

A Visual Approach for Exploring Computational Design

Introduction

1. General introduction
2. Framework

Chapter 1

Learning about Computational Design

1. What is Computational Design?

- The difference between computation for designing and computation for modeling designs
 - Using computers to facilitate and enhance the design process
 - Using computers as a medium for representation of design ideas
- The Role of Creativity in Computation
- Using computation to do architectural (or other) designs

2. How this approach differs from other systems of computational design: Visual vs. Programming

- Emphasize how this thesis differs from other approaches to computational design, which utilize symbolic or 'traditional' computation, whereas the *Shaper2D* approach is inherently visual. *Shaper2D* is an exclusively visual environment, although it is arguably more restrictive and constraining workspace than the other approaches.
- Give specific examples of approaches:
 - Maeda's *Designing by Numbers* is interested primarily with the final design, or product, not the design process
 - Yakeley's *Digitally Mediated Design* is concerned with the processes involved in generating a design—designer programs designs, albeit in an unrestricted environment.
 - *Shaper2D* is an exclusively visual environment, although it is arguably more restrictive and constraining workspace than *Digitally Mediated Design*
- Discuss how each of the aforementioned approaches creates its own 'microworld', in order to focus the exploration of a specific idea.
- Using approaches for learning mathematical concepts (primarily Euclidean geometry), draw analogy with difference between *Logo* and *Geometer's Sketch Pad*

3. Background to learning methods for "learning by doing"

- Constructive learning
- Situated Learning
- Collaborative and Cooperative Learning

4. Computer mediated design

- Hand computation and Computer computation

5. Shape Grammars

- Using shape grammars as an example of computational design
- Computer-mediated shape grammars

Chapter 2

Shaper2D: An dynamic shape grammar application

1. Previous computer implementations of shape grammars

- Discuss previous implementations: what they sought to achieve, what audience were they trying to engage, strengths, limitations.
- Specific examples, that relate to Shaper2D, of previous implementations
 - Tapia: *GEdit*
 - Wang: *3D Shaper*
- How the implementation of Shaper2D differs from these implementations, and why.

2. What is Shaper2D?

- How Shaper2D was initially conceived of as a teaching tool, and its subsequent evolution to its eventual status as a tool for facilitating learning and understanding of shape grammar concepts
- Is Shaper2D a design or a learning tool... or both?
- How it relates to constructive learning

3. Hand vs. Computer Computation, with reference to Shaper2D

4. Design-mediating software

- An intuitive visual "architect friendly" interface

5. Developing and implementing Shaper2D:

- The interface design
- Interactivity
- The application and applet

6. How to use Shaper2D:

- Overview
- Spatial relationship
- Rules

- Design
- Exporting

7. Using Shaper2D with other programs

Chapter 3

Shaper2D in the studio

1. The Workshop: Hand then Computer Computation

- 1.1. Aims of the workshop
- 1.2. The teams
- 1.3. The modalities of communication
- 1.4. The software
- 1.5. The project
- 1.6. Role of Shaper2D
 - Tutorial
 - Exercises
 - Assignment
- 1.7. Role played in final project
- 1.8. Conclusions drawn from workshop

2. Data Collection

- 2.1. Experimentation with basic shape grammars using Shaper2D during the first week: tutorial, in-class exercises and assignment.
- 2.2. Survey during the second week of the workshop.
- 2.3. Informal follow-up interviews with MIT students during the week following the end of the workshop.
- 2.4. Informal follow-up chat with Miyagi students several weeks after the end of the workshop.

3. Data Analysis

- 3.1. Introduce methods used and why
- 3.2. Explain limitations of study in terms of people and time, and suggest further tests/experiments that could be done.
 - It would have been preferable to have two groups of students - one half who were exposed to hand computation prior to using the computer and one half who were not. However, due to the time constraints (and limitations?) of the workshop, this was not possible.
 - Perhaps better to have interviewed the students before the workshop began?

4. The Follow-up Study: Computer then hand computation

4.1. Why necessary

- Follow-up study using Shaper2D to test hand vs. computer question, and to serve as additional feedback for learning/understanding aspects of the program.

4.2. The people

- Include note, stating awareness of possibility that results may have been affected by the fact that the subjects were colleagues of the investigator.

4.3. Description of study, (include consent form etc. in appendix)

4.4. Observations of how the exercises were approached and executed

4.5. Conclusions drawn from study

Conclusions

1. Implications and conclusions

- When considering the role of computers in constructive learning of mathematics, can this approach be used for design? How does learning about design differ from learning about mathematics, given that design is inherently a hands-on activity?
- Does computation enhance design, or is it merely a catalyst for the design process?
- Do computers enhance the design process, or merely mimic it?
- How do computers facilitate computational design?

2. Future work

- Development / extensions to Shaper2D
- Discuss the limited (restricted) nature of the studies using Shaper2D, and propose further studies that could be conducted to answer some of the questions raised by this thesis.

References

1. Picture Credits
2. Bibliography

Appendices

1. Description of mathematical approach used in programming Shaper2D
2. Shaper2D Java Code
3. MIT Study Documentation: Outline of Study and Consent Form
4. List of all previous computer implementations of Shape Grammars