Turmitecture

A generic approach for autonomous topological generation

Stephan Droste¹
¹University of Stuttgart, Germany
¹http://www.casino.uni-stuttgart.de
¹stephan@casino.uni-stuttgart.de

Abstract. The paper describes and discusses a generic approach for generative design by Turing Machines operating on a three-dimensionally folded surface. It is part of an on going research on concurrent cooperative design processes in architecture. During the development of systems for simplified generic interaction in spatial design, it turned out that the basic operations are applicable to be processed by non-sophisticated automata. If the spatial configuration is interpreted as the medium for an ordinary state machine, the whole system adds up to a kind of Turing Machine. Since 2D-Turing Machines are often referred to as “turmites”, and the proposed system is based on a yet three-dimensional folded, but still two-dimensional surface - the automaton will be called “Turmitect”.

Keywords. Collaborative Design; Generative Design; Design Concepts; Shape Studies; Virtual Architecture.

INTRODUCTION

This paper presents and discusses an experimental research on generative architecture. A new kind of spatial automata is proposed and its capability of application in the process of form finding is explored.

Context

Unlike classical CAAD (for drawing and construction), the support of the early design stages – especially the creative process – is in fact discussed for decades, but still in the need of easily utilizable tools. Even the currently fashionable Parametric Design is mainly an approach of constructing carefully projected designs and lacks the intuitive interaction with the planner. The design itself still emerges from the ingenious mind of the architect.

Yet by simulating the growth and the emergent process of environmental structures, it should be possible to suggest designs that are neither deliberate nor anticipated. Thus, the designer becomes more a supervisor, trainer and judge for generated propositions, elevating Alexander’s much-cited army of clerks to the next level.

Outline

In this text, firstly the related work will be summarized, followed by the exposure of own assumptions, which lead to the basic concept. Subsequently an experiment is evolved and evaluated to check the suitability of the supposition. The paper concludes with the next preferable steps of the on-going examination.
PRELIMINARY WORK
The appliance of automata in the architectural process is already widely discussed. The idea of programming simple design tasks and the emergence of design solutions matching complex constraints seems promising since computer systems are capable of processing architectural models.

Architecture Machines
In his “Evolutionary Architecture”, John Frazer (1995) gives a comprehensive overview on the topic. Even if the efficiency of the machines has matured since then, the principles of generative space are still the same.

Beside the calculation of complex three-dimensional graphs, essentially interacting agents and generative automata, as well as genetic strategies are the approaches in computable architecture. It seems that in most cases either complex modeling of specifications are developed or on the other hand simple, idealized mathematical models are translated into architecture.

Both directions are complicated. The one needs sophisticated analysis of the specifications and the set of solutions. The other has to interpret the given patterns in an architectural way.

Simplicity and Complexity
While not in the domain of architecture, a new insight concerning creative machines was contributed by Wolfram (2002). His approach is not to design automata with an intention how they have to work, but creates classes of automata to investigate their behavior. It is shown that similar to the well-known “Langton’s Ant”, automata of different types with simple set of rules generate complex patterns in many cases and can be computationally universal.

BASIC ASSUMPTIONS
The basic approach of this work is to simulate the process of modelling, instead of defining constraints. If it is possible to break this process down into simple operations, it might be possible to develop powerful, learning programs for its support.

The fundamental determination hereby is the detachment of object and subject. While many generative concepts in architecture propose intelligent systems (based on agents or conditions), I propose an architectural space model as the object and a universal automaton as the subject.

Spatial Concept
The fundament of simulation is the modeling of a capable concept. Since this work is focusing on architectural form finding processes, there is a need for a spatial model and the possibility to operate on it. The basic idea is the model of a continuous surface, which separates void and volume (consequently, they are continuous themselves). The only operation is to sculpt this surface by simple concave or convex distortion. If to convex areas converge, they form a “bridge” and increase the topology of the surface (as converging conceives do via a “tunnel”). The result is a mass model comparable to classical plaster models, which have no (visible) separated voids and no independent (flying) volumes.

Design Operations
On the proposed spatial model, an acting subject has to operate. This configuration reminds of a two-dimensional Turing Machine, with a head running on a surface, using it as the storage. Since these machines are often referred to as “turmites”, the proposed one is called “Turmitect”.

The space and its configuration is the medium to be explored and designed by an automaton. The automaton is an ordinary finite-state machine as in the constitution of a Turing Machine. Though it is not capable of writing and reading symbols on a tape, but it distorts and interprets the sculpted surface.

This automaton can be seen as a cursor moving on the specified surface. The cursor has the capability of input, which is its own state and an interpretation of its position on the surface. The output is the operation on the surface, in principle counting five actions: go ahead, turn left, turn right, push and pull.
the surface at the actual position (in fact, the turning action could be merged into one, since the angle to the left equals the one to the right subtracted from $360^\circ$). For convenience, the operations that lead to inconsistency of the surface are ignored.

In a prototypical implementation, this cursor was also interactively controllable by a user via joystick. While the creation of buildings is time-consuming inconvenient, it could be proofed that this operations suffice to model the surface universally.

**Implementation**

The realization of the concept is done using a binary voxel space. While voxel data seem inflexible on the first glance, they have several advantages. The size of the stored data is constant for any spatial complexity. If implemented as cubic voxels, the turning and modeling actions are discrete and easily understandable, so the programming of the automaton is straightforward. Additionally, an existing voxel engine was available to render the process.

**EXPERIMENTATION**

**Objective**

In order to verify that simple automata operating on the given surface can show complex behavior, a simple experiment is proposed. A class of elementary Turmitects is created, so that the amount of possible sets of rules is very limited. Then, all possible automata are tested and logged. The results are inspected and classified by complexity.

![Example set of rules and resulting surface (Turmitect No. 94)](image)
Setup
Since the testing is still manually, the attempt was to create a Turmitect, which could be, encoded in just one byte, hence a number of 256 possible configurations. With this precondition the states and the operations of reading and writing must be reduced to a minimal set of rules.

Surprisingly this can be achieved with an automaton featuring two states. Firstly, the reading is radically simplified, just differentiating horizontal and vertical ground. Secondly, the actions are combined, so that there are solely two different actions:
• Turn left – Pull – Go ahead
• Turn right – Push – Go ahead

Thus, with to possible inputs per state and the action and sequential state coded each in a single bit, this class of Turmitects can be stored in one byte (see figure 1, bottom). Each sample can be named by the decimal interpretation of its eight-bit configuration, hence from #0 to #255.

The setup of the experiment sets a run with a specific number of steps aside for each of the possible Turmitects. After each pass the results are evaluated. The promising samples with non-trivial behavior are forwarded to further cycles.

Evaluation
As expected a predominant number of sets of rules led to no or simple repetitive patterns. Yet 36 Turmitects featured a non-trivial behavior, which did not end in a deadlock during the first 64 steps. Out of these 26 passed 128 steps without simple repetition. After 4096 steps, there are still 6 Turmitects without deadlock and chaotic, complex output. Of these 6, 2 pairs (#14&44, #74&104) are obviously equivalent. After more than 4096 steps #62 starts to produce a repetitive non-trivial pattern, similar to #107 after more than 1024. Despite, these samples appear interesting in terms of architectural criteria, because they all spread on the surface leaving a flat relief. Noticeable in contrast is a number of Turmitects producing spatial complex structures, before they reach a deadlock. For example #94 passes more than 256 steps before ending in an endless circular operation (see figure 1). Similar but with less states are #114 and #141. Several Turmitects fall into a simple repetition after the creation of complex patterns: #11, #26, #77, #79, #111, #118, #203, #205.

Conclusion
The tested class of Turmitects is obviously not capable of creating valuable architecture. Yet as the simplest representative of its species, it shows that complex behavior by simple rules can be translated to spatial design. With the result of 14% of non-trivial samples in the first steps it seems rather promising to investigate this topic. The conceptual description of the Turmitect and the implementation should be sufficient to build on.

OUTLOOK
The results of the first testing suggest further work.

To achieve more diverse behavior, it is necessary to extend the automaton. Basically, this can be done by the use of more possible states. Moreover, the reading possibilities have to be refined. Supposable, this includes distinguished orientations, reading of the last step (is it straight, convex or concave?) and more complex inputs, like spatial configuration (“closed cave”, “hill”, illumination, etc.). Furthermore, if the minimal storage is not required, the five possible actions can easily be differentiated.

The testing of the Turmitects has to be more extensive by varying the constraints. Conceivable are preconfigured environments, maybe also existing architecture. Then, multiple Turmitects can operate on one surface, provoking interaction.

Finally, it might be tested if the automaton fits into the design workflow. An integrated or automatic evaluation of the Turmitect’s quality is essential. If this can be achieved, even evolutionary developing automata are easily imaginable.

REFERENCES

