An Enchanted Toy Based on Froebel's Gifts: A computational tool used to teach architectural knowledge to students

Hyong-June Park, Emmanuel-George Vakaló

Keywords
bottom-up approach, formalization, data abstraction, communication, basic transformation functions, syntactic interventions, isolated island of automation, Feedback and error-elimination

Abstract

Assuming that students can require architectural knowledge through direct manipulation of formal objects, this paper introduces a computational toy as a means for teaching knowledge about composition and geometry to students of architecture. The bottom-up approach is employed in the manipulation of the toy. The toy aims at recovering and nourishing the students' creative spirit and enriching their vocabulary of forms and spaces.

Introduction

Friedrich Froebel, the pioneer of the kindergarten movement, exemplifies the nineteenth century's interest in educational toys. In 1837, Froebel introduced what he called the Ten Gifts, a set of increasingly complex play materials, to which kindergarten pupils were to be exposed sequentially.

Since Frank Lloyd Wright acknowledged, in his biography, the influence of Froebel's kindergarten method on his design, the kindergarten method became famous among architects and designers. As it turns out, even after becoming an architect, Frank Lloyd Wright used building blocks to derive design solutions.

Based upon the possibilities of exploring spatial design with Froebel's gifts, Stiny, in his article "*Kindergarten Grammar*,"[1] develops a visual grammar to facilitate a vocabulary of building elements and a system of categories of forms in languages of designs. The languages are formed by combining or augmenting other languages of designs in terms of various language-theoretic operations such as substitution, Boolean operations (union, intersection, difference) and basic transformation functions (translation, rotation, mirroring, and scaling). Stiny shows possible techniques of specifying a spatial relation actually occurring in Froebel's gifts with his kindergarten grammar. After this, a number of efforts have been made in making an application of a three-dimensional shape grammar. A possible way of recognizing three-dimensional shape grammar and a framework of application are sketched by Piazzalunga and Fitzhorn.[2] Agarwal, Cagan, and Constantine[3] suggest the idea of optimizing production system according to the feedback about each separate stage of designing. Nevertheless, the use of shape grammar still has been difficult for a student and designer who are more familiar with manipulating formal objects than with making a rule and applying it.

This paper introduces the algorithm and programming of design as Froebel's 11th gift4. The proposed enchanted toy translates user's direct manipulation of objects into a set of rules organized in sequential order of design resolution, which is known as schema. In return, schema illustrates the student's form-making process so that the student can modify his/her design result by changing the illustrated process. Thus, the toy provides not only what is built but also how it is constructed. An individual way of designing, tacit knowledge, becomes an object of play.

An Enchanted Toy

Fundamentally, the toy employs the notion of " *object*" [5] and suggests a way of understanding design as a process of making "*a meaningful order.*"[6] The toy regards all the components used to generate a design as a set of *objects*, which are organized in sequence of design resolution. The basic structure of object consists of state and behavior. State contains geometric entities as attributes of object. Behavior has basic transformation functions. In addition, the relation between different objects defines a step. The relation includes addition, delete, and Boolean Operations. Hence, the toy allows students to define a new object. Each stage of generating a new object is recorded as a step. The step, which is the design resolution, clarifies schema. Therefore, the student who uses the toy will get data of visual artifact and information pertaining to form-making process of the artifact. Based upon the proposed idea of basic structure, the framework of the toy consists of **formalization, data abstraction,** and **communication**.

1. Formalization

Formalization allows students to define formal objects as *objects.* When the student makes a substitution (addition or delete) of objects on the playing panel in **Designer** interface with prototypes from instance libraries in **Designer** or previously defined objects on the playing panel, the *objects* are instantiated by the toy. The attributes of an *object*, which are geometric entities, are established either by the student or defined as default value initially by the toy. Either the toy or the student gives corresponding label or name to the *object*. However, the identity number of the *object* is defined only by the toy. In addition, the behavior (basic transformation functions) of the *object* is defined by the student's direct manipulation of the object and organized by the toy. Also, the student creates a new object by making a relation, which includes Boolean Operations. The created objects are also stored as instances in instance libraries.

2. Data Abstraction

Data abstraction enables students to isolate "how a compound object is used from the details of how it is constructed from more primitive objects. "[7] Assuming the toy provides the basic transformation functions of spatial entities, the instantiated *object*s can be assembled with various sets of formal relations defined by the students. The toy organizes each step of creating another/new object according to the students' assembling of objects. Illustrating steps of the student's design process in **Grammar** interface, the toy provides an understanding of the connection of the object on the playing panel in **Design** and a set of rules that generates the object. Therefore, the sequence of making new *objects* is displayed in step classes, and the transformation of *objects* is explained in *object* classes. It allows the students to assemble their formal pieces not only in constructing a compound artifact within a visual state but also in modifying the process and transformations of the artifact.

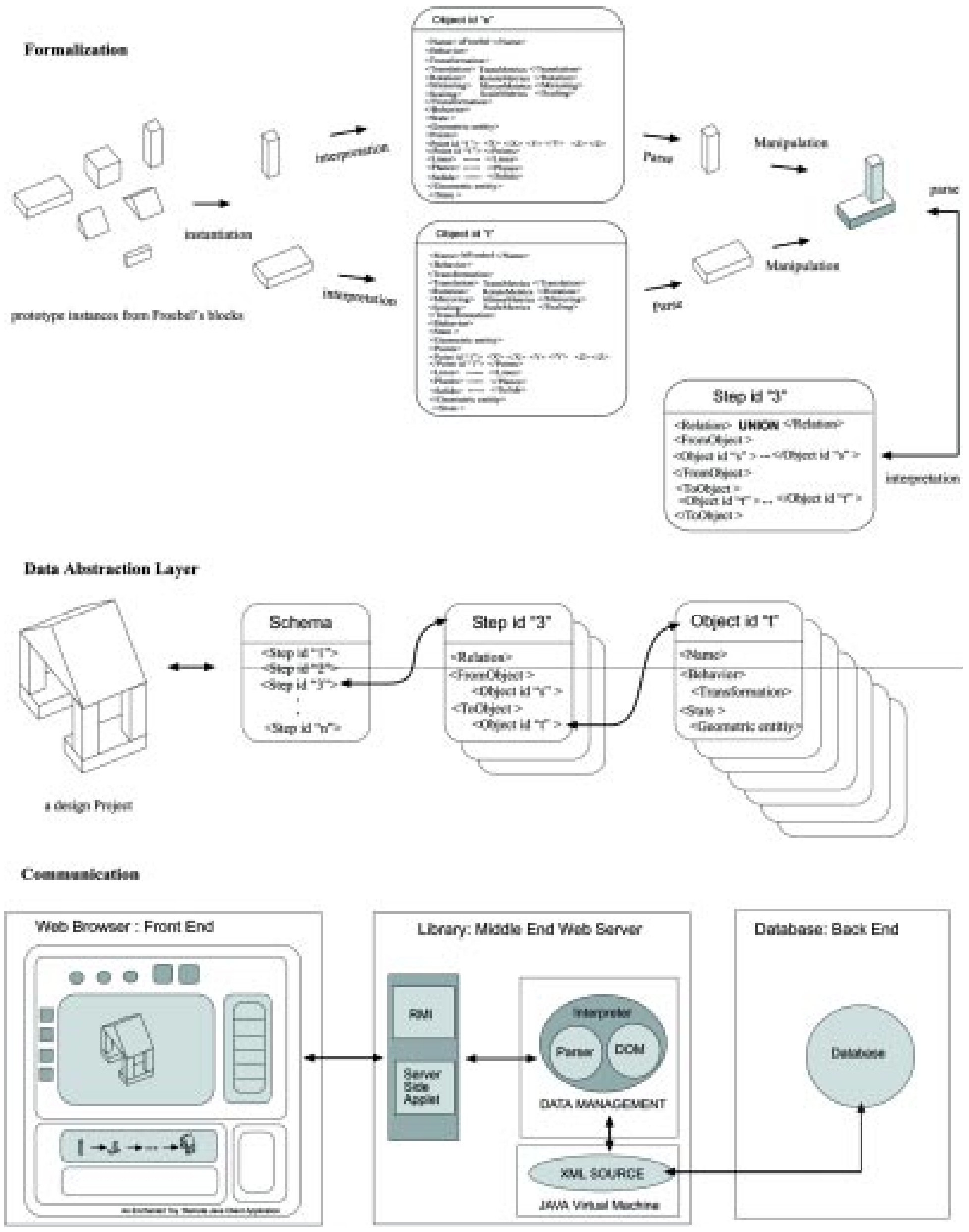


Figure 1:
Algorithm
of an Enchanted Toy

An Enchanted Toy Based on Froebel's Gifts… 37

3. Communication

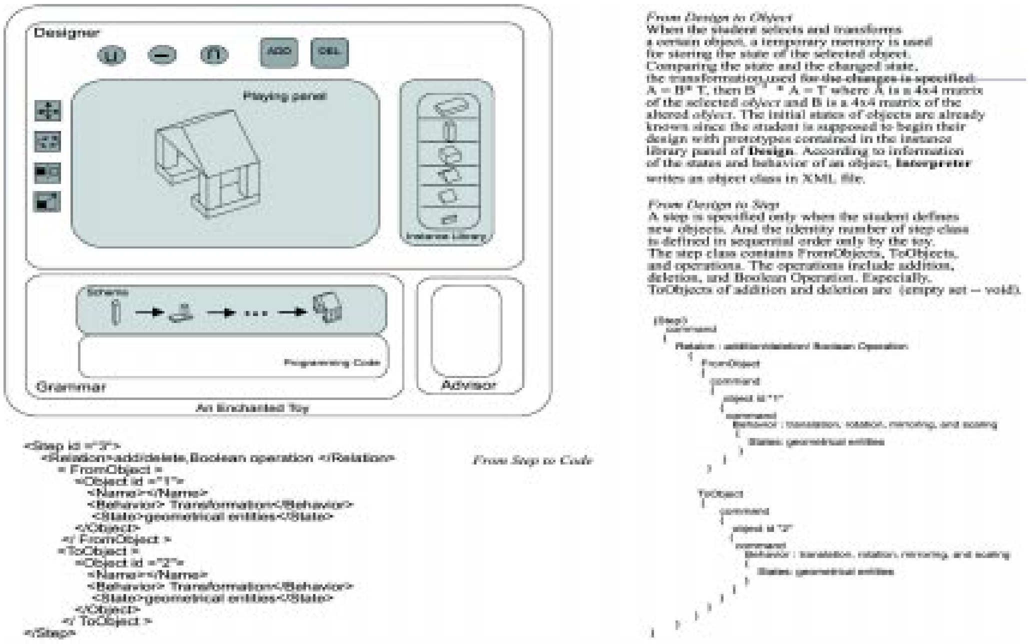
Communication in the framework facilitates, through the web-browser, the exchange of encoded or encapsulated information with other designers. Feedback and error-elimination are requested to the student. According to the student' decision, the result of communication effects the change of the state and composition of the artifacts through altering the *object* and step classes parsed inside the database of the enchanted toy.

4. Implementation of 11th Gift

The significant components to implement the 11th gift consist of Library, Interpreter, and User-interfaces.

Library & Interpreter

Initially **Library** contains classes of prototype *objects*. The classes of prototype instances are predefined. However, in **Library,** the number of *object* classes and step classes increases when a new object is constructed in **Designer** interface. **Interpreter** translates, according to changes occurred either in **Design** or in **Grammar,** the object constructed by the student into *object* class and step class using "Extensible Markup Language - XML."[8] XML provides the advantage of describing metacontent, publishing the database contents, and communicating data using a messaging format. XML also provides Data Object Model (DOM) for efficiently understanding the data structure. In re-



From Design to Object
When the student selects and transforms a certain object, a temporary memory is used for storing the state of the selected object. Comparing the state and the changed state, the transformation used for the changes is specified: A = B* T, then B⁻¹ * A = T where A is a 4x4 matrix of the selected *object* and B is a 4x4 matrix of the altered *object*. The initial states of objects are already known since the student is supposed to begin their design with prototypes contained in the instance library panel of **Design**. According to information of the states and behavior of an object, **Interpreter** writes an object class in XML file.

From Design to Step
A step is specified only when the student defines new objects. And the identity number of step class is defined in sequential order only by the toy. The step class contains FromObjects, ToObjects, and operations. The operations include addition, deletion, and Boolean Operation. Especially, ToObjects of addition and deletion are (empty set – void).

```
(Step)
  command
  {
    Relation : addition/deletion/ Boolean Operation
      {
        FromObject
        {
          command
          {
            object id "1"
            {
              command
               Behavior : translation, rotation, mirroring, and scaling
               {
                  States: geometrical entities
               }
            }
          }
        ToObject
        {
          command
          {
            object id "2"
            {
              command
               Behavior : translation, rotation, mirroring, and scaling
               {
                  States: geometrical entities
               }
            }
          }
        }
      }
  }
}
```

From Step to Code

```
<Step id ="3">
  <Relation>add/delete,Boolean operation </Relation>
    < FromObject >
      <Object id ="1">
        <Name></Name>
        <Behavior> Transformation</Behavior>
         <State>geometrical entities</State>
      </Object>
    </ FromObject >
    <ToObject >
      <Object id ="2">
        <Name></Name>
        <Behavior> Transformation</Behavior>
         <State>geometrical entities</State>
      </Object>
    </ ToObject >
</Step>
```

Figure 2:
Library, Interpreter, and User Interface

turn, the interpreter illustrates the parsed *object* on the library panel in **Designer**, and the proceeding steps as schema on the grammar panel in **Grammar**. The interpreted *objects* in XML are subclasses that make up a design project. It leads the student to find out that his or her main project consists of sets of individual objects. In addition, **Interpreter** re-organizes the parsed step file in LSP format as a program. In return, the programming code itself is appeared on the code panel in "Grammar." The contents of data exchange file (DXF) of a project are divided as *objects* and *steps* in the format of XML during the design process. With network communication based upon "Remote Method Invocation (RMI)"[9], in **Designer** or **Grammar** the student can use the basic transformation functions and relations, which are transferred to the database in the back end server.

User- Interfaces
User interfaces consist of **Grammar**, **Designer**, and **Adviser**.

Conclusion

An enchanted toy has been developed as a migration of Nine Square Grid Composition (NSGC), which was designed within AUTOCAD environment in AutoLisp and DCL, to a web-based application. The basic structure of NSGC is rooted from research on student work in traditional studio class.[10] The most polemic point taken from the research is that rational discussion between student and instructor about a design is not possible without records of the form-making process.
In this paper, we tried to explain how the design process could be programmed using the concept of *object*s and *step*s. Without the burden of understanding a programming language, students create easily their own programming of what they design, and investigate their algorithm of form-making process. Also, this allows them to modify their design easily when needed.

" In design, we are not teaching and studying what an engine is but how to make a form of the engine."

Notes

1 Stiny, George **Kindergarten Grammars: Designing With Froebel's Building Gifts.** Environment and Planning B: Planning and Design7, pp 409-462, 1980
2 Piazzalunga, U & Fitzhorn P I **Note on a Three-dimensional Shape Grammar Interpreter.** Environment and Planning B: Planning and Design25, pp 11-33, 1998
3 Agarwal, M., Cagan, J & Constantine, K.G. **Influencing generative design through continuous evaluation: Associating costs with the coffeemaker shape grammar.** Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) Volume 13 Number 4, pp 253-275, 1998
4 Froebel's 10th gift is a reconstruction with formal objects.
5 An object always has two characteristics: state and behavior. For example, Bicycles have state (current gear, current pedal cadence, two wheels, number of gears) and behavior (braking, accelerating, slowing down, changing gears)
6 Papanek, V. **Design for the Real World.** Van Nostrand Reinhold, 1984
7 Abelson and Sussman **Structure and Interpretation of Programs** . The MIT press. pp. 79-93, 1996.
8 Maruyama, Hiroshi & Tamura, Kent & Uramoto, Naohiko **XML and Java .** Addison Wesley Longman, Inc. pp. 14-30, 1999.
9 Reese, George **Database Programming With JAVA** . O'Reilly & Associates, INC. pp139-169, 1997
10 http://www-personal.umich.edu/~egvakalo/nsgc/teaching/design.htm

Hyong-June Park, Emmanuel-George Vakaló
Doctoral Program in Architecture
Taubman College of Architecture and Urban Planning
The University of Michigan
Ann Arbor, Michigan, 48109-2069, USA
archphj@umich.edu, egvakalo@umich.edu