ABSTRACT

The day is not far off when autonomous, artificially intelligent agents will be employed in creative industries such as architecture and design. Artificial intelligence is rapidly becoming ubiquitous, and it has absorbed many capabilities once thought beyond its reach. As such, it is critical that we reflect on the relationship between AI and design.

Design is often tasked with pushing the envelope in the quest for novel meaning and experience. Designers can’t always rely upon existing models to judge their work. Operating like this requires a curious and open mind, a willingness to eschew reward and occasionally break the rules, and a desire to explore for the sake of exploring. These behaviors fly in the face of traditional implementations of computation and raise difficult questions about the autonomy and subjectivity of artificially intelligent machines.

This paper proposes computational play as a field of research that covers how and why designers roam as freely as they do, what the creative potential of such exploration might be, and how such techniques might responsibly be implemented in computational machines. The work argues that autotelism, defined as internal motivation, is an essential aspect of play and outlines how it can be incorporated in a computational framework. The work also demonstrates a proof-of-concept in the form of an autonomous drawing machine that is able to plot a drawing, view the drawing, and make decisions based on what it sees, bringing computational vision and computational drawing together into a cyclical process that permits the use of autotelic play behavior.
INTRODUCTION

Architecture has traditionally been slow to adopt new technology. Tools are incorporated into an office only after careful vetting, to ensure they do not compromise the creative processes of design. This hesitance reveals a persistent misalignment between the capabilities of the tools and the reach of the design process. Such a gap often results in notions of how technology will influence design—but not the reverse. With a few notable exceptions (Sutherland’s Sketchpad, Gehry’s Digital Project, etc.) discussions of design rarely influence computational development.

One area that is particularly lacking in contributions from the design community is artificial intelligence. AI, as a major focus of computational development, is rapidly becoming ubiquitous. It is employed to some extent in almost every professional field, and the day is not far off when intelligent computational agents will be widespread in creative industries such as architecture and design. Whether designers are comfortable with the prospect or not, these computational design agents will demonstrate some degree of autonomy. If the prospect of autonomous, creative AI is not a matter of if but when, then it is also critical for us to ask how it will come about.

Currently, artificial intelligence is developed on the premise that intelligence is primarily about problem-solving. While this strategy is appropriate for the majority of AI’s current applications, it does not capture the full scope of design thinking. Designers demonstrate a curiosity and internal motivation that is critical to the creative process—they play with even the most serious of topics. This behavior escapes traditional models of rational problem-solving and seems at odds with computation as we have come to understand it. How can we model the kind of playful behavior that designers exhibit in our computational implementations of artificial intelligence?

The objective of this work is twofold: First, the work will present a theoretical framework for implementing playful design behavior in low-level, autonomous computational agents. Specifically, the work focuses on the motivation of the agent, advocating for the necessity of autotelism (internal motivation) in play. Autotelism’s role in the creative process is discussed, and the computational implications of implementing autotelism are outlined.

Second, the work also discusses the construction of an autonomous, autotelic drawing machine, which acts as both a testing ground for the ideas presented as well as a proof-of-concept of the computational framework. The machine uses a cyclical process of machine vision, rule application, and drawing construction to iteratively progress through a drawing exercise. Together, the computational framework and drawing machine promote a dialogue into the nature of computational play.

It is important to note that this avenue of research does not just contribute to the field of computer science. Equally important is the reciprocal effect upon design. Just as the study of artificial intelligence is also an in-depth study of human intelligence, the study of computational play will illuminate what it means to play as humans and as designers. Doing so will improve the ongoing dialogue about design thinking, the creative process, and how design and computation can be brought together in novel and productive ways.
BACKGROUND
Design Problems as Indeterminate
According to design professor Richard Buchanan, design is full of wicked problems. Such problems are ill-defined, indeterminate, dynamic, and confusing (Buchanan 1992). Discussing such problems in the context of professional fields, philosopher and urban planner Donald Schön describes indeterminate zones of practice as situations that “escape the canons of technical rationality.” These zones of practice are areas of “uncertainty, uniqueness, and value conflict” that are common across all professional fields (Schön 1990). The problem space of an architectural design, for example, is a complex, fluctuating landscape that incorporates often-conflicting considerations of aesthetics, culture, structure, and budget, among a myriad of other factors. In dealing with such issues, designers explore with limited or no knowledge of what the end goal should be, or even if the considerations of today will be the same as those of tomorrow. Because of this, designers must be ever on their toes, ready for change. Schön describes the designer’s ability to deal with surprise as reflection-in-action, a type of improvisational problem-solving and on-the-spot experimentation that allows designers to try previously untested ideas (Schön 1990).

Navigating indeterminate spaces requires the designer to both come up with the question and provide an answer. Sociologist Richard Sennett reflects on a similar scenario at the heart of craftsmanship, stating that the experienced craftsman must be able to localize, to question, and to open up a problem (Sennett 2008). Schön captures these ideas in his description of problem-framing. He also notes that in philosopher John Dewey’s view, the designer is “one who converts indeterminate situations to determinate ones” (Schön 1990). All of these depictions involve creating a path for inquiry and investigation. This is a back-and-forth investigation of ways to ask the question and ways to answer it. As the design progresses, both the problem space and the solution space evolve. During this process, the designer must remain open to the possibility that the solutions may have an impact on the initial question. Knowing that the initial question might change at any moment drastically reduces the ability of the designer to rely upon the initial question for guidance. Thus, exploring the current space of possibilities is useful not just for its potential to solve the problem, but for its potential to shed light on the question.

Design as Artistic Creation
At the same time, design is a form of artistic expression. While designers have numerous practical, real-world constraints to attend to, they also share in art’s search for novel meaning and experience. Architecture, for instance, is far more than problem-solving. Buildings must satisfy a range of technical criteria, but we prize them for their ability to transcend these utilitarian roles and serve as powerful expressions of personal creative vision (Schön 1990). Anthropologist Tim Ingold describes the creative process as “an ongoing generative movement that is at once itinerant, improvisatory, and rhythmic.” He later states that practitioners “are wanderers, wayfarers, whose skill lies in their ability to find the grain of the world’s becoming and to follow its course while bending it to their evolving purposes” (Ingold 2010). Both Schön and Ingold use the same term for this creative activity: making (Ingold 2010; Schön 1990). Schön states that:

...designers construct and impose a coherence of their own. Subsequently they discover consequences and implications of their constructions—some unintended—which they appreciate and evaluate...Their designing is a web of project moves and discovered consequences and implications, sometimes leading to reconstruction of the initial coherence—a reflective conversation with the materials of a situation. (Schön 1990)

This process of seeing and moving in design pairs well with professor of design and computation George Stiny’s description of embedding and fusion in shape grammars. Both insist on design as an ongoing, cyclical construction of meaning (Figure 4).

Characterizing Design
Characterized this way, design is an iterative, meandering search for interesting questions and possible values in an ambiguous and dynamic space of possibilities. Designers make their own way through this indeterminate space, often with very little guidance. In order to do so, designers cultivate certain behaviors—of curiosity, open-mindedness, and freedom from external constraint. These behaviors indicate a willingness to change that stands contrary to traditional notions of problem-solving. As will be discussed later, these behaviors fall nicely into a description of play.
Play

Play is the autotelic behavior of a subject temporarily exploring a system of rules. This definition bears a strong resemblance to game designer Brian Upton’s depiction of play as “free movement within a system of constraints” (Upton 2015). Playful behavior is meandering and exploratory, but also very orderly, as it is based on rules (Gadamer 2004). The rules form the playground, or setting, for the play. Dutch historian and early play theorist Johan Huizinga refers to this as the magic circle of play (Huizinga 2014). Players are driven by an inner curiosity and motivation, often eschewing external guidance in favor of their internal interests (Bruner 1979). This characteristic is captured in play’s autotelism, which specifies that play has no goal other than self-perpetuation (Hein 1968).

As an autotelic activity, play permits the agent to wander, following paths that might otherwise have been ignored or even changing the rules to enhance the play. In play, the agent can break the rules without any justification other than curiosity. The player remains open to the possibility of questions that haven’t been asked yet, and solutions that haven’t been considered. This type of autotelic activity can be difficult to fit into dialogues dominated by problem-solving and optimization, but it is essential to both play and the creative process. The autotelism in play allows the designer to suspend external considerations in favor of free exploration.

Play also provides a natural bridge for communicating between design and computation. The terminology used in discussions of theoretical play echoes that used by both fields. For instance, rules form the backbone of play. The rules of play demarcate the playground and establish what is and is not allowed to happen. Rules are specifications of activity that focus on process over product. As such, they are especially useful in discussions of design activity, where curiosity and improvisational exploration often precede the ability to make logical decisions. Rules also relate directly to computation, where they refer to the rule-based systems that underlie many computational models. Such systems specify behavior for the computer to carry out when a certain condition or set of conditions is met. Play’s alignment with both design and computation situates it as a promising method for bringing the two fields into closer collaboration.

METHODS
Theoretical Framework: Computational Play

Several strategies for implementing play in a computational framework are outlined below. These techniques incorporate both the observable and inferred qualities of play activity and address the player as a critical component to the play (Figure 5).

First, computational play is facilitated by modal representation conducted by the agent. This can take a variety of forms, but the process must be significant enough that it requires the agent to translate its own work between different modalities, interpreting the content along the way. An example is the designer’s tendency to sketch. This externalizing process is key to the designer’s ability to shed a cognitive bias and approach a topic from a different perspective (Suwa and Tversky 2002). Insisting on multiple modes of representation acknowledges that the process of translation often ignites our subconscious, causing the kind of insight that often proves so fruitful to the design process (Stiny 2015).

The second characteristic of computational play is its focus on generative techniques that do not rely upon results for justification. Designers—as well as many other artisans, craftsmen, and creative professionals—improvise. Rule-based behaviors work well in this regard, as they specify behavior without relying upon efficiency or other heuristics for guidance.

Third, computational play is iterative. Combined with the insistence upon modal representation, this lends play a cyclical structure reminiscent of the design processes outlined independently by both Schön and Stiny (Schön 1990; Stiny 2011). Iteration opens the design process up to shifting contexts and changing requirements. Iteration grants the same level of importance to both question and problem. At the same time, it also results in a natural discretization of the creative process that is conveniently applicable to computation.

Finally, and most importantly, computational play should be autotelic. While the three traits described up until this point provide a strong framework for observable playful activity, autotelism is the key to untethering the designer and encouraging the kind of curious exploration at the center of play.
Autotelism’s importance stems from the fact that it taps into both the objective and subjective qualities of the player-at-play. Autotelism is driven not only by the number of possibilities available to the player, but also the player’s willingness to explore. These two qualities are intertwined but ultimately independent, and at each moment they provide a rich ground of possible motivations for the designer. Plotting these two variables as independent axes of a behavioral region illustrates this range of possible outcomes. The predictability of the behavior tends to be higher in the upper-right and lower-left corners, where the number of possibilities and the interest level of the agent are similarly extreme. In the opposite corners, the scenario is less predictable: The agent has an equal chance of quitting or continuing the process (Figure 6).

If we begin to study intentionality and attribution, however, the graph changes. All four corners are easily attributable, as the extreme values (from one or both axes) yield straightforward cues to behavioral choices. Only in the center, where neither axis holds significant sway over the decision, does attribution dissipate (Figure 7).

Including attribution draws attention to the upper left and lower right corners, where the behavior is highly unpredictable, yet easily attributable. Adding color to this diagram along that axis better displays the range of interesting possibilities that autotelism is able to bring to the decision-making process. An agent that has few possibilities left but an intense desire to continue may break or change the rules in order to keep the play going. Conversely, an agent that has plenty of room left to explore, but no real willingness to do so, may quit playing prematurely (Figure 8).

Overall, this framework for computational play focuses the process on the ever-changing possibilities of the present, rather than the most likely or most successful future outcome. Doing so acknowledges the role of designers as those who pursue the possibilities that could be, rather than the ones that work based...
on their current understanding. Computational play encourages agents to push the envelope and pursue novelty simply for the sake of novelty.

In addition, this framework acknowledges the subjectivity of the machine. This is another time that we must not ask if it will exist, but when. Concepts like machine choice, preference, and belief may still be far from any real-world computational implementation, but it is no longer hard to imagine a day when subjectivity is shared by humans and machines alike. These concepts play a key role in the creative process, and we must begin to explore what happens when they are implemented computationally.

Implementation: Hardware
The framework discussed above might be difficult to reconcile with our traditional understanding of computation, but it is not inherently difficult to implement. This section will discuss the construction of an autonomous, autotelic drawing machine that demonstrates computationally playful behavior.

The machine hardware is a Makeblock XY Plotter (V2.0) resting on a 2 x 2 ft acrylic base (Figure 9). The Z-axis of the machine is outfitted with both a dry-erase marker and an eraser. The stepper motor for this axis is fixed with an elliptical rotor. Resting on this rotor are two separate 3D-printed mechanisms that individually hold the marker and the eraser. As the motor turns, the elliptical rotor causes one mechanism to rise while the other falls; turning in the other direction reverses the motion (Figure 9, upper right.

8 Autotelic behavior space, with behavior predictability and attributability noted. The brightness (amount of black or white) of any point in the space indicates the likelihood that the activity will continue. The blue and green colors begin to indicate the emergence of interesting, unpredictable, attributable behaviors.
corner). When the rotor is positioned so that the long axis of the ellipse is horizontal, both marker and eraser are lifted above the acrylic base (Figures 10–12). In this way, the machine is capable of both additive and subtractive drawing procedures.

A webcam sits near the center of the machine, approximately 250 mm above the surface of the acrylic base (Figure 2). In order to ensure an evenly lit drawing surface, LED lights line the underside of the machine around the entire perimeter (Figure 3).

A connected computer acts as the digital "brain" to this plotter "hand" and webcam "eye." All of the computational processes—the webcam feed, the digital drawing manipulations, and the G-code processing—are managed in Grasshopper for Rhino. The components for receiving the video feed and communicating with the Arduino are provided through Firefly. Once the G-code has been determined, it is communicated from Grasshopper through Firefly to the Arduino, which then uses pre-loaded Grbl software to translate the G-code into commands for the plotter’s motors.

**Implementation: Software**

Echoing the design process espoused by both Stiny and Schön, mentioned earlier, the machine works through an iterative cycle of seeing and drawing. The computational framework is divided into three stages: machine vision, rule application, and plotting. These stages roughly align with the seeing-drawing cycle of design, with the intermediate rule application section representing the conscious processes that designers use to choose what to draw. The resulting cycle is see-choose-move (Figure 14).

In the first stage, the machine views its drawing. The webcam feed that is passed to the computer is immediately processed into a more usable format through brightness and threshold controls. Next, the computer uses a set of custom machine vision scripts to extract edges from the image and collect them into digital curves. This rough geometry is then smoothed into a cleaner interpretation. A separate data tree is constructed that describes the connectivity of the curves: each branch of the tree is an intersection of two or more curves, and each leaf is a neighboring intersection. Using this data tree, the computer determines every possible closed, non-self-intersecting path that exists in the drawing. After removing duplicates, the computer uses the unique paths to build closed shapes, which are passed to the next stage in the computation.

In the second stage, the machine determines all of its possible next moves, decides whether or not it will continue, and chooses a move to make. The moves are determined through the use of drawing rules that take as input a certain shape or drawing and output a modified shape or drawing. For instance, a simple triangle rotation rule looks for an equilateral triangle of any size and rotates it around its centroid (Figure 13).

**rule application:**

\[ \triangle \rightarrow \triangle \]

"rotate a triangle 180 degrees around its centroid"

13 Rule application. The left side of the rule specifies what shape to look for, while the right side specifies what shape to draw. In this case, once an equilateral triangle is found, it is rotated around its centroid (erased and redrawn).
The machine matches the curves supplied from the previous stage against all of its internal rules. Having discovered all of the shapes that match a particular description, the actual rule applications are carried out, resulting in a number of possible next moves. One of these shapes is then selected to be drawn. The rule application portion of the computation also includes the implementation of autotelism. This component takes as input both the number of possible next moves as well as a measurement of the machine’s own internal interest in continuing the activity in order to determine whether or not to continue.

In the third and final stage, the computer instructs the plotter to draw the new geometry. First, the digital lines are connected together into a toolpath, complete with initial travel distance, intermediate travel steps, draw and erase information, and concluding travel distance. This toolpath is then broken into individual points, which are converted into corresponding G-code and sent to the machine. At the end of the drawing process, the plotter returns to the home position, at which point the entire computation iterates, returning to the machine vision stage.

**Encouraging Play**

This relatively low-level implementation is designed specifically to address the four components of the computational play framework. First, the machine is **multimodal**. By forcing the machine to reinterpret and reconceptualize its drawing with each iteration, the process pivots from likely defaulting to what it already knows to likely seeing alternative possibilities. Surprising new configurations are not avoided, but pursued.

Second, the machine’s behavior is driven by well-defined rules. These drawing rules are **generative**, as they specify all possible moves without pre-assigning value.

Third, the entire process is broken down into an **iterative** sequence of seeing, choosing, and moving. Each iteration forces the subject to step back and analyze the content anew, thereby avoiding the bias inherent in sticking with one perspective and opening the exploration up to new insight.

And finally, the machine is **autotelic**. It considers both the number of available possibilities as well as its own interest level in determining whether or not it will continue. The machine makes its own path through the space of possible moves, which, while entirely unpredictable, is still completely attributable to specific causes.

**RESULTS AND DISCUSSION**

The machine demonstrates a cyclical drawing process that is entirely reminiscent of a designer iteratively constructing a drawing. After using machine vision to recognize the current geometry, the machine cycles quickly through its possible options, before choosing one and proceeding to the plotting stage. All three computational stages blend into one continuous process.
In early drawing experiments, the physical setup of the machine resulted in messy drawings that did not reflect the projected path of the computational process. Instead of quitting, however, the machine handily incorporated the leftover lines, resulting in unpredictable drawings (Figures 15–17). This capability to incorporate emergent values is far from a mistake; coupled with the subjectivity of the machine, this open-ended approach to novelty better approximates the design process.

CONCLUSION

While the machine does demonstrate the observable qualities of computational play (multimodal, generative, iterative behavior), the implementation of autotelism is harder to witness. This characteristic is the most fertile area for modification and development. The machine may choose to continue against all odds, but what should it do next?

Several interesting behaviors have been mentioned as possible products of autotelism. For instance, if the interest level of the machine is high, but the number of possibilities is low, then the machine might erase certain portions of the drawing that it can’t understand, in order to focus on that which it can—or it may choose to start over entirely. The machine may continue a drawing even despite the lack of progress it seems to be making. And of course, if the number of possibilities is high but the interest level is low, then the machine may choose to quit.

Perhaps most interesting, however, is the possibility that the machine may break or change the rules to be able to recognize or create different content. In one possible implementation of this, the rules of the play might themselves be parameterized. The computer would be able to tweak the parameters that define either the shape to be recognized or the transformation to be applied, resulting in entirely different drawing outcomes. For instance, if we parameterize the triangle rotation rule, as is shown in Figure 18, then we might suddenly be able to recognize pentagons instead of triangles [1], draw a triangle at a different angle of rotation than before [2], or even drop the erasing factor altogether and maintain both triangles [3].

Another implementation might involve the computer swapping the left- and right-hand sides of two different rules. If the transformations involved are general enough, the rules could feasibly intermingle into new possibilities. In Figure 19, a triangle inscription rule and a pentagon rotation rule are swapped to create a triangle rotation rule and a pentagon inscription rule.

This machine is a proof-of-concept of the proposed computational play framework. It demonstrates an autonomous, playfully creative drawing process. The machine is meant to provoke discussion as much as it is meant to provide useful technical insight. As AI moves into widespread use in creative design environments, it must begin to employ behaviors demonstrated by creative designers. With enough development, such machines could one day be tasked with design projects of all scales, from architectural details to entire building designs.
Until such a vision is realized, however, creative AI will most likely be used in collaborative settings. Even in collaborative design scenarios, many of the behavioral characteristics of play are maintained. An autonomous subject acting in concert with a human designer could prove to be an excellent way to study the system at close range. Furthermore, such a subject could provide more interesting and novel prompts in a designer's software UI.

Play need not be a high-level, all-encompassing strategy: even non-autonomous tools (especially in design environments) could benefit from the implementation of play behavior.

ACKNOWLEDGEMENTS

This work is part of an ongoing thesis at MIT. The work wouldn't be possible without the oversight of Terry Knight, Stefanie Mueller, and George Stiny. In particular, the theoretical work draws heavily from Stiny’s work on shape grammars, and the physical implementation wouldn’t be possible without the community of makers and hackers behind Makerblock, Grashtopper, Firefly, et al.

REFERENCES


IMAGE CREDITS

All drawings and images by the author.

Scott Penman is a graduate student in the Design and Computation program as well as the Electrical Engineering and Computer Science program at MIT. While his education and formal training is in architecture, he has maintained an interest in all things technological since he was a kid. His interest in both design and tech brought him to MIT, where he is investigating the intersection of design theory, artificial intelligence, and creative drawing practices.