TEACHING SYSTEMS-THINKING WITH ALGORITHMIC PROCESS

Introduction to computation and programming with Processing programming language

CARL R. LOSTRITTO
University of Maryland
College Park, Maryland USA
Email address: carllos@umd.edu

Abstract. This research investigates how algorithm design and scripting as pedagogy can affect generalized design ability and understanding. Logical, systematic thinking is considered foundational in developing architectural design aptitude and is explicit when designing algorithms. The course work presented mandates the construction of process rather than product. Scripting is implemented not as a means to an end but rather a medium for exploration. More valuable than formal generator or problem-solver, these scripted designs test direct aesthetic implications. Further tested is the role of animation in de-linearizing the design process. By isolating the algorithm as topic, technique, and concept, scripting skills and the produced artifact are extendable and are translatable to other media. Algorithm design is presented as a 2-dimensional but temporal endeavor: students script an animate, interactive vector-based image. This facilitates the transition from algorithm to spatial experience while also readying students for form-based explorations. The 2-d temporal exercise is of a similar order of complexity to a 3-dimensional static condition. Pieces of the animation structure are provided as a canvas, specifically the ability of the viewer to manually control a single parametric variable that affects the visual output through a user-interface element. The following and final project of the course expands upon the technique of scripting image in the design of an experience by collaging video, images and animation.

1. Course Context

"Introduction to Digital Media" is a design seminar that seeks to leverage media exploration and skills teaching toward a critical approach to architectural design process. Students in the course are tasked with creating an interactive animation using the Processing programming language. It is the first course in the curriculum dealing expressly with the implications of digital media technique. The course balances between a need to convey technical knowledge with an academic mission of analysis, speculation and conceptualization. Scripting tests this paradigm given the nature of languages in this context. In architectural education, scripting and especially programming are often treated as an advanced and specific application of extra-disciplinary knowledge. Still, algorithmic concepts are historically and conceptually rooted in multiple canons of architectural thought, even if the term is itself absent. Programming is the most direct connection available to apply algorithmic processes. Further, programming as technique has obvious implications in digital media design processes if parameterization is valued. Terzidis appropriately shifts the discourse from "computerization" to "computation" and beyond in noting, "An algorithm is not only a computer implementation, a series of lines of code in a program, or a language, it is also a theoretical construct with deep philosophical, social, design, and artistic repercussions. (Terzidis, 2006)

Processing, "the first full-featured programming language end environment to be created for artists by artists," (Greenberg, 2007) provides an environment ideal for teaching the concepts of algorithmic design without syntax obstacles of many languages. Processing's strengths go beyond pedagogy. Processing can be fruitfully extended into a multi-media design process. As an open source project, Processing's development tends to eliminate disciplary biases. In a course where the primary focus is digital modeling, the geometric class structure provides a reduction in the time spent introducing the language. For many students, this exercise is their first foray into coding of any kind. Processing can be used to generate vector data for fabrication or modeling or to generate a standalone Java Applet able to incorporate human interaction.

The programming exercise shown here lasts for one week of the course and leads to a three-week project. The exercise outcome and methods are leveraged in the design of an animate spatial construct as the final project.

2. Exercise specifications

A pre-made processing "sketch" is provided to the students. It produces no visual content but establishes the framework of the applet. Developed by the author for the students in the template are a slider, the exclusive user-interface element in the generated applet that affects a global variable, 'T' and two nested For-loops. The loops are associated with x and y variables and provide the underpinnings for the generation of a grid. Students add code within one and/or both of the loops. They are required to create a graphic construct that "significantly" varies with 'T." Students are required to use at least one conditional, "if-then" statement. The only subjective directive given to the students is to, "strike a balance between perceived order and perceived chaos at certain or all ranges of 'T'."

2.1. CANVAS AND TOOLSET

Students are exposed to a resource of geometric functions (point, line, polygon, rectangle, circle, ellipse, spline and Bezier curve) and manipulator functions (tone and hue). The key to this exercise, however, lies not in the application of these built-in functions. Instead, it is the straightforward accessibility of these geometries that allows them to be ignored. This is a study of computation, not of scripting or programming skills. The operand or content is largely, if not completely, irrelevant in the process. Sometimes this content becomes invisible in the product as effects dominate the aesthetic. Exploration and play can occur based on computational adjustments. For example, an adjustment in the increment of one of the two loops affects the spacing between one axis of the grid. A non-linear increment produces a gradient effect.

Students publish the applets online as the formal completion of the exercise. These are available at http://0095b6.com/lostritto/?p=263. The experience of interacting with peer projects directly, as opposed to a conventional demonstration-based method, is an important pedagogy. In design education this is a rarely intimate opportunity to establish connections across contrasting design processes, as the animation here is explanatory and conceptually reductive. An assertion made through this research is that animation more clearly represents the computational relationships than does the source code. This is in contrast to the default paradigm of digital animation: as a means to vary parameters towards an exponential versioning experienced virtually simultaneously.

Figure 1. "Canvas" code provided to students.

3. Programming as a medium, computation as a creative process

Student work in Figure 1 demonstrates how the programming canvas provided positively provides structure but not direction. Student success is dependant upon the selection and execution of a spatial intent.

Often a programming is treated as a tool, a means to carry out or execute a function. In many cases programming has proved its value in this regard. As a medium however, programming itself can provide meaningful aesthetic content. In this exercise the programming media provides the student direct feedback: by testing the applet and affecting the single parameter the students understand computational operations in terms of their visual corollaries. As examples, the modulus operator establishes a rhythm; a range-based condition can denote an area; and addition creates a geometric translation. (Terzidis 2006)

While critical to linking computational technique and compositional concept, this relationship is often not apparently direct. This exercise seeks to provide students an environment for experimentation and play though coding. Even in a more specific or architectural application of the programming process where deductive reasoning characterizes the generalization necessary to design an extendable algorithm, this foundation allows for aesthetic considerations to remain at the forefront. Also prevalent is the maintenance of aesthetic architectural values through a

process performance or functional articulation that characterizes most scripting and programming applications.

Viewed as a set, the work in figure 2 reveals certain patterns inherent to algorithmic design. Taken as a whole, the gallery provides significant insights into the nature of the roots of digital media. Part-whole relationships are paramount. Also noticeable is a tendency towards self-similarity. These characteristics define the nature of a system.

4. Algorithm as Media

This research addresses a "chicken and egg" paradox with regard to teaching and applying computational design and parametric processes: if this methodology is to resonate deeper than an architectural expression of process, the meaning of the process must be understood prior to or in parallel with technical learning. Simply put, students often have difficulty conceptualizing computational implications prior to engaging in computational methodologies directly. However, some pre-rationalization is required to direct these techniques toward fruitful outcomes. This pedagogy seeks to address this by providing process-based direction towards divergent outcomes that collectively express the potential of algorithm as media.

Emergence is perhaps the most widely used terms to blanket the broad implications of digital media. As is often the case with higher order concepts it is difficult to leverage productively by students. Emergence is a necessary topic in the intellectualization of digital media. By tying multiple conditions directly to the parameter, T, secondary and tertiary effects become compositionally primary. The bottom set in figure 2 demonstrates and emergent circular shape. Sometimes emergent effects are similar to their constituent parts—dozens of larger circles define the visible (and emergent) circle form, for example. Another common effect is a Moiré pattern as multiple layers of offset grids overlap. The ability to animate in a controlled fashion directly leads to the ability to capture and control these effects. Emergence, when understood by students can shift from an artifact of digital process to a controllable means to achieve temporal transparency. These effects are more useful as topologies in a design process than subject (or variables) affected by the algorithm (or script). Without animation these effects are often ineligible and untranslatable to other contexts or applications.

McLuhan, in thoroughly documenting his famous truism, "The Medium is the Message," makes a distinction between "hot" and "cold" medias. This scale associates "high definition, low-interaction" media as "hot" (As an example, photography or film). Conversely "cool" media are "low definition" and thus require "high interaction," (examples are the cartoons

or the telephone). (McLuhan, 1964) Digital media is traditionally "hot" by this measure. For architects, the ability to model and visualize in four dimensions clearly extends human thought with significant but sometimes cripplingly linear detail and formal resolution. This exercise seeks to "cool" digital media by requiring speculation and interpretation based on its inherent qualities and patterns through temporal interaction. Certainly a "software made it" rationale, or worse—label, is intellectually lacking (Meredith, Maeda, Terzidis), yet a control of this medium can be fruitful in engaging media to new ends. In writing about the work of Aranda/Lasch Balmond explains the significance of algorithmic media as, "alive and spontaneous, capable of self-ordering, embedded with vision…" (Balmond 2005).

5. Simultaneously reduce and extend: systems in design

Commonalities exist between all programming and scripting languages. This nature facilitates the translation of algorithms across varying syntaxes and allows for students, after this exercise, to rapidly learn advanced languages or scripting environments. More valuable in this exercise though is the broader implications on the design process. Scripting, programming or even more broadly algorithmic design necessitates a systemization of function, form and space: rules and parameters create a whole from a set of parts. This process mandates a level of abstraction whether through topology, typology, classification or categorization. Simultaneously, a multiplicity of specific outcomes can be explored in a non-linear fashion in the projects animate qualities; artifacts have value out of context; translation can occur between media by adding meaning.

This value is isolated from an overtly architectural language in that the artifact produced is more appropriately described as animate drawing rather than animate architecture. This process places considerable value on representation. Representative media can, as is the case here, indirectly translate to architecture in a meaningful way despite physical qualities.

In Figure 2 it becomes evident that the product, already presented in terms of pedagogical value, also has architectural value as graphic non-repeating patterns have implications to surface and space.

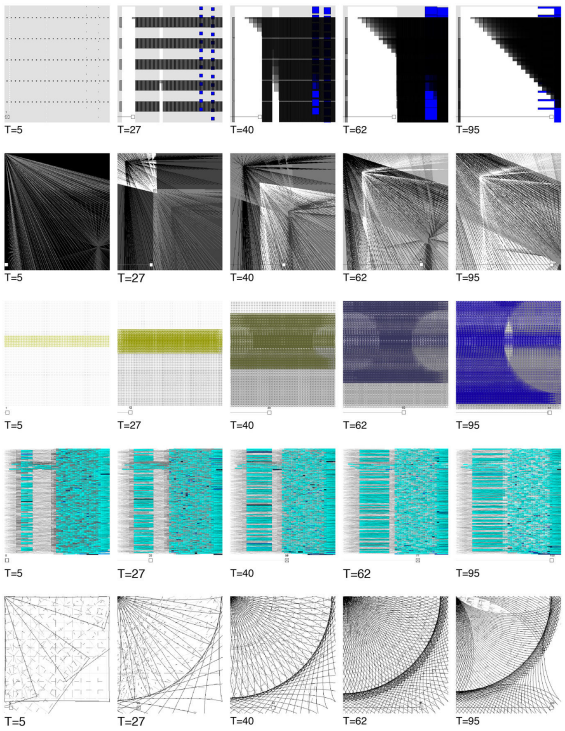

Figure 2. Frame captures of student-designed applets as user manipulates the value of 'T'

Isolating algorithmic study is beneficial in circumventing potential ideological shortcomings of parametric design as its own end. To that point Meredith posits, "Parametric design fits within an architectural discipline that is simultaneously searching for a unified organizational clarity (the diagram, parti, etc.) and visual complexity (Venturi). But no matter how patterned, totalizing and parametric it is, architecture is inevitably a fragment…It requires differentiation for it to become Architectural, and it is the socio-political that allows it to escape the emptiness of objects." (Meredith, 2008) Systemization is in this context the mechanism for eventual meaningful parameterization. These scripted images are not Architecture and require a level of abstraction before it can be translated in contexts. Further, algorithmic thinking by students, even (and especially) if the result is not formally scripted, can promote operational strategies in the design process. The value of systems thinking toward rational creativity is ironically absent in a "scripting toward an end" mindset that is required in the application of algorithms in machining or fabrication of surface-derived forms.

6. Low-level process; higher order thinking

As software becomes more specialized the low-level processes are further from immediate grasp. Without entering the relevant debate as to whether form should "follow software," this exercise and the body of knowledge embedded in the course seek to empower students to develop an instinctive understanding of software. John Maeda notes that pervasive acceptance of designers ignorance of basic process is unique to design. He notes that "Computation…is often misunderstood because the complexities o software and the process of creating it are invisible….this ignorance is forgivable in the layperson, but has been oddly tolerated in a field like visual design, rooted in the craft of printing, where basic knowledge of ink and paper is requisite. The digital medium still remains only vaguely understood beyond the tools that are used." (Maeda 21) Higher-level processes tend to produce outcomes that are more equal part authored by software engineer and user of that software. Because the underlying algorithmic processes are so specific the potential output is narrowly predictable. Higher-level processes also tend to be procedural from the point of view of the user. Procedural processes are less valuable learning experiences as they are less likely to be re-applied out of context. The work presented here seeks to reinforce deductive reasoning, logic and rationalization through graphic, spatial and aesthetic means.

7. Algorithm to animation

This exercise serves as the course transition from physical media to animation methods and techniques. The product is animate but is neither film nor video. In laying the foundation for animation as a process towards the creation of animate architecture students become equipped to animate form and space in meaningful ways. That the animation is user-controlled reinforces the potential of the medium itself beyond a hyper-perceptual virtualization of human first-person movement.

Again using McLuhan's hot-cold scale, animation is made to shift from a hot to cold medium. Animation extends beyond a "high definition" representation of perceptual experience into constructs which with ambiguity and potential as a lens.

Subsequent projects allow students to apply new technical knowledge towards animate form and/or to translate found or emergent effects into truly three-dimensional environment.

7.1. ANIMATING SPACE FINAL PROJECT

The final project of the course directs students to "explore the implications of composing from the first-person perspective by overlapping video and a series of static 3-d images (the resolution of their variance will determine to what extent they are perceived as animate…The scripted interface exercise explored implications of a graphic experience varying with a parameter. Conditions from that exercise can be translated into a animate condition if that parameter is applied to the time as represented in video." (project brief)

Exporting video from the live frames is one technique available to students. In the project some students record interaction with the applet in constructing source video. Another means of translation lies in the ability to export vector graphic output from a frame, series or sequence of frames. This information can be used towards the construction of a digital model or towards the extraction or overlay of still images.

Beyond providing the technical connection, the exercise lays the conceptual foundation for animation. The parameter, 'T,' when understood as linear time reveals to students that the algorithmic structure of each construct and that of animation software, while vastly different in terms of complexity and scope, are fundamentally similar. In this way animation is made accessible as a medium for creation.

Figure 3. Static capture of two frames from student project

8. Conclusion

The exercise and project presented here serve as successful introductions to programming techniques by linking algorithms and animation. Students of multiple levels are able to, in relatively short period of time, visualize and express computation structures. By isolating the meaning of this process, methodologies and priorities are translatable and extendable. The artifacts

produced are equally valuable in other contexts especially when considered as indirectly representational. Fundamentally this research presents animation and programming as media. In this way they become pedagogical devices with interconnected meaning allowing technical lessons to translate into an approach to design that values rational systematic process in abstract contexts.

Acknowledgements

This research and Introduction to Digital Media course have been facilitated by support and mentorship from Assistant Professor and Studio Coordinator Michael Ambrose.

References

Maeda, J., 1993. *Design By Numbers*. Cambridge, Massachusetts: MIT Press.

Meredith, M., 2008. Never Enough. In: From Control To Design, Parametric/Algorithmic Architecture. SAKAMOTO et al ed. Barcelona, Span: Acta-D.

Mcluhan, M., 1964. *Understanding Media, The Extensions of Man*. Cambridge, Massachusetts: MIT Press. Media Hot and Cold, 22-32.

Terzidis, K. 2006. *Algorithmic Architecture*. Oxford: Elsevier Ltd. Prologue, xi-xv; Scripts, Algorithms and other predicaments, 75-103.

Balmond, C., 2005. Foreword. *In: Pamphlet Architecture 27, Tooling*. Aranda/Lasch ed. New York: Princeton Architectural Press, 7.

Berkley California: Friends of ED, xx.