Do You Like This Art I Made You: Introducing Techne, A Creative Artbot Commune

Johnathan Pagnutti, Kate Compton, Jim Whitehead
University of California, Santa Cruz
1156 High St
Santa Cruz, CA, 95064
(831) 459-0111
jpagnutt,kcompton,ejw@ucsc.edu

ABSTRACT

Artbots have found limited fame—from wordplay bots on Twitter with thousands of followers, to bots made to show off the latest machine learning and AI techniques. Such bots may sometimes interact with human input, but almost never interact with other bots. When they do (Madrigal 2014), they do not learn from the other bots. This is a far cry from how real creative communities work, where practitioners learn from some of their colleagues and can inform the creative processes of others. What would a community of artbots look like if they could communicate and learn like a community of human artists?

In this paper, we present Techne, an agent-based bot-communication platform designed to give artbots the ability to communicate and share artistic practices with each other. We use the grammar-based generation tool Tracery to create art-generating grammars for the bots, and these grammars also serve as a "lingua franca" which allows any bot to understand (and borrow from) another artbot's generative code. This communication between individual bots enables us to build up the social features we would expect from a "real" artistic community: Techne bots create art to win the approval of other bots in the community, create art that other bots hate but they personally love, and help each other (through sharing art-making "process") reach some artistic ideal. This paper presents the current work done on building this platform, and some of the lessons we've learned from watching the first Techne communities grow and trade art.

KEYWORDS

multi-agent systems, generative methods, computational creativity

INTRODUCTION

Often, when human artists create new artifacts, their work is informed by a larger artistic community of practice. Established communities serve an important role in creativity: they provide known ways of making art and well-vetted examples of good work, and provide critique and support to newly-created works (Csikszentmihalyi 1996). Whether a circle of close friends, a writing club or a formalized artist commune, these social structures to provide critique and feedback are important to the process of making new things.

Proceedings of 1st International Joint Conference of DiGRA and FDG

Part of the field of computational creativity is the practice of applying existing theories of human creativity to computational generators. There are many examples of content-creating generative methods, algorithms that create some new artifact, but these often model creativity as a lone encapsulated "artist in a box" (Kosak 2005). A creative human rarely creates work isolated from the feedback or inspiration of their community, so algorithmic artists may benefit from creative community as well.

This paper goes over work on Techne, a communication protocol and platform for the development of creative agent AI that engages in these social operations. In Techne, each agent not only has its own generator, but its own heuristic for evaluating artwork. This isn't unique to multi-agent creative systems—however the heuristics in Techne are completely separate from the generation process. This lets Techne bots 'try on' different generative strategies until they start creating art they like.

Techne bots can also critique artifacts generated by another agent in the commune, and furthermore, Techne bots can use that critique to change how they generate new art, and in future work, how their heuristic functions. We've started by building lightweight prototype of Techne using Tracery. In this work, we uncovered shared art that was surprising and delightful.

RELATED WORK

Creative agent AI is not a new field. Agents have been used in generative methods before, from creating new gridfonts (Hofstadter and McGraw 1993) to Mario levels (Kerssemakers et al. 2012). In these contexts, however, a set of agents work together to model a single generative process. In Techne, each agent has its own way to generate new artifacts—the agents aren't explicitly working together to come out with a single master artwork.

Agent networks, where each agent has its own generative process, is also not entirely novel. For example, (Saunders and Gero 2001) describes a multi-agent system, where each agent uses a genetic algorithm to create new artwork. Agents share art according to the "law of novelty", which is a drive to innovate in their own artwork, and share artwork that they find innovative.

In addition, (Gabora 1995) describes an agent network that uses a process of simulation, imitation, and innovation to show how successful dance behaviours spread quickly through the agent population. Rather than building a network to look at a how a particular theory of creativity or cultural idea transmission performs, Techne aims to develop a platform for creative agent work. This platform is focused on being extensible and accessible for wide, non-technical audience.

This puts this work very similar to (Corneli and Jordanous 2015), which details a design specification of the writing workshop process for creative agent AI. Despite pulling much of the theory from writing workshops, the system itself is designed to be domain-agnostic. However, in the 5-step communication protocol detailed, agents take several steps that are meta-data focused (asking questions about critique, for example). This is different than Techne's approach, which is grounded in communicating through the art the agent creates.

Finally, (Cook and Colton 2015) describes an evolutionary system that generates code snippets that model preferences in a bot. A preference, in this case, is a comparison between two objects of the same class. The code snippets generated are measured in terms of specificity, transitive consistency, reflexivity and agreement. The preferences that are created here are entirely different from a system that might generate particular artifacts– all it knows about are the artifacts at hand. Techne takes a similar approach with separating a bot's generator and evaluator.

GUIDING PRINCEPLES

Heterogeneous Bots

Techne is about building an ecosystem of heterogeneous bots. A single bot can be composed of many discrete, different parts, and each of these parts communicates through a set of common sockets. We tend to think of our bots like legos—there might be a whole set of generator bricks, a whole set of evaluator bricks, a whole set of sharing bricks, and Techne is the thing that lets these various bricks work together. We propose to do this by defining a common set of operations between bots and bot parts.

It is also important to realize that each brick can be realized in many different ways. From simple noise functions or particle systems to complex genetic programming or machine learning, a generator brick or an evaluator brick is treated like any other brick of the same type. Techne bots can even have components that only partially utilize Techne, bots do not need to implement the entire protocol.

Duck Typing

Because bots are heterogeneous, no one part of a bot can be certain how another part functions, or if a bot even has particular part. Therefore, bot components are not strongly typed. Instead, we take the duck typing approach: if it looks like a generator and acts like a generator, it's probably a generator.

Tied in with using duck typing to identify components, Techne bot operations should fail gracefully. When a bot component doesn't understand how to process something, the bot shouldn't stop operating. In our current implementation, bots silently drop information they don't know how to process, and no bot components expect any other component to operate in a particular way. This often means a round of data cleaning before and after anything external to a particular bot component is called upon.

'You Communicate Through What You Make'

Techne is about building artbots, therefore, all communication should be focused on art. All messaging passing between bots (and even within bot components) should either be a representation of an artwork, or something 'one step away' from an artwork (a critique, for example). This is mostly a scoping decision on our part—we do not aim to build a general purpose way for bots to communicate. This restriction helps us focus on what we find important: bots being able to share and evaluating art made by bots.

BOT DESIGN

Bot Generator

Although not a part of Techne as a communication protocol, in order to create several bots and have them communicate with each other, we needed to make a bot generator. The

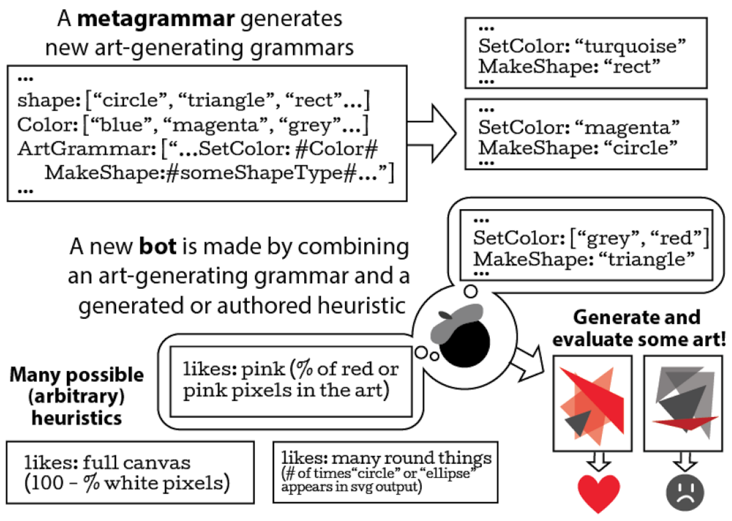

A metagrammar generates new art-generating grammars
...
shape: ["circle", "triangle", "rect"...]
Color: ["blue", "magenta", "grey"...]
ArtGrammar: ["...SetColor: #Color# MakeShape:#someShapeType#..."]
...
SetColor: "turquoise" MakeShape: "rect"
SetColor: "magenta" MakeShape: "circle"
A new bot is made by combining an art-generating grammar and a generated or authored heuristic
SetColor: ["grey", "red"] MakeShape: "triangle"
Many possible (arbitrary) heuristics
likes: pink (% of red or pink pixels in the art)
likes: full canvas (100 - % white pixels)
likes: many round things (# of times "circle" or "ellipse" appears in svg output)
Generate and evaluate some art!

Figure 1: Techne Bot Composition

core part of this was a generator that made art generators, as shown in Figure 1. This lent cohesion to the various generators bots were armed with—although the art that they made might look different, it all came from a basic set of operations. We can, for experimental and showcase reasons, also control how much generator variety there is between artbots in a commune. This 'meta-cohesion' in bot generators will be relaxed more and more as we develop Techne further and further. Our goal is to completely relax it, as per our design principle of heterogeneous bots.

Currently, this is realized using a metagrammar that creates SVG Tracery grammars, like the kind in Figure 2. Tracery is a JavaScript, context-free textual grammar engine that allows for specifying grammars rules as a JSON object in a simple syntax.

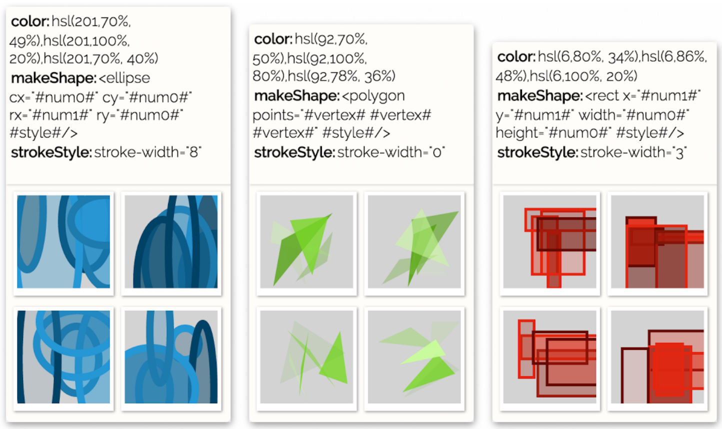
Bots Make Art

Each Techne bot has the potential to make art. Importantly, the way a bot makes art is completely separate from how it evaluates art. Additionally, a bot should be able to change and tweak how it makes new art. Generators should not be frozen in time, but instead should allow tweaks and changes based on author-specified impetus.

For our bots, each generator is a Tracery SVG grammar, as shown in Figure 2. Each individual bot's grammar might not use the entire alphabet of potential symbols or the entire library of potential rules at any one given time, but because the grammar is specified in an updatable JSON format, a bot can tweak it, adding or subtracting symbols to the grammar.

Bots Evaluate Art

Each bot also has an evaluator on what makes good art. How this evaluator is built (for example, what heuristic it uses) is up to the bot creator; however, it should be completely separate from the generator. Because Techne bot generators should be mutable in response to internal and external pressure, an evaluator that is baked into how the generator works may fail if the generator changes. Additionally, evaluators may be called on to try and evaluate another bot's art.



```
color: hsl(201,70%, 49%),hsl(201,100%, 20%),hsl(201,70%, 40%)
makeShape: <ellipse cx="#num0#" cy="#num0#" rx="#num1#" ry="#num0#" #style#/>
strokeStyle: stroke-width="8"

color: hsl(92,70%, 50%),hsl(92,100%, 80%),hsl(92,78%, 36%)
makeShape: <polygon points="#vertex# #vertex# #vertex#" #style#/>
strokeStyle: stroke-width="0"

color: hsl(6,80%, 34%),hsl(6,86%, 48%),hsl(6,100%, 20%)
makeShape: <rect x="#num1#" y="#num1#" width="#num0#" height="#num0#" #style#/>
strokeStyle: stroke-width="3"
```

Figure 2: A simplified version of the Tracery SVG grammar.

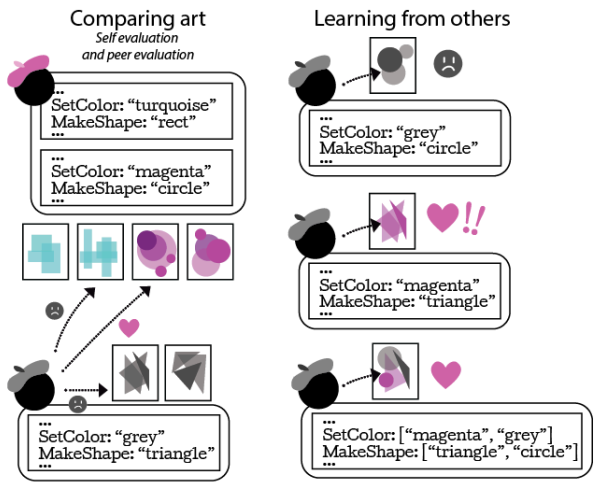


Figure 3: Techne bots evaluate art. The gray hat bot really likes the art the pink hat bot makes. It also doesn't like any of its art. By permuting its own grammar, or learning about potential symbols from another bot, it may start generating art that it likes.

Our system has not completely divorced evaluation from generation. When a new Techne bot is created, it is given two symbols it looks for in a Tracery SVG grammar. An example is provided in Figure 2. These symbols come from the generic alphabet of potential symbols, and may or may not be actually present in the bot's own grammar. When a bot needs to evaluate an art, it does a search through the Tracery trace representation of that art, and sums up all the times it found a symbol it likes.

The two symbols we give to bots come from the same two symbols we allow to vary from grammar to grammar in our bots—shape and color. Because the symbol the bot is looking for may not be present in its own art generation grammar, the bot may generate art it hates, and may only be able to generate art it hates. But it can learn.

Bots Can Try Out Other Ways of Making Art

Because of mutability of generators, Techne bots can try out different ways of making art. Because any generators are separate from any evaluators, bots updating their own generators from their evaluators is essentially a blind search to find art that they like. This lets a bot discover how to generate to its heuristic, but also discover new ways of making art from other bots.

For our bots, because we have the whole alphabet of symbols and potential rule expansions to use, we can have bots randomly try different symbols until their heuristic starts to give them a "good" score. However, if we restrict certain bots to certain parts of the space, and then give them a heuristic for part of the space they have no way of accessing, then they're forced to learn from other bots through social exchanges of art and critique.

COMMUNITY DESIGN

The other important part about Techne bots is that they share the art they make, and critique other bot's art. Techne bots can encode social state, if they want, but Techne itself doesn't define any sort of bot state. This lets bot-makers be flexible, and allows for bots that keep deep histories of art given to them by other bots in a network able to interact with someone's first attempt in botting.

Give Art

Techne bots can give art to other bots. When giving an art, each bot also provides some way for another bot to respond back with a critique, from an index in a list to an IP address of a cloud-hosted server. Bots can use the structure of the artwork, ala duck typing, to see if they can understand it enough to critique it.

For our Tracery-powered Techne bots, the art provided is a trace through the Tracery SVG grammar. This lets our bots quickly realize they can respond to the art, and form a response as a critique.

Give Critique

Critique, in Techne terms, has two parts—a score, and something for the original artist to use to update their own generator and/or evaluator. As stated earlier, Techne bot evaluators have some heuristic that can score art. This gets returned as the score part of the critique.

The next bit is harder, and requires a unified way to talk about artworks of any particular class. Development of such a unified language is still ongoing, but any full specification of Techne would have some way for any bot to give critique that any bots who make similar kinds of art can understand.

A bot can then use a critique to tweak their generator and/or evaluator. This lets bots authenticate to communities of other bots, but always dislike the art they make (using critique to just update a generator). This also lets bots update their evaluator without updating their generator; a bot can fall in love with another bot's art but never figure out how to replicate the other bot's art itself.

Critique is currently implemented by having the bot share one of its symbols that it's looking for in a SVG Tracery grammar trace. This is highly domain dependent, and we'd like to move to a more general way of formulating critique.

Update Generator / Evaluator

Having a bot be able to update its generator / evaluator is a key part in how Techne operates. Bots can critique their own art, and use it to improve how they generate art in the future, although they don't have to listen to anything the critique says. The key part of an update operation is to take the actionable part of the critique (the second part, not the score), and change the generator or evaluator in some way to more closely match the kind of art that the critiquing bot would give a high score too.

This operation is complex, and in our current Techne prototype, we've only implemented generator updates. A bot can update its generator by incorporating the rule from the actionable half of a critique. Currently bots do this by swapping out a potential expansion for the rule with the one mentioned in the critique.

CURRENT INSTALLATION

We built two versions of artbot communes that follow the principles outlined here. The first is a local commune, where all bots are stored in memory. These bots create and swap art in a more controlled environment, and this commune was useful for prototyping out bot strategies, and seeing how state changed the bot art created.

The second is a long running commune, where each bot is an individual Node.js process. These bots are hosted on a cloud server, and communicate over the Internet [1]. Looking at these two communes we can see surprising art (one commune settled on pink being the only acceptable color for art, regardless of individual preferences, because of a pair of bots that very negatively voted down all non-pink art), bots sharing new techniques for making art, and some current limitations.

Art Evolution

One of the important aspects is that artbots change how they make art to fit either their own heuristic, or another bot's heuristic with a critique. Figure 4 shows a bot trying out new ways to make art until it makes art it likes, and also shows a bot creating art that it doesn't particularly like, but the other bot in its commune is giving higher and higher marks for.

FUTURE WORK

We would like to add an authoring interface for Techne bots. Techne leaves a lot up to a bot maker, but enabling people to write simple Techne bots quickly could get more people into botting. Such an interface could allow for people to see what kinds of output their bot will make, before they let it out into the wild and it starts to update its generators and evaluators.

We have found Tracery to be immensely powerful in developing Techne this far. We aim to make a 'Techne pipeline' to automatically create Techne bots from sets of artifacts where the artifact can be represented as a tree. By creating a minimum spanning super tree form the set of tree representations of the artifacts, then building a Tracery grammar from that tree gives us most of the machinery required. It becomes a case of finding nodes in that tree to vary from bot to bot, and using those nodes to create evaluators. Work is currently undergoing to implement this pipeline for the Berkley Annotated Recipe Corpus (Tasse and Smith 2008).

1. Check out how that commune is doing here: http://45.55.28.224:8100/

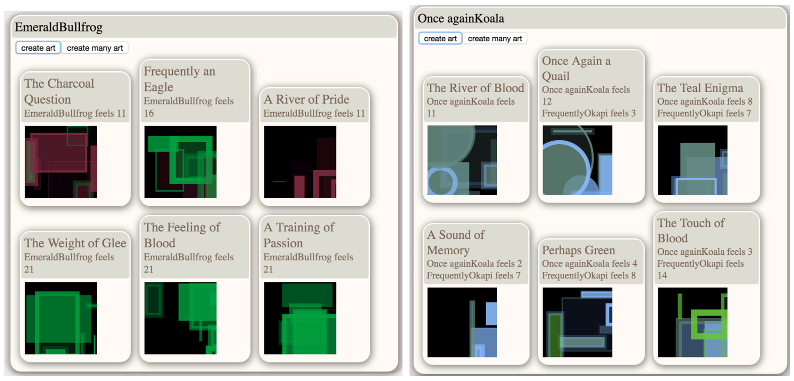

EmeraldBullfrog
create art
create many art
The Charcoal Question EmeraldBullfrog feels 11
Frequently an Eagle EmeraldBullfrog feels 16
A River of Pride EmeraldBullfrog feels 11
The Weight of Glee EmeraldBullfrog feels 21
The Feeling of Blood EmeraldBullfrog feels 21
A Training of Passion EmeraldBullfrog feels 21
Once againKoala
create art
create many art
The River of Blood Once againKoala feels 11
Once Again a Quail Once againKoala feels 12 FrequentlyOkapi feels 3
The Teal Enigma Once againKoala feels 8 FrequentlyOkapi feels 7
A Sound of Memory Once againKoala feels 2 FrequentlyOkapi feels 7
Perhaps Green Once againKoala feels 4 FrequentlyOkapi feels 8
The Touch of Blood Once againKoala feels 3 FrequentlyOkapi feels 14

Figure 4: Bot making art. The bot experiments with different grammar permutations until it starts really liking the art that it is creating. In a different commune, one bot makes art it dislikes more and more, but the other bot likes.

BIBLIOGRAPHY

Cook, Michael, and Simon Colton. 2015. "Generating Code For Expressing Simple Preferences: Moving On From Hardcoding And Randomness." In *Proceedings of the Sixth International Conference on Computational Creativity June,* 8.

Corneli, Joseph, and Anna Jordanous. 2015. "Implementing feedback in creative systems: A workshop approach." *arXiv preprint arXiv:1505.06850.*

Csikszentmihalyi, Mihaly. 1996. "Flow and the psychology of discovery and invention." *New York: Harper Collins.*

Gabora, Liane. 1995. "Meme and variations: A computational model of cultural evolution." *1993 Lectures in complex systems:* 471–485.

Hofstadter, Douglas, and Gary McGraw. 1993. "Letter spirit: An emergent model of the perception and creation of alphabetic style." Citeseer.

Kerssemakers, Manuel, Jeppe Tuxen, Julian Togelius, and Georgios N Yannakakis. 2012. "A procedural procedural level generator generator." In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on,* 335–341. IEEE.

Kosak, Dave. 2005. *Will Wright Presents Spore... and a New Way to Think About Games,* March. `http://www.gamespy.com/articles/595/595975p1.html`.

Madrigal, Alexis C. 2014. *That Time 2 Bots Were Talking, and Bank of America Butted In,* July. `http://www.theatlantic.com/technology/archive/2014/07/that-time-2-bots-were-talking-and-bank-of-america-butted-in/374023/`.

Saunders, Rob, and John S Gero. 2001. "The digital clockwork muse: A computational model of aesthetic evolution." In *Proceedings of the AISB,* 1:12–21.

Tasse, Dan, and Noah A Smith. 2008. *SOUR CREAM: Toward semantic processing of recipes.* Technical report. Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.