

CARNEGIE MELLON UNIVERSITY

School of Architecture

College of Fine Arts

Thesis

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computational Design

Generating, Simulating, Interrogating:
A Computational Design Thinking Framework


Scott Parker Donaldson

ACCEPTED BY ADVISORY COMMITTEE:




Daniel Cardoso Llach Principal Advisor

12 / 15 / 17
December 15, 2017



Molly Wright Steenson Advisor

12/15/17
December 15, 2017



Stuart Candy Advisor

12 / 15 / 2017
December 15, 2017

Acknowledgments

Were I to follow my own recommendations in this research, there would exist some script to automatically generate and simulate these written acknowledgments. Instead, what follows is a decidedly non-computational expression of my gratitude.

First and foremost, thanks belongs to my advisor Daniel Cardoso Llach, who welcomed me into this program, provided research opportunities, and encouraged and guided me throughout the thesis process (not to mention shaping, through his own work, much of the theoretical background here). My thesis committee members also deserve thanks. Molly Wright Steenson drew my attention to cybernetic discourses in architecture and design, as well as highlighting pedagogical models relevant to the computational design thinking framework. And Stuart Candy helped shape my view of simulation in the context of futuring, as an exploratory, speculative design method.

This thesis also could not have been completed without the cooperation of the various people I interviewed, whose work has been deeply inspiring and central to the ideas I put forward here. In addition, the participants in the various design workshops I conducted helped shape the case studies in the Generating and Simulating chapters.

I thank my peers in the M.S. Computational Design pro-

gram: Javier Argota, Adie Al-Nobani, Camille Baumann-Jaeger, Hetian (Darcy) Cao, Cecilia Ferrando, Atefeh Mahdavi Goloujeh, Yuqian Li, Yingxiu Lu, Rachael Tang, and George Zhu, for their collegiality and support. In particular, I'm grateful for attending ACADIA 2016 with Javier, Camillie, and Cecilia, and to Atefeh for being my partner in conducting the design workshop I describe in the Simulating chapter (I especially look forward to Atefeh's upcoming thesis on participatory simulations). Outside of my cohort, I appreciate conversations in and around the CodeLab with Ardavan Bidgoli, Pedro Veloso, Harshvardhan Kedia, Ian Friedman, and others, as well as with Dan Taeyoung of Columbia GSAPP.

I'm grateful to my parents, Pam Parker and William Donaldson, for putting up with (and usually supporting) my many shifts, from mathematics to art in undergrad, from web development to architecture, and now to computational design. I can't promise that the future will be any more linear, but I look forward to sharing it with them.

Finally, this thesis is dedicated to my wife and enduring partner, Lisa Otto, who has been my greatest supporter personally and emotionally (and whose own design research and ideas have and continue to contribute to my intellectual growth). Lisa, I look forward to designing our future together.

Contents

Abstract	5	Activity 1: Negotiating an Intersection	36
Introduction	6	Activity 2: Generating Factors and Affinity Diagramming	37
Personal History	6	Activity 3: Loopy	38
Background	7	Activity 4: Design Interventions	39
Methods	11	Findings	39
Computational Design Thinking Framework	15	Future Work	41
Generating	16	Conclusion	42
Overview	16	Interrogating	43
Rulemaking	16	Overview	43
Exploring (or, Searching)	18	Hacking	44
Automating	19	Unlearning	46
Case Study: Worldmaking	21	Transforming	47
Background	21	Case Study: Coons Patch Reconstruction	49
Participants	23	Background	49
The Silent Game	23	Interface	49
The Reference Game	25	Discussion	50
Discussion	27	Conclusion	51
Conclusion	29	Conclusion	53
Simulating	30	Contributions	53
Overview	30	Discussion & Future Work	53
Emerging	31	Appendix	56
Futuring	33	Interview Subjects	56
Case Study: Reimagining Urban Intersections	35	Bibliography	57
Introduction	36		
Participants	36		

Abstract

Computational design is often depicted as an instrument for analysis or production, but it is also a space in which to explore and create new ways of working and thinking. This thesis explores how, through critically engaged practice, designers working computationally are uniquely able to envision and work toward desirable futures, challenging a techno-utopian status quo and projecting humane alternatives. What computational design methods, approaches, and strategies can help to bring about these desirable futures?

Through primary research involving interviews with computational design practitioners, developing interactive software prototypes as investigative tools, and conducting design workshops, I investigate various modes of working computationally. Building on this research, I propose a

three-part framework that synthesizes high-level approaches to computational design work. The first component, generating, reveals how computation enables the designer to work at various levels of abstraction, navigating large possibility spaces. The second, simulating, provides a frame for envisioning and modeling potential interventions in complex systems. Finally, interrogating, drawing from both Schön's 'reflective practice' and Wark's 'hacker ethos,' encourages computational designers to critically question their tools and practices in order to discover new ways of working and thinking. I conclude by discussing potential embodiments of this framework in computational design education.

Introduction

Personal History

I was in the second semester of a 3-year Master of Architecture program and was struggling. I had been accepted into the program a few years out of a liberal arts undergrad, and, having spent most of the intervening time working as a web designer and developer, considered it a pivotal shift for me. Architecture culture was, however, completely unfamiliar. In drawing and representation courses, as well as the core studio, I was continually frustrated by the seemingly arbitrary nature of architectural form, and perceived a lack of scientific rigor in analyzing and solving design problems through it. Architecture defied the systematic rationalization I was used to in digital design contexts. So I gravitated toward projects where I could apply my background in math and programming to architectural design — for example, representing the paths of a city park with an undirected graph structure, building a custom software tool to generate rectilinear forms from sketches, and likening the functioning of spaces in a building design to ‘cooperative parallelism’ (as in multithreaded computing). Of course, a park is not a graph, and a building is not a computer, but these metaphors helped me to comprehend the world of architecture — and made me a misfit among aspiring architects.

Not long after my decision to take a leave of absence from

the M.Arch. program, I discovered the Computational Design track at Carnegie Mellon University and enrolled. Here was a world of people whose work and research didn’t fit neatly into the traditional disciplines, who were using computation and digital technology not only as instruments for doing design, but as lenses and metaphors for design, and as areas to be explored, to bring back new ways of thinking and designing. The Master of Science in Computational Design program has given me a wide latitude for my research, and the opportunity to take courses from architecture and computer science, design and human-computer interaction, all under a multidisciplinary umbrella. Through it I have come to see computational design (like architecture) as an openly promiscuous field, borrowing from diverse disciplines, with the potential to apply its approaches to larger questions of technology, design, and society.

In this thesis, my aim is to bring together the methods, techniques, and strategies of computational design that are particularly oriented toward addressing problems of social complexity, and that provide productive frames for thinking about and doing design. My hope is that this work will combine the most fruitful practices of both computation and design, and in doing so, show that these worlds are not at all disparate, but significantly overlap. My goal is to provide illustrative approaches for those with a compu-

tational mindset interested in learning to work in ambiguity and socio-technical systems as well as designers who want to incorporate computation into their practices without giving up their unique, human role.

Background

Although mathematical practices in design extend to antiquity (Vitruvius wrote extensively on proportion and symmetry in *Ten Books on Architecture*¹), a potential genesis of computational design might be traced to Ivan Sutherland's 1963 PhD dissertation, *Sketchpad*². Now widely recognized as the first computer-aided design program, *Sketchpad* presented not just a new platform for designing, but a new *paradigm* for designing. A digital representation of points, lines, and shapes decoupled from drawings or physical models promised a new mode of plasticity for the architectural designer. By visualizing and manipulating objects in digital space through a computer interface, the designer gained a detached, abstracted view of their work. Sutherland and his fellow researchers were fully aware of (and striving toward) the transformative potential of their work, a history charted by computational design scholar

(and my master's thesis advisor) Daniel Cardoso Llach in his 2015 book *Builders of the Vision: Software and the Imagination of Design*.³ Notably, Sutherland's PhD advisor, mathematician and designer Steven A. Coons, in a paper published the same year as Sutherland's dissertation, wrote: "The design process is unpredictable. Indeed part of the design process consists in *designing new ways to perform the design function itself*"⁴ [emphasis mine]. *Sketchpad* was never used in industry, but succeeded in projecting the future of computational design. While technology has advanced greatly in terms of efficiency, ease of use, and computational power, the innovative features of *Sketchpad* remain central to 3-dimensional modeling and computer-aided design software today.

However, with increasingly powerful computation came an ethos that declared that the driving force behind new design possibilities was technology alone. Philosophically, Coons, as well as his student Nicholas Negroponte (founder of the MIT Media Lab and the One Laptop per Child initiative), were highly optimistic about the role of technology in driving the future. For example, as detailed by designer and historian Molly Wright Steenson in *Architectural Intel-*

1 Pollio, Vitruvius. *Vitruvius: The ten books on architecture*. Harvard University Press, 1914.

2 Sutherland, Ivan E. "Sketchpad: a man-machine graphical communication system." *Transactions of the Society for Computer Simulation* 2, no. 5 (1964): R-3.

3 Cardoso Llach, Daniel. *Builders of the vision: Software and the imagination of design*. Routledge, 2015.

4 Coons, Steven Anson. "An outline of the requirements for a computer-aided design system." In *Proceedings of the May 21-23, 1963, spring joint computer conference*, pp. 299-304. ACM, 1963.

ligence: How Designers, Tinkerers, and Architects Created the Digital Landscape, Negroponte envisioned “an [artificially] intelligent environment that we would all eventually inhabit and that would eventually surround all of us.”⁵

Outside of architecture discourse, futurists such as Ray Kurzweil see the history and future of humanity as determined by technological advances, proceeding predictably and mechanistically, without critically considering the social and political forces at play. Computer scientist Stephen Wolfram, in his 2002 book *A New Kind of Science* promotes the notion that computational processes might replace the traditional, analytical scientific method. In turn, adapting Wolfram’s ideas to the world of architectural design, historian Mario Carpo writes of a ‘Second Digital Turn’ in architecture, following the widespread adoption of digital practices in the 1980s and ‘90s, wherein designed forms can be generated and optimized by algorithms.⁶ In his telling, the future of architectural design is written in increased computational efficiency. A view of technology as an autonomous force leaves one wondering: What is the role of the designer? Is there room for humanism?

As discussed by Cardoso Llach in *Builders*, a related (but distinct) school of thought to technological autonomy is

5 Steenson, Molly Wright. *Architectural intelligence: How designers, tinkerers, and architects created the digital landscape*, MIT Press, 2017.

6 Carpo, Mario. *The second digital turn: Design beyond intelligence*. MIT Press, 2016.

the practical, everyday view that computational technologies are neutral tools. This theory holds that in architecture, for example, it is possible to directly translate a designer’s vision onto paper or into a digital model, and that different design softwares attempt to make this process more smooth and seamless. While this view undoubtedly places more emphasis on human agency when using technology (arguing that architects, not algorithms, design buildings), it is equally pernicious in claiming that technologies do not possess inherent values or biases. Historian Leo Marx points to the 19th-century development of a notion of ‘technology’ (constituting not only machines themselves but systems of technical and social complexity) as a moment which allowed ‘neutral tools’ to “distract[] attention from the human — socio-economic and political — relations which largely determine who uses them and for what purposes.”⁷ In particular, as further detailed by Cardoso Llach in *Builders*, discourses of computers and software today are driven by frames of autonomy and neutrality, a dichotomy which “hides a great deal... By construing software systems either as autonomous agents or as neutral tools for design, we shut down their politics... [and] their poetics... as territories of creative exploration.”

Another, more extreme version of technological determin-

7 Marx, Leo. “‘Technology’: The Emergence of a Hazardous Concept.” *Social Research* (1997): 965-988.

ism is the cognitive theory of computationalism. As digital humanities scholar David Golumbia writes in *The Cultural Logic of Computation*, computationalism is “the view that not just human minds are computers but that *mind itself* must be a computer — that our notion of intellect is, at bottom, identical with abstract computation.”⁸ Similarly, in their 1986 book *Understanding Computers & Cognition*, Terry Winograd and Fernando Flores carefully situate computers within the rationalistic tradition of thought. Concerned with the design of computer systems, their work is deeply relevant to broader design problems (including and especially computational design). They write, “An understanding of what a computer really does is an understanding of the social and political situation in which it is designed, built, purchased, installed, and used.”⁹ Clearly, approaches to computational design that avoid the pitfalls of both technological autonomy and neutrality must also eschew a primarily rationalist worldview. While computers are essentially machines that perform logical calculations, human interaction with them is socially and materially contingent. As anthropologist and human-computer interaction researcher Lucy Suchman writes, computers are situated in and inextricable from socio-technical systems.¹⁰

8 Golumbia, David. *The cultural logic of computation*. Harvard University Press, 2009.

9 Winograd, Terry, and Fernando Flores. *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.

10 Suchman, Lucy A. *Plans and situated actions: The problem of human-*

Logical, rationalist thinking about computation and design cannot easily tackle problems of significant social complexity. Instead, systems thinking and cybernetic philosophies provide a path toward working with and understanding complex systems. Texts such as physicist Fritjof Capra’s *The Systems View of Life*¹¹ describe how the forms and affordances of material objects have unpredictable effects at scale, in turn being affected by their contexts. Environmental scientist Donella Meadows introduces the powerful notion of leverage points¹² — the idea that, by analyzing the systems one is working in, one can predict what will be a more successful type of intervention (or at what *level* to intervene). Computationalism points to the lowest level in Meadows’ hierarchy of leverage points — constants, parameters, numbers — as an adequate site of intervention. It suggests that data is objective, and that conclusions can be derived from data that translate into actionable, instrumental designs to solve well-defined problems. For a designer with a notion of working within socio-technical systems, this clearly is a gross oversimplification.

machine communication. Cambridge University Press, 1987.

11 Capra, Fritjof, and Pier Luigi Luisi. *The systems view of life: A unifying vision*. Cambridge University Press, 2014.

12 Meadows, Donella. “Leverage points: Places to intervene in a system,” 1999.

Fortunately, a body of design and computation researchers and scholars are underscoring socio-technical complexity in their work. As mentioned above, Lucy Suchman's notion of situated design technologies provides a strong counterargument to the false dichotomy of autonomy and neutrality. Design researcher Donald Schön describes the act of designing as a 'reflective practice,' with the designer engaged in a continuous conversation with their materials.¹³ Together, these theories have been highly instructive in formulating the third component of the computational design thinking framework put forward in this thesis. In addition, historians, scholars, and theorists such as Daniel Cardoso Llach, Molly Wright Steenson, Yanni Loukissas, Sherry Turkle, Kazys Varnelis, and Tara McPherson productively challenge and complicate received narratives of computation, architecture, digital technology, and design. And practitioners such as Laura Kurgan of Columbia University's Center for Spatial Research, Taeyoon Choi of the School for Poetic Computation, and the Dark Inquiry collective of technologists, artists, and writers, all put forward work that critiques existing modes while also projecting new ways of being: Desirable, humane futures to strive toward. For example, a recent app created by Dark Inquiry, called Bail Bloc,¹⁴ applies spare computing power on its

13 Schön, Donald A. *The reflective practitioner: How professionals think in action*. Basic Books, 1984.

14 Bail Bloc, <https://bailbloc.thenewinquiry.com/>, accessed December 1, 2017.

users' laptops to mine cryptocurrency, which is exchanged for U.S. dollars on a monthly basis and put toward paying onerous bail funds for those awaiting trial. The project redirects existing technologies which might otherwise exacerbate inequality toward more egalitarian, just ends.

In addition, two recent and timely events seem to offer good omens of things to come. The first is a symposium called *Computational Design: Practices, Histories, Infrastructures*, held in October 2017 Carnegie Mellon University in Pittsburgh in conjunction with the exhibition *Designing the Computational Image: Imagining Computational Design*.¹⁵ Tracing the roots of computational design history to mid-20th century work by Coons, Sutherland and others, the exhibition also highlighted contemporary work by artists, architects, and designers. The symposium brought together many of these individuals in a rich series of talks and discussions covering their own architecture and design work and socio-historical examinations of technology

15 In full disclosure, I was not only an attendee to the symposium and exhibition, but a research assistant who worked on two interactive software installations at the gallery, as well as a co-host of a design workshop held the weekend of the symposium. Both events were also curated and organized by my thesis advisor and M.S. Computational Design track chair, Daniel Cardoso Llach. I can't claim to be an objective observer, but one who was deeply involved with the events of both the exhibition and symposium, and who benefited from the confluence of work, people, and ideas at them. As such, part of my goal in this thesis is to synthesize the conversations I observed and was a part of at these events, and to share the knowledge with a broader audience.

and digital culture. In doing so, as well as by highlighting the material foundations of current and historical design research, the symposium charted the fluid space of computational design as a field of practice.

The second event is the *Cybernetics Conference*, held in November 2017 in New York City. Hosted by Prime Produce, a non-profit ‘guild for social good,’ the conference aimed to bring “scholars, technicians, activists, and artists in dialogue to consider the ways informatic systems shape social organization.”¹⁶ As if responding to the call made by Tara McPherson in her 2012 essay “U.S. Operating Systems at Mid-Century” for “hybrid practices: artist-theorists; programming humanists; activist scholars; theoretical archivists; [and] critical race coders,”¹⁷ the speakers and participants at the *Cybernetics Conference* are extremely difficult to categorize by discipline. The conference’s three organizers¹⁸ alone do work as diverse as critical digital art, game design, social simulation, curatorial research, geonomics, and (of course) event organizing. The conference serves as case in point that strong work and research is not merely aesthetically interesting or analytically sound,

but challenges the boundaries of disciplines and received narratives of technology, information, and society.

We cannot afford to be neutral on the issue of the social context of our work. Even the most mundane works embody some vision of the future, and designers and technologists should do everything in their power to ensure that it represents a desirable future, or else it can (and will) be co-opted for other purposes. Contrary to tenets of technological determinism, autonomy, and computationalism, powerful computation has a productive role to play in the creation of designing in complexity, for multiple desirable futures. In fact, in this thesis, we will see that computational designers possess unique abilities in thinking and working this way.

Methods

In order to project future ways of working, it is necessary to understand how and why designers and technologists work the way they do today. However, I am not attempting a broad survey of styles and techniques. Rather, I am interested in close research with people who exhibit *positive deviance*. As described in a 2009 sociological article on the subject, positive deviance is “the observation that in most settings a few... individuals follow uncommon, beneficial practices and consequently experience better

¹⁶ The Cybernetics Conference, <http://cybernetics.social/>. Accessed December 1, 2017.

¹⁷ McPherson, Tara. “US operating systems at mid-century.” In *Race after the Internet*, 2013.

¹⁸ Sam Hart, Melanie Hoff, and Francis Tseng, who are also associated with the School for Poetic Computation and Dark Inquiry, both mentioned above.

outcomes than their neighbours.”¹⁹ Toward the ends of this thesis, I am interested in artists, architects, designers, and technologists whose work may not lead to ‘better outcomes’ according to traditional, capitalist definitions (they won’t be showing up on the cover of magazines, for example), but who operate outside of the norm and project unique, provocative futures. I sought out such individuals in order to understand their work, and more importantly, *how* and *why* they work and think the way they do.

Over this summer and fall, I have conducted nine one-hour interviews with subjects working in architecture, design, technology, and art (although most lack a job title identifying them explicitly with one of those fields). Again, were I to be studying these fields writ large, nine people would be an insufficient sample size to examine. Instead, my investigation into these interviews is presented as a close study of a small group that is not representative, but which might lead to general insights. The interview subjects include architectural designers, software developers, game designers, graphic designers, digital artists, technologists, and students (with significant overlap and blurring of professional lines). All of the subjects are young (in their 20s or early- to mid-30s), and at the time of this

research based on the East Coast or Midwest of the U.S. Six identify as male, two as female, and one as nonbinary. Five are personal acquaintances of mine; I was connected to the remaining interviewees through other personal acquaintances, through a technology/design forum, and through the October 2017 computational design symposium. Wherever I quote from or reference interviews in the chapters that follow, I use pseudonyms for each interview subject, and have occasionally made slight adjustments to how they describe their work in order to preserve anonymity.

In addition to the interviews, and alongside the formulation of a framework for thinking about computational design, I worked on software prototypes and conducted design workshops, which are used as case studies supporting the framework. I call the two software programs ‘prototypes’ to emphasize the fact that they are not intended as commercial apps or products, but as embodiments of the particular component of the framework I associate them with, as well as research instruments. They are full-fledged, interactive (web-based) interfaces for exploring computational design principles, and I was able to build them not only as a result of my professional background as a web developer, but with the opportunity, through this program, to take courses in strictly-typed imperative programming, computer graphics, and human-computer

19 Marsh, David R., Dirk G. Schroeder, Kirk A. Dearden, Jerry Sternin, and Monique Sternin. “The power of positive deviance.” In *BMJ: British Medical Journal* 329, no. 7475 (2004): 1177.

interaction. The third case study, a design workshop, relies on an open-source software program as a research device, and also benefited from my collaboration with my classmate, Atefeh Mahdavi Goloujeh, as well as an additional cross-disciplinary course I took on social innovation and group facilitation in design.

Synthesizing this research, the framework for thinking about computational design rests on three components: Generating, simulating, and interrogating. Unifying theories are always just out of grasp, but I found it helpful to begin to think about my own work in this way, and to start to view other work through this lens. However, the results of the interviews and the case studies together illustrate that it's impossible to conceptually or practically separate one approach from another: Generative work can be greatly enhanced through simulation techniques; neither simulating nor generating will cohere without an interroga-

tive mindset; and designers working technically and materially are better positioned to critically interrogate using computational simulation and generating methods.

Through this framework, supported by examples from the interviews and case studies of my own work, I present a proof-of-concept for a new way of understanding and practicing computational design. I hope that the framework, as a general and flexible lens for work and research, will lead to new pedagogies of computational design and, ultimately, bring about novel, desirable futures.

Computational Design Thinking Framework

The way we think about and conceptualize computational design tangibly and pervasively influences the work that designers working computationally produce. In turn, the results of such work shapes the way we think about the discipline it belongs to. This dialectic is succinctly highlighted by linguist and philosopher George Lakoff, who writes, in *Metaphors We Live By*, “New metaphors are capable of creating new understandings and, therefore, new realities.”¹ What syntheses and metaphors exist for imagining computational design today? What could there be? Could new understandings of computational design spread outward to influence other disciplines?

A recurring theme in my interviewees’ accounts of design was the role of rules — that creativity is impossible without setting constraints — and the work of devising a conceptual framework is no different. In order to realize this framework, I had to articulate certain boundaries. Such an understanding of computational design couldn’t be tied to specific technologies or software paradigms, or it would only address a small subset of the field (and, in all likelihood, would soon become dated). It also couldn’t simply result from an analysis of historical precedents, whether a challenge to dominant narratives or a counterhistory — my work is also projective, and the framework is intimately

tied to my own work as a designer. Finally, the framework could not propose a design methodology, that is, a procedural, step-by-step plan for addressing design scenarios. As with a technical framework, an abstracted procedural methodology would still limit, rather than expand, the possibilities for computational design.

The framework I present is comprised of three components of computational design: Generating, simulating, and interrogating. Although I list them in this order, it does not imply a linear causality or hierarchy among them. Each is indispensable; each relies on and supports the other two. Like a poorly engineered building, I imagine computational design as an unstable practice without them. They are lenses through which to do and think about computational design. While a computer algorithm to create, for example, arbitrary house plans is certainly a generative program, it is also useful to adopt a mindset of simulating and interrogating when working with or on such software, and to borrow liberally from techniques belonging to the other pillars.

¹ Lakoff, George, and Mark Johnson. *Metaphors we live by*. Chicago: Chicago University Press, 1980.

The following chapters each focus on one of the three pillars, including an overview, a section on specific methods, techniques, and strategies from the interviews, and a case study of my own work. Again, the three sections may be read in any order — you might find it helpful to jump from one case study into the methods from another chapter and back, or read the overviews of all three before diving into the case studies.

Generating

Overview

The act of designing involves the creation of something new, whether an object, an image, or an idea. I call the first pillar of the framework ‘generating’ as opposed to ‘creating’ to highlight how computation allows designers to work at a level of abstraction from singular artifacts. An artist working alone might create a painting or a drawing. If, however, they were to generate it, that would imply an underlying structure and logic capable of producing more artworks, each qualitatively different from the others but arrived at through the same mechanism. Generating also implies a complexity in the production of a generative process that is gradually insurmountable by human effort alone. In his 1968 essay “Systems Generating Systems,”¹ American architect and theorist Christopher Alexander differentiates ‘systems as a whole’ from ‘generating systems.’ The former are complex assemblages, not singular objects, that are nevertheless characterized by some “holistic phenomenon.” The latter are also not individual things, but groups of objects and forces “with rules about the way these parts may be combined.” Set into motion, they produce complex outputs that are more than the sum of the parts: Generating systems generate systems as a whole. Alexander closes his essay with a call to action for

designers and architects: “To make objects with complex holistic properties, it is necessary to invent generating systems which will generate objects with the required holistic properties.”

If we grant that the work of designers has become more complex — think not only of large software programs or buildings with demanding energy requirements but also the ways in which designed artifacts are situated in socio-technical contexts — then it is important for designers to work generatively. Through computation, this is intuitive for technologists; something that is hard-coded is inherently limited to a certain scope, whereas abstractions allow for more flexibility and multi-purpose reuse. Computational design brings certain key methods to aid the designer in generating systems. In particular, through *rulemaking* processes, designers can more easily work at the level of the system. By *exploring* (or, *searching*) they can make sense of the potentially overwhelming space of possibilities that is generated. Finally, in *automating* aspects of their work, the goal for designers is not so much efficiency or optimization as the ability to shift between various levels of abstraction.

¹ Alexander, Christopher. “Systems generating systems.” 1968.

Rulemaking

Alexander's generating systems rely on a notion of "rules about the way... parts may be combined." A logical starting point for designers working generatively is to delineate those rules, but that is hardly ever done in a void. A process of rulemaking often begins analytically, almost scientifically, observing the functioning of an existing system in order to derive elementary objects and rules. In 2- and 3-dimensional formal design, this has been explored in the work of computational design researchers George Stiny and James Gips and their notion of shape grammars.² Shape grammars are abstractions of geometries, described by a set of rules that define transformations. A simple ruleset might only encode a few recognized shapes and basic transformations, such as scaling or rotating. A more sophisticated ruleset, such as Stiny and William J. Mitchell's 1978 "Palladian grammar,"³ might be capable of generating complex, recognizable forms, such as floor plans in the style of 16th century Italian architect Andrea Palladio. Derived from analysis of a larger body of work, Stiny and Mitchell's Palladian grammar contains the potential for new works that would be recognized as mem-

bers of the same stylistic family as the original villas. More recently, in his 2006 book *Shape: Talking about Seeing and Doing*, Stiny has argued for designing shape grammars as a new pedagogy: "Creative design can be taught like language and mathematics in school, with examples, rules, and practice and the opportunity to experiment freely."⁴

Many of the practitioners I interviewed use rulemaking techniques in their work. Paul, an architect and technologist, puts the tools of his day job to work toward more visually aesthetic ends as experimental artworks. Like Stiny, he argues that a rule-based logic can be used as an expressive medium in design and art. Describing a specific piece, Paul says, "The logic [here is] starting with a cube and then randomly subtracting cubes from that cube. Starting at the lower-left corner and moving all the way up, each one uses a different set of random cubes... I was interested in testing it out, seeing what graphic outcomes it gave me."⁵ Through the iteration of a subtractive rule, and by varying parameters (the size and position of the subtraction), Paul's system is theoretically able to generate an infinite number of possible outcomes. Another interview subject, Anna, a digital artist and game designer, describes a project, a "generative experiment" that spun

2 Stiny, George, and James Gips. "Shape Grammars and the Generative Specification of Painting and Sculpture." In *IFIP Congress* (2), vol. 2, no. 3. 1971.

3 Stiny, George, and William J. Mitchell. "The Palladian Grammar." In *Environment and planning B: Planning and design* 5, no. 1 (1978): 5-18.

4 Stiny, George. *Shape: Talking about Seeing and Doing*, 2006.

5 Skype interview with Paul, August 3, 2017.

off of a larger game they were working on, that took the form of a Twitterbot that periodically posts generative drawings of winged animals. Anna notes, “The interesting thing about that project is the way it works, which is very lo-fi in some ways. It draws each [animal] much like a person would draw something, in that it places individual pixels by rulesets to generate these patterns and these shifts.”⁶ Anna’s analytical ruleset is derived from the visual language of animal life, coupled with a procedure modeled after human drawing. The space of possibilities formed by Anna’s ruleset, while also infinite, is intuitively larger than the space of Paul’s cube-based artwork — the generated animals vary in size, color, shape, texture, and pattern as opposed to simply geometric form. In both cases, however, the number of possible outcomes exceeds the ability of a single designer to ever observe. These examples each use pseudo-random selection to choose from generated possibilities. In aggregate, with enough sampling, this will eventually provide a qualitative sense of the possibility space: A rough feel for its size and the variations it permits. However, there are other, more directed ways of traversing a generative possibility space.

6 Skype interview with Anna, July 18, 2017.

Exploring (or, Searching)

Generating through rulemaking is a powerful tool for designers working computationally. But the sheer size of the output poses a new problem: How does one navigate an immense field of possibilities? For digital architecture historian Mario Carpo, the answer lies in increasingly powerful computational search algorithms, such as those used by Google to find information from among the billions of indexed web pages on the internet.⁷ In this technologically determinist view, if one knows that a certain optimized solution exists, it is possible to arrive at it through algorithmic processes. While computational techniques like genetic algorithms and neural nets trained on relevant datasets are adept tools for parsing the results of a generative procedure and determining optimal results, a technology-first approach downplays the role of the designer in the process. Rejecting the dominant narratives of data and computation as panacea to all problems (design or otherwise), artists, architects, and designers are devising approaches to exploring generative possibility spaces that harness technology while transforming the role of the human.

Natalie, a software developer and designer who creates

7 Carpo, Mario. The second digital turn: Design beyond intelligence. MIT Press, 2016.

interactive, web-based visualizations, subverts established generative practices to restore agency to the artist. Describing a piece that resembles a modular, gridded, sprawling circuit board, she reveals that, “Although at first glance it does look pretty symmetrical, pretty generative, if you were to really dig in there... There are minor inconsistencies and stuff like that, which I think lends it that organic-ness and its realness. There’s a way to do it with code, but do I want to? No.”⁸ Natalie uses an aesthetic associated with generative art (which I mistakenly assumed the piece was in our interview) but maintains complete, manual control over the creative process. Her work critiques a view of the role of the designer as merely selecting from a deterministic set of choices, trading away creative agency in exchange for breadth of possibilities.

However, for others, giving up control over individual objects is desirable for increasing corresponding agency at a level of abstraction. In this view, exploring a possibility space is neither randomly selecting from it nor exhaustively searching it via algorithm. Instead, it can be an aesthetic experience that provides a starting point for further iteration, whether manual or generative. Max, an artificial intelligence researcher, worked on a project to generate plunderphonics music tracks — a style which samples and combines various audio sources into a musical collage —

from a ‘seed’ song. Using, for example, an album by Prince as an input, the software might pick up on the use of falsetto, drum machine, and synthesizer (that is, the rules of a generating system), and assemble tracks with overlaid, found audio clips approximating that sweet Minneapolis sound. Importantly, while the algorithms perform a great deal of ‘labor’ in data analysis, searching for audio clips and assembling them according to rules, what to do with them from there is expressly left to the designer-listener. The produced track comes with an index of its source clips, making it not so much a listenable artifact as a new starting point. Max elaborates, “It’s like a little toolkit where you have the samples, you have the track listing, and you have one possible track that could be created with the samples, and you can create something totally different from that if you want.”⁹ In his view, exploring a generated space of possibilities isn’t about arriving at one optimal solution so much as discovering something with potential to be further investigated and expanded on by the human designer.

Another aspect of this example stands out: As opposed to the work of Anna, Paul, and Natalie, who all dictate the initial rules themselves to a generative algorithm, Max’s software takes over the analytical step. Automating information processes is central to the fields of artificial

8 Google Hangout interview with Natalie, August 1, 2017.

9 Phone interview with Max, August 2, 2017.

intelligence and machine learning, and while there are certainly risks involved, it also serves as a strategy that allows designers to generate systems in new ways.

Automating

There is an anxiety often expressed around automation, that computers will replace humans in daily life. Robotic production in factories, for example, has led to a transformation of the American industrial labor economy in the last decades of the 20th century. Figures from architecture and engineering worlds have at times flirted with design automation, a narrative traced in Daniel Cardoso Llach's *Builders of the Vision*.¹⁰ He presents the figures of computer engineer Douglas Ross and mathematician/designer Steven Coons as embodying opposing views of automation and augmentation. Ross, along with his research group at MIT, Electronic Systems Laboratory, understood design "as a noun: a geometric specification that could be calculated... if the design problem was adequately represented in a formal — as opposed to natural — language." Coons, meanwhile, recognized design as "a verb: an open-ended and essentially human activity," wherein the computer of the 1960s could play an increasingly supportive role but never fully replace the human. Half a century later, this debate has not been resolved — but I argue that

only one of the positions suggests a multiplicity of futures and a meaningful role for the human, and that it is the latter path we should follow.

This section began by defining design as a fundamentally creative act, with the hypothesis that computation can enable designers to better approach complexity in their work. But what does creativity mean when a computer can exhibit it? For a designer, automating away every step of their work would be a nightmare — a critique Natalie makes in her manually 'generated' artwork. Even in the era of Ross and Coons, computers could generate multiple forms from a set of encoded rules. Today, it seems that machine learning, in taking on the previously exclusively human role of analyzing data and synthesizing rules, removes the designer from that side of the equation. On the opposite end, with automated algorithms for searching and selecting from a generated possibility space, it might seem that there's nothing left for the designer to do whatsoever. But interrogate this narrative further, and cracks in the façade appear — new entry points and levels at which the designer can work.

A recent technological development will serve as an example to explore the changing role of the designer. In a 2014 paper, Ian Goodfellow and researchers at Université de Montréal introduced generative adversarial nets

¹⁰ Cardoso Llach, Daniel. *Builders of the Vision*, Routledge, 2015.

(GANs),¹¹ a novel framework for neural network-based machine learning. As the name implies, one major component is a generative model: A technical structure that, after processing and analyzing a large collection of data (a process called ‘training’), can subsequently generate new objects that cohere stylistically with the given data. Much recent work has centered on image generation — for example, training a generative model on millions of Google Street View photos in order to create new, plausible, but completely imaginary Street View images. The other key component is a discriminative model, which appraises the output of the generative model and classifies the objects as ‘real’ or ‘generated.’ Both models are incentivized to optimize their performance: The generative model to generate objects that are classified as ‘real,’ and the discriminative model to maintain accuracy in its classifications. It is a conceptually sound framework, and would appear to automate a design-adjacent process of rulemaking, generating, exploring, and curating. Where does the human fit in? The example of Max’s music generation software is instructive: The designer no longer works directly with objects (images, audio clips), but instead acts as a coordinator of the computational technologies. To draw another musical parallel, the human is less a virtuosic player of an

instrument than a symphony conductor, signaling and directing flows. By specifying and providing the data that the generative model is trained on, and tweaking the nuances of the discriminative model, the designer becomes almost a facilitator of a conversation between the technological agencies, a role rich with possibilities for working within and toward generative systems.

Case Study: Worldmaking

In this case study, I describe *Worldmaking*, a software prototype for 3d modeling I designed and developed, and a series of design activities (framed as ‘games’) conducted with pairs of designers using the software. The 3d modeling environment, while poorly suited to architectural design or building modeling, serves as a platform to explore a given possibility space generated by the interface, affordances, and constraints of the program. In conducting the activities specifically as games to be played by a pair of designers, extra rules were introduced that further refined and shifted the space of possibilities and how the designers explored that space — not only individually, but in a dialectic process of conversation with each other and the technology.

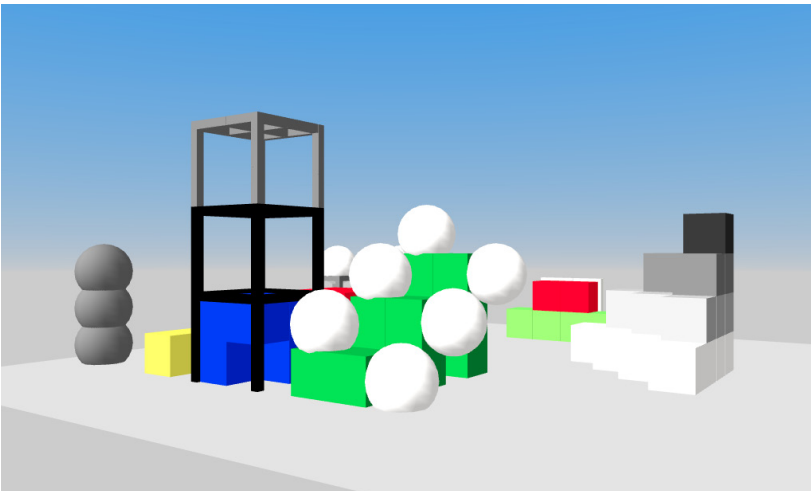
11 Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets.” In *Advances in neural information processing systems*, pp. 2672-2680. 2014.

Background

I began working on this software as a final project for a course in the second semester of my Master of Architecture program on architectural drawing and representation, and continued developing it after leaving that program and joining my current program. I further developed (and renamed) it as a Teaching Assistant for a computational design seminar course, *Inquiry into Computation, Architecture and Design*, taught by my advisor, Daniel Cardoso Llach. The software, *Worldmaking*, is a 3d modeling environment with a limited vocabulary of geometric shapes — voxels (cubes), spheres, and beams (lines) to be placed on and around a 2-dimensional plane at the center of the

environment. A designer-user can add these shapes, in a color of their choosing, and can also delete them. No other design actions (such as moving, rotating, scaling) are implemented. While at first blush this might appear overly limiting (and it certainly would be as an architectural design or building modeling program), one precept that became clear to me through a generative approach was the productive application of constraints and rules in the design process. The three shapes in the environment inscribe an infinite possibility space, a world for the designer to create and explore.

The name of the software, shared with the name of a two-week module in the Inquiry course, comes from the opening chapter of philosopher Nelson Goodman's 1978 book *Ways of Worldmaking*.¹² While Goodman's focus is mainly linguistic — how different methods and styles of speech and text can form different abstract 'worlds' — his ideas are readily applicable to formal design. In particular, he describes how different worlds are shaped by unique logics and truths that constitute that world's 'reality,' a parallel notion to how generative design rules shape a possibility space. Design researcher Donald Schön has also invoked a notion of 'design worlds' as formal environments constituted by (and in dialogue with) types and rules.¹³ The



An example 'world' in the 3d modeling environment

¹² Goodman, Nelson. *Ways of worldmaking*. Hackett Publishing, 1978.

¹³ Schön, Donald A. "Designing: Rules, types and words." In *Design studies* 9,

software *Worldmaking* permits a narrow range of types (objects) and formal rules, but includes other features to grant further agency to designer-users and shape the possibility space within the constraints of the technology.

A turning point in the development of this project came when I implemented a real-time interface allowing multiple designers to work in the same 3d environment simultaneously. While not as direct or embodied as sketching on the same paper,¹⁴ this new affordance does make it possible for two or more designers to engage in a conversation implicitly, through the negotiation of forms in space, and also explicitly, through a built-in chat window.¹⁵ It would have been possible to investigate a generative approach to design through this software without having real-time interaction between remote designers, but the possibility of communication and miscommunication, of cooperation and conflict, makes this a much richer study. A frame for communication between designers comes from linguist Michael Reddy's critical notion of the 'conduit metaphor'¹⁶ — the implicit belief that it is possible to directly and seamlessly translate one's thoughts into verbal or written

communication that will then be interpreted unambiguously by others.¹⁷ In both the chat interface and through implicit communication through the interplay of forms in the modeling environment, it is impossible to definitively communicate one's design intent to a partner, who reconstructs the meaning behind communications based on their own worldview, experience, and intent. However, the potential misunderstandings between designers represent not a failure to communicate, but a space rife with generative possibilities. The two games conducted in this study each represent one of the possible communication paradigms, implicit and explicit. The games themselves are adaptations and further explorations of work by design researchers N. John Habraken and Mark Gross: the "Silent Game" and the "Reference Game."¹⁸ In their original work, as in the *Inquiry* class, the games are played with physical pieces (such as LEGO bricks); in the *Worldmaking* software the rules are identical but take place in digital space. The games are structures introduced not as design tools, to produce aesthetic or functional forms, but as aides for research into the design process, for "demonstrating and testing design concepts."

no. 3 (1988): 181-190.

14 But certainly more egalitarian than a 'collaborative' model where one designer looks over the shoulder of another, working at the computer.

15 A setting allows the designer-user to specify their name, a nickname, or to remain anonymous, identified only by a randomly generated ID.

16 Reddy, Michael. "The conduit metaphor." In *Metaphor and thought* 2 (1979).

17 There are strong parallels between the conduit metaphor and the notion of technological neutrality described in the introduction to this thesis — both understate how difficult the process of embodying one's thoughts is (which anyone taking a drawing class for the first time will agree is a constant struggle).

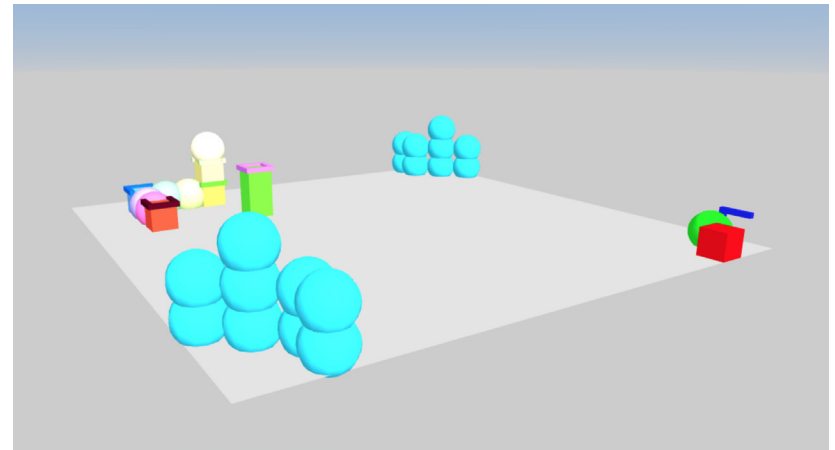
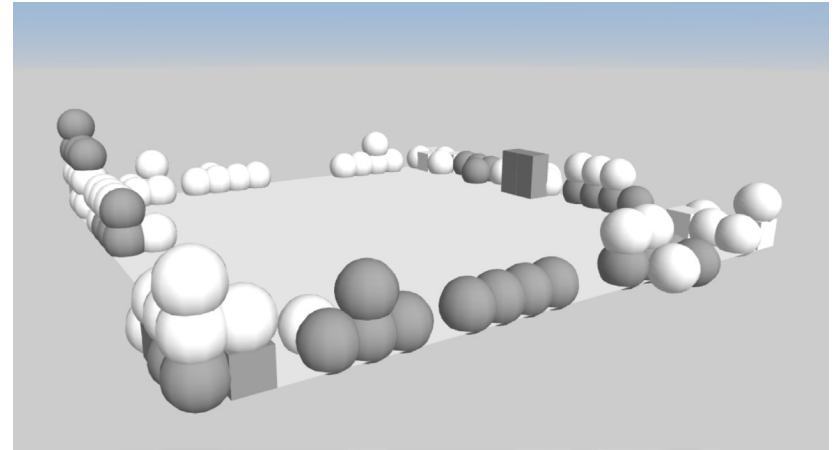
18 Habraken, N. John, and Mark D. Gross. "Concept design games." In *Design Studies* 9, no. 3 (1988): 150-158.

Participants

I conducted the pair of games with two different pairs of designers. They are current architecture and design students at Carnegie Mellon University, and one is a recently graduated design student. All the participants have prior experience with design and/or 3d modeling software, and prior to the games, were given a 'sandbox' environment to familiarize themselves with the *Worldmaking* interface. They were randomly paired with each other based on availability and assigned specific roles (A and B) for the games.

The Silent Game

In the Silent Game, written communication between participants through the chat interface is discouraged (except for saying, "I'm done," or asking clarifying questions). Player A establishes a design intent by placing up to 5 shapes in one corner. Player B places up to 5 shapes in the opposite corner, in order to demonstrate that they understand Player A's intent. They then repeat this process with free range over the environment, further elaborating and exploring the design intent. There are no goals such as filling the available space or making a specific pattern; the game is open-ended and simply ends after 30 minutes, leaving time for discussion with the participants.



Screenshots taken at the end of both Silent Games

In the first pair, Player A interpreted '5 shapes' to mean collections of voxels and spheres, as opposed to the individual objects. They embodied their design intent with adjacent groups of objects resembling 3-dimensional

Tetris pieces. Player B followed this intent without copying it exactly, modifying some pieces while respecting the ‘language’ established by Player A. After the initial stage, both players kept to the edges of the building plane in the environment. While they switched shapes often, they each kept their own color throughout (gray for Player A, white for Player B).

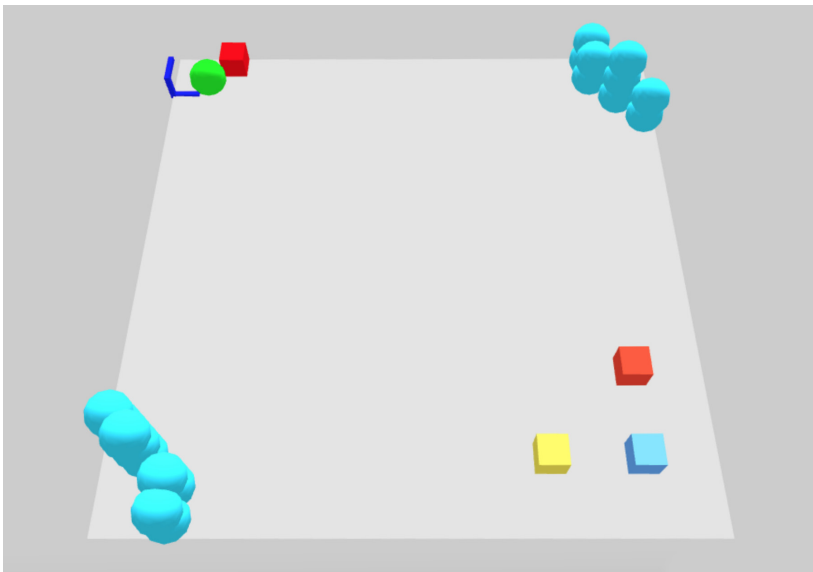
In the second pair, Player A drew five turquoise spheres in one corner, and Player B replicated the arrangement, rotated 180 degrees, in the opposite corner. Player A built

a second level onto this structure, and Player B copied it again. At this point I intervened, worried that Player B would continue to simply imitate Player A’s forms, and I asked Player B to establish a new design intent in one of the remaining corners for Player A to follow. B drew a green sphere, red voxel, and blue triad of beams, and Player A followed this with three distinct voxels in the opposite corner, forming the corners of a right triangle. Finally, in the free drawing stage, both players focused their efforts around this latest development, Player A’s three voxels, connecting them with spheres, and building a two-height voxel form with beams as a ‘cornice,’ as well as on the original three voxels.

The Reference Game

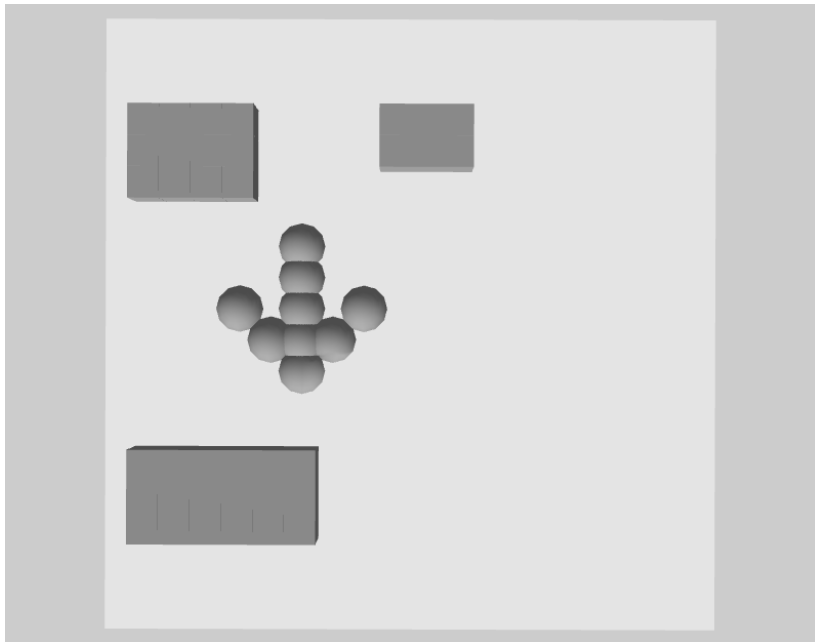
In the Reference Game, with the same players as A and B, Player A becomes the ‘doer’ and Player B becomes the ‘talker.’ This means that Player B can no longer add or delete shapes in the environment, and instead gives written instructions to Player A via the chat interface, who attempts to follow Player B’s instructions and interpret their design intent. Player B may then clarify or provide further instructions to Player A, who continues drawing in the environment.

In the first pair, Player B asked Player A, *Can you draw-*



The 1st design intent in the lower-left and upper-right corners, the 2nd design intent in the upper-left followed by the lower-right corner

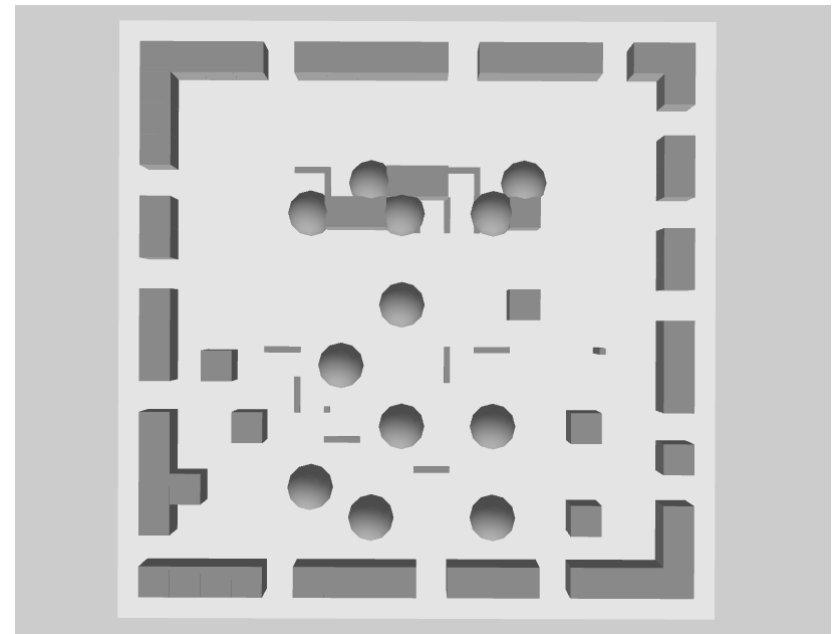
ing something “democratic”? [sic].¹⁹ In response, Player A drew two differently sized rectangles using gray voxels, an arrow out of spheres, and a larger rectangle, symbolically imply a causal relationship between the two smaller rectangles and the larger one.



Player A's response to Player B's first instruction

¹⁹ All quotes taken from the chat interface are in italics, with typos/grammatical choices left intact.

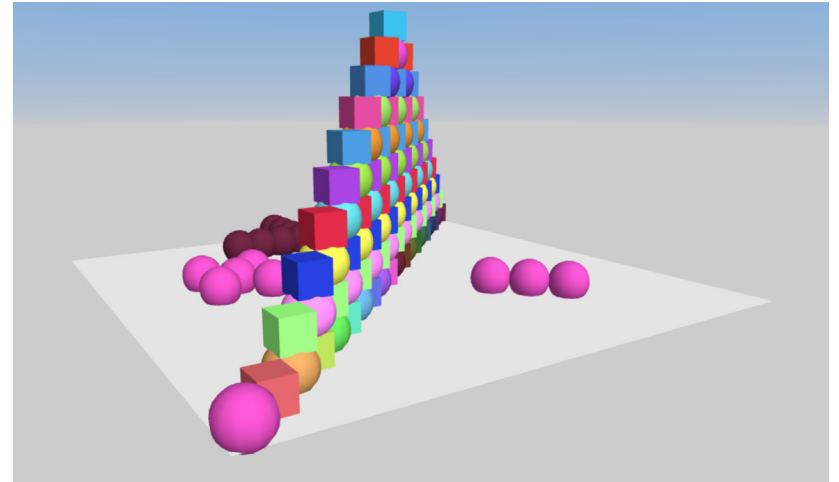
On observing this, Player B was not satisfied, and said, *The 3d model is quite different from what I have imagined.* They clarified that it should be more scattered. Player A deleted the large rectangle and arrow and added individual voxels, dispersed across the surface. At this, Player B further requested, *Can you mix different shapes? ...it looks too generalized individuals.* Player A then deleted some of the voxels and replaced them with spheres and scattered beams. At this, Player B was excited, and encouraged more conceptual exploration: *it would be also great if the model shows how individuals reach a consensus!*



Final state of first reference game

Player A responded by reincorporating some larger groups of shapes, but this time using combinations of voxels, spheres, and beams. For unknown reasons, Player A also built a voxel border around the edges, and Player B hinted, *if i draw it, I would make the border a bit loose*, to which Player A deleted some of the shapes to dematerialize the boundary.

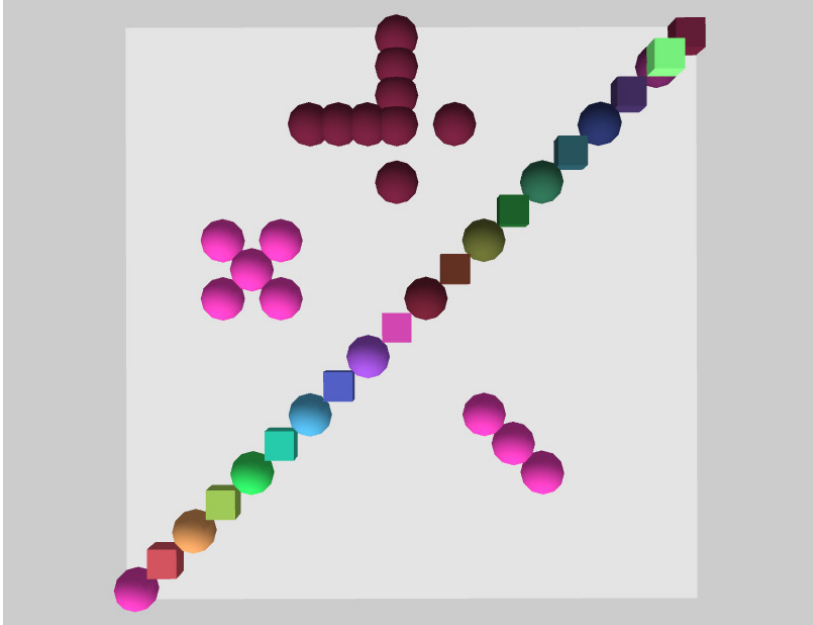
In the second pair, Player B gave a starting instruction that appeared to offer little room for creative interpretation:
Form a line of alternating voxel in rainbow color gradient



Final state of second reference game

starting from the one corner. After Player A built a diagonal line of alternating voxels and spheres from a corner to the center, B added: *Please continue on with the alternating voxel until the opposing edge is reached... In a muted reverse rainbow color... muted = Darker.* After this, B gave more ambiguous instructions: *Please draw something “dark” on the darker spectrum of the rainbow... “dark” = “sad”... divide up the space into positive and negative spaces.* In response, Player A drew a symbolic ‘sad’ face (in dark purple) and plus and minus signs, all out of spheres.

Ambitiously, Player B continued: *put another set of alternating block on the top of the current block with a differ-*



Second reference game after a few instructions

ent type of voxel. if the bottom is sphere, a cube should be placed... repeat this until two or one block remains on the current row. Player A gamely agreed, building a large, multicolored, triangular wall that spanned the environment's modeling surface diagonally, and writing afterward: *whew done.*

Discussion

Habraken and Gross, reflecting on their research with the Silent Game and Reference Game (using physical game pieces such as nails and washers), write that “Players must cooperate and try to understand each other’s intentions.” This is the case whether both players are ‘silent’ and communicate only through form, or when one is allowed to speak/write. While the ‘talkers’ in the Reference Game appear to have more agency than the ‘doers,’ who follow their instructions, in both games the instruction-givers engaged in a negotiation with their partner, clarifying and furthering certain points. However, it is necessary to clearly define the relationship between the players. In the first pair, after finishing both activities, Player B described mixed feelings on the roles of the Reference Game: *I feel a bit bad about that I kind of order him/her. I wanted to accept their imagination. I think that is what collaboration is.* In future iterations of the Reference Game, it might be helpful to reframe the role of the instruction-giver to

encourage exploration of emergent forms that Player A designs in the modeling environment.

There are trade-offs in following rules literally or interpretively, and in specifying narrowly-defined or loosely-defined rules. In the second pair’s Silent Game, Player B copied Player A’s forms exactly; in some contexts understanding rules is synonymous with following those rules precisely. When the roles were reversed, Player A took significant creative liberties in interpreting Player B’s intent. At first it was unclear why Player A was adding these particular shapes, but their later explanation demonstrated an abstracted interpretation of B’s intent. As a result, the game entered a new, more dynamic phase, with both players creating novel patterns in response to each other’s forms. In a context of cooperative designing, divergent interpretations can be used generatively. In the first pair’s Reference Game, Player B’s prompt to draw something “democratic” — an abstract notion with no obvious embodiment — served as a simple rule that (through Player A’s work and the resulting interplay) generated unexpected possibilities. This case study uses a digital platform to explore this mode of communication, an openness to possibility and willingness to explore. However, a notion of ‘generative dialogue/conversation’ has a significant history in the (non-computational) worlds of group facilitation, decision-

making and participatory design.²⁰ How does this mode apply to computational design specifically?

In the context of automated computational technologies, one could imagine versions of the Silent Game and the Reference Game where one ‘player’ is a set of algorithms that can ‘learn’ through playing the games. Importantly, as the games have no winners or losers, a computer player’s goal would be two-fold: To ‘understand’ the human player’s design intent (encoding it in some digital representation) and to act dialectically alongside the human partner, sparking new ideas from unexpected interpretations and actions. How these new ideas would be qualitatively different from those that arise through human-to-human communication is an open question. In the next chapter, *Simulating*, I will make the argument that computer simulations, specifically, are useful in producing unexpected or emergent properties to be considered and acted on by a human designer.

Conclusion

Working generatively can be a completely manual process of determining and specifying rules, and then exploring the possibility space they produce, or a heavily automated process where machine learning technologies take over these tasks, in turn creating a new role for the computational designer. However, more likely is that a generative process lies somewhere between these two extremes; designers pick up or discard digital tools as needed, always keeping in mind the agency that software and algorithms bring to the process. Generating is not a simple, cause-and-effect method with a ruleset deterministically leading to a collection of outputs, from which one optimized solution is selected. Rather, it is a complex feedback loop and negotiation between the designer, their tools and systems, and the data they create or provide. As shown through the *Worldmaking* case study, the process becomes even richer when multiple human designers work together in a dialectic conversation, generating new possibilities through productive (mis)communication. Working together, and with critical engagement with their tools at various levels of abstraction, designers may subtly and tangibly shape the holistic properties of systems.

²⁰ See the work of David Bohm, William Isaacs, and Otto Scharmer’s *Theory U*.

Simulating

Overview

Working with computation, designers can more easily generate complex systems (as opposed to creating individual objects), explore spaces of possibility, and engage with their work at new levels. But while generating often implies the formulation of an entirely new system, the creative work of designers always exists within an exterior environment (technical, social, economic, material, etc.). Even the most speculative, implausible architecture or design projects presuppose a cultural context, perhaps also imagined, in which they take shape. A work, perhaps presented as purely formal or aesthetic, that ignores its context, lies through omission. Generative techniques give form to complex systems; simulating allows designers to model existing systems, to experiment through interventions in those systems, and to imagine from them new structures, goals, and paradigms.

Reacting against the computational paradigm described in the literature review, the term ‘simulation’ should be clarified. A computer simulation is a representation of some aspects of reality — a computational, mathematical model — usually with a temporal dimension and rules for how the model changes over time. Processes of simulation are generating systems, giving rise to holistic properties that, to varying degrees, align with those in the referent of

the simulation. However, in the context of computational design, simulations are perhaps less valuable as purely analytical, predictive tools, than as spaces in which to explore and understand systems. Especially when visualized or embodied in order to tangibly observe the holistic properties of systems, simulations (like automated generative techniques) become powerful allies that allow the designer to operate at a level of abstraction. Coupled with an interface that allows for intervention into the system, a designer can experiment and speculate practically *ad infinitum*, testing their theories in a miniature world before implementing them in the real world.

All this does rest on the notion that a computer simulation can, to some degree, precisely and accurately represent aspects of reality. Few would argue that physical equations, say, for velocity over time (resulting from acceleration due to gravity), don’t resemble and predict phenomena observed in the real world. But when it comes to simulations of large, socio-technical systems, which are far more complex and unpredictable, the output of simulations should be met with skepticism. As artist and game designer Paolo Pedercini points out in his keynote talk at the 2017 *International City Gaming Conference*, simulations can be wielded as instruments of propaganda.

Pedercini says:

“[Simulations] for city planning are often presented with a neutral technocratic language: ‘Let’s try to explain to common people the complexities of urban development’... But I swear, I can design you a [simulation] that subtly leads people to whatever “solution” you want... You can easily create formal mathematical relationships that reinforce your agenda. The more complex a simulation is, the more obfuscated is the data it is based on, the harder is to analyze it, fact check it, and criticize it.”¹

Through a lens of technological neutrality, simulations are promoted as descriptions of reality used to provide recommendations for future action. But as Pedercini notes, they are always constructed by individuals with a (perhaps unconscious) agenda, and contain the biases of their creators. As digital media researcher Yanni Loukissas further describes in *Co-Designers: Cultures of Computer Simulation in Architecture*, simulations are also prone to differing or conflicting interpretations. They act as “spaces of exchange... open[ing] up zones in which design participants can coordinate... without sharing the same concep-

tions about those designs.”² Additionally, the technical infrastructure and specifics of how a simulation is encoded shapes a space of possible interactions. Any simulation of a socio-technical system put forward as purely objective is an outright lie. However, whether a simulation is ‘right’ or ‘wrong’ is not the salient point; it’s what we can learn from it about ourselves and the world that is. The simulating processes described by my interview subjects and in the case study at the end of this chapter appear less like a Nostradamus predicting the future than an Octavia Butler or Ursula K. Le Guin imagining one of many possibilities. For designers, this happens through a modeling process that allows for unexpected patterns of behavior *emerging* in the simulation, and through a mindset of *futuring* — seeking out certain unintentional, desirable outcomes as projective futures to work toward.

Emerging

As described by Christopher Alexander in “Systems Generating Systems,” we can only recognize ‘systems as a whole’ by virtue of some observable, holistic property that emerges from the unpredictable interactions between constituent elements of and behaviors within the system. By simulating systems, one can draw causal relationships

1 Pedercini, Paolo. “SimCities and SimCrises,” 2017.

2 Loukissas, Yanni Alexander. *Co-designers: cultures of computer simulation in architecture*. Routledge, 2012.

between low-level behaviors and high-level patterns. In fact, simulations, as philosopher Manuel DeLanda argues, give conceptual legitimacy to the notion that observable complex phenomena can emerge from interactions between the elements of a system at all. In *Philosophy and Simulation: The Emergence of Synthetic Reason*, he writes, “Simulations can play the role of laboratory experiments in the study of emergence complementing the role of mathematics in deciphering the structure of possibility spaces.”³ In this view, simulating is a necessary partner to generative work, as it provides a method for designers to explore and draw conclusions about the relationship between the outputs of generating systems and their constitutive rules and elements.

In a canonical example from 1971,⁴ economist Thomas Schelling created a simplified model of cities as a 2x2 grid, with individual pieces representing citizens. In Schelling’s model, a citizen of a certain ‘type’ will move around on the grid until their neighbors are composed of a minimum percentage of their own ‘type.’ With two ‘types’ in play, black and white, the model demonstrated that a certain phenomenon would inevitably arise — the board (or city) tending toward starkly divided areas (or neighborhoods)

defined by ‘type’ (or race). The parentheticals denote the findings of Schelling’s model: Systems-level effects such as racial segregation might arise from the actions and interactions of individual actors regardless of their intent. This model (which, astonishingly compared to capabilities today, actually was implemented hundreds of times using physical pieces on graph paper) helped to set the stage for a key computational technique in simulating emergence: Agent-based modeling (ABM). Computer scientist and ABM researcher Uri Wilensky defines it as “a form of computational modeling whereby a phenomenon is modeled in terms of agents and their interactions.”⁵ This dry definition belies the real potential of agent-based modeling. One of my interview subjects, Max, an artificial intelligence researcher, describes the values of ABM in contrast to other approaches:

*“Most machine learning is oriented towards producing an answer of some kind... Agent-based modeling is more about understanding how all of the different components of a system interact. So it’s less about the final state that the simulation produces, but how it got there, and why it got there.”*⁶

3 DeLanda, Manuel. *Philosophy and simulation: The emergence of synthetic reason*. Bloomsbury Publishing, 2011.

4 Schelling, Thomas C. “Dynamic models of segregation.” *Journal of mathematical sociology* 1, no. 2 (1971): 143-186.

5 Wilensky, Uri, and William Rand. *An introduction to agent-based modeling: Modeling natural, social, and engineered complex systems with NetLogo*. MIT Press, 2015.

6 Phone interview with Max, August 2, 2017.

Agents are individual actors in a simulation, with behavioral rules that drive their actions within an environment. ABM has been used to model such diverse phenomena as urban form,⁷ the spread of infectious diseases,⁸ and labor markets.⁹ Agents do not necessarily represent individual humans, however. David, a PhD student in a technology-oriented architecture program, described a project he worked on to use ABM in an urban design context. The agents, in this case, are building footprints:

“I created a prototype to take maps of a city, recognize the buildings, and make the buildings behave as a physical entity in a physical simulation environment... The interesting part is that, even with such a simple behavior — they’re reflex agents — you start to find out really interesting clusters and patterns of how you can map the city. Even with no intelligence on the side of the agent.”¹⁰

7 Batty, Michael. *Cities and complexity: Understanding cities with cellular automata, agent-based models, and fractals*. The MIT press, 2007.

8 Perez, Liliana, and Suzana Dragicevic. “An agent-based approach for modeling dynamics of contagious disease spread.” *International journal of health geographics* 8, no. 1 (2009): 50.

9 Neugart, Michael, and Matteo Richiardi. “Agent-based models of the labor market.” LABORatorio R. Revelli working papers series 125, 2012.

10 Skype interview with David, July 24, 2017.

David underscores one of the most compelling aspects of agent-based modeling — individual agents typically have very limited awareness of the system beyond their immediate ‘neighborhood’ (defined by the designer). However, the rules for how they behave in relation to each other and to their environment almost always result in emergent patterns at the scale of large groups of agents. Emergence is unpredictable, and opens a new, productive line of inquiry into the mechanics of and relationships between individual agent behavior and systemic patterns.

Another interview subject, Ken, a technologist and organizer, describes an experimental project to simulate house parties. In this case, agents represent individual attendees to a party, with behaviors that include socializing, eating, drinking, and going to the bathroom. In calibrating the agents and the simulation, Ken and his collaborators ran into some unexpected emergent patterns: “The design process wasn’t form-based, it was system-based, so it’s kind of like trying to tweak this system that’s going off the rails. We had problems like, ‘People are going to the bathroom constantly in this simulation! We can’t stop them!’”¹¹

Although this is a humorous example, it’s also indicative of the capability of the emergent properties that arise to surprise the simulation designers. An awareness of emergence as a phenomenon can aid in using simulations as a

11 Skype interview with Ken, July 17, 2017.

research tool. Ken explains how, as a result of his experience with ABM, he works differently now: “Maybe one of the ways in which agent-based simulation has affected my thinking is that I see inside, mentally, in physical spaces, all the varied agents doing whatever they would want.” He encodes behaviors at the level of individual agents, but has gained the ability to imagine systemic behaviors that arise from the agents’ interactions in simulation space. Closely tied to systems thinking and socio-technical complexity, exploring emergence through ABM is a potent technique for designers, architects, and technologists.

Futuring

Simulating allows for a level of predictability and repeatability, a rigor that is rarely possible in built architectural or design work. For example, an urban planner considering alternate traffic signals could hardly expect to implement and empirically test dozens of different intersection designs on actual city streets. Instead, through a careful modeling process, the planner could compare various designs in a simulated environment to observe their effects. It is important to underline the fact that simulations are not substitutes for real, lived experience. Rather, through considered computational modeling, simulations based on mechanism-independent components of emergent properties might provide evidence to make arguments about pos-

sible interventions in systems. Simulations need not be (and can never be) comprehensive. Like the map in Jorge Luis Borges’ short story “On Exactitude in Science,” which becomes more and more detailed until it grows to be the size of the territory it charts, a simulation which corresponds 1-to-1 with reality is a fiction. Certain parameters must be selected as the key elements of the simulation, and the rest is noise. As DeLanda writes, “The process [of simulation modeling] may... change in an infinite number of irrelevant ways, the art of mathematical modeling being based in part on the ability to judge what changes do, and what changes do not, make a difference.” A simulation being plausible or completely impossible when compared against reality might hinge on a single parameter held to an inflexible degree of precision. One dangerous pitfall is that both realistic and unrealistic simulations might operate based off of the same internal logic (they might even differ by a single number), a logic that is easily mistaken for completeness and accuracy.

Technology and psychology researcher Sherry Turkle, echoing architect Louis Kahn’s question, “What does a brick want?” asks, “What does simulation want?” In her critical 2009 book *Simulation and Its Discontents*, she offers a simple answer: Immersion. A computer model based on and resembling reality offers its own simplified reality with an internally coherent logic. A facility with simulat-

ing might lead one to forget about outside dimensions that are left unmodeled, and the ways in which they could come into play. Turkle writes, “In simulation, architects feel an initial exhilaration because of the ease of multiple iterations. But at a certain point... possibilities can feel like inevitabilities.”¹² Simulations of scientific phenomena often model in order to predict the performance of, for example, a building’s energy efficiency, and immersion in such simulations to the neglect of outside factors would be a misstep. In a computational design context, however, the immersiveness of simulations can be helpful. Rather than predictive, decision-making tools, simulations can be interpreted as discursive fictions on how the world might be, as opposed to how it is. If a designer can maintain an interrogative, skeptical stance while, at the same time, suspending disbelief about the simplifications necessary in order to model certain phenomena, immersion in a simulation can lead to new ideas and understandings of reality.

For some of my interview subjects, immersion in simulation is desirable in order to see possible futures from potentially unrealistic (non-normative) emergent patterns. A simple reframing of simulations on the part of the

observer can lead to novel interpretations. In the case of Ken’s party simulation described above, with partygoers constantly going to the bathroom, one interpretation is as a glitch in the simulation code. Another is that Ken has inadvertently created a world where, perhaps, personal hydration has become a cultural imperative and recurrent bathroom-going is one logical side effect of this norm.

Max, the artificial intelligence researcher, told me about a project he worked on to model labor economics. The simulation uses census and American Community Survey data to model a large U.S. American city around the turn of the 21st century. The individual citizens, agents in the simulation, are motivated by abstracted economic behavior, like looking for a job and buying food for themselves and dependents, and in turn the economy of the simulation is affected by the millions of actions taken at the individual level. However, unlike a project undertaken by a government or a think tank, Max describes this work as speculative and exploratory: “We wanted to push it in this direction where it was a simulation where these other ways of addressing problems or even defining what problems are is a lot more open-ended.” For example, a viewer of the simulation can adjust city-wide parameters such as healthcare cost (or universal healthcare) to see the potential impact on the agents and the system. But despite the software’s numeric encoding and outputs, it is

12 Turkle, Sherry, William J. Clancey, Stefan Helmreich, Yanni A. Loukissas, and Natasha Myers. *Simulation and its discontents*. Cambridge, MA: MIT Press, 2009.

not meant to provide solutions to problems, but to explore possible futures.

Unlike a centralized, predictive model that tells us what the future will look like, Max's project is best viewed as a provocation, asking what the future might look like. Higher levels of technological automation coupled with universal healthcare might lead to an egalitarian, post-work utopia, while other simulation runs end in a late-capitalist societal collapse. On top of this, in Max's project, a viewer can return to the level of the agent, experiencing futures through the eyes of an individual, and see how various scenarios affects the lives, jobs, and health of simulated citizens. In fact, he had originally planned to "partner with various science fiction authors... We would give them a simulation run, and let them develop a richer narrative around that." Simulation space acts as a means of envisioning and extending the possibilities of real, physical and social spaces. Similar to his generative music project (described in the Generating chapter), Max sees the outcome of his work not necessarily as a finished product, but a jumping off point. After setting the environment, one can "see how the behavior of these simulated agents evolves and unfolds and from that. Maybe you can develop interesting narratives... out of the configurations that seem the most interesting." A simulation itself can act as a generative device, sparking the imagination and leading to the creation of new realities.

Case Study: Reimagining Urban Intersections

Introduction

With a fellow computational design researcher, Atefeh Mahdavi Goloujeh, I co-hosted a design workshop titled *Reimagining Urban Intersections through Systems Thinking*. Our workshop used traffic intersections as a site for potential design interventions, and introduced an open-source software, *Loopy*,¹³ to simulate factors influencing intersections as a system. We found that, in addition to being an effective tool to model emergent behavior in a system, *Loopy* serves to clarify and make explicit biases and assumptions of the designer. When used by a group of designers, this forms the basis of a conversation among the designers as well as between the designers and the simulation itself.

Participants

Participants are all attendees to the *Computational Design: Practices, Histories, Infrastructures* symposium, held at Carnegie Mellon University in October, 2017. The three participants in our workshop are students (two undergraduate, one graduate) enrolled in design or architecture

¹³ Case, Nicky. *Loopy*, <http://ncase.me/loopy/>, accessed September 21, 2017.

programs. All three identify as female, and are in their early- to mid-20s. Per the results of a survey distributed before the workshop, all three typically walk or take the bus to navigate their respective cities (this likely stems from their status as transient students, and may also be a factor self-selection toward this workshop in particular). When asked to name their most memorable experience at an intersection, all three gave negative responses, such as nearly being struck by a vehicle and “hat[ing]” a particular intersection in their hometown. With this question, we hoped to begin priming participants to think about negative events whose motivating factors they might model in a simulation.

Activity 1: Negotiating an Intersection

We began the workshop with a physical activity, both an icebreaker and a means to encourage different ways of thinking about intersections. We laid four cardboard ‘teardrop’ shapes, about 2 feet in diameter, on the floor, to suggest a cloverleaf interchange with the negative space.

Then, in a physical simulation with human bodies representing vehicular traffic, we asked the participants to begin walking through and around the intersection. The rules for agents in this simulation were that they should strive for continuous (not necessarily fast) motion and that they could not communicate with each other with speech.



Physical simulation of vehicular traffic with workshop participants in a cloverleaf intersection

After about one minute of this, we brought the cardboard shapes closer together on the floor to ‘tighten’ the intersection. After another minute, we declared that one path would become ‘one-way,’ allowing movement in one direction but not the other. Finally, co-investigator Mahdavi

Goloujeh and I stepped onto the cardboard, and acted as 'pedestrians' to the participants' 'drivers,' stepping in front of their paths between the cardboard shapes.

Activity 2: Generating Factors and Affinity Diagramming

Then, we provided some examples of real-world factors influencing intersections – amount and types of traffic lights, the presence of sensors on cars providing drivers with feedback, and a regional phenomenon known as the 'Pittsburgh left' – and we asked the participants to spend five minutes listing on Post-It notes as many other intersection factors they could think of.

After this, we asked the participants to bring their factors to a large whiteboard and look at all the factors. We asked them to move the Post-It notes around on the whiteboard to group similar factors into an affinity diagram, meant to draw out patterns of thinking. This led to a discussion about the different types of factors they had come up with, noting the common themes that had arisen, and their perspectives on these factors. After about 15 minutes of conversation, we asked the participants to identify five factors that they wished to focus on for the coming activities.



Factors influencing urban intersections

Activity 3: Loopy

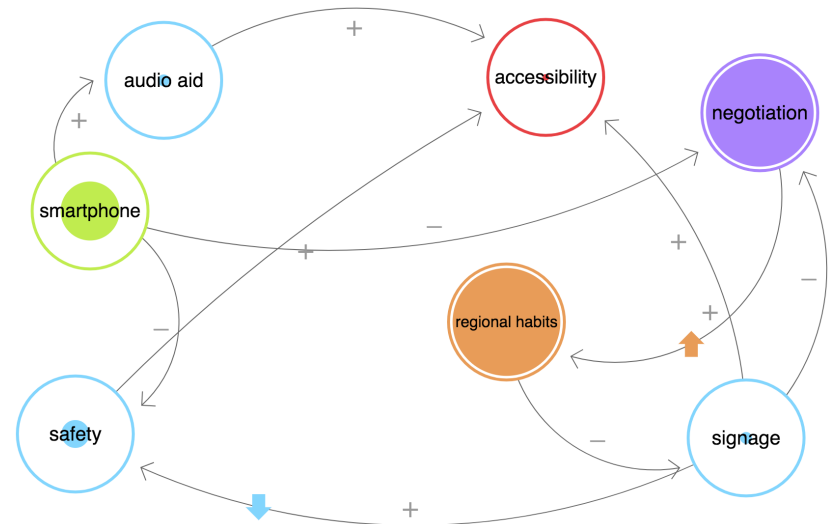
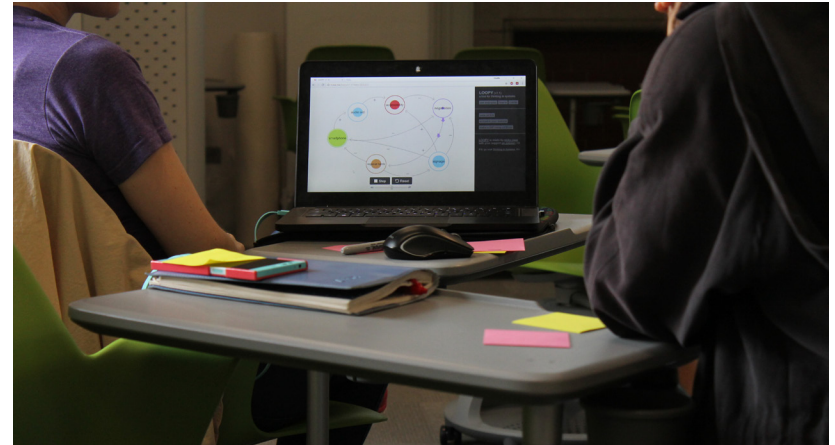
We introduced Loopy as another type of simulation which models the high-level relationships between various system factors. Loopy is not a tool to model specific types of systems, such as ecological or economic, but systems in general, as relationships of and flows between entities. In Loopy, a designer-user draws circles which represent factors, and then draws arrows between them to represent the interaction between these factors: for example, in a

basic ecological simulation, one might include ‘predators’ and ‘prey’ as the two factors. Two arrows would be drawn: A positive relationship from prey to predators (more prey leads to more predators), and a negative relationship from predators to prey (more predators leads to less prey).

For the intersection simulation, we encouraged the participants to think of their factors not necessarily in strictly quantitative terms, but also qualitative — a positive relationship might mean that factor A leads to ‘better’ or ‘stronger’ factor B (as opposed to more of it). The factors that the participants chose to simulate were:

- Pedestrian attention/distraction (articulated through ‘smartphones’)
- Accessibility (for both cars and pedestrians)
- Negotiation between drivers and pedestrians
- Regional habits
- Signage

The participants began by drawing circles for each of these factors, and then drawing arrows to model the relationships between the factors. For example, in their telling, smartphones have a negative relationship to negotiation between drivers and pedestrians — more smartphones results in less negotiation/communication. After setting up the relationships between factors, the participants ‘ran’ a



The participants' factors influencing intersections, modeled as a simulation in Loopy

simulation by making a change to the system: Increasing the number of smartphones in use. They then observed

the resulting, proliferating changes to the system. They found some of the results unrealistic, and changed some of the relationships in order to run the simulation again and observe these changes in action.

Eventually, they added a sixth factor, ‘safety,’ to serve as a yardstick by which to measure their simulations. After iterating on the simulation a few more times, they moved on to the final activity, proposing potential design interventions that could shift their modeled system in desirable directions.

Activity 4: Design Interventions

Finally, the participants used the whiteboards to propose some design interventions based on their system simulation. They first rewrote their initial factors on the board, and one of the participants drew a storyboard for an embedded sign meant to address smartphone users. They also sketched an intersection with smartphone-linked signage. The last design intervention was a system of signs aimed at cars approaching intersections that would state the number of pedestrians at the coming intersection.

Findings

From observing the participants using *Loopy* to model factors of urban intersections in a system, we draw three

overall conclusions. First, *Loopy* is an effective tool to simulate the unexpected consequences of changes in a system (that is, to simulate emergent behavior). Second, *Loopy* is most useful when viewed as a dialogue between the software and a designer or, preferably, a group of designers. Finally, *Loopy* helps designers to clarify and make explicit certain assumptions they hold, but it also obfuscates other biases.

Throughout the activity, the phenomenon (and observation) of emergent behavior played a significant role in the participants’ usage of the tool. *Loopy* requires the designer to make some change to the system in order to begin a simulation run. On the first iteration, the participants increased ‘smartphones.’ This soon led to an undesirable scenario, best told in the words of Participant 1: “Oh, no! No signage and no accessibility!” Later, after noting the increased levels of ‘smartphones’ and ‘regional habits,’ she described the effects as “...a bunch of people on their smartphones doing the Pittsburgh left.”¹⁴ All the participants noted how a small change in one area of the system could have ripple effects that defied their expectations

¹⁴ Typically, in U.S. American cities, a driver at a red light who is turning left, when the light turns green, will wait for others passing straight through before turning. In Pittsburgh, the first driver at a red light turning left takes precedence over others passing straight through (although a second or third driver turning left will wait). While there appears to be little literature on the Pittsburgh left, it is a well-recognized, if illegal, local behavior.

qualitatively and quantitatively. The concept of emergent patterns resulting from interactions at other levels in a simulation seemed to be reinforced through *Loopy*.

The participants were engaging in a back-and-forth dialogue with *Loopy*, using it as a discursive tool rather than a means to analyze or quantify behavior. Throughout the duration of the simulation activity, they ran at least eight iterations on their initial system design. From the second to third iteration, they didn't change any factor or relationship of the system, but initially increased 'signage' instead of 'smartphones,' and noted that the resulting scenario was preferable, with high levels of 'accessibility.' Participant 3 observed: "Oh, this is so much better." Looking back on this iteration a few minutes later, Participant 1 wondered, "Is there a way to max out smartphones that also increases accessibility?" Observing unexpected emergent behavior led the participants to trace certain effects, analyze their model, and make adjustments. As Participant 1 put it after a later iteration: "I guess deleting that link between smartphones and signage had a huge impact. Our design intervention is not working as well." *Loopy* is far from 'black box' software, which provides an output for an input without revealing its functionality. Instead, it allows designers to peer into causal relationships within their simulation, and encourages self-conscious iteration on their work.

Through discussion of how to model factors and relationships, designers working collaboratively must articulate certain assumptions they hold. However, *Loopy* also appears to reinforce other biases, and hides the effects of any un-modeled factors or relationships. Early in the activity, as they were beginning to draw relationships between factors, Participant 2 justified one link by saying: "Perhaps more signage would lead to less smartphone [usage], because people are like, 'Aw [expletive], I gotta look.'" Had the participants been provided with datasets related to the system factors, this would have provided fodder for the simulation. However, in this workshop, the participants' beliefs about how and why intersections function provided justification for their system model, and they had to communicate these to each other. Later in the activity, however, the simulation appeared to take on more and more agency, and conversation around the faithfulness of their model to reality took a backseat to creating desirable scenarios. In later iterations, working at a faster pace, the participants forgot what exactly they had changed from one iteration to the next and for what reasons. Conversely, they also increasingly began to see the factors they had modeled as comprehensive — an example of behavioral economics researcher Daniel Kahneman's phenomenon of 'What You See Is All There Is,'¹⁵ and a negative version of Turkle's "immersion" in simulations. A particular chal-

15 Kahneman, Daniel. *Thinking, fast and slow*. Macmillan, 2011.

lenge for designers working with *Loopy* or other simulation software is to remain engaged but skeptical throughout the process.

Future Work

Mahdavi Goloujeh and I plan to hold further versions of this workshop to address some issues we noted. First, we would like to conduct a workshop with a group of non-designers and non-students to discover how people who have not been exposed to systems thinking (as all of our participants had been) would use a tool such as *Loopy*. In the future, we will also encourage the participants to choose five *specific* factors to model in the simulation, as opposed to *general* factors such as ‘accessibility’ — this should be broken down into concrete examples, among which one could be chosen to stand in for the whole. However, we are intrigued by the participants’ introduction of a new generic factor, ‘safety,’ to serve as the ultimate goal of their system, and wonder how future participants might assess their simulations. To discourage the participants from making arbitrary changes to their system based on such a goal, we might ask them to track all their changes in a related document, and to justify them with stated assumptions.

We believe that *Loopy* is an effective tool for performing high-level, qualitative modeling of systems and their

behavior. For designers, it is especially useful as a discursive tool when used collaboratively, encouraging designers to make explicit their assumptions about systemic factors and their relationships. It also allows designers to model certain interventions into the system, such as (in this workshop) increased signage or smartphone usage. There are drawbacks to *Loopy*, which are likely common to any simulation software, such as designers ignoring possible factors that have not been modeled. This might be addressed through prompts in the software that ask the designers to consider what other effects or what other factors might exist in this system, or (to compensate for increasing system complexity) to include a separate area for listing unmodeled but relevant factors.

Conclusion

Computer models and simulations can present an immersive space that correlates with aspects of reality. Dominant narratives presenting computation as purely ‘objective’ data operated on by algorithms ask us to see simulations as mirrors of reality, that with increased computational power we can predict the singular future. However, as shown in the case study with Loopy, software for simulation can also act as a mediator, provide a means to make explicit certain understandings and biases that might otherwise remain hidden, and a space to model and explore these ideas. As my interview subjects show, it can be productive to engage with and interrogate simulations as discursive platforms, as representations of multiple futures that might be. Through simulating processes, designers can develop ways of envisioning and working toward those futures.

Interrogating

Overview

Interrogating might seem like a misfit among the other pillars of the computational design thinking framework. While there exist clear software programs for doing nothing but generating or simulating, computation for interrogating seems more nebulous (or brings to mind a dimly lit room and polygraph machine — which is not what I’m suggesting here). That’s because, of the three, interrogating is the most like a general approach to design work across situations and practices. It’s easier to form a notion of simulating through a collection of simulation softwares and technologies than it is for interrogating, which lacks obvious, material tools. However, as I will show, interrogating, as a critical questioning, is not only a productive mind-set for designers when computation enters the picture, but it is one that computational designers are uniquely able to adopt and operationalize.

Researchers have long theorized the practice of design as demanding criticality and introspection on one’s work. Design researcher Donald Schön advocates a ‘reflective practice,’ in which one is continually considering (and reconsidering) the steps one takes in the work one does. Through a process of critical questioning, of interrogating, one can improve on past work, and learn and develop new strategies for future work. For example, Schön describes

designers sketching out ideas as “having a conversation with the drawing.”¹ The act of drawing is not — can not — ever be a direct translation of immaterial ideas onto paper or a screen. While drawings are certainly informed by the designer’s thoughts, they are also mediated by the tools, the surface, the infrastructure, the language (whether programmed or spoken), the lighting, the mood, the environment, the political climate, etc. Some of these elements are unconscious or ambient background effects. Others, such as one’s tools and thoughts, are open to ongoing negotiating and interrogating throughout the design process. For artists, architects, and designers, the tools today are often digital and screen-based. Computation as commercial, user interface software often takes the form of programs that reveal no trace of their inner workings, but designers, working with and through computation, are particularly empowered to interrogate their tools through subverting and *hacking* their defaults. In addition, by discarding (or *unlearning*) certain rigid, computational mindsets they gain freedom from thought processes that are not conducive to the work of design. However, computational technology and concepts can also help in *transforming* designers’ projects, helping them to see and understand their work in new ways.

1 Schön, Donald A. *The reflective practitioner: How professionals think in action*. Basic Books, 1984.

Hacking

As noted by technology and psychology researcher Sherry Turkle, contemporary software often appears as opaque ‘black boxes.’ In the introduction to the 2004 reprinting of her 1984 book, *The Second Self*,² Turkle notes the reframing of the term ‘transparency’ in computing over the last two decades of the 20th century. Previously, using command line-based systems like DOS, “things felt transparent when computer use felt analogous to working on a traditional mechanical device, like a car.” By the mid-‘90s, however, with the advent of graphical user interfaces (GUIs) with ‘file’ and ‘desktop’ metaphors, “when people said that something was transparent, they meant that they could immediately make it work, not that they knew how it worked.” The ease of use of modern software means that, for example, Facebook boasts that it connects over two billion users globally as of June 2017.³ But it also introduces a more hierarchical model where the companies developing such ubiquitous software prevent their users (consumers) from inspecting or probing the underlying programs and algorithms. Still, an oppositional ethos persists in the form of ‘hacking’ culture — not in the sensationalized

sense of mysterious coders bringing down governments and corporations, but in the curious and critical work of everyday people, questioning the nature of the reality presented to them. Media theorist and philosopher McKenzie Wark, in his 2004 work *A Hacker Manifesto*, writes: “Whatever code we hack, be it programming language, poetic language, math or music, curves or colorings, we create the possibility of new things entering the world.”⁴ Hacking, then, is as much critical and subversive as it is projective, revealing the previously unseen; that is, it is an act of design.

Across my interview subjects, whether the term hacking is used or not, a mindset of hacking is clearly present. Anna, the digital artist and game designer, depicts the chasm between programmers working on products for commercial consumption and hackers as a fundamental difference in interests: “You have some folks who are very, very interested in making a beautiful, perfectly functioning, glitchless [software]. And then you also have people who are... interested in the thing itself because it has this weird logic and poeticism intrinsically.”⁵ This Warkian ‘weird logic’ and ‘poeticism’ lurking behind seamless interfaces proves an attractive line of inquiry for designers working with technology. For Anna, such investigations

2 Turkle, Sherry. *The second self: Computers and the human spirit*. MIT Press, 2005.

3 Facebook, “Two Billion People Coming Together on Facebook.” <https://news-room.fb.com/news/2017/06/two-billion-people-coming-together-on-facebook/>, Accessed November 6, 2017.

4 Wark, McKenzie. *A Hacker Manifesto*. Harvard University Press, 2004.

5 Skype interview with Anna, July 18, 2017.

lead to artwork self-consciously using code as an expressive medium, including pixel art, Twitterbots, and browser extensions that modify the user experience on certain websites (for example, by replacing one common phrase with another). Common in all these projects is a usage of software tools in ways that their developers did not necessarily intend. Used toward creative ends, Anna's subversive hacking practices can result in new forms of aesthetic expression.

Other interview subjects harness hacking toward more practical ends in their work. For some, it's a necessary step to doing design work at all. David, the PhD student in a technology-oriented architecture program, succinctly describes how given digital tools inevitably fail to address certain design scenarios: "There's many parts of computer science that are basically trying to understand the problems, break [apart] the problems, and solve the problems... [But] especially if you work with architecture or art, you cannot make that into a rational problem."⁶ When a design problem has not been (and perhaps cannot be) completely, computationally rationalized, an alternative for a technologist is to seek different ways of using their given tools, and in the process transforming them. Speaking about the boundaries of an architectural design software, Revit, Paul, the architect and technologist, said:

"When you're doing more radical design, you're constantly forced to trick [it]... There are so many different ways that you can just push a little bit or misuse a feature or hack or tweak in order to extend or expand the capabilities of these existing design platforms to get them to do what you want."⁷

In the architecture and construction industries, Revit is known as a software with a wide vocabulary — it models not just abstract geometrical forms, but also stairs, doorways, heating and cooling equipment, etc. However, the breadth of its knowledge is always incomplete, and technically-informed designers supplement it, simultaneously constrained by and creatively empowered by the limitations of the software. Paul describes an example from his work at a large architecture company: "I saw a tutorial today on our intranet that was on how to use curtain walls to model railings. A curtain wall is not a railing... But from Revit's standpoint, in terms of this element as being a collection of behaviors... a curtain wall is actually a really good representation of a railing." A wall becoming a railing: One element stands in for another, and in doing so, the designer questions both elements, interrogates the defaults of their tools, and sees the tool and the work in new ways. However, it is admittedly a challenge to question and

6 Skype interview with David, July 24, 2017.

7 Skype interview with Paul, August 2, 2017.

overcome the defaults of software that is presented as comprehensive in its scope. By extension, designers working computationally must take a second glance at notions of computational logic itself, which appears as a complete, closed system, in order to become more adept interrogators of technology.

Unlearning

As laid out in the introduction, a certain strain of computational thinking is presented as a necessary and sufficient basis for knowledge altogether. In recent years, ‘computational’ as an adjective has emerged to describe new approaches to various older fields — among them, computational biology, computational linguistics, and of course, computational design. Ironically, given post-Enlightenment progressivism and the hegemony of scientific knowledge, for some, computation even threatens the scientific method (see Stephen Wolfram, *A New Kind of Science*). However, both rational scientific analysis and computational processes tend to disregard or undervalue practices that cannot easily be quantified; for example, tacit/embodied knowledge, divergent thinking, and framing problems. Computational designers bring valuable approaches and strategies to the greater fields of design and technology, but there is also a distinct friction arising from the

intermingling of computation and design, processes with particular (and perhaps opposing) ways of knowing.

A few of my interview subjects addressed the sometimes arduous process of reconciling computational thinking with design. Ken, an adjunct professor of architecture and a technologist who learned programming at an early age, described challenges he faced specific to his background in his architectural education. He points to handling ambiguity in the design process in particular, confessing that he “couldn’t handle not having a process.”⁸ To wander in search of a solution, and sometimes in the process, reframing the question itself and starting over again from the beginning was onerous: “How could you just scrap your whole plan and start again?... That’s so inefficient and nonlinear from a code perspective.” Unexpected discoveries in every field, of course, come about only through nonlinear, emergent practices, as opposed to tried-and-true methods and techniques. For someone schooled in computation and programming practices, successful design work requires a simple (and not at all simple) mindset shift. Ken concludes, “I had to unlearn a lot of things about programming in architecture school... To acquire a logic of design was to abandon certain logics of programming.”

8 Skype interview with Ken, July 17, 2017.

Another interview subject, Maria, a design instructor, notes how certain ways of thinking are also related to age. A notion that, in design, one not only solves a given problem but frames it, is one that somehow grows more difficult with age. Having taught design to both elementary-level and high school students, she says:

“The 8 to 14 year-old range [are] actually much more open-minded... For the high schoolers... Some of them are against that ideology. They’re used to being given the topic. It’s hard, sometimes, for them to step back and really have an open mind about ways of solving for problems that are not one single thing.”⁹

Like Ken’s experience in architecture school, Maria’s students face difficulties not so much in the technical knowledge of software, but in the more nuanced worldview of designing generally. For Maria, as a teacher, one of the greatest challenges is instilling this spirit: “To teach a student to look at a problem from multiple perspectives, really analyze a problem critically, and then solve for that problem, through something that’s designed.” In addition, for students with an all-or-nothing approach to their work, the nonlinearity of certain unsuccessful designs can be

discouraging. Maria points to the fear of failure as another hindrance to embracing designerly ways of working: “A lot of these students are afraid of failing, they’re afraid of getting bad grades, so it’s been kind of indoctrinated into the way they’re thinking. It’s sort of like I’m unteaching them. They have to unlearn some of that to do this studio.”

While certain approaches to programming do run contrary to the design process, for designers who can pragmatically discard the constraints of programming logic, computation can also meaningfully and even poetically situate their creative practice. Beyond specific technologies and software programs, the ability to imagine and see transformations in objects and systems greatly expands the space of possibilities for designers.

Transforming

A critical, reflective practice requires contemplation and reconsideration of one’s work — of seeing in new ways — and technology also provides a means of achieving this. English art critic and historian John Berger, in his 1972 television series (and later book) *Ways of Seeing*,¹⁰ describes how the advent of photography altered perceptions of art: “The invention of the camera has changed not only what we see but how we see it... The painting on the wall...

9 Skype interview with Maria, September 10, 2017.

10 Berger, John. “Ways of seeing.” London: BBC. 1972.

can only be in one place at one time. The camera reproduces it, making it available in any size, anywhere, for any purpose.” For Berger, the camera transforms art, allowing us to see it and understand it in qualitatively different ways. Computational technology extends this phenomenon, transporting (and transforming) not only images, but data, information, and knowledge.

In conjunction with his hacking of Revit and architectural design software, Paul, the architect, proposes a way of working that involves multiple mediums. He describes a consistent experience in reviewing building designs: “We export our models from Revit into other platforms, like virtual reality... in order to walk around them or experience them in other ways... Every single time we’ve done that... [the designer] will notice something wrong with their model that they didn’t know was wrong.” Working in one medium alone limits the perspective of designers in such a way that their work is harmfully (not creatively) constrained. It’s preferable, instead, to reconsider one’s work, and different technologies serve as tangible environments in which to do that. Paul continues, “The best thing you can do to resist the specific limitations of a piece of software is to try it in another piece of software... use as many lenses as you have access to.” Interpreted as platforms, environments, and lenses, designers can see their computational tools not only as instruments for producing

work, but as mediating experiences that can lead to new understandings.

Just as palpable digital technologies shift our thinking, immaterial constructs and concepts taken from computation can also act as lenses, providing new mental models for one’s work. Ken, for example, has not completely ‘unlearned’ a computational way of thinking, but also uses the language of variables and abstraction to describe a process of framing a design scenario: “‘A something is an X that’s a Y’... I will play with that statement. ‘What’s not an X and a Y? What is an X and not a Y?’ Like: A bus stop is an outdoor shelter where you wait for something. What’s an outdoor shelter where you don’t wait for something? What’s an indoor shelter where you wait for something?” The use of syllogisms as a logical, rhetorical device dates to antiquity, but Ken’s example is notably computational. A designer could write a computer program, encoding and storing concepts like ‘outdoor shelter’ and ‘waiting’ as variables in a procedure which generates a space of possibilities and simulates various explorations of that space in order to manifest design solutions to that particular framing. But such a hypothetical program might not be strictly necessary; perhaps more important to designers is the ability to see and play with abstractions from particulars. Computational thinking is only a rational, problem-solving strategy, but can be discursive: ‘How might I think about

this object? What if I treated it as a variable? A constant? What parameters could be adjusted? What functions does it permit and what effects would they have on other objects?’ When asked about what he’s interested in learning next, Ken’s answer reveals a strong urge to see and know in new ways: “I would want to learn category theory and functional programming, because I think it would afford really interesting... conceptual metaphors to think about the world... I’m really curious [as] to what [else] I would learn that would help me see the world differently.” The tools designers use influence the way they think, but those tools also embody (implicitly or explicitly) a way of thinking — a lens that can then consciously be adopted or discarded in various design scenarios. The logics and truths of computation can serve as tools themselves in a designer’s repertoire. Alone, as David reminds us, computational thinking is insufficient for “architecture or art, [which] cannot [be made] into a rational problem,” but complementing other ways of seeing, it greatly expands the capabilities of the designer and the possibilities for design as a discipline.

In the following case study, I’ll describe my work as a software developer on an interactive project meant to elucidate the structure and underlying principles of a mathematical formula for computer-aided design. Both explanatory and exploratory, the software presents encoded 3-dimensional geometries in an embodied way, and offers

a space in which users may themselves act as interrogators of the constructs of mathematics and design technology.

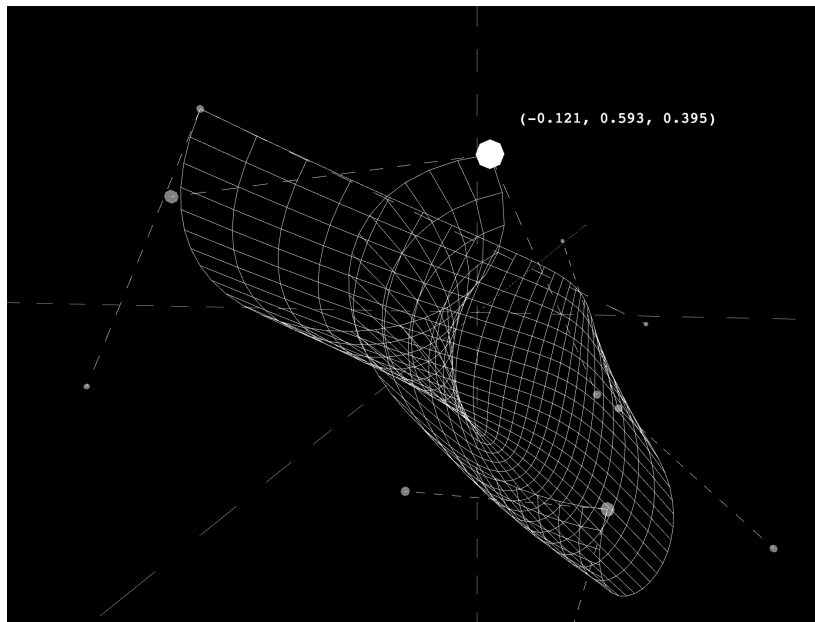
Case Study: Coons Patch Reconstruction

Background

As a research assistant for an exhibition on post-War computational design practices, with the curator (and my master’s thesis advisor) Daniel Cardoso Llach, I worked on a software program to visualize a mathematical formula for creating freeform, 3-dimensional surfaces. The formula, which takes in four boundary curves joined at the endpoints, produces from them a surface called a *Coons patch*, named for mathematician and designer Steven A. Coons. Coons patches (or surfaces) provide a succinct digital representation of complex forms. Rather than storing all the coordinates of a large number of points on a freeform surface, as in a mesh representation, Coons patches are defined by the parametrization of the surface’s boundary curves (which might be comprised of as few as 3 or 4 points in 3-dimensional space) and the patching formula. In this software program, our goal is to create an interface to reveal the underlying mathematical structure behind these elegant, complex surfaces, and to allow non-technical users to interrogate the relationship between the two.

Interface

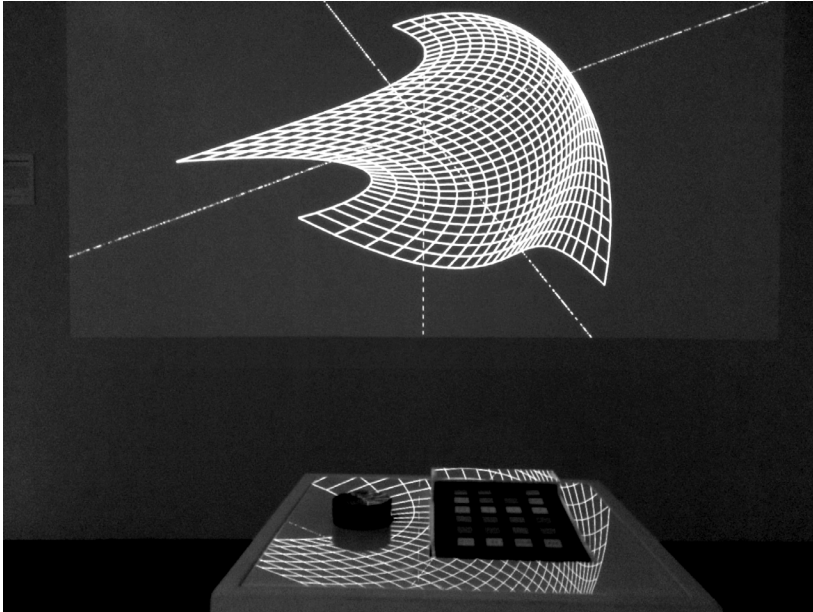
A single patched surface is presented as a grid of white curves on a black background, paying homage to videos from the early 1970s from the Ford Motor Company's work using Coons patches. However, where the designers at Ford had to manually encode each coordinate of all the points on the boundary curves inscribing the surface, users of this software benefit from a more embodied, interactive experience. A single button, for example, 'morphs' the surface from its current state. The code for this feature



Screenshot of the Coons patch software reconstruction

chooses random values within a bounded range for each of the endpoints and control points of the boundary curves and moves the points until they reach the new values. Morphing the surface enables a user to quickly see some of the many possible forms a Coons patch might take.

A central element of the interface is a point toggle and the ability to move individual endpoints or control points. Whereas morphing the surface discards the current state in order to present a new, random surface, with this feature a user gains a fine-tuned level of control over the form. From a random surface, a user might decide that they want to make changes to a specific region. By turning on point controls and toggling through the endpoints and control points, they can select the ones that are closest to the region they want to adjust. Then, they can choose an axis (x, y, or z) along which to move the point, using a tactile control knob. Minute adjustments of a point along an axis affect the area closest to that point, while more extreme adjustments affect almost the entire surface. The whole time, the coordinates of the point the user has selected are displayed, revealing the mathematical representation behind the points, curves, and surface, as well as allowing the user to examine the relationship between numeric values and geometric form.



Photograph of Coons patch software showing control knob and keypad

Discussion

Interrogation, in this project, is not only on the part of the users prying open the hood of opaque geometries to see the mathematical machinery underneath, but also in my role as the software programmer, against dominant narratives of interface design. In my past experience as a web designer and developer at and for various organizations, a common refrain is ‘ease of use.’ A popular book on web usability, *Don’t Make Me Think*, promotes the philosophy that user experience should be as seamless, obvious,

and easy as possible. Surface morphing certainly falls under this approach. But with point toggling and adjusting, I take the stance that, when the ‘user goal’ is not simply transactional, but toward acquiring and expanding knowledge, a degree of awkwardness and interruption can be more memorable and helpful than a perfectly smooth experience. In fact, the experience of fiddling with a single endpoint or control point, moving it along the x-axis, then the z-axis, then a bit back on the y-axis, etc. is very similar to the experience of a beginner programmer, manually experimenting with variables and loops, or that of a designer slowly acquiring facility with a 3d modeling software. The Coons patch formula, supported by its proof, and a few illustrations of unique surfaces might provide an abstract, factual knowledge of how it works, but having an embodied interaction — transforming and hacking the surface by hand — brings about a deeper understanding and helps to engender a mindset of critical questioning.

Conclusion

Dissatisfied with contemporary notions of ‘transparency’ in software, rigid computational logics, and the resulting limitations for design, computational designers critically question their work through interrogative processes. Paul’s subverting and hacking Revit revealed cracks in the façade of the architectural software in which to work. Maria depicted the open mindset of younger children as preferable to one which merely solves problems. And Ken playfully and poetically illustrates how computation can serve as generative metaphors for design thinking. In interrogating, computational literacy and thinking skills become valuable not only for producing optimized, efficient solutions to problems, but for expanding spaces of possibility and arriving at new understandings. The Coons patch software reconstruction serves as an example of an interface through which seemingly opaque geometries might be interrogated by non-technical users. In teaching basic programming and computational thinking, this fact should be stressed: That computation is as much a mode of problem-solving as it is problem-framing. A solution to a complex problem arrived at through computation alone inevitably misses the bigger picture. However, approaching computation as an interrogator, empowered to critically question the story it tells, a designer working computationally can discover and operationalize new ways of seeing the world.

Conclusion

Contributions

In opposition to reductive, determinist notions of computation, this work shows how computational design offers valuable methods, approaches, and strategies to working in socio-technical complexity. In this thesis, I have presented a three-part framework for understanding computational design as a situated, contingent, and evolving set of practices and approaches. The first component, generating, reveals how computation enables the designer to work at levels of abstraction, gaining facility in shaping and navigating large possibility spaces. The second, simulating, provides a frame for exploring complex systems, and envisioning and modeling potential interventions in those systems. Finally, interrogating, drawing from both Schön's 'reflective practice' and an ethos of hacking, encourages computational designers to critically question their tools and practices in order to discover new ways of working and thinking. I support each component of the framework with background texts from computation and design, interviews with individuals demonstrating positive deviance in their creative work and research, and case studies of my own investigative software prototypes and design workshops. The computational design thinking framework, supporting interviews, and case studies are the culmination of this thesis, but I see an almost overwhelming space of possibilities for further research.

Discussion & Future Work

First, some limitations and shortcomings of this work should be addressed (which might also suggest paths for future work). Notably, there is a U.S. American (and specifically East Coast) bias among my interview subjects, with New York City, Cambridge/Boston, and Pittsburgh especially overrepresented relative to the rest of the world. In addition, fully two-thirds of my interview subjects identify as male, a disparity I would love to see reversed. In addition to striving for more egalitarian representation, it is likely that further examples of positive deviance are to be found among groups that have traditionally been marginalized. The goal would not be to co-opt their practices, but to amplify those voices, and to bring about more diverse, thoughtful work toward the creation of desirable futures.

Technically, the software prototypes among my case studies are myopic with respect to the graphic user interface. Again, with my background as web developer, working within the computer screen is my comfort zone. However, there is also ample room for work taking a more embodied approach, through physical computing, virtual or augmented reality, and embedded interfaces (for just a few examples). I believe that software as research tools within those paradigms would help to further nuance the components of the framework, providing different computational

perspectives on and approaches to generating, simulating, and interrogating.

In terms of future work, I see this framework as having great potential in pedagogy. While I have depicted computational design at various points as a set of overlapping, neighboring practices and approaches to design, borrowing heavily from other fields, it is also the case that (for example) Computational Design is a singular track in the School of Architecture at Carnegie Mellon University. Two hours east of here, at Penn State University, is the Stuckeman Center for Design Computing. MIT, meanwhile, houses the Design and Computation Group within their architecture school. If, in these examples (among many others) from the academy, it is to be presented as a cohesive area of study, I propose that this framework provides a high-level overview of productive approaches to computational design. Irrespective of technological shifts and advances in the coming decades, I see this framework as remaining relevant as an argument in favor of unique human agency in the tide of advancing computational efficiency (especially in the age of artificial intelligence and machine learning). The evidence I have offered in this thesis suggests that it has viability at both the graduate and advanced undergraduate levels in design studios to provide a framing of the 'how' and 'why' of computational design. Devising a seminar syllabus, a design project or

series of projects, or using the framework to restructure an existing course would all be viable means of testing the efficacy of the framework in computational design education. Concretely, a studio instructor could strive to self-consciously instill a mindset of generating and simulating in their students toward the formulation of design questions and the iteration and refinement of design interventions, all the while maintaining an interrogative stance toward the student's materials and tools. In doing so, this will also engender a mindset of critical questioning as well as openness to shaping and exploring spaces of possibility.

Another direction for the framework lies outside of the academy and traditional design learning contexts. Architecture is an intellectually open but practically closed field, with a grueling internship and licensing process and appalling representation of women and minorities among its professional ranks. Computational design, as an area of study typically housed within architecture schools, suffers from many of the same problems (again, as does this thesis research). To address this, there must be opportunities outside of traditional universities and academic programs. The two events described in the introduction, the computational design symposium and the Cybernetics Conference, were each affiliated with various institutions, but opened doors to the wider public. In addition, each demonstrated impressively diverse representation (both

demographically and according to discipline). Other organizations, such as the School for Poetic Computation and Learning Gardens, a community of “self-organized learning groups,”¹ provide promising alternatives to the university classroom or design studio as well as to for-profit programs aimed squarely at career placement. As the computational design framework outlined here is itself critical and projective, it would be appropriate for its pedagogical embodiment to also chart a new educational path forward.

1 Learning Gardens, <http://learning-gardens.co/>. Accessed December 1, 2017.

Appendix

Interview Subjects¹

<i>Aaron</i>	master's student in technology-oriented architecture program
<i>Anna</i>	digital artist, game designer
<i>David</i>	PhD student in technology-oriented architecture program
<i>Ken</i>	adjunct professor, community organizer, technologist
<i>Maria</i>	recently graduated architecture student and educator
<i>Max</i>	artificial intelligence researcher
<i>Natalie</i>	software developer and designer at technology company
<i>Paul</i>	architect and design technologist
<i>Zach</i>	recently graduated graphic designer

¹ Names as shown are pseudonyms and, in some places, I have made slight adjustments to how interview subjects describe their work in order to preserve anonymity.

Bibliography

Alexander, Christopher. *Systems generating systems*. 1968.

Batty, Michael. *Cities and complexity: Understanding cities with cellular automata, agent-based models, and fractals*. The MIT press, 2007.

Berger, John. "Ways of seeing." London: BBC. 1972.

Capra, Fritjof, and Pier Luigi Luisi. *The systems view of life: A unifying vision*. Cambridge University Press, 2014.

Cardoso Llach, Daniel. *Builders of the vision: Software and the imagination of design*. Routledge, 2015.

Carmo, Mario. *The second digital turn: Design beyond intelligence*. MIT Press, 2016.

Case, Nicky. Loopy, <http://ncase.me/loopy/>, accessed September 21, 2017.

Coons, Steven Anson. "An outline of the requirements for a computer-aided design system." In *Proceedings of the May 21-23, 1963, spring joint computer conference*, pp. 299-304. ACM, 1963.

DeLanda, Manuel. *Philosophy and simulation: The emergence of synthetic reason*. Bloomsbury Publishing, 2011.

Golumbia, David. *The cultural logic of computation*. Harvard University Press, 2009.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672-2680. 2014.

Goodman, Nelson. *Ways of worldmaking*. Hackett Publishing, 1978.

Habraken, N. John, and Mark D. Gross. "Concept design games." In *Design Studies* 9, no. 3 (1988): 150-158.

Kahneman, Daniel. *Thinking, fast and slow*. Macmillan, 2011.

Krug, Steve. *Don't make me think!: A common sense approach to Web usability*. Pearson Education India, 2000.

Kwinter, Sanford. "The computational fallacy." In *Thresholds* (2003): 90-92.

Lakoff, George, and Mark Johnson. *Metaphors we live by*. Chicago: Chicago University Press, 1980.

Loukissas, Yanni Alexander. *Co-designers: cultures of computer simulation in architecture*. Routledge, 2012.

Marsh, David R., Dirk G. Schroeder, Kirk A. Dearden, Jerry Sternin, and Monique Sternin. "The power of positive deviance." *BMJ: British Medical Journal* 329, no. 7475 (2004): 1177.

Marx, Leo. "'Technology': The Emergence of a Hazardous Concept." In *Social Research* (1997): 965-988.

McPherson, Tara. "US operating systems at mid-century." In *Race after the Internet*, 2013.

Meadows, Donella. "Leverage points: Places to Intervene in a System," 1999.

Mitchell, William J. *The logic of architecture: Design, computation, and cognition*. Cambridge, MA: MIT press, 1990.

Neugart, Michael, and Matteo Richiardi. "Agent-based models of the labor market." LABORatorio R. Revelli working papers series 125, 2012.

Pedercini, Paolo. "SimCities and SimCrises," 2017.

Perez, Liliana, and Suzana Dragicevic. "An agent-based approach for modeling dynamics of contagious disease spread." *International journal of health geographics* 8, no. 1 (2009): 50.

Pollio, Vitruvius. *Vitruvius: The ten books on architecture*. Harvard University Press, 1914.

Reddy, Michael. "The conduit metaphor." In *Metaphor and thought* 2 (1979).

Schelling, Thomas C. "Dynamic models of segregation." *Journal of mathematical sociology* 1, no. 2 (1971): 143-186.

Schön, Donald A. "Designing: Rules, types and words." In *Design studies* 9, no. 3 (1988): 181-190.

Schön, Donald A. *The reflective practitioner: How professionals think in action*. Basic Books, 1984.

Steenson, Molly Wright. *Architectural intelligence: How designers, tinkerers, and architects created the digital landscape*, MIT Press, 2017.

Stiny, George. *Shape: Talking about seeing and doing*. MIT Press, 2006.

Stiny, George, and James Gips. "Shape Grammars and the Generative Specification of Painting and Sculpture." In *IFIP Congress* (2), vol. 2, no. 3. 1971.

Stiny, George, and William J. Mitchell. "The Palladian Grammar." In *Environment and planning B: Planning and design* 5, no. 1 (1978): 5-18.

Suchman, Lucy A. *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, 1987.

Sutherland, Ivan E. "Sketchpad: A man-machine graphical communication system." In *Transactions of the Society for Computer Simulation* 2, no. 5 (1964): R-3.

Turkle, Sherry. *The second self: Computers and the human spirit*. MIT Press, 2005.

Turkle, Sherry, William J. Clancey, Stefan Helmreich, Yanni A. Loukissas, and Natasha Myers. *Simulation and its discontents*. Cambridge, MA: MIT Press, 2009.

Uptis, Alise. "Nature normative: The design methods movement, 1944-1967." PhD diss., Massachusetts Institute of Technology, 2008.

Wark, McKenzie. *A Hacker Manifesto*. Cambridge, MA: Harvard University Press, 2004.

Wilensky, Uri, and William Rand. *An introduction to agent-based modeling: Modeling natural, social, and engineered complex systems with NetLogo*. Cambridge, MA: MIT Press, 2015.

Winograd, Terry, and Fernando Flores. *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.