REFLECTION IN ACTION

An educational indie video game with design schema

ALICE SANDSTROM[1] and HYOUNG-JUNE PARK[2]
[1,2]*University of Hawaii at Manoa*
[1,2]*{alice3|hjpark}@hawaii.edu*

Abstract. This paper outlines the development of an educational indie video game in which a set of design rules are generated as a schema from player actions with the spatial components of architectural precedents in a given library. Each player's outcome is scored with its comparison to the functional sequences of the original precedent and its formal arrangement. The implementation of the proposed game within UNITY is introduced.

Keywords. Shape Grammar; Indie Game; Schema; Design Rules; Scoring.

1. Introduction

Shape grammar is a means of calculating with shapes in order to generate new and unique designs by applying a set of shape rules, generated based on the spatial relationships formed and transformations undergone between a set of shapes, to an initial object (Stiny 1980a, Stiny 2006). This approach to design can also be used retroactively to analyze precedent architectural styles to understand their spatial compositions and to create new architectural designs that fall within that style through the development of specific shape grammars (Duarte 2005, Koning and Eizenberg 1981, Stiny and Mitchell 1978, Tepavčević and Stojaković 2012). Shape grammar has also been one of the approaches to design education which tacit knowledge in architecture is capable of being taught. This form of tacit knowledge relies on learning in action, which occurs in design through the act of designing without an awareness of learning contents themselves (Schön 1983). However, most applications that employ the use of shape grammar require a user to first create a rule prior to its application within the design. This order is opposite to how people naturally want to develop and use shape grammar where user does the designing *making* playing first and develops the design rules and their schema based on the analysis of their design actions (Piazzalunga and Fitzhorn 1998, Tapia 1999, Park and Vakaló 2001, Trescak et al 2010, Grasl and Economou 2013). The design rule based shape grammar's usage within the design process has also made it difficult to develop such a program for use in design practice. A fair amount of understanding morphological transformation functions is required before dealing with the generation of and experimentation with shape grammars in design process (Tepavčević and Stojaković 2012). These issues of shape grammar's inaccessibility for beginners limits the number of people who can learn about and experiment/play with shape grammars. This further limits its use in the field of both design and design education (Fischer and Herr 2001).

2. Methodology

To solve the problems outlined above we are developing a simple Unity-based game whose purpose goes beyond entertainment and includes education. Unity is a game engine by Unity Technology which supports 2D and 3D games across multiple gaming platforms, including mobile, web, pc, and console, and was chosen due to its flexibility as a game engine and the online resources available to us during the development process. We develop a video game as a potential solution to these problems as they are highly interactive, allowing the user/player to manipulate virtual 3D objects within the game while having the game analyze and provide feedback to the player throughout the whole process, in real time. This real-time analysis allows for the generation of design rules based on a player's design activity and their resulting schema to occur while the player is actively playing with their design rather than requiring them to define the rules and schema prior to the making of their design. This real-time feedback to players present in games allows the player to review a score for their design solution, guiding the player as they play through meaningful feedback (Deterding et al. 2011).

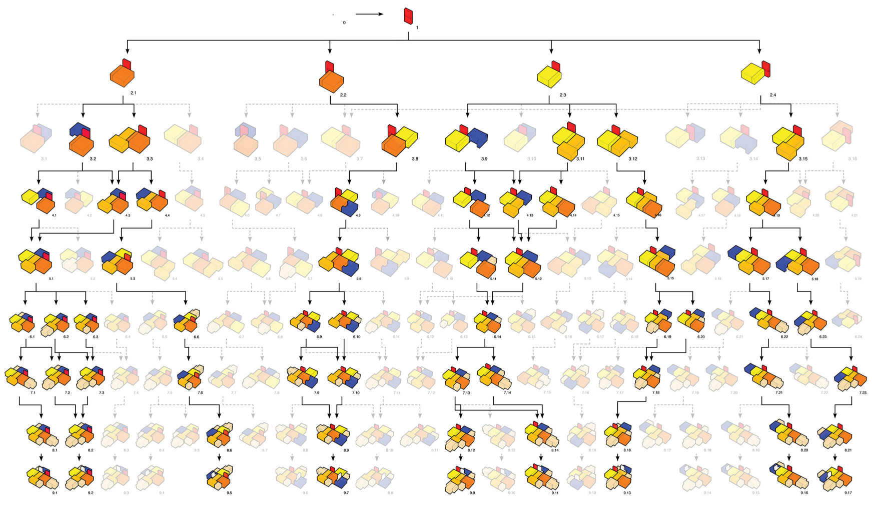

Figure 1. Various Ground Floor Schemas of Martin House within Game.

The proposed game is, at its core, a 3D puzzle game where the player is guided through a series of levels (also referred to as *scenes*) each based on a specific architectural precedent. In each of these levels, the player is provided a set of puzzle pieces (*labeled placeholders*), which function as abstractions of a precedent's different rooms and/or formal/functional delineations. The players progress through each level by arranging the puzzle pieces provided to them, creating their design variations. While the player is going through the process of arranging these puzzle pieces, the game tracks their process of doing so as a list of design rules. These lists of design rule are then compiled into an overall design schema list of that considers where and when a player changes their course of action by undoing previous decisions they had made, see Figure 1 for a visual representation of this idea.

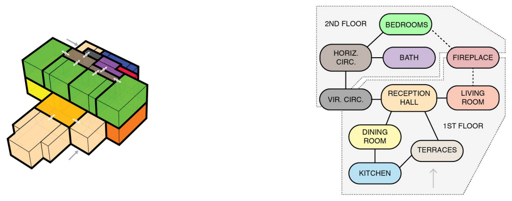

Figure 2. Original Arrangement & Functional Diagram for Martin House.

Our proposed game contains a library of architectural precedents ranging in building typology, style, time period, etc. which the player will be able to tacitly learn about as they progress through the game. These precedents serve as the basis for a puzzle given to the player in each game level. They determine the set of puzzle pieces and labels (space functions) the player is provided with at the start of the level, as well as serve as the basis for scoring the player after they solve the levels' puzzle, providing them meaningful feedback on the actions and choices they make. One of the initial puzzle levels currently being developed for testing is inspired by a paper written by Koning and Eizenberg titled ”The Language of the Prairie: Frank Lloyd Wright's Prairie Houses” where they detail a parametric shape grammar which can be used to generate a design within the style of Frank Lloyd Wright's Prairie Homes.

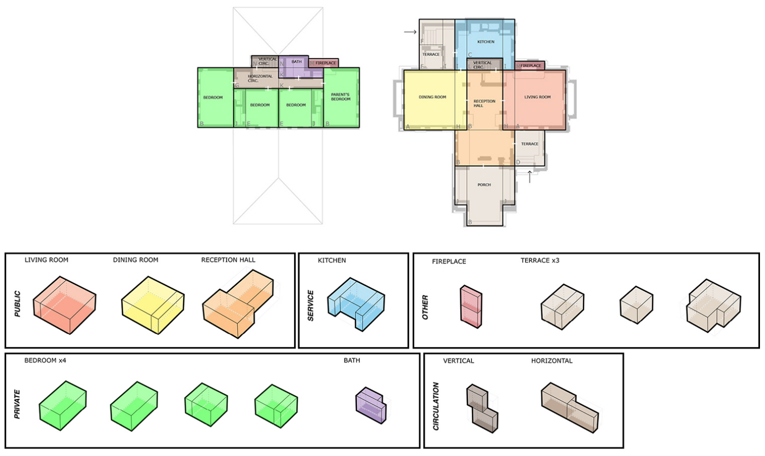

Figure 3. Abstracted layout of Martin House and its 3D placeholders .

Puzzle pieces are derived from the abstraction of a precedent's spatial and functional layout, and act as placeholders for architectural elements. There are four main categories of puzzle piece shapes used in our game: basic, which are simple cuboids; compound, which are comprised of a series of basic geometric forms which act as the components; unique, which consist of non-cuboid forms; and unique compound, which are identical to compound shapes but are comprised of both basic geometric forms and unique forms. These four types where the result of a series iterative abstractions for precedents selected for the games initial prototyping. The following shows the placeholders from an abstract layout of a selected precedent.

Alongside the abstraction of the formal aspects of a precedent design, the connections between the different functions assigned to each abstracted shape are articulated. These functional sequences are compared against the original precedent's functional connects which serve as the basis for the functional scoring criteria. The scoring criteria provide the player a guideline for designing with meaningful syntax while providing the player with the level of design freedom when assembling a solution to each puzzle given to them.

3. Gameplay

Our game serves two purposes: one at the player level and one at the researcher/developer level, both looking at different key motivational factors. At the player level, the game serves as an educational tool for the player to study and learn about architectural precedent styles and their vocabulary and design rules in order to obtain tacit knowledge of their design. A player's motivation within the game would vary from one player to another, much like in all games. This variation in player motivation affects how different players would approach the puzzles we give to them. For example, while one player might care more about obtaining a high score at the end of each level, seeking to create solutions which perfectly match the original precedent another player might care more about the freedom in form making. At the researcher/developer level, the game serves as a tool for larger scale data collection on the player's step-by-step process and their subsequent design schema developed as they move through the various precedent levels (Park and Vakaló 2001).

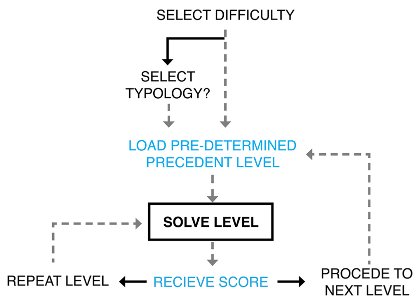

Figure 4. Game Overview Flowchart.

At the start of each level, the player is provided with a set of labeled puzzle pieces (game objects assigned a room function). The player then goes through an iterative process of object *instantiation* and *manipulation*, and *connection* using said puzzle pieces. *Implementation* refers to the initial placement of a game object in the active scene. *Manipulation* refers to the basic transformation of a game object within the scene (translation, rotation). *Connection* refers to the designation of two adjacent game objects as being ‘connected’, i.e. that a person could move between the two rooms. These three action types, in addition to other types of actions involved in the exploration of design schema and the recreation of a player’s prior puzzle solutions, form the *action-cycle* of our game. At the end of each cycle, the actions taken by the player are compared against the original precedent’s solution to score the player’s most recent actions in addition to saving said actions as a *design rule*, which are then added to a list of design rules that form a *design schema*.

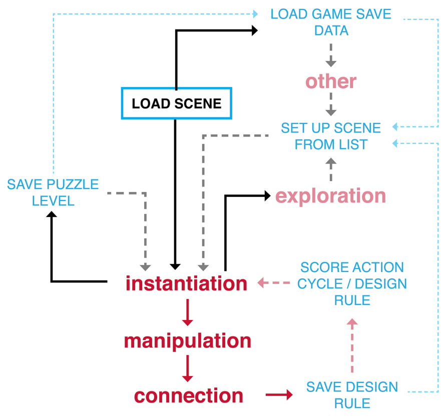

Figure 5. Action Cycle Overview.

Currently, we are looking at scoring in terms of the player’s outcome’s accuracy in terms of two different aspects of design: 1) overall design - how closely does their outcome match up with the original precedent’s formal arrangement? and, 2) functional relationships - how closely does the player’s connections between different room functions match to the originals? Not every player is going to prioritize obtaining a perfect score, caring more about the freedom of form-making. The ability for players to create design variations becomes an important part of setting up each scene. The elements of a precedent’s abstraction will affect how easily a player will be able to create a variation which makes sense using the set of objects provided to them. The process of generating design alternatives by the player is saved as a design schema which is a morphological sequence in the process.

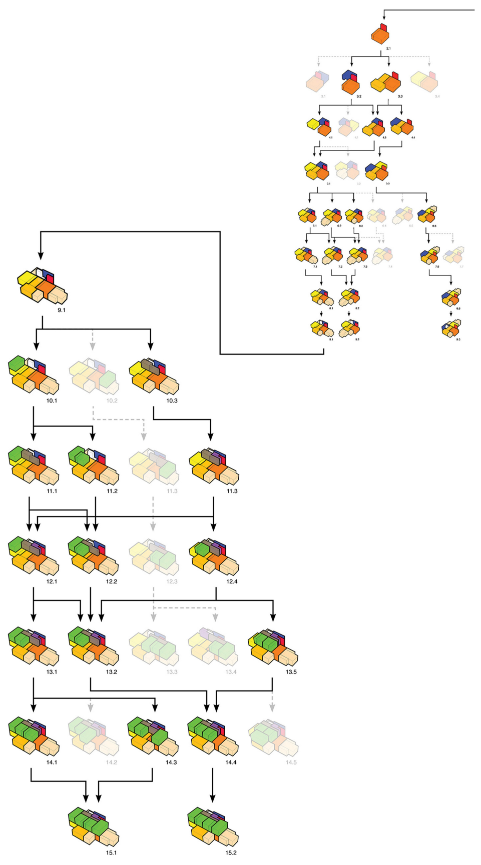
Figure 6. Generation of Design Alternatives.

One of the key issues that came up early in the development process was the question of what information is the player provided with at the start of each level? There are three types of information that are currently being explored in their potential for implementation in future development: 1) information embedded within the objects themselves, providing the player with not just the room geometry, but its respective function name; 2) images and information about the precedent given to the player just prior to the start of the level, but not accessible within the level itself (isometric drawings, building pictures, etc.); 3) information contained within the scene which could help guide players if they get lost or do not know where to start. The last type of information will convey the overall flow through the precedent spaces and provide basic functional organization between public, private, and service spaces.

The player controls are most directly tied to a player's ability to instantiate game objects within the scene, and manipulate properties and values assigned to each game object. Instantiation is done through the instantiate method native to *Unity*. Manipulation of the game object includes the translation and rotation of the said object. Currently, translation and rotation are done through the adjustment of said object's transform component which is responsible for defining the position, rotation, and scale of the game object within the active scene (the current level) based upon a set of constraints (i.e. corner-to-corner object snapping). Corner-to-corner snapping is exactly what it sounds like. The player can adjust a selected object's position by aligning one of its corners with the corner of another object within the scene. Connection is done through the detection of touching surfaces and player input on which game objects whose surfaces touch, should the game consider "connected".

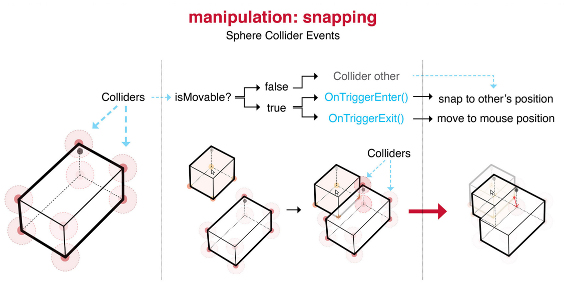

Figure 7. Manipulation: Snapping.

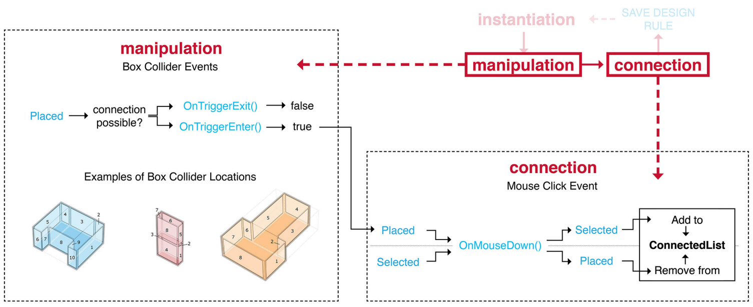

Figure 8. Connection.

There are several outcomes on the player's side of our proposed game. The most immediate two relate directly to two of the core player motivations looked at earlier in this paper: scoring and design variations. Another potential outcome relates to the idea of form as a placeholder. The game objects that constitute the puzzle are generated, on the game development side, through an abstraction process whereby architectural details were removed. These game objects could go through the opposite process on the player's end: applying architectural details to the puzzle pieces after the player has arranged them in a way they see fit. In this way more and more of the meaning behind the puzzle pieces and their arrangements could be unveiled to the player. Similar to the exploration of gameplay difficulty options, the potential for the later application of architectural elements to a player's puzzle solution has not been implemented within our game but is being explored for its potential implementation in future development.

4. Implementation

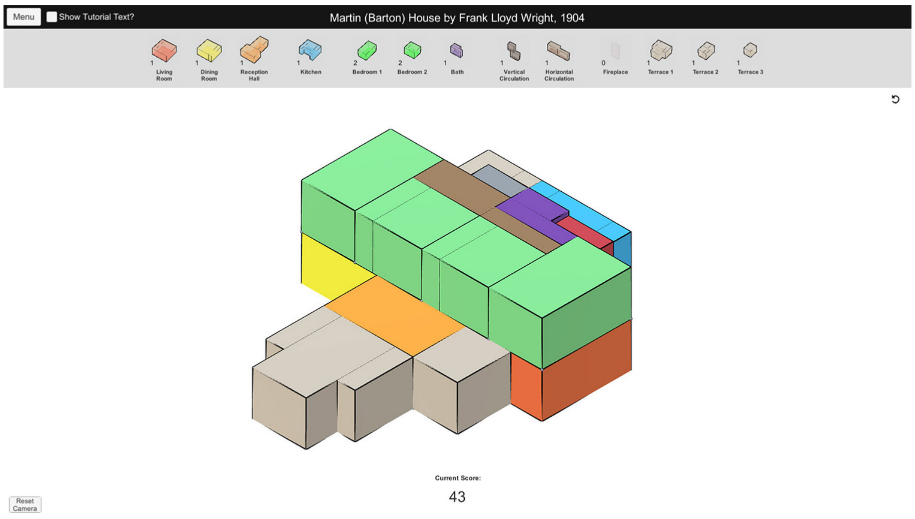

Figure 9. Screenshot of User Interface .

Instantiation is done using the *Intantiate* method native to the *Unity* game engine. When the *Intantiate* method is called, an instance of a prefab game object located within the game files is instantiated at a designated position (*Vector3*) and rotation (*Quaternion*) within the currently active scene.

Manipulation is done in *Unity* by changing the transform component of the game object using the dot operator (*transform.position* and *transform.rotation*). The specific constraints placed on each game object's potential positions are done through a series of trigger colliders attached to specific corners of each game object, resulting in the desired corner-to-corner snapping.

Connection is done using various trigger colliders that are attached to the surfaces of each game object that tell the game when two game objects come into contact. The game then conveys this information to the player in the *connection* stage, at which point a player clicks on the game objects they wish to connect to the piece they just placed (manipulated). Lists of which game objects the player as designated as "connected."

Design rules are saved every time a player completes an *action-cycle*. The game does this by saving a series of variables into a list. These variables are assigned based off of the puzzle piece instantiated in the most recent *action-cycle* (referred to as the current puzzle piece): information specific to both the prefab puzzle piece game object as well as its specific instance within the scene (like its function, shape, and id number specific to each individual game object); the position and rotation of the current puzzle piece within the game world space; and a list of puzzle piece game objects that where connected to the current puzzle piece (the one instantiated this *action cycle*).

Design Schema refers to the list of player's generated *design rules* that are structured in such a way that they show the path the player took to arrive at their solution. The *design schema* shows not only the choices the player chooses to keep but also the ones they discard, i.e. where they backtracked. This list is continuously updated during gameplay which each new *design rule*.

Scoring looks at the variables saved to the *design rules* and it compares them to predefined lists based on the formal and functional properties of the precedent the current game level's puzzle is based on.

Library refers to the set of scenes which contain each precedent puzzle. These scenes of an instantiation UI specific to the set of labeled puzzle pieces for the precedent, the list used for scoring, and any hints specific to the precedent. The geometric models are imported into *Unity* by the developer, defined by them during each scene's construction. Accordingly, the labels and scoring criteria are defined per each scene in the library.

5. Discussion

Shape grammar is a rule-based design system in which shape rules are created and applied based on the spatial relationships formed and transformation undergone by and between a set of placeholders within a given style. The basic structure of shape calculation lends itself to computer applications within the field of design, however, at present this requires making and applying design rule first in the design process rather than engaging the practice of design. Furthermore, a fairly high level of understanding morphological transformations necessary for applying design rules has been problematic for the usage of shape grammar in the process. These limit shape grammar's use as both a design tool as well as an educational tool when it comes to understanding the spatial relationships found within a given precedent style.

Our proposed game seeks to address the problems and provide an educational tool for learning about formal and spatial relationships found in a variety of architectural precedents with translating design actions into design rules and accumulating those rules as a design schema. The game provides a rigorous way of producing design variations in the guideline of the meaningful syntax with maintaining the level of design freedom to assemble given placeholders/objects, scoring the players' outcome, providing them feedback based upon the precedent they are working with, also allow the player to backtrack on a recorded design schema. The proposed game's ability to provide immediate player feedback based on their actions along with their playful nature allows us to a sense of reflection in action in the design process.

We are currently in the pre-production phase of development, with the goal of creating an alpha build (an early prototype). This level will be based on the Martin (Barton) House by Frank Lloyd Wright. This alpha level build will be used to perform preliminary, small scale player testing in order to determine any further tweaks to the overall gameplay mechanics prior to the further implementation of additional precedent levels.

References

Bojan, T. and Stojaković, V.: 2012, Shape grammar in contemporary architectural theory and design, *Facta Universitatis-series: Architecture and Civil Engineering*, **10**, 69-178.

Deterding, S., Dixon, D., Khaled, R. and Nacke, L.: 2011, From game design elements to gamefulness: defining "gamification", *Proceedings of the 15th International Academic MindTrek Conference*.

Duarte, J.P.: 2005, Towards the Mass Customization of Housing: The Grammar of Siza's Houses at Malagueira, *Environment and Planning B: Planning and Design*, **32**, 347-380.

Fischer, T. and Herr, C.: 2001, Teaching Generative Design, *The Proceedings of the Fourth International Conference on Generative Art 2001*.

Grasl, T. and Economou, A.: 2013, From Topologies to Shapes: Parametric Shape Grammars Implemented by Graphs, *Environment and Planning B: Planning and Design*, **40**, 905-922.

Koning, H. and Eizenberg, J.: 1981, The Language of the Prairie: Frank Lloyd Wright's Prairie Houses, *Environment and Planning B: Planning and Design*, **8**, 295-323.

Park, H.-J. and Vakaló, E.-G.: 2001, A Form-making Algorithm. Shape Grammar Reversed, *Proceedings of the Ninth International Conference on Computer Aided Architectural Design Futures*, 453-466.

Piazzalunga, U. and Fitzhorn, P.: 1998, Note on a Three-Dimensional Shape Grammar Interpreter, *Environment and Planning B: Planning and Design*, **25**, 11-30.

Schön, D.A.: 1983, *The Reflective Practitioner : How Professionals Think in Action*, New York : Basic Books.

Stiny, G.: 1978, The Palladian Grammar, *Environment and Planning B: Planning and Design*, **5**, 5-18.

Stiny, G.: 1980a, Introduction to shape and shape grammars, *Environment and Planning B: Planning and Design*, **7**, 343-351.

Stiny, G.: 1980b, Kindergarten grammars: designing with Froebel, *Environment and Planning B: Planning and Design*, **7**, 409-462.

Stiny, G.: 2012, *Shape : talking about seeing and doing*, MIT Press.

Stiny, G. and Mitchell, W.J.: 1978, The Palladian Grammar, *Environment and Planning B: Planning and Design*, **5**, 5-18.

Tapia, M.: 1999, A visual implementation of a shape grammar system, *Environment and Planning B: Planning and Design*, **26**, 59-73.

Trescak, T., Esteva, M. and Rodriguez, I.: 2010, Shape grammar interpreter for rectilinear forms, *Proceedings of. 4th Int. Conf. Design Computing and Cognition*, Stuttgart.