the following.

1. When directly understanding the shape rules for input into the rule-based interpreter, participants interpreted shapes differently, and the learning-based interpreter could provide other shape rules.

About the rule-based interpreter, people who focused on the overall shape easily understood the shape rules and generated design patterns without considering the rate of the change. On the other hand, people who examined the shape rules specifically found it challenging to detect the rules due to the ambiguous rate of the change in rule 2. The small rate was hard to recognize. Meanwhile, while the random variations in rule 2 were designed to make rule recognition difficult, it was unexpected that many participants recognized rule 1 as their familiar design logic, fractals. Therefore, the experiment in Session 1, which aimed to recognize the shape rules through the rule-based interpreter, demonstrated the potential for the learning-based interpreter to expand the rule space. This is because designers could obtain rules that are different from what they initially thought by selecting the point data in the learning-based interpreter. However, this relied on basic knowledge of data.

2. When designers use the learning-based interpreter, they aim to input data for rule generation based on the relationships between data, unlike the rule-based interpreter. However, they can interpret and control the shape rules only if they have prior knowledge.

Initially, I thought the learning-based interpreter would rely less on designers' design knowledge. Participants with knowledge of machine learning could understand data sufficiently, select data by considering the relationships between data, and interpret shape rules based on the design outcomes. Therefore, they evaluated the learning-based interpreter for providing various

unexpected but interpretable shape rules. On the other hand, most participants who lacked this knowledge found it frustrating that they could not understand the process behind why the rules were generated in this way, making it uncontrollable. Some of them input data for obtaining shape rules using their design experience in generating shapes with other CAD tools. Even designers with limited experience in CAD tools selected data without any logic, similar to random selection. However, many participants mentioned that it would have been easier to understand the interpreter and data logic if they could try various rules. By experimenting with different shape rules, they could understand it better and expand the rule space. This demonstrates the potential of the learning-based interpreter as an educational tool for understanding shapes through data.

3. Through the experiment, it was evident that selecting only point data in the learning-based interpreter is an easy method for generating rules without much effort. However, ease of action is not equal to ease of design decision-making. Those without knowledge of data found it challenging to make decisions.

4. Since shape grammar is a generative system where shape rules are the key factor for exploring design solutions, expanding the rule space can give users satisfaction by generating various designs (Strobbe et al., 2015).

The characteristic of shape grammar interpreters, which generate shapes based only on the given rules, coupled with the fact that only two shape rules were given in the user study, led some participants to feel less enjoyment in the design method. They stopped designing more patterns despite having remaining time. However, participants found satisfaction and joy in the learning-based interpreter's ability to generate unexpected shape rules. Many participants, whether knowledgeable about data or not, were interested in experimenting, like playing a game. This highlights the potential of the learning-based interpreter as an enjoyable tool for easily exploring shape rules. It allows designers to generate shape rules beyond those they typically conceive, expanding the rule space and potentially resulting in a wider range of design outcomes, ultimately enhancing designer satisfaction.

5.2 Limitations

5.2.1 Design of tool

In this user study, the investigator designed the interpreter using the simple geometrical point coordinates data, and the interpreter was trained through points selected by users as low-level data manipulation. This is because of the assumption that data is not a familiar area for designers and the time constraints. As a result, users cannot actively engage in data manipulation, which constrains the understanding and implementation of the logic of recognizing shape rules. In the future, the learning-based interpreter could be improved in the way that users actively engage in data manipulation.

The original shape grammar interpreters recognize all sub-shapes in the initial shape that match the LHS shape and change them into the RHS shape. However, the learning-based interpreter in the research compares the entire initial shape with the LHS shape. Therefore, strictly speaking, shape recognition in the learning-based shape grammar interpreter is not sub-shape recognition. The mathematical method of accurate sub-shape recognition in 2D vector geometry, used in the Shape Machine, is not proper for the probabilistic shape rule recognition of the learning-based interpreter. Meanwhile, object detection algorithms in deep learning enable sub-shape recognition with image data. However, to build a model that allows object detection with users' image data, preprocessing marking the location of each object in the image data as bounding boxes is needed. This process is not practically user-friendly as it requires marking each object individually, making it impractical.

The learning-based interpreter is not an intuitive method for shape recognition because users cannot directly understand the shape rules by selecting points. When using the rule-based interpreter, users clearly know the shape rules and input them into the interpreter by drawing the shapes. However, with the learning-based interpreter, users can visualize the trained shape rules after inputting any initial shape. Both the rule-based and the learning-based interpreter require inputting the initial images to visualize the applied results, but with the learning-based interpreter, shape rules are recognized without visually knowing them. This can be frustrating when using shape rules.

The learning-based interpreter, constructed with deep learning algorithms, fundamentally relies on probabilistic predictions. Therefore, in user tests, even if the same numbers are selected,

the results applying the trained shape rules were all slightly different. In cases where there were errors in the training, the results differed significantly from the expectations, and users could not understand why such results occurred and could be dissatisfied with the generated shapes.

5.2.2 Design of User Study

The user study comparing two shape grammar interpreters has some symmetry problems in the design and application of the interpreters in the experiment.

Although the rule-based interpreter Grasshopper components like 'Sortal GI' are available in Rhino Grasshopper, I designed the first session using not the computational tool but the hand-drawing method. As mentioned in the Methods chapter, focusing on shape rule recognition, this corresponds to a learning-based interpreter that can recognize and apply shape rules even if the shapes are not geometrically identical. Moreover, this was also due to constraints on experimental time and participants. The user study for the rule-based interpreter was initially designed to draw shapes in the Grasshopper interface computationally. However, even for computational design students, who may not always handle such tasks, it was hard to use it within a short timeframe. Therefore, the experiment was redesigned to be more accessible to the general designers, and users could not generate design alternatives computationally in the new study design. As shown in Figure 32, the participants could generate patterns even with inaccurate rules, and some of them enjoyed designing patterns by hand drawing. While there was no issue with evaluating the main idea of recognizing shape rules, as the steps became more complex, it was challenging to develop the patterns due to the gaps caused by inaccurate shape rules. Some participants thought it was not suitable for design generation due to the redundant and unclear drawings. If the first session had been designed with a computer-based interpreter, it might have achieved better balance, even though it would not have been appropriate for general designers. Instead, in this case, designers would have to input the rule accurately, which could provide designers with a different experience.

There is also some imbalance in the interface of the user study. When representing data samples, point data was displayed in the learning-based interpreter. This is because point data was only used in Session 2 as interpreter input values. Even in the rule-based method, represented points can serve as guidance for recognizing the shape rules. Therefore, the different

representations of information in both sessions may act as different conditions for the participants.

6. Conclusion

6.1 Summary

In summary, this research proposes another approach to input the shape rules through the learning-based interpreter rather than the process of recognizing the input shape and outputting the modified shape, which the existing shape grammar interpreter research has focused on. Instead of inputting shape rules by directly drawing the shapes, designers input shape data, and deep learning algorithms analyze the shape rules. To evaluate the proposed interpreter, simple geometry vectors were used as input data, with an interface designed to allow designers to select point data for shape rule recognition, considering accessibility. The interpreter, trained with the input data, can change the initial images using the recognized shape rules. Although this interpreter is a limited tool that does not support sub-shape recognition and the data and interface used in this research are experimental settings, it demonstrates an alternative approach to design generation by inputting shape rules through data. Depending on the designers' data selection, different rules are input, showing that designers can generate different results through another approach. While the learning-based interpreter based on inductively predicted value may have lower accuracy compared to the rule-based interpreter in accurately recognizing and generating precise shapes, it offers a broader range of recognizable shape rules. This suggests the potential for learning-based interpreters to support existing rule-based interpreters.

The user study showed designers' experience with each shape rule recognition. Using a rule-based interpreter for rule generation is a direct and straightforward approach to design problems. With the guidance of the shape grammar system, designers can freely generate complex shapes reflecting their intentions, which results in a sense of satisfaction and enjoyment. However, design solutions may unintentionally be limited by their experience. Additionally, some designers consider the design generation system with shape rules as restrictive and may feel bored when unable to generate new designs.

While it is important for designers to approach design problems by reflecting the design intentions in their style, they also crave generating designs with new ideas. The learning-based interpreter generates shape rules based on the data selected by designers. Although the

interpreter's processing is like a black box, the process of data-driven rule generation differs from the designer's rule recognition process, leading designers to approach problems differently and potentially produce unexpected shape rules. This can complement the rule-based interpreters by expanding the rule space. However, designers need to understand how the interpreter recognizes the rules and the underlying data. In addition, as the interpreter recognizes shape rules based on probabilistic predictions from data, it doesn't generate 100% accurate shapes. Therefore, it can be valuable in the ideation stage for designing shape rules.

The learning-based interpreter in this study can only recognize shape rules from the preprocessed dataset and model structures I generated. In reality, acquiring the desired design data and processing it in the desired way for shape rule recognition has many challenges. Gathering data is one challenge, and preprocessing the data takes a lot of time. Additionally, depending on the type of data, training can be time-consuming. If one cannot handle the data, it's impossible to design an interpreter. Therefore, using existing design data for the interpreter is not easy. Instead, following the advantages of shape grammar that fit well with educational purposes, the learning-based interpreter can be applied as an educational tool for studying design logic by finding shape rules from the shape data.

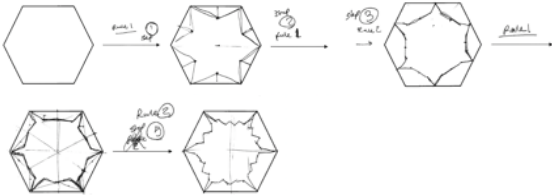
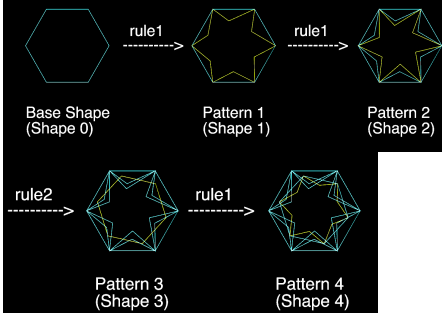
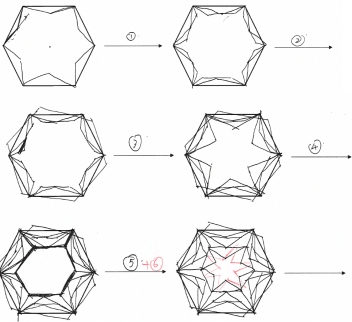
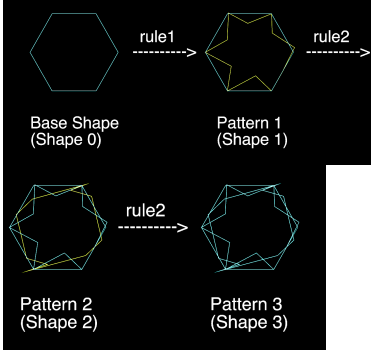
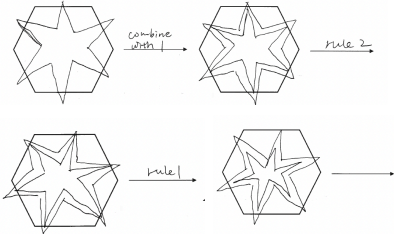
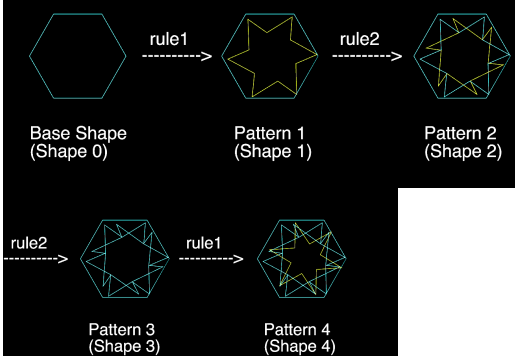
6.2 Future Works

Depending on the design context, recognizing shape rules based on the designer's expertise may bring designers closer to the desired design outcome. However, according to the user study, designers also value the unexpected design generated by the learning-based interpreter. Feeling limited by the diversity of their design generation, some designers expressed the desire for computers to aid it. The geometric patterns in the experiment served as simple examples for many design cases, showing the potential to enrich and detail the rule space by simultaneously utilizing human-generated flexible design and computer-generated unexpected design through the two interpreters. In the future, by building a system that computationally utilizes both approaches simultaneously within the same design space, we can examine how meaningful it is for designers to expand their ideas about shape rules.

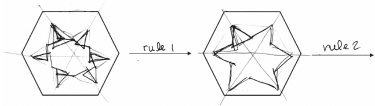
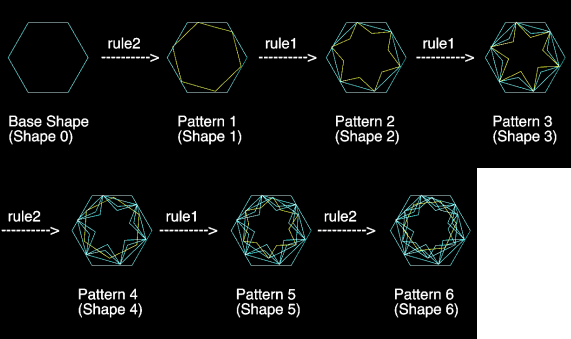
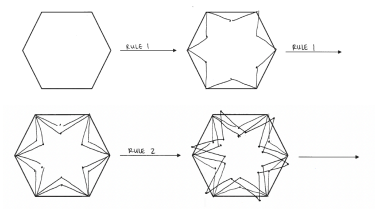
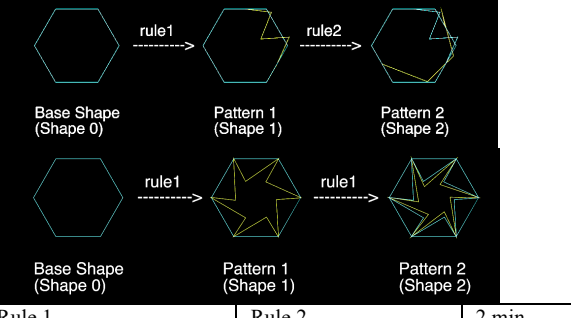
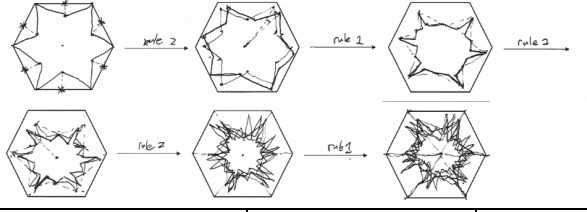
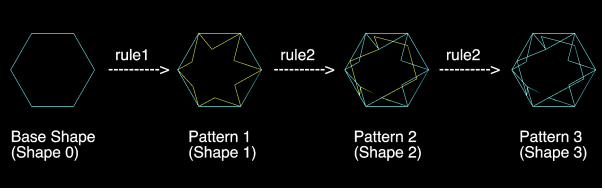
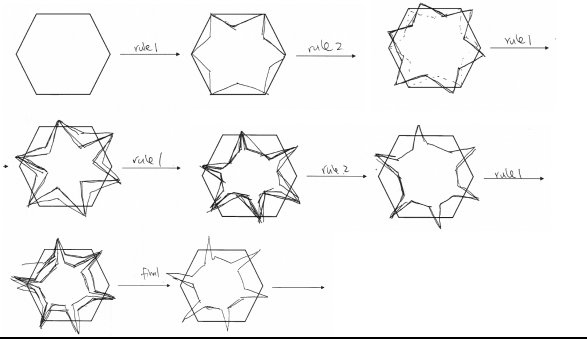
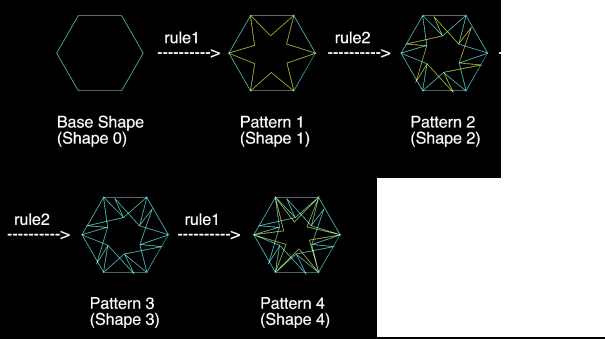
In fact, for most designers, data is a foreign concept. Many participants mentioned gaining a new perspective on recognizing shapes by selecting and training data through a learning-based interpreter. Also, they said that their understanding of the data processing methods of the learning-based interpreter was enhanced. It may be possible to develop the learning-based interpreter into educational tools for data-driven design for future work. The interpreter designed by the investigator in the user study has a restricted format, allowing participants to select only point data of geometry. However, in the future, designing an interpreter that enables designers to manipulate shape data actively can provide an advanced understanding of shape data and develop the learning-based interpreter into a tool capable of generating more diverse shape rules.

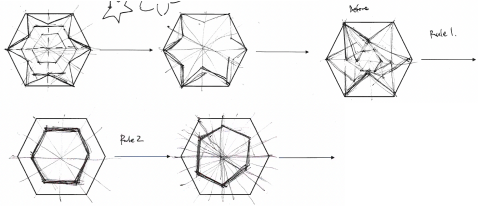
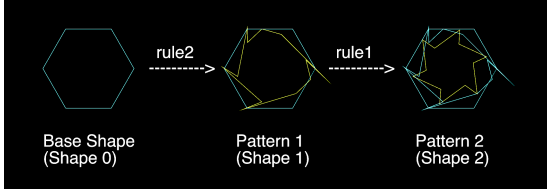
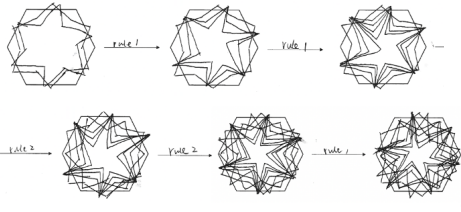
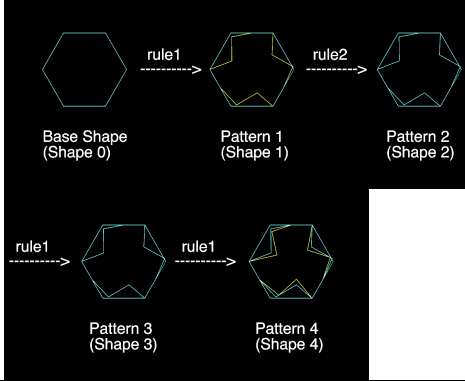
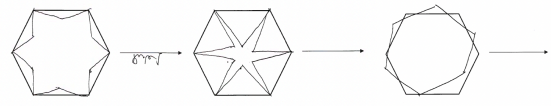
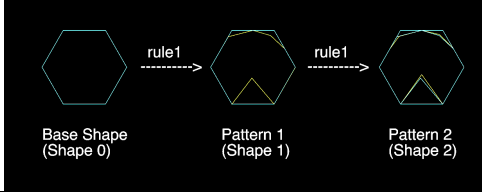
Appendix

Table 5 Design Outcomes of participants

Session 1. Rule-based interpreter			Session 2. Learning-based interpreter		
Rule 1. The rule takes the middle point on each edge and shifts it toward the center to get the after shape.	Rule 2. Before shape is rotated by 15-30 degrees clockwise around the center.	5 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	2 min to find rules
					
Rule 1. For every pair of edges, middle point goes to center	Rule 2. Rotate	2 min to find rules	Rule 1. 2, 4, 6, 8, 10	Rule 2. 0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12	1 min to find rules
					
Rule 1. The red one is always 12 lines, and it will be fold into center as the blue one fold into center.	Rule 2. The red and blue are like cover together and the red one rotate and the borders come out a little bit outside.	3 min to find rules	Rule 1. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	Rule 2. 0, 1, 3, 5, 7, 9, 11	1 min to find rules
					

Rule 1. Share the convex as Before shape, move to center to be proportionate in the side length.	Rule 2. Don't know the geometrical relationship between vertices considering mathematical calculation.	Could not find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 3, 5, 7, 9, 11	5 min to find rules
X					
Rule 1. From centroid of the hexagon to midpoint of every side, there's a string connection. Every time the string pulls midpoint of the side inward. Therefore, each side dives into angled 2 sides creating a star-like shape eventually.	Rule 2. Each turn before shape rotates in a certain angle and creates other shape.	5 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 3, 5, 7, 9, 11	1 min to find rules
Rule 1. Each edge makes angle smaller (e.g. 180 -> 165)	Rule 2. Rotate -10	3 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 0, 1, 3, 5, 7, 9, 11	2 min to find rules
Rule 1. the y gets larger when it increases the # of pair number.	Rule 2. 1) Rotate 2) scale down	5 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	1 min to find rules

					
Rule 1. Concentrate inward	Rule 2. rotation in clockwise, but don't know how much	10 min to find rules	Rule 1. 0, 1, 2, 3	Rule 2. 0, 1, 9, 10, 11, 12	2 min to find rules
					
Rule 1. Find a interpolate point on each edge of the input shape / Move the interpolate point towards the centroid of the input shape[val: random] / update the input shape	Rule 2. Find the centroid of the input shape / Scale down the shape by a given factor / Rotate around the centroid of shape	5 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 2, 3, 5, 6, 7, 9, 11, 12	2 min to find rules
					
Rule 1. Find the concave point of the given shape and indent inside	Rule 2. Rotate the same shape by around 20-25	3 min to find rules	Rule 1. 2, 4, 6, 8, 10, 12	Rule 2. 1, 3, 5, 7, 9, 11	1 min to find rules
					
Rule 1. Divide into different section, equally, Curve	Rule 2. Rotation of shape, border, Relationship in	10 min to find rules	Rule 1. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	Rule 2. 0, 2, 3, 4, 6, 8, 10, 11, 12	2 min to find rules

scale different, hard to see the difference in curve	scale + rotation, rotate counterclockwise				
					
Rule 1. to concave becomes sharper	Rule 2. the shape rotate some angles maybe 30 or so	3 min to find rules	Rule 1. 0, 1, 2, 5, 6, 9, 10	Rule 2. 1, 2, 3, 5, 6, 7, 9, 10, 11	2 min to find rules
					
Rule 1. Red shape is bounded by blue shape, distance from center point to vertices is changing in some rules	Rule 2. same shapes, but rotating	10 min to find rules	Rule 1. 0, 3, 5, 10	Rule 2. 1, 5, 9	1 min to find rules
					

Reference

- Beirão, José, José Duarte, and Rudi Stouffs. “An Urban Grammar for Praia: Towards Generic Shape Grammars for Urban Design,” 575–84. Istanbul, Turkey, 2009.
<https://doi.org/10.52842/conf.ecaade.2009.575>.
- Chau, Hau Hing, X. Chen, Alison McKay, and Alan de Pennington. “Evaluation of a 3D Shape Grammar Implementation,” 357–76, 2004. https://doi.org/10.1007/978-1-4020-2393-4_19.
- Chen, Wei. “Data-Driven Geometric Design Space Exploration and Design Synthesis.” [object Object], 2019. <https://doi.org/10.13016/QWKQ-QMTW>.
- Chiou, S-C, and Ramesh Krishnamurti. “The Grammar of Taiwanese Traditional Vernacular Dwellings,” 1995. <https://journals.sagepub.com/doi/abs/10.1068/b220689>.
- Duarte, José P. “A Discursive Grammar for Customizing Mass Housing: The Case of Siza’s Houses at Malagueira.” *Automation in Construction*, Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2003), Digital Design, 14, no. 2 (March 1, 2005): 265–75. <https://doi.org/10.1016/j.autcon.2004.07.013>.
- Economou, Athanassios, and Thomas Grasl. *Paperless Grammars*, 2017.
- Eloy, Sara, Maria Ângela Dias, and Pieter E Vermaas. “User-Centered Shape Grammars for Housing Transformations: Towards Post-Handover Grammars.” In *Blucher Design Proceedings*, 156–64. São Carlos, BR: Editora Blucher, 2018.
<https://doi.org/10.5151/sigradi2018-1248>.
- Eloy, Sara, Pieter Pauwels, and Athanassios Economou. “AI EDAM Special Issue: Advances in Implemented Shape Grammars: Solutions and Applications.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 32, no. 2 (May 2018): 131–37.
<https://doi.org/10.1017/S0890060417000634>.
- Frid-Adar, Maayan, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. “GAN-Based Synthetic Medical Image Augmentation for Increased CNN Performance in Liver Lesion Classification.” *Neurocomputing* 321 (December 2018): 321–31. <https://doi.org/10.1016/j.neucom.2018.09.013>.
- Grasl, Thomas, and Athanassios Economou. “From Topologies to Shapes: Parametric Shape Grammars Implemented by Graphs.” *Environment and Planning B: Planning and Design* 40 (January 1, 2013): 905–22. <https://doi.org/10.1068/b38156>.
- Gürsoy, Benay, and Mine Özkar. “Visualizing Making: Shapes, Materials, and Actions.” *Design Studies*, Special Issue: Computational Making, 41 (November 1, 2015): 29–50.
<https://doi.org/10.1016/j.destud.2015.08.007>.
- Haakonsen, Sverre Magnus, Anders Rønquist, and Nathalie Labonnote. “Fifty Years of Shape Grammars: A Systematic Mapping of Its Application in Engineering and Architecture.” *International Journal of Architectural Computing* 21, no. 1 (March 1, 2023): 5–22.
<https://doi.org/10.1177/14780771221089882>.

- Harrison, Laura, Iestyn Jowers, and Chris Earl. "Defining Rules for Kinematic Shapes with Variable Spatial Relations." In *Computer-Aided Architectural Design. "Hello, Culture,"* edited by Ji-Hyun Lee, 444–58. Communications in Computer and Information Science. Singapore: Springer, 2019. https://doi.org/10.1007/978-981-13-8410-3_31.
- Hasey, Michael. "The 3D Form Analysis of Regional Architecture Using Deep Learning: A Case Study of Wood Churches from the Carpathian Mountains." Thesis, Carnegie Mellon University, 2022. <https://doi.org/10.1184/R1/21291672.v1>.
- Hong, Tzu-Chieh. "SHAPE MACHINE: Shape Embedding and Rewriting in Visual Design," 2021.
- Jowers, Iestyn, and Christopher Earl. "Implementation of Curved Shape Grammars." *Environment and Planning B: Planning and Design* 38, no. 4 (August 1, 2011): 616–35. <https://doi.org/10.1068/b36162>.
- Jowers, Iestyn, David Hogg, Alison McKay, Hau Hing Chau, and Alan de Pennington. "Shape Detection with Vision: Implementing Shape Grammars in Conceptual Design." *Research in Engineering Design* 21 (October 1, 2010): 235–47. <https://doi.org/10.1007/s00163-010-0088-z>.
- Kingma, Diederik P., Danilo J. Rezende, Shakir Mohamed, and Max Welling. "Semi-Supervised Learning with Deep Generative Models." arXiv, October 31, 2014. <http://arxiv.org/abs/1406.5298>.
- Knight, T W. "Color Grammars: Designing with Lines and Colors." *Environment and Planning B: Planning and Design* 16, no. 4 (December 1, 1989): 417–49. <https://doi.org/10.1068/b160417>.
- Knight, Terry. "Applications in Architectural Design, and Education and Practice," April 1999.
- . "Transformations in Design: A Formal Approach to Stylistic Change and Innovation in the Visual Arts." *Choice Reviews Online* 32, no. 11 (July 1, 1995): 32-6041-32–6041. <https://doi.org/10.5860/CHOICE.32-6041>.
- . "Transformations of Languages of Designs: Part 1," 1983. <https://journals.sagepub.com/doi/abs/10.1068/b100125>.
- Knight, Terry, and George Stiny. "Making Grammars: From Computing with Shapes to Computing with Things." *Design Studies*, Special Issue: Computational Making, 41 (November 1, 2015): 8–28. <https://doi.org/10.1016/j.destud.2015.08.006>.
- Krishnamurti, R. "The Arithmetic of Shapes." *Environment and Planning B: Planning and Design* 7, no. 4 (1980): 463–84. <https://doi.org/10.1068/b070463>.
- . "The Construction of Shapes." *Environment and Planning B: Planning and Design* 8, no. 1 (March 1, 1981): 5–40. <https://doi.org/10.1068/b080005>.
- Krishnamurti, R, and C Giraud. "Towards a Shape Editor: The Implementation of a Shape Generation System." *Environment and Planning B: Planning and Design* 13, no. 4 (December 1, 1986): 391–404. <https://doi.org/10.1068/b130391>.
- Krishnamurti, Ramesh. "SGI: An Interpreter for Shape Grammars." *Technical Report, Centre for Configurational Studies, Design Discipline, The Open University*, January 1, 1982.

- Krishnamurti, Ramesh, and C F Earl. "Shape Recognition in Three Dimensions," 1992.
<https://journals.sagepub.com/doi/10.1068/b190585?icid=int.sj-abstract.similar-articles.3>.
- Krishnamurti, Ramesh, and Rudi Stouffs. "Spatial Change: Continuity, Reversibility, and Emergent Shapes." *Environment and Planning B: Planning and Design* 24 (May 1, 1997): 359–84. <https://doi.org/10.1068/b240359>.
- . "The Boundary of a Shape and Its Classification." *J. of Design Research* 4, no. 1 (2004): 0. <https://doi.org/10.1504/JDR.2004.009843>.
- Li, A. "A Shape Grammar for Teaching the Architectural Style of the Yingzao Fashi," 2001.
<https://www.semanticscholar.org/paper/A-shape-grammar-for-teaching-the-architectural-of-Li/77a87f49e2ba6332fc12e5106967ab68222be470>.
- Li, Andrew I-kang, and Rudi Stouffs. "Towards a Useful Grammar Implementation: Beginning to Learn What Designers Want." In *A New Perspective of Cultural DNA*, edited by Ji-Hyun Lee, 55–64. KAIST Research Series. Singapore: Springer Singapore, 2021.
https://doi.org/10.1007/978-981-15-7707-9_5.
- Ligler, Heather, and Athanassios Economou. "From Drawing Shapes to Scripting Shapes: Architectural Theory Mediated by Shape Machine," n.d.
- Llach, Daniel Cardoso. "Sculpting Spaces of Possibility." In *The Routledge Companion to Artificial Intelligence in Architecture*, edited by Imdat As and Prithwish Basu, 1st ed., 13–28. Abington, Oxon; New York: Routledge, 2021.: Routledge, 2021.
<https://doi.org/10.4324/9780367824259-3>.
- Noel, Vernelle A.A. "The Bailey-Derek Grammar: Recording the Craft of Wire-Bending in the Trinidad Carnival." *Leonardo* 48, no. 4 (August 2015): 357–65.
https://doi.org/10.1162/LEON_a_01089.
- Rhee Jinmo, Veloso Pedro, and Krishnamurti Ramesh. "Three Decades of Machine Learning with Neural Networks in Computer-Aided Architectural Design (1990–2021)." Accessed November 15, 2023. <https://www.cambridge.org/core/journals/design-science/article/three-decades-of-machine-learning-with-neural-networks-in-computeraided-architectural-design-19902021/766645E6931ACF8B5B813FD3792A726D>.
- Rhee, Jinmo, Pedro Veloso, and Ramesh Krishnamurti. "Integrating Building Footprint Prediction and Building Massing - an Experiment in Pittsburgh," 669–78. Bangkok, Thailand, 2020. <https://doi.org/10.52842/conf.caadria.2020.2.669>.
- Steinfeld, Kyle. "Imaging Place Using Generative Adversarial Networks," 2019.
- Stiny, G. "Introduction to Shape and Shape Grammars." *Environment and Planning B: Planning and Design* 7, no. 3 (September 1, 1980): 343–51. <https://doi.org/10.1068/b070343>.
- . "Spatial Relations and Grammars." *Environment and Planning B: Planning and Design* 9, no. 1 (March 1, 1982): 113–14. <https://doi.org/10.1068/b090113>.
- . "Weights." *Environment and Planning B: Planning and Design* 19, no. 4 (August 1, 1992): 413–30. <https://doi.org/10.1068/b190413>.
- Stiny, George. "Pictorial and Formal Aspects of Shape and Shape Grammars." Basel: Birkhäuser Basel, 1975. <https://doi.org/10.1007/978-3-0348-6879-2>.

- . “The Algebras of Design.” *Research in Engineering Design* 2, no. 3 (September 1, 1991): 171–81. <https://doi.org/10.1007/BF01578998>.
- Stiny, George, and James Gips. ‘*Shape Grammars and the Generative Specification of Painting and Sculpture.*’ *IFIP Congress*. Vol. 71, 1971.
- Stouffs, Rudi. *On Shape Grammars, Color Grammars and Sortal Grammars: A Sortal Grammar Interpreter for Varying Shape Grammar Formalisms. Digital Physicality - Proceedings of the 30th eCAADe Conference - Volume 1* (Eds. H. Achten, et al.), Pp. 479-487, Faculty of Architecture, Czech Technical University, Prague, Czech Republic, 2012, 2012. <https://doi.org/10.52842/conf.ecaade.2012.1.x.j7v>.
- Stouffs, Rudi, and Ramesh Krishnamurti. “A Uniform Characterization of Augmented Shapes.” *Computer-Aided Design* 110 (May 1, 2019): 37–49. <https://doi.org/10.1016/j.cad.2018.12.004>.
- Strobbe, Tiemen, Ronald De Meyer, and Jan Van Campenhout. “A Semi-Automatic Approach for the Definition of Shape Grammar Rules,” 437–43. Vienna, Austria, 2015. <https://doi.org/10.52842/conf.ecaade.2015.1.437>.
- Tapia, M. “A Visual Implementation of a Shape Grammar System.” *Environment and Planning B: Planning and Design* 26, no. 1 (February 1, 1999): 59–73. <https://doi.org/10.1068/b260059>.
- Veloso, Pedro. “An Academy of Spatial Agents - Training an Ecology of Computational Agents for Spatial Synthesis with Fine-Grained Interaction,” 2023. <https://doi.org/10.1184/R1/22105298.v1>.
- Yue, Kui, and Ramesh Krishnamurti. “Tractable Shape Grammars.” *Environment and Planning B: Planning and Design* 40, no. 4 (August 2013): 576–94. <https://doi.org/10.1068/b38227>.

List of Figures and Tables

Figure 1. Timeline of Shape Grammar research.....	12
Figure 2 Principle of shape grammar interpreter computation.....	13
Figure 3 Examples of design application of Shape Machine (https://shape.design.gatech.edu/index.html)	15
Figure 4 Interface of the interpreter (Left), Example of rule application (Right) (Jowers et al., 2010).....	16
Figure 5 Comparison of the overall design process for rule-based shape grammar interpreter and learning-based shape grammar interpreter	18
Figure 6 The designer's perspective of the problem is reflected in the design of the data and results. Predominant visual properties of places worked by Steinfeld (Left), building massing design using GIS data of building footprints and parcels worked by Rhee et al. (Right)	19
Figure 7 Data generation of the first experiment (2K)	21
Figure 8 Model structure of the first prototype	22
Figure 9 Result of the first experiment.....	23
Figure 10 Data generation of the second prototype (10K)	23
Figure 11 Model structure of the second and third prototypes.....	24
Figure 12 Result of the second prototype.....	24
Figure 13 Data generation of the third prototype (90K).....	25
Figure 14 Result of the third prototype.	26
Figure 15 User interface of the data display part (Grasshopper screen). Participants controlled the sliders to see shape data.....	28
Figure 16 User interface of the data display part represents the pair shape samples (Rhino screen).....	28
Figure 17 User interface of shape rule recognition part (Grasshopper screen). Participants connected the point number to the collector.....	29
Figure 18 User interface of shape generation part (Grasshopper screen). Participants designed patterns through the interpreter by selecting rules and inputting the names of initial shapes.....	30
Figure 19 The logic of shape rule generation (Description used in this user study)	31
Figure 20 User interface of the shape generation part represents the patterns (Rhino screen)	31

Figure 21 Error-checking block.....	32
Figure 22 User interface of rule-based interpreter test module (Grasshopper screen). Participants controlled the slide bars to see samples of pair shapes.....	32
Figure 23 User interface of the rule-based interpreter test module represents the pair shape samples (Rhino screen).....	33
Figure 24 Task of the user study in two sessions	34
Figure 25 Section 1-2 of the questionnaire, asking about experience with design and data	35
Figure 26 Handout design for session 1	36
Figure 27 Section 3 of the questionnaire, asking about their indirect experience with the rule-based interpreter test module.	38
Figure 28 Handout design for session 2	39
Figure 29 Section 4 of the questionnaire, asking about their experience with the learning-based shape grammar interpreter.	40
Figure 30 User study was conducted in person individually.....	41
Figure 31 Participants' experience with design and data/AI	43
Figure 32 Design outcomes by some participants in Session 1.....	44
Figure 33 Design outcomes by some participants in Session 2.....	45
Figure 34 Evaluation about usability.....	48
Figure 35 Analysis of participants' satisfactory.....	49
Figure 36 Analysis of participants' enjoyability	50
 Table 1 Given rules and challenges.....	 27
Table 2 Design of the questionnaire	36
Table 3 Schedule of the study.....	41
Table 4 Some comments from participants about two shape grammar interpreters	47
Table 5 Design Outcomes of participants.....	59