Beyond the Creative Species

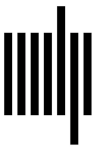
Making Machines That Make Art and Music

By: Oliver Bown

Citation:

Beyond the Creative Species: Making Machines That Make Art and Music
By: Oliver Bown
DOI: 10.7551/mitpress/10913.001.0001
ISBN (electronic): 9780262361750
Publisher: The MIT Press
Published: 2021

The open access edition of this book was made possible by generous funding from University of New South Wales.



The MIT Press

6 Putting Computational Creativity to Work

> Just by rotating this knob, any one of you can produce up to three sonatas per hour. Yet consider how hard it was for your ancestors. They could be creative only be driving themselves into fits of "inspiration"—an unknown form of epilepsy.
> —Yevgeny Zamyatin[1]

While there are many algorithmic methods to choose from to create computationally creative systems, they each have specific limitations and affordances that need to be worked with to get anything done. There are no magic bullets or general purpose solutions, no general intelligence systems. For the moment, AI solutions are highly domain-specific tailored tools with quirks and constraints that influence how they might be used. They may be operated by professional programmers who run custom scripts and output the processes of their generation for others to access, or through one-button-interfaces, simple navigation interfaces, or text-based search bars that an unskilled user can explore. Since machine learning is simultaneously so successful and so complicated, some researchers and companies have set out to make general-purpose creative machine learning toolkits that can be more easily used by end-users. The Runway toolkit is described as "an interface and framework that orchestrates the training, use and deployment of artificial intelligence models in design and creative platforms,"[2] taking an ecosystemic approach where different contributors can provide models and templates that can be used by others. Machine learning interaction design poses complex conflicting objectives. In their overview of machine learning approaches to music generation, Briot and Pachet define four key challenges in designing usable generative music tools:

- Control, such as tonality conformance, maximum number of repeated notes, rhythm
- Structure, versus wandering music without a sense of direction
- Creativity, versus imitation and risk of plagiarism
- Interactivity, versus automated single-step generation[3]

This chapter gives an overview of what people are doing with computationally creative systems, how they are being used in creative projects, and how businesses are beginning to apply computational creativity to commercial projects.

Chapter 5 discussed many examples of computational creativity systems, focusing largely on the algorithms themselves. The discussion of these examples inevitably touched on the *use cases* we find emerging around the application of computational creativity to real creative tasks. This chapter turns to this topic proper and accordingly draws in another academic discipline: the study and design of technology use and interaction, which includes the fields of human computer interaction (HCI)[4] and interaction design (often abbreviated IxD).[5]

I will leave a more thorough discussion of the methodologies and theoretical foundations of these subjects and their application to computational creativity to the following chapter, which discusses evaluation. However, it is important to mention that in these fields the subject of distributed agency, discussed in chapter 3, is as important as it is in the social sciences, since questions of where actions originate, how intentions are formed, and what systems or networks can be identified as beneficiaries of various interactions are central to understanding the design of interactions with technology. Lucy Suchman[6] pioneered the application of such thought to design in the 1980s, proposing "situated action" as an alternative to intention, whereby action involves complex and collaborative interplay between people and things, noting that "plans are inherently vague." This work has been developed in the intervening period by many others. Werner Rammert[7] develops the thesis that the role of machines in networks of distributed agency already has tangible effects, and Susanne Bødker[8] identifies a "third wave" of HCI thinking and research, in which greater attention is paid to the emergent nature of interaction.

For now, however, we immediately pick up from the last chapter's survey of computational creativity algorithms to look into how these are being put to work in practice.

Approaching Interaction with Computationally Creative Systems: Interaction Metaphors and the User's Understanding

Interacting with machines and other objects has for a long time been centered largely on acts of direct manipulation. From can openers to cars (older cars, at least), each part of the system has a traceable structure or behavior that you can directly control. Computers, as the logical progression of a shrinking, encapsulation, and complexification of machine behaviors, have brought us to a point where this is no longer true. We have no idea what most machines are doing underneath their shells, and little hope of intuitively working it out by tinkering. Modern machines are becoming in interactive terms closer to biological organisms, which are also doing complex things under the hood that we cannot directly manipulate. The philosopher Dan Dennett[9] has a useful way of talking about such understandings in terms of the different "stances" we take toward different types of object. As far as most of the world goes—that is, the abiotic, non-human-made world—a good understanding of physics, and thus a "physical stance," goes a long way. Humans are good throwers, and dogs are good catchers: both have the cognitive machinery to plan and predict the movement of a ball through the air.

But an understanding of physics does little to help predict the movement of the dog (except when it too is flying through the air). For this we apply a different stance, where animals are understood in terms of goals, perception, and intelligence. We know we can attract the dog with food, and we can train it. We think about whether it is smart enough to open the fridge door or find its way home. With other humans, we are highly adept at modeling the behavior of others, enabling forms of trust and deception not known elsewhere in the animal world. This stance Dennett calls the "intentional stance." The human-made world, according to Dennett, invites a third stance. When wondering what a certain button does on a pocket calculator the logic is to approach it as something that has been designed with a function in mind. We expect its behavior to be predictable and thus usable, and we apply the "design stance." As systems become endowed with increasing levels of computational intelligence, the design stance potentially becomes fragmented and the intentional stance may come into play; confusion or a sense of either the profound or the uncanny may ensue.

This is discussed by authors such as Philip Auslander, and Hollington and Kyprianou (as quoted by Auslander), who remark that an uncanny experience

> occurs when animate and inanimate objects become confused, when objects behave in a way which imitate life, and thus blur the cultural, psychological and material boundaries between life and death, leading to what [Ernst] Jentsch called "Intellectual Uncertainty"—that things appear not to be what they are, and as such our reasoning may need re-structuring to make sense of the phenomenon.[10]

Despite the rapid increase in complexity and opacity in human-made computational systems, we still frequently design systems so that anyone using them has a direct manipulation relationship with the system. I am directly typing characters into a document (although it is occasionally correcting me); a graphic designer drags, drops, and draws onto a virtual canvas; a composer places notes and adjusts intensities on a timeline. Accordingly, good design practice is predicated on the idea that a user must be able to maintain a good understanding of what the system is doing and the system must support them to intuitively achieve goals. If they are ever confused about what is going on or how they can manipulate an object then the system has failed. As we saw in chapter 3, the idea of affordances[11] has been central in helping designers frame both how to integrate heterogeneous components into efficient designs and how to communicate the intended use of something to a user. Designers also talk about constraints and mappings, things that guide the user or reveal intuitive relationships between entities.

In chapter 3, I described the concept of affordances as providing a way for us to think about inanimate objects directing the behavior of smart, intentional users, balancing the asymmetry of agency relations between people and objects, and I considered how material engagement provided a model of co-creative exploration even when the objects involved were wholly passive. The design of such system affordances underlies the notion of usability, which is generally associated with task performance and its measurement. A typical way designers understand usability is through personas and user scenarios. A persona is a portrait of a typical user, detailing their qualities and goals. For example, Instagram users can be classified according to different personas—the professional photographer, the teenager, the amateur artisan. A user scenario describes a specific situation in which the user must interact with an artifact or piece of software, detailing

what they are trying to do, what they know, what the particular demands are, such as time constraints, or the potential for distraction.

An example in computational creativity is the work of Aengus Martin[12] examining the use of a system for automating the performance of arrangement-level musical decisions (the arrangement of individual musical elements—drum patterns, chord sequences, bass lines, and so on—into entire compositions). In Martin's case, the system is trained not from a large data set of existing music, as is commonly the case, but from the user's own live arrangements of a musical work. The aim of the system is to accurately model the user's arrangement style so that it can produce new variations of that style, exhibiting an understanding of the structural properties of the user's arrangements. The system is incorporated into the Ableton Live digital audio workstation so that it can be used in a context that is natural to digital musicians. It uses variable order Markov models to record transition probabilities for each individual track, but also a constraint solver system to discover the rules underlying the relationship *between* tracks. The latter system's job is to identify if, for example, certain groupings of tracks always played together or never played together, and override the Markov processes where one of the rules it had learned was being broken.

The important issue with such a system is that it doesn't work perfectly with whatever data it is given to learn. Its learning performance will vary greatly in different musical contexts, and a lot of the work of doing good machine learning, even in the age of sophisticated deep learning algorithms, lies in tweaking the learning system to suit the context. Martin's system's performance depends on a number of parameter settings, and the user is in a position to vastly improve performance by guiding the learning, for example by indicating where the system is likely to find relevant association rules, a process known as feature selection. Thus a machine learning system here is seen like any other piece of software in that it must be operated by a user to work effectively, and the user may need to learn how to use the system effectively; the system isn't necessarily easy to use. Because it is a generative music system, the user must also audition the results, which may take a long time. These considerations begin to paint a picture of user experience and usability in the world of computationally creative systems.

Martin's research then considers what interface is needed for effective user control of feature selection. A default design would involve simply presenting all of the system parameters via various widgets, which the user

can manipulate. This could be considered the expert interface. Users would need to invest in reading documentation, following tutorials, and experimenting in order to gain an understanding of what each control does. Alternatively, one could aim to reduce and conceal these issues as best as possible, for example by providing a set of presets with clear user contexts (such as the genre-specific EQ settings found in car radios; rock, pop, classical, etc.). They could provide a wizard-type interface that asked the user a series of questions sequentially. These various interface options offer different levels of detail, rapid access to results, intrusion into the user's workflow, and so on. Examining users working with different prototype manifestations of the interface can reveal which strategies might be best for this particular system. Rather than actually programming all of these candidate interfaces, a Wizard of Oz approach can be used in user studies, where the developers are acting behind the scenes to fill in for the software where it is not fully implemented.

With regard to auditioning the results, Martin found that, predictably, a user would spend less time at the early stages of listening through different candidate agents (an agent is a specific instance of a system for generating the arrangement) and increasingly more time as they narrowed in on agents whose behavior they liked. This can be explained by the fact that it is very quick to pick up when the system is doing things that are clearly wrong, and as the system improves it takes more time to spot issues. However, it could also be that at an early stage the user is in a more exploratory mode, quickly looking through different options, and then later narrowing in on what they think is their preferred behavior. In this latter case, the system needn't get better for this increased attention over time to play out.

Dahlstedt's work on interactive genetic algorithms for sound synthesis,[13] discussed in chapter 5, embodied in the software for the commercial Clavia Nord Modular G2 synthesizer, investigates similar usability issues. In the MutaSynth software, users can interactively evolve synthesizer sounds; given a certain arrangement of oscillators and effects, all of the numerical parameters (the knobs and sliders controlling each component) are compiled into a genotype which can be interactively evolved. Dahlstedt looked at user interface issues such as the ability to remember individual sound configurations or to return to previous configurations. He developed visual icons that represented the genetic code of each synthesizer configuration, so that users had a visual cue to represent the sound. He also provided tools

to allow users to easily navigate the search space and store and retrieve synthesizer configurations as they searched. Since this work was deployed in a commercial software tool, Dahlstedt's user research was based largely on responses from users through online feedback, with users identifying the system as effective for rapid undirected search.

These examples continue the paradigm of a user operating a system, albeit a sophisticated system with domain-specific intelligence. A serious goal for computational creativity, and one that is becoming ever present with AI-powered conversational user interfaces, is to reconfigure that interaction into one that better resembles forms of human-to-human interaction. This is a theme that has a lineage in earlier thinking in cybernetics, courtesy of the cybernetician artist Gordon Pask. As documented by Usman Hacque,[14] Pask, in the spirit of the cybernetic revolution, trod a richly multidisciplinary path, creating interactive artworks that communicated and explored. His Musicolor system, for example, was a light artwork that responded to the sound of a live musician, demonstrating a basic understanding of the musical signal and mimicking human traits such as expressing boredom.

More recently, the creativity researcher Todd Lubart[15] proposed four metaphors based on human roles that elaborate on ways in which the computer might act as a collaborator: computer as nanny, computer as pen pal, computer as coach, and computer as colleague. The nanny metaphor uses a slightly odd terminology—I would suggest that "manager" or "facilitator" might be clearer. It refers to the activity of structuring creative processes, such as brainstorming, so that the human can get more out of their creativity, so to speak. The pen pal metaphor refers to something similar in a group setting, with human user and computer iterating through design ideas, with the focus of this interaction being the question of how creative ideas are represented. The coach metaphor refers to the use of the computer supporting an individual to use creativity techniques that stimulate ideas. It is the computer as colleague metaphor, of course, that alludes to a more creatively symmetrical, partnership between user and system, with all of the issues of cognition and communication that this entails. A pertinent question with all of these metaphors is how they might be *partially* realized in computationally creative systems and, if so, to what extent such a metaphor holds. Ultimately, existing metaphors of interaction can be subsumed by new design concepts once the field of practice develops its own

directions, and these metaphors can be seen as initial provocations. Other contributors have suggested similar metaphors of human roles. In his work, Lomas[16] refers to the computer as an assistant, which he in fact categorizes as being devoid of any creative autonomy (perhaps akin to Lubart's "nanny"), with the potential through learning and evolution to advance to the status of a collaborator. McCormack refers to the notion of the computer as an equal partner and "first class citizen" in the creative process.

Computational Creativity Activities

A good way to think about the application of computational creativity in the hands of users is to identify the distinct activities that take place during computational creativity algorithm use.[17] I therefore outline nine distinct activities that a person working with a computational creativity system might engage in: selecting algorithms; selecting representations and data formats; manipulating algorithms; generating outputs; reviewing and analyzing outputs; selecting and giving feedback on outputs; feeding source artifacts into a system; specifying goals and constraints; and manipulating outputs.[18] All of these activities can be seen elsewhere besides in computational creativity use cases, but each has a more specific definition and set of design concerns in the computational creativity context.

Note that each of these perspectives conforms to a standard human user perspective, with the human strictly in charge and driving the process. We could consider these interaction scenarios, for example, in terms of Norman's (very much single-user-centric) four stages of interaction: forming an intention, selecting an action, executing the action, and evaluating the outcome.[19]

Selecting Algorithms

Any computationally creative task begins with the selection of algorithms or tools to complete the job. It is worth noting that the selection of the algorithm is itself a task performed by an operator, possibly a creative practitioner, which significantly affects the type of output produced. Whether the practitioner is hand-coding the system themselves or using an out-of-the-box web service, they must make a choice about which tools will best serve their needs. This will depend heavily, perhaps almost exclusively, on what has been demonstrably achieved by others in the past. In this way, algorithm selection is a highly cultural activity, and the concept of a

"community of practice" applies equally well to computational creativity practice as to any other area of creative activity, with its associated schools and genres. Students learning computer music have learned about Markov models for decades and their early application in pioneering works such as the compositions of Iannis Xenakis.[20] They have applied them in their own practice and passed them on to their students. One redefines one's goals based on the observation of others around them, as well as hands-on experience. Competing algorithms may need to signal their value with respect to current knowledge about the state of the art, and in the computational creativity domain this may come about through creative trends as much as scientific evidence. The affordances of such algorithms likewise come to be understood in communities of practice. For example, neural network approaches have proven themselves as effective mash-up tools but are not so well proven as strong ideation tools; but that's not to say a community of practice might not emerge around this use case in the future.

In the increasingly hybrid methods of algorithmic generation, artists may also need to select algorithmic combinations or other ways of constraining generation. We previously saw examples of constraints applied to music sequence generation. Alexandre Papadopoulos, Pierre Roy, and François Pachet[21] explore ways to constrain a Markovian music generation process to avoid plagiarizing content in the original dataset, and in cases such as this the user may be involved in actively configuring algorithmic combinations according to their needs.

It is also possible that the selection of algorithms is itself automated. As we expect the world of computational creativity to be fragmented into many domain-specific tools, meta-tools that help choose between them or that distribute tasks to multiple worker algorithms are likely to be an important part of the mix. Standards that support swappability between algorithms will enhance how easily this can be done (as can be seen in practice in the Runway platform), and mutually add to the ease of adoption for all such algorithms.

Selecting Representations and Data Formats

Machine learning and evolutionary computing researchers put great effort into exploring the ways that the data they are working with are represented. The representation of the data fed into a machine learning model or used as a genotype in an evolutionary search has a significant impact on the outcomes of the generation. Making decisions about data representations is

perhaps one of the more obscure ways in which an end-user might interact with a generation process. Largely, the impact of the data representation is hard to predict and requires experimentation. However, there are some relevant general issues. One is the question of the granularity of the data. In music, a successful approach is to sequence together larger readymade units from a database of existing short melodic expressions,[22] rather than trying to create sequences from the ground up from individual notes, and similar representations exist in other domains. This "unit selection" approach has the obvious advantage that the raw ingredients of the generated sequences are themselves quality phrases (e.g., extracted from the original database), and the goal of generation is to sequence them together. While the promise of strong deep learning and related deep algorithms is that they can work from the lowest-level raw ingredients—pixels, waveforms, or individual musical notes—inferring multiple stages of structure generation, such modular methods can lead to practical improvements in output, generation time, and variability. This may be a user's preference, possibly even giving them precise control over the seed set of primitives used in generation (e.g., using their own database of musical licks to mix together). There are many other more nuanced issues of data representation. Some might be reduced to control parameters that a user can manipulate to influence the generation. Largely, however, the best data representations are likely to become common standards and users would become trained in their use, just as musicians are familiar with the MIDI protocol or piano-roll notation.

Manipulating Algorithms

All algorithms have parameters that can be manipulated that alter the way they perform. For example, Markov models have an *order* that indicates how many steps backwards the model looks when making predictions: variable-order Markov models exist, but still have parameters such as *maximum order*. Such parameters may be reduced to a simple GUI of knobs, menus, and sliders that can be tweaked graphically. In the MutaSynth software, discussed in chapter 5,[23] one could directly interact with an interface that controlled the breeding of new synthesizer sounds, specifying algorithmic parameters such as mutation rate. In many of the online examples of work using Google's Magenta machine learning tools,[24] interface elements are offered that control the parameters of sequence generators, such as a neural network's *temperature*, or set model parameters prior to training, such as

number of layers. These control points may be more or less intuitive to the user, who may have more or less technical understanding of the system. They will also inevitably be often highly specific to the algorithm in question. In all cases, someone—either the system programmer or a user—will inevitably have to configure the algorithm to work as desired. More powerful algorithms may reduce this need but are far from eradicating it.

Often the manipulation of algorithms needs to be more open-ended than a simple GUI will allow, requiring a programming environment to write scripts or programs that direct the process. For now, therefore, it is still predominantly those who can code who have access to the operation of computational creativity systems. Powerful open-source machine learning and evolutionary computing libraries allow creative practitioners who code to construct arbitrary configurations and variants on existing algorithms. This includes chaining different processes together or nesting one process inside another: for example, a system that uses evolutionary search, with a fitness function defined by a trained neural network that performs object recognition. Configuration of such sorts can also be done using simpler data-flow tools or graphical patching environments. The FloWr framework[25] allows users to plug together small data processing modules to create simple computationally creative algorithms by hand. An important factor here is to find suitable data formats, communication protocols and levels of abstraction that enable a modular approach in which users can easily configure hybrid algorithms.

Again, the manipulation of algorithms can be automated in meta-processes. In some of the scenarios we've looked at, we have seen layered systems, where one process configures another, as in the use of evolutionary algorithms to optimize neural network configurations.[26]

Generating Outputs

Perhaps the most ubiquitous stage in a computational creativity workflow is the running of a generative algorithm to create outputs. Typically this will be an iterative process involving generating multiple outputs that the user may select from or possibly give feedback on. At one extreme we have the ideal of the casual creator,[27] in which the space of possible outputs is highly constrained and generally benign (a majority of points in the space looks or sounds nice). Here the search of the space itself is of less importance. Anything goes. The most extreme cases are those like MIT Media Lab's

generative logo, where any combination is as good as any other—it is the collective effect that is of interest. We also have examples such as the control of a GAN's latent space,[28] where generation of outputs might be directly controlled by some intuitive, smooth user interface; perhaps the user can precisely specify the properties of the thing they want generated, even if they don't know how the outcome will turn out. Memo Akten, Rebecca Fiebrink, and Mick Grierson[29] discuss interface designs that allow a user to design trajectories through an image-generating GAN latent space to produce time-based media. In other cases, such as sequential music or script generation by neural networks, the output would typically be seeded or controlled by some input, possibly a random input, or possibly some source material that should steer the output. Inputs to an algorithm may also be presented in other ways, such as the joke-generating system STANDUP, which allows themes or words to be included in the joke.[30] At the other extreme lies the domain of interactive genetic algorithms; each candidate is a simple mutation or recombination of elements from the previous generation, and the search proceeds in a gradualist manner, the user giving feedback to the system in the form of selections rather than specifying operation parameters.

Important factors in the interaction design experience of generating outputs include: the speed at which the generation can be performed; the speed with which a user can evaluate candidates, or more generally the effort involved in that evaluation; the overall success rate of the algorithm; and how the user can control the generation process or give feedback on the results in further iterations. Different use cases also demand different types of output set. An ideation task may require high diversity, while the production of a final product may require greater convergence on a specific outcome with tighter constraints.

In his own practice-based work, Lomas[31] identifies four modes of exploration: initial exploration, in which the user explores and gains a cursory understanding of the space; secondary exploration, in which the user still engages in a broad search, but better directed by goals derived from an understanding of the space; refined focus, in which the user focuses their search on a narrower space and more specific goals; and looking for novelty, in which the user returns to seeing what novel outcomes lie beyond the spaces they have come to know.

It is also important to note that a user may access generated outputs without having generated them themselves but may still select or give feedback

on them (actively or passively). For example, in distributed interactive evolutionary systems like Picbreeder, DarwinTunes[32] and Electric Sheep,[33] the user may or may not be a participant in the generative process but still accesses the outputs of that process. In Steffan Ianigro's distributed genetic algorithm for generating sounds, Plecto,[34] an algorithm pre-generates many prototypes using novelty search, that a user can then search through using a more manual browsing process, sorting the outputs by properties and sifting through them.

Reviewing and Analyzing Outputs

The evaluation of generative outputs often simply means looking at (or listening to or reading) those outputs. Review and analysis factors can be critical in interaction design when, as is likely to be the case, the system does not consistently and immediately produce the desired output. A system's success rate might be relatively low, but it can still be useful if the work involved in identifying good outputs is manageable. Again interactive genetic algorithms are an extreme case where the user's repeated feedback is the means by which the next round of outputs is generated. In all cases, outputs need to be reviewed, but different use cases will demand different approaches to that review, from a rapid skim through results to a more considered analysis.

It is also possible that algorithmic approaches can be used to support the evaluation of outputs, without going so far as actually automating evaluation. Evaluating generative music or other time-based outputs, or long-form literature, can be time consuming, so basic measures and simple visualization techniques can be used to simplify the results. The context of the evaluation or usage of outputs can also be controlled to better fit workflows or to be most convenient or effective. As described above, passive scenarios involving large audiences can be an effective way to trial lots of candidate outputs. Paid "human computing" methods like Amazon's Mechanical Turk[35] can also be used.

Selecting and Giving Feedback on Outputs

Once a generative process has been run and there are candidate outputs that have been reviewed, several things can happen next. The user might simply select one or more final outputs and end the process, or reject all outputs and iterate the process from scratch. They could also interact with the process at

this stage by performing more fine-tuned selection or feedback. The most common form of fine-tuned feedback is the rating of some or all elements in the output set, or at least sorting them into good and bad categories or otherwise tagging them. In an ambient interaction paradigm (described below), passive measures might be used instead of active measures, such as how long a person spends interacting with a given output. Furthermore, the evaluation of outputs might go deeper into the specific detail of the output, such as identifying exactly which aspects or regions of the output are good, or giving feedback that indicates overall issues with the set of outputs.

It is important to remember that selection and evaluation are not only about thinking in terms of rating goodness but taking a multifaceted response into account. For example, in a distributed search process, someone might identify an output that is not what they are looking for but that is nevertheless of interest for some reason. To take into account such situations, it is better to think of evaluation as being fluid and having multiple possible manifestations and orderings, resulting from an ongoing interaction between users, systems, and their outputs.

Feeding Source Artifacts or Data into a System

In many computationally creative systems, some form of input data is needed. This may take the form of training data for a machine learning algorithm or target outputs for an evolutionary algorithm to match. In more complex scenarios it might take the form of instances with respect to which a system might aim to demonstrate novelty. Input data may be submitted in order to say "match this" or "avoid this." In style transfer, the user blends different aspects derived from two or more different sources. In various training strategies, we might also combine different datasets together in structured ways, such as training a system on a wide training set first and a more specific set later.

An important factor to consider is where the input data comes from. Does the user create it? Do they have access to various corpuses of data? Is the algorithm capable of trawling the web for publicly accessible resources, or does the data come prepackaged in the system, as in pretrained networks, in which case the user may have limited knowledge about what data the system has been trained on (which is increasingly common)? In some cases, such as in live music performance systems, the data might be input in real time. The system may even explore the world itself, for example in

a robotic system that can move around and explore its visual environment, as in the creative robotics work of Petra Gemeinboeck and Rob Saunders, where robots learn patterns in their environment, driving their innate (that is, programmed-in) curiosity. This in turn has significant impacts on the attribution of the output and may constrain what the user can do with the trained system. A trained system might be deemed plagiaristic if it regurgitates outputs from its training set, but that depends on the context. Also important is how easily the data can be organized. How well annotated is it to allow grouping into different categories? Is it in a form that allows different aspects of the data to be presented to the algorithm? Finally, how easy is it to understand how the system has responded to the data? For example, if a machine learning algorithm is failing to reproduce a certain style, is it easy to understand what might be changed in the training set? This impacts how the user can respond and alter the training set or the algorithm itself.

Allowing users to train systems on their own data holds enticing potential. An example of a platform for facilitating this is the Wekinator, created by Rebecca Fiebrink. The Wekinator is named after a machine learning library called Weka, upon which it is built. It allows any user to rapidly associate inputs such as physical gestures with outputs, such as synthesizer parameters. As Fiebrink demonstrates in various studies,[36] "One of the most immediately apparent benefits of using Wekinator to build mappings is the speed and ease with which composers can build a new instrument and modify it." This, importantly, applies to nonprogrammers, expanding access to these tools. She continues with an important observation about how this in turn transforms thinking around music system creation:

> Another critical difference between designing instruments using machine learning and designing instruments by writing code is that composers are able to use their bodies directly in the design process. Instead of reasoning about what sort of movement-sound relationships he might want in an instrument, then deriving a mathematical function that he thinks will facilitate those relationships in a mapping, a composer can simply demonstrate examples of movements and movement-sound pairs that feel and sound right to him.[37]

Specifying Goals and Constraints

With systems that are goal-based or are able to perform some kind of evaluation on their own, it is common that we want to specify goals to the system. In target-based evolutionary search, this means specifying a fitness

function or providing example target behavior. I don't include examples such as training neural networks here, because in this case the data is not used to specify a goal but to feed a learning process (although the difference can be minor depending on the circumstances, and reinforcement learning is an exception).

A common way to describe goals is in terms of constraints, such as properties that the output shouldn't contain, or relations that should pertain in the results. An important question is how goals or constraints can be specified in a user-friendly manner. Graphical user interfaces are again possible for this purpose, but also limited in this respect. We can specify constraints in terms of logical relations between elements in some formal language. Example-based approaches are also already very successful since they are intuitive (as in Fiebrink's observation above). We can expect that natural language-based approaches would be very powerful in this regard, since it is natural to specify goals in terms of descriptive language, and we are beginning to see systems that are successful at mapping between natural language descriptions and cultural products. At the same time, we cannot always expect that rules can neatly be derived from data; such processes may not always be successful.

Another question is exactly how the system responds to user-specified goals, such as how it presents or orders the results. A system can spew many results out and leave the user to deal with this mass of options, or be more prescriptive, offering a small number of results (even if they are not evidently better than others). This relates to factors such as how open a user is to alternatives. If the system presents a single output as being preferred, we may be influenced by that analysis, placing trust in the system (and perhaps inadvertently assigning it greater creative authorship). Multi-objective strategies integrate different objectives to come up with a new, narrower search space. The setting of constraints and the sorting of results can be made iterative in various ways, for example by allowing a user to filter results by iteratively adding new constraints, or correcting others. Such technologies that steer or filter generative production without necessarily converging on specific solutions may become critical to good interaction design.

Manipulating Outputs

Lastly, once a system has generated outputs, a user may want to take over and finish the job themselves, which as we have seen is a likely scenario,

given the limitations of existing algorithms. This requires simply that outputs are generated in, or easily convertible to some common format that a user can manipulate. This cannot always be guaranteed. For example, if the output being generated is a piece of interactive software, possibly involving a neural network or other complex behavior, then this object may not be represented in a very usable form or be easily manipulable by a user. It may remain opaque to the user, presenting potential situations where the generated artifact is nearly but not quite acceptable, with no easy way to better refine it. For example, the outputs created by Picbreeder are complex neural networks that generate bitmap images. The bitmap images are editable in programs like Adobe Photoshop but only at the pixel level; there are no layers or brushstrokes or other underlying objects that make up the image. Meanwhile, the neural networks that did the image generation are effectively black boxes; there is little hope of an artist manually tweaking their contents in a controlled manner.

In other cases, manipulation may be trivial, for example if a system generates MIDI or vector graphics data, which can be read and manipulated in many different programs. Even then, two other consideration still apply: is the generated data still presented in the most ideal way for manipulation? If it is vector graphics data or 3D point data, is it ordered sensibly? Secondly, if the user manipulates the data, can it then be further manipulated by the algorithm? Is there a two-way exchange between human and machine generated artifacts? For example, if a user edits a Picbreeder-generated image in Photoshop, there is no way for that newly edited image to be fed back into Picbreeder for continued evolution. Picbreeder would first need to find the correct neural network configuration that could generate the new image, which may not be possible.

Computational Creativity Interaction Paradigms

For each of the algorithms discussed in chapter 5, any or all of the above activities may be involved and manifest in very different ways. From this discussion of user interaction activities in computational creativity, we can also identify three broad interaction paradigms.[38]

Operation-based interaction: In some cases, a user operates advanced AI or search algorithms largely as they would any other digitally creative tool—setting parameters, loading training data, running a generator, evaluating

the outcomes, iterating, and so on. In music generation, Briot and Pachet[39] discuss the ways in which various algorithms provide "entry points" to their control. For example, "The WaveNet architecture uses conditioning as a way to guide the generation, by adding an additional tag as a conditioning input." Good operation-based interaction requires good design of the interfaces to these algorithmic entry points. Intelligent use of presets and other constraints that simplify the interface may radically improve interface design.

Request-based interaction: In other cases, the system affordances may mark a radical departure from this more typical control-based paradigm, and we may interact with an algorithm via requests for outcomes. This can be described as a service-based or request-based paradigm, which resembles the user experience we find in search engines and natural language interfaces. Direct control of an algorithm is replaced with indirect interaction with the algorithm via the definition of goals and the evaluation of the responses. Consider the Google search bar, or assistants like Siri or Alexa, to be a model for how this might look; the primary interface is a free or constrained text field (or voice equivalent) that spawns a series of results, with which other interface elements can be used to interact (clicking links, setting filters, and so on). By subsuming the direct control of the generative algorithms, a request-based system can potentially be more intuitive and flexible than an operation-based system, but may be harder to understand in terms of potential affordances.

Ambient interaction: In other cases still, the algorithm operates in the background and it is not directly operated or requested by the user. In some of the music performance scenarios we will look at below this is the case: a live musician jamming with a piece of software can take a stance toward that piece of software in which it is human-equivalent (not necessarily humanlike, but performing the same role as a human co-performer). The user does not necessarily control the software or "submit requests" to the software. We could equally describe this as a partner paradigm and see it as an equally radical shift to the request-based paradigm. In other cases, the software may be making suggestions in the background, or filling in or predicting steps in the process, much as a text autocomplete system works.

These concerns lead to other design issues that underlie the usability or user experience for someone operating a computationally creative system: How does the system support playful interaction and divergent goals?

What intuitive interfaces or programming libraries support the programmatic design of behaviors? And how can distributed multiuser creative systems support communication and collaboration between different users? These topics are considered in the following chapter on evaluating computationally creative systems.

Computational Creativity Application Domains

We can also class the application domains of computational creativity into a number of broad categories. In the remainder of this chapter I outline several types of creative use contexts which apply computationally creative systems of varying degrees of sophistication, applying different algorithmic and interaction design paradigms. In one direction, we look at automating relatively routine tasks, such as completing a visual pattern or applying standard audio mix strategies to a piece of music, tasks which are already commonplace. In the other direction, we glance at increasingly anthropomorphic scenarios involving questions around the embedding of systems as participants in cultural contexts, with attributions of creative agency. However, this is far from a smooth, linear spectrum of application domains. For example, performance systems, such as improvising music systems, exist in such a different domain of creative experience and evaluation, focused on real-time collaborative creation, that they shift the type of considerations of agency we take into account. Some application areas, such as distributed evolutionary systems, depend upon, modulate, and will potentially ultimately transform how different creative practitioners collaborate, while for other application areas this is less significant.

Automating Routine Tasks

The most long-standing way in which AI has been applied to digital creative work is in the automation of relatively routine tasks which can empower creative practitioners. These are often not overtly computationally creative, in that they do not directly involve generation of original content, nor evaluation, but they do contribute to a foundational infrastructure that can support the emergence of computationally creative systems. A simple example from music is the ability to perform manipulations on a sound file that require some degree of perceptual intelligence, such as finding event onsets (the moments in an audio recording where a sonic event starts) or

understanding which notes make up a given chord. Imagine a note being struck on the piano in a jazz club, against a backdrop of other instruments, clinking glasses, and chatter. From the sound, you as a human listener are able to identify that a discrete event has taken place. Modern algorithms can do this with relative ease, and variants of this kind of functionality have been in place in commercial computer music production tools for decades. Similarly, in graphic contexts, an equivalent would be the ability to trace the outline of an object in an image.

Music information retrieval—the study of extracting information from musical data such as audio files—offers sophisticated computational methods for extracting events, understanding pitch and harmonic content, separating different sound sources or musical streams, inferring the underlying beat and metrical structure of a piece of rhythmic music, identifying phrase boundaries, recognizing styles and assigning genre classifications, recognizing individual instruments, and building models of the long term structure of music. All of these techniques are in use in some form or other in various creative software products today. Similarly, techniques that allow us to pull apart and manipulate existing audio *as if we were creating it from scratch*—techniques such as additive, granular and concatenative resynthesis—mean that static sound files become dynamic, editable entities. By modeling the underlying components of the sound, we can manipulate the sound in powerful ways. As stated, none of this directly performs the work of computational creativity—most of what I have just described can be characterized as cognitive work that a skilled human could do but that doesn't require creativity—but it provides a set of sophisticated ways in which algorithms are already at work, in use in current creative practice, into which computational creativity methods can tap. The more fluidity that computational intelligence can bring to the digital world, the more easily we can apply computational creativity methods.

Consider as an intermediary example the Sony CRL team's Reflexive Looper,[40] which employs machine intelligence to enhance the usability of the standard looper pedal. The looper pedal is a simple technology in the audio effects domain that is a mainstay for many performing artists, allowing them to create entire tracks by looping and layering elements in real time, particularly by overdubbing loops of themselves playing. The Reflexive Looper applies analysis to work out what parts are being played and when they begin and end. This is a natural extension of the kinds of machine

listening applications we've already seen—the ability of music software to understand instrumentation, key, tempo, meter, and distinct events in audio—working toward higher-level music knowledge such as understanding phrase structure, maintaining a statistical model of style, and thus offering new affordances for how the musical content is manipulated by the user.

Meanwhile, the field of creative AI includes a wide array of novel concepts for the application of AI to creative work. As we incorporate new machine perception capabilities and understanding into software as they become available, new system affordances alter how we approach creative work. Smart object or event recognition, or smart modeling of content, has the immediate effect of facilitating new possibilities for creative production, opening up new ways of working and possibly leading to entirely new genres of creative practice.

Creative AI, for example, enables new types of audio or image synthesis. In concatenative synthesis, or audio mosaicing, small grains of audio are gathered into massive databases and then reconfigured to construct new sounds. We can arrange the databases of grains according to each one's low-level acoustic features and then do things like create interfaces that represent these grains in scatter plots, as in Diemo Schwartz's performance system CataRT, or use similarity matching to simulate one sound using the grains from another, as in Michael Casey's SoundSpotter software. Techniques such as dynamic time warping, the squashing and stretching of material in time to try to find more appropriate matches of patterns, improve the success of the matching.

Some techniques are pseudogenerative. They do not exactly generate new content but enable the advanced manipulation of existing content because they are capable of accurately modeling that content. Adobe's Content Aware technology in its Photoshop program, for example, fills out backgrounds after an object has been removed, using a model of what the pattern properties of the background are. Similarly, the Melodyne software for audio editing is capable of pulling apart the notes in a chord from an audio recording and tuning individual notes. Depending on the situation, the results in both cases can be very successful, but might still fail if the scenario falls outside a common set of expectations.

In games and virtual environments, the production of scenes has long been a target of procedural content generation. Introducing variation in trees and mountain ranges can be done easily using rule-based

generative systems. Increasingly, generative AI tools are continuing this trend. Cityscapes can be filled out autonomously: given a city plan, the height, structure, facade details, and texturing of the city's buildings can be filled out using similar predictive algorithms. Style transfer–like methods can be used to apply transforms to a scene. The structure of one scene can be applied to content from other sources: a car changes make and model, people change clothes, Paris morphs into Phoenix.[41]

Closer to the more lofty end of computational creativity, we are witnessing the automation of tasks such as the mixing and mastering of music, where creative judgment is required. While for many, such as myself, these are tasks for which taste and creative style matter, and significant training makes a difference, they can also be construed as formulaic, scientifically grounded, and objectively good or bad. As discussed extensively in chapter 4, we can avoid the argument about which of these is really the case by recognizing this as a question that has different answers depending on cultural context. In some musical domains, the sonic production of work, including mixing, equalization, and the application of effects such as reverb, is a quintessential zone of creative activity.[42] In other musical domains, this entire process is a more suppressed area of creative focus. For live, unamplified acoustic music there is no mix, and in many amateur subcultures of digital music, tracks are posted to the web straight out of the creator's music workstation. The same analysis can be applied to other dimensions of creativity. Some musical artists might never have considered the question of what tuning system to use, while for others this might be what marks them out as an innovator.[43] Many bands have simple demands, wanting mixing and mastering that make their songs sound clean and loud, which could conceivably be performed by a perfectly formulaic mastering algorithm. Just as some hi-fi amplifiers offer different default EQ settings, such as for rock, pop, or classical, we can accept off-the-shelf mix and master algorithms that offer different styles. These could take the form of different explicitly stated algorithms, just as each individual digital effect has a clear underlying algorithm with various controls and presets, such as an algorithm that tries to maximize the separation of elements or an algorithm that applies a standard techno mix palette. Or they could be based on machine learning—trained on famous studio engineers, for example—or on evolutionary methods, where over time the algorithms are evolved in response to what is most popular. An example in practice is the work of Abreu, Caetano,

and Penha,[44] who used a target-based evolutionary search method to find orchestrations of musical works based on target spectral features.

In short, while in some corners of creative practice the mix might be everything, this is a situation where the creative talent of expert human sound engineers may not figure in the bottom-line considerations of many artists. In a *prosumer* culture, where we write music digitally and immediately upload it, the advantage of such tools in achieving rapid turnover may outweigh the existing constraints on their ability.

Generative Art and Casual Creators

Despite the many advanced methods for applying AI to creative tasks, the more naive end of the spectrum, where simple generative processes could be hand-coded very quickly, is also an area of significant transformation and great immediate application. This is lowly, or *merely generative*, computational creativity, and explicitly so. The creative coder whips up an algorithmic process that generates patterns in a short sketch, perhaps only using it to generate a single output. They may do this as a convenient way to try out multiple variations or simply as a form of material engagement, working using code and algorithms in order to discover new structures that they would not have encountered otherwise, and tapping into the long historical lineage of mathematically driven design. Galanter[45] describes a "complexism" turn in digital art that responds to the nature of complex systems as a building material for behaviors, resembling the "truth-to-materials" principle of the brutalist movement. Here the software is enacting minimal agency in the creative process. Students are often taught to code in this way using creative coding environments such as Processing (a programming environment for artists and designers with a shallow learning curve), learning to appreciate how algorithms work and how they can be effective to produce outputs that wouldn't come about by hand.

Nevertheless, these use cases are important as forms of generative design and creation are becoming increasingly practiced across creative spheres. This has been a well-established field of activity for some time in graphic art and music and is becoming increasingly prominent in design and architecture, where advanced computer-aided manufacturing techniques are freeing us from the constraint to create using only a simple palette of shapes. The challenge is putting the act of casual algorithmic creation into the hands of nonprogrammer users. Related to this are questions around the contexts in

which casual creation is of interest. Compton and Mateas[46] touch upon the potentially great cultural value of rapid personalized creation using creativity support tools. This has started to become a site of interest for a new generation of creative tools in which the act of creation itself is gamified. Nelson et al., for example, present Gamika, a system for metacreating simple 2D computer games (think Pong, Asteroids, or ball-in-maze games),[47] which they believe will support popular forms of social engagement such as jamming (in the hacker sense, creating new games in a social environment) and sending novel artifacts to friends, possibly with a surprise or challenge involved.

Such casual creation systems are also important because several of the techniques described in chapter 5 still need some kind of content-creating subsystem in order to operate. Evolutionary computing approaches require a genotype-to-phenotype transformation that usually has some degree of complexity. As we saw, it is not normally effective to evolve phenotypes directly (such as mutating individual pixels within an image or individual samples within an audio file). The effectiveness of the content creation system at producing richly varied and complex structures will influence the effectiveness of the overall algorithm. Thus even if generative methods themselves are not particularly powerful in the hands of a programmer, finding effective generative methods underlies many potential advances in computational creativity.

For example, a recent trend has been for conceptual creative works that take as their focus the remixing of large databases of existing content. *Of Oz The Wizard* is a beautiful example, a remix of *The Wizard of Oz* movie with the entire movie rearranged so that the script is in alphabetical order. *Double Oh* is a mash-up of every instance of everyone saying "Double-O" in every James Bond movie ever. Christian Marclay's *The Clock* is a 24-hour video mash-up gallery exhibit made entirely from famous movies, where at each moment there is a clock depicted or a person stating the time, telling the correct time of day as seen by the viewer. Not all of these works were made by particularly advanced means (in Marclay's case, a team was recruited to watch movies and note when such events appeared). Nevertheless, AI techniques such as automatic subtitle generation or event and object detection will advance the kinds of tricks we can use and the speed at which we can work in the creation of such data-driven artworks, leading to new aesthetic concepts.

Likewise, new creative practices, particularly in the area of remix and mash-up, are opened up by commercial software that enables the conversion of a recorded sound or image into a more readily editable model of that

sound or image—for example, music software that can convert a recorded piece of music into a series of virtual instruments and score information that recreates the music as well as determining the tempo and meter. Thus a novice musician could produce an accompaniment to a piece of recorded music simply by transforming and filtering the note information that is extracted from it. If such techniques seem too commonplace or mundane to call them AI, remember from chapter 1 that our perception of AI is in a constant state of change, the goalposts forever shifting—AI defined as everything we don't yet know how to do.

Other types of AI systems reveal their own peculiarities that have been the focus of aesthetic exploration. In 2015, a series of images created by "deep dreaming" neural networks went viral on the Internet.[48] These images were derived from a neural network that had been trained to recognize certain things such as dogs. As with the deep learning image processing systems described earlier, these neural networks work directly on image pixel data, meaning that you can feed the pixels of an image file directly into them, from which the object recognition occurs. An interesting feature of such networks, therefore, is that they can be run backward, making alterations to the pixels themselves. For example, you can iteratively make small adjustments to the image pixels that increase the network's perception that there is a dog in the image. By doing so, any minor curve or color gradient that vaguely resembles some part of a dog gets gradually sculpted so as to look more doglike. There's no better word to describe the results than "psychedelic." A regular image becomes peppered with dog hallucinations, seamlessly injected into the surface details. In other cases, clouds become sheep and more abstract effects occur—contours and edges become flowy, almost painterly. Such tools have been made available for public access and use, the result being *lots* of images with trippy dog hallucinations added to them. Importantly, then, these systems are highly narrow in their application: they are powerful tools but still tools, with a characteristically domain-specific nature. AI-powered effects now form part of a growing palette of visual processes used by artists and designers. For example, a recent makeup photo shoot of model Kylie Jenner was adorned with AI-generated "makeup" effects, a custom image transform that mutated the face in curious ways.[49]

Likewise, it may even be the error in the algorithms, their failure to accurately reproduce outputs, that becomes of greater interest, either because the resulting outputs are structurally interesting in their own right or because of

the poetic dissonance of such a creative process. There are many examples of creative work that purposefully draws to light the more daft or unnatural outputs of generative algorithms. In a discussion paper on creative music composition using AI by Bob Sturm and a number of collaborators,[50] Sturm says of his own compositional work: "I did not limit myself to 'cherry picking,' but also challenged myself to work with generated material that was not immediately sensible. … I was specifically looking for failures of the model." Collaborator Oded Ben-Tal says, "Fairly early on in the composition process I realised I was more interested in exploring the edges of the model, and not its more typical results." In both cases, the artists further manipulated the outputs of the system in the production of the final compositions to greater or lesser extents.

Producing Generative Works and Embedding Generative Variation in Interactive Experiences

From Radiohead's pay-what-you-want online record release *In Rainbows* to U2's notorious collaboration with Apple to bundle a copy of their album *Songs of Innocence* free with iTunes, technology now relentlessly insists on shifting the relationship between creative producer and creative consumer. Bjørk's 2011 *Biophilia* album was a high-profile case of using the computational medium, primarily the tablet, as a way to expand the possibilities of music experience into an interactive realm. Years earlier, Brian Eno produced albums based on generative processes and with other artists began to explore the affordances of the CD-ROM, the medium that heralded the mass distribution of software programs as artworks.

Despite the persistence of older formats such as vinyl, we now listen to music primarily through general purpose computational devices such as computers and smartphones, and thus there is in principle no technological barrier to any artist choosing to release music that employs generative or interactive elements (for example, on the iOS platform, it is simply the difference between releasing an album on iTunes and an app on the App Store). There is a usability barrier, as users tend to consume music through preferred and trusted playback apps such as iTunes or Spotify, and of course there is a creativity barrier, as the means to produce generative music are not yet ready to hand for producers. The same applies to video content.

Why would anyone want to produce generative art, and why would anyone want to experience it? Even in domains such as video games, which

would seem to establish the perfect context for generative music, there seems only a weak appetite for generativity. Notions of authorship and association play an important role here; while in some commercial or functional contexts, such as relaxation music or logo design, authorship is not an important factor in the consumption of the creative work, in other contexts it is key. A challenge lies in reconciling the lack of control that comes with generative production with the need for authors to stamp a work's identity. The activities described above lead down different avenues with respect to this challenge; an artist may produce a casual creator, which contains a high degree of their own creative autonomy and can then be placed in other people's hands to arrive at the finished product. Here the task of building a generative system remains very much in the artist's control, whereas using an evolutionary or machine learning process might undermine (or appear to undermine) that control, since they more radically disrupt the autonomy of the creator in their generate-and-test process; if the creator is only playing the role of evaluator and not that of generator, building with their own hands, so to speak, this may challenge their authorship, and once again this will correspond with different attitudes to authorship in different domains.

In 2011, I worked on a project in the band Icarus, a duo consisting of myself and Sam Britton, in which we set out to produce "an album in 1,000 variations." We composed an album of electronic music (our seventh album, so we were well practiced at the electronic music production component of this process), and in addition, we created our own software tools to enable systematic variations to be introduced into each track. A simple kick drum pattern, say, could have four or five variations: one completely regular four-to-the-floor pattern, something with a bit more swing, something with a syncopated offbeat on each fourth beat, and so on. Our system would allow us to set up these prototypes and then smoothly interpolate between them in custom ways. At a higher level, we also set up a similar process regarding the progression of the track through its constituent elements. One version of the same track might start with drums and bass, while another version started with horns and guitar, before following some mutable trajectory through the track's components.

The creative challenge was to find a way to make smooth variations where each variation was coherent in its own right. Just because two kick drum patterns are good, it doesn't follow that a kick drum pattern that is half-of-one-half-of-the-other is also good. Creatively then, the project

involved composing music and seeking out interpolation strategies that maintained coherence. The result was 1,000 minutely different copies of the album. Such an approach could be used to allow endless variation. It could be used to allow a listener to play a record many times without hearing it sound exactly the same each time—a common ambition for generative music producers. However, our interest was in the opposite direction, that each listener might own their own unique experience of the album that they get to know like any other album, but never quite sharing the experience of other listeners. We sold the records so that each purchase was unique and was not available to anyone else, a sort of limited edition. This also enabled us to explore issues of music ownership, to say that the purchaser of each copy actually *owned* a share of the copyright of that version, meaning not only the listening rights to a copy of a mass produced record, but partial ownership of the recording. This also meant that the record existed in the form of regular digital files, not a generative software system, making it more compatible with existing music distribution infrastructure.

However, it seems plausible that we will shift to a situation in which producing a piece of music or a film for a service like Spotify or Netflix will include the option to compose interactive and generative experiences. This is technologically entirely possible now, even if it is questionable what kind of adoption might take place and where. The web leads the way. The sophistication of HTML's media tools makes it trivial to create custom media players that combine layered video, audio, animation, and interactive elements. Indeed, if we think of the domains of design and layout evolving from the fixed media of magazines and posters, then the browser itself has already been serving this purpose since its inception.

Of interest are the kinds of formats that will emerge to accommodate creative production in a generative world. At one extreme, a player could simply embed a rich media webpage or other plug-in architecture that allows creators to submit programs of any kind; the producer would be entirely in control of the format of their creation, with both the freedom and burden of choice that comes with this. At the other extreme are simple audio formats such as Native Instruments' STEM format, which specifies that a musical track be made up of four remixable elements, such as bass, drums, keys, and vocals. This allows a very crude level of on-the-fly remixability, useful for expanding the boundaries of DJing, but perhaps too limited to be of

widespread appeal. Such a proposition also runs up against an acceptance hurdle for artists, who may be reluctant to give access to the constituent components of their work (and if they did want to then they could do this easily using standard audio files and still cater for a large number of uses), and raises quality issues such as how the artist might apply global effects such as compression.[51] Intermediate formats might take the form of simple instruction sets that define the rules for the music's progression, stochastic systems, or controllable parametric systems, and a good general purpose format would cater for all of these scenarios.

In practice, the more features any such format accumulates, the closer it will come to being a general purpose programming environment. It is not too hard for practitioners with basic coding skills to author reasonably advanced generative works in an environment such as HTML5, and generative-focused authoring tools targeting a general purpose platform such as HTML5 can be developed, around which communities of practice and expertise can form. The FLoWr framework is one such example. In the case of the web, where any paradigm is possible, the concept of a *document object model* is extremely helpful in bringing together the most obvious and functional aspects of page layout. It is also apparent in games engines such as Unity and Unreal. These support general purpose software programming manipulation but have common features and file formats that align with expertise and task decomposition, such as the production of 3D models or the scripting of character behaviors.

Extending this idea, generative and interactive systems open up different affordances in different consumption contexts. In a world of ubiquitous or pervasive computing, the technology enabling generativity can be found everywhere. All modern media devices are computers, all computers can be media devices, and all objects can be imbued with computational capabilities and digital interfaces. This expands the purposes for which we might employ generative media. A compelling example is Ben Houge's work on "Food Opera." Houge, an experienced computer game music composer, began to explore where his generative compositions might be applicable outside of the games world. He turned to the dining experience, long associated with music, and asked how the art of a game music composer might be applied to personalized, adaptive composition-for-food. In his resulting operas, collaborated on with world-leading chefs, he places a speaker at

each seat in a collective dining experience and composes adaptive music in which each channel is matched to the menu but also coordinated rhythmically and harmonically with the collective.

Games themselves present several application areas for generative content where the generative production of content is not only beneficial but somewhat necessary. A recent, much discussed example is the game *No Man's Sky*, which uses procedural techniques to generate an astronomical number of planets for players to visit. Planet generation is deterministic, not random, meaning that in a sense these planets *actually* exist, so to speak, as points in the game space; their existence and properties are predicated by the algorithm, rather than being created on the fly. This means that any player could decide to visit a previously unexplored planet, for which they have the pleasure to name it forevermore. There they will see configurations of plants and animals and atmospheric content and other game elements that nobody, not even the game creators, has ever seen, although these will certainly be identifiable as simple combinatorial and exploratory

Figure 6.1
Ben Houge's Food Opera is a novel application of generative music in which adaptive compositions are coupled with dishes and coordinated among the diners in a distributed fine dining audio experience. Image credit: Ben Houge.

variations on what has been seen before. This generative capacity affords a certain game experience, that of authentic discovery.

Lastly, the field has possible functional applications beyond entertainment scenarios. Requena et al.[52] discuss the use of generative music produced adaptively to alleviate pain or attention toward pain in a scenario such as receiving an injection. Similarly, Stefan Ehrlich, Kat Agres, and Gordon Cheng showed that participants could control the mood of music played to them via a brain-computer interface, providing "a tool for listeners to mediate their own emotions by interacting with music."[53]

Creative Blends and Mashups

Style transfer, introduced in chapter 5, is the automated application of a style to an artifact. The classic example is the application of a painterly style such as that of Van Gogh or Munch to a photographic image. From photo filters in Instagram to groove quantize features in digital audio workstations which apply specific rhythmic grooves to musical patterns, the idea of sequentially applying transforms to an aesthetic object to give it a certain style is now completely familiar in digital creative practice. The meaning of style is of course highly variable. It may be acceptable to say that an image filter applied to a photograph is a style applied to the photograph's content, but in reality the photograph embodies a multitude of stylistic components—the choice of subject matter, angle, composition, and so on—and our context influences what draws our focus when we talk about style. Once again, while there may be a natural logic to the relationship between style and content, there is a great deal of domain specificity about where a boundary is drawn between such entities. A neural network performing painterly rendering will have been trained on works of art to derive a model of the fine-grained qualities of brush strokes, and so can modify a photographic image by making it better fit what it expects of this fine-grained detail.

Like other examples discussed earlier, style transfer does not in itself provide either processes of generation or evaluation but instead systematically transforms existing work. Used in this way, style transformations are functionally equivalent to photo filters or groove quantizers, but the potential of the underlying technology is qualitatively different. Systems such as neural networks that are trained on raw image data provide an incredibly fluid toolkit for manipulating style. They have inferred the stylistic qualities

themselves, and this learning can be adapted to target different levels of content and composition. They can be used to reverse-engineer individual creative stylistic features at different levels and then mix these together generatively. They speed up the rate at which we can generate outcomes in a certain style. They support the generation of multiplicities. For example, an artist might generate a style filter as a creative output, deployed in the context of a real-time experience or an endless body of work applied to existing datasets of images. An example is such work is the creative practice and associated technologies developed by Parag Mital in his PhD thesis.[54]

More generally we can say that style transfer is a form of blending where an artifact is created through the composition of two or more different processes by a user. In the style transfer examples discussed, this process is sequential, with the application of one style to an existing artifact, but in other examples of blending it needn't be. For example, Singh et al.[55] looked at the capacity for variational autoencoders (VAEs) to blend concepts in image sketches.

Style transfer performs a task that a skilled artist might have been needed to perform and also speeds up that task. Style transfer may not be obviously creative in itself—it is systematic, predictable, and does not involve search—but one of its interesting potential applications is in its potential to *force* quality outputs: a make-everything-nice machine that democratizes creative production. If we can apply a formulaic process that "fixes" the stylistic qualities of an artifact, then we can be more free in our use of the source artifacts. A mundane street photograph rendered in Van Gogh's brush strokes may take on new qualities. The potential implications of this are diverse. Reducing the skill required in a production task makes it easier, of course, and is also likely to make each such outcome considered less valuable. Thus the domain itself, including the criteria for judgment in the domain, is likely to be transformed by the available technology.

Co-Creative Generative Ideation

This book has been particularly concerned with creativity as a social process. In many fields, collaboration is key and strategies for successful collaboration are a subject of critical research. An example is in the design research of Cross and Cross.[56] They identify a number of informal roles design collaborators might take, along with their formal roles, such as in problem clarification, timekeeping, or facilitating. In co-creativity, a human and a

computational system collaborate in creative production, meaning that the machine is in some way actively involved in the production of creative outputs, interacting during the process with a human co-creator.[57] The human may be in charge and creatively dominant, but they may be steered by the output of the system.

A key application area is generative ideation. This includes any situation where the machine is used to get some ideas on the table, such as to break a writer's block (recall this is the motivation David Cope gave for embarking on work in generative music). The ideas might not be great, but they might be good enough, contain the spark of an idea, or simply trigger the creative practitioner to develop a new direction. The Aiva music composition system, for example, is based on the observation that commercial composers often need to rapidly turn over a rough idea to a client at the first stage of a pitch before developing it further. Flicking through several generative examples for something to get started on can be more effective than a blank slate. I have found this to be the case in my own creative music practice: getting started always means just getting something on the page, anything, and then manipulating and iterating it.

This we may consider the current dominant model of human-computer co-creativity. The human user by one method or other requests suggestions from the system. The system's output does not need to consist of complete works, and the system itself does not need to perform any evaluation of these outputs, but it is capable of producing things in the right ballpark, which the user might be able to iterate. Schematically, the computational system is a generator in a larger, adaptively creative, human-computer generate-and-test complex.

This approach suits a number of the algorithmic methods we have seen so far, and it also fits well with aspects of creative practice that have been discussed previously. Firstly, the user remains in charge of the overall process. Secondly, the system serves a clearly defined part of the creative process, the exploration of a space of possibilities. Here we can see the system both as a material in the hands of a creative individual—like the potter interacting with clay, the user leads the process but is also led by what the system feeds back—and as a cognitive agent collaborating with the user as part of a process of extended cognition, where the tasks of creative ideation and evaluation are split between system and user. As we have seen, distributed evolutionary systems have demonstrated some success as generative

ideation tools, as have neural networks and rule-based systems, all of which can mass produce outputs according to targets, feedback, training sets, or defined constraints.

We have also already seen the most elementary form of co-creation in interactive genetic algorithms where the user selects and the algorithm seeks new random or cross-bred variations within the space that the user is searching. Target-based evolution can also provide co-creative support: as the user is creating something, the algorithm is searching the related space of possibilities for outcomes that suit certain criteria. This was applied in a game level-design scenario by Yannakakis, Liapis and Alexopoulos where the user designs a game level map while the system proposes suggestions based on a hardwired model of playability that it used to test levels with and evolve optimum designs.[58]

Machine learning algorithms such as autoencoders and GANs provide interesting variations on the user experience of co-creation. These enable training on an input set as well as interaction via latent or embedding spaces which gives the user some control over the generation process but with a high likelihood of surprising outcomes (whether good or bad). In Singh et al.'s VAE sketch system, discussed above,[59] the authors argue that the potential conceptual blends the system can produce are valuable for ideation, stimulating the user with unimagined outputs.

Co-creativity can also involve co-development of an output in which the user and the system iterate. This might simply mean that the system generates and the user selects, gives feedback, and alters settings, but it could also involve a more symmetrical relationship resembling a negotiation in which the user may adapt their understanding and goals, and the system may give feedback and, so to speak, alter the settings. Anna Kantosalo and Hannu Toivonen[60] propose the concept of alternating co-creation to describe this situation, distinguishing it from task-divided co-creation. They specifically define symmetric co-creation as that in which the system is able to make fundamental transformations to its own rules for generation.

Kaz Grace and colleagues[61] have explored such "conversational" or "dialogic" AI systems in sketching. A trained sketch-generating neural network uses a model of typicality to generate more or less wild suggestions for designs. These were shown to prompt the designer to come up with new ideas in a manner that evokes classic divergent-thinking stimulation.

"Explainable" AI and the theme of framing in computational creativity (discussed in chapter 7) are important here. A good dialogic interface may require the system to provide additional explanatory content for what it created. Wagstaff et al. showed that a scientific discovery engine performed better in collaboration with scientists if it better explained its discoveries.[62]

Through the above examples, such systems can be seen as standard production tools, much as a Google search might be used to request creatively stimulating or appropriate content, but they are of a particularly creatively potent nature by virtue of their ability to mass produce variants. They are generatively creative in a sense; novel artifacts are produced, regardless of whether they are deemed good or not. These candidates might be immediately abandoned, being deemed of no value, but they might also cause the evaluator, the human, to alter their value system.

The coupled human-computer system, meanwhile, is adaptively creative. The user, even if he or she does not have well-formed outcomes in mind and is engaged in open-ended search, is likely to have motivations and longer-term outcomes in mind such as to impact an audience.

Breeding Creatures

Evolutionary computing approaches, as we have seen, are associated with a slight variant of this relationship between a user and a generative process. The three basic ways that I have discussed evolution being used—targeted fitness-based evolution including multiobjective search, interactive evolution including multiuser evolution, and novelty search—offer different use cases with specific affordances. These approaches, as their inspiration in biology suggests, involve a form of interaction that is slightly more involved than running a generator with given settings. Instead we might think of more responsive and experimental forms of interaction—nurturing of a complex system, akin to selective breeding or the gardening of experimental ecosystems.

Targeted search works when it is possible to define a goal for the system in an algorithmic form. This is particularly applicable in those areas where externally imposed demands must be addressed: object design, interaction design, architecture, graphic design, music for film, and so on. Such objectives can be more or less explicitly stated. An object design task may have specific structural stability requirements, whereas a music composition task

may have an expected mood which may be open to some interpretation and contextual influence. In architecture it is now normal practice to run building designs through a building analytics system that reports on measurable qualities of the design: how much sunlight an apartment gets at different times of day, how long it takes to get from one part of the building to another, and so on.

Such analytics frameworks, especially as they become standardized, mean that any number of automatically generated designs can be rapidly evaluated in terms of their performance on these various metrics, and practitioners can set acceptance thresholds for designs or seek optimal compromises according to multiobjective criteria. Remember that using multiple criteria typically results in a spectrum of solutions rather than one single solution. Each point on that spectrum represents a solution that is in some sense optimal: there is no other solution that is better according to each and every one of the criteria, although there may be solutions that are better according to *some* criteria. Recall the example in chapter 5, in building design, of a tradeoff between sunlight exposure and insulation. The more area covered by windows, the greater the exposure but the lower the insulation. Of course *where* the windows are placed matter a great deal in each case. There are better or worse solutions, but there is no objectively best solution that maximizes sunlight exposure and insulation. Nevertheless, the solution space can be reduced by removing *any solution for which there is at least one other solution that is better in both sunlight exposure and insulation* (this resulting set of partially optimal solutions is the Pareto front, introduced in chapter 5). The user can then look along that spectrum and decide where the best payoff is between different criteria, perhaps based on further criteria. This could even be a choice that is handed to the customer, buying their apartment off plan with an option to choose specific design details before it is built.

The fact that such analytics systems don't cover every aspect of the design quality, especially aesthetic criteria, does not mean that targeted evolutionary search cannot take into account such factors. For example, a designer might create a parametric system that they consider to produce elegant structures of a certain desired style, and then seek optimal outputs from that system using analytic criteria. The result is heavily constrained by the designer's initial design but is still the most performant it can be given that constraint. And of course it may be that, in the spirit of form

following function, the automated adaptation of designs to satisfy explicit performance criteria tends to result in designs that people find pleasing. The trend toward biologically inspired design is indicative of this. A building that optimizes sunlight exposure might end up with pleasing sinusoidal curves, and a building that optimizes travel time between points might end up with a pleasing radial form. Such methods can effortlessly tap the beauty in nature's "algorithmic design."

Evolutionary optimization can be powerful to the point that it can even break the fitness function. In chapter 5, we saw anecdotes of how evolution effectively outwitted the creators' fitness function designs, finding holes that were unintended. In a more sublime case, a researcher was evolving designs for hardware components and wanted to evolve a series of transistors. As they explained, after a successful evolutionary run, they discovered that "some circuits had amplified radio signals present in the air … to give good fitness scores. These signals were generated by nearby PCs in the laboratory where the experiments took place."[63] The circuit simply didn't work if it wasn't in exactly the environment in which it had been evolved. In my own first ever implementation of genetic algorithms for music, I had endless trouble trying to find fitness functions that did what I meant, not what I said. The algorithm would constantly optimize things that weren't the things I actually wanted to optimize.

Another application of target-based evolution is to match specific targets based on similarity. The utility of this is not immediately obvious, because it is hardly creative to merely recreate something that already exists. As we saw in chapter 5, Yee-King's EvoSynth[64] uses evolution to tweak the parameters of synthesizers to recreate natural sounds. Once this has been achieved, the types of manipulation that can be performed on the synthesizer are very different to the types of manipulation that can be performed on the sound sample itself, so something has been gained. An obvious application is to be able to create smooth, interesting, or perceptually effective interpolations between one sound and another, and the same process could be applied to images. Another application lies in mosaicing (or "musaicing" in the case of sound), where a target output is recreated using components of a specific type. A classic example is the mosaicing of thousands of tiny color photographs to recreate the image of another color image, an effect commonly seen in ads and animated graphics. Each photograph's average color content is used to place it in the larger composite image. In musaicing, small sound

fragments are woven together to recreate other sounds. This can be put to conceptual effect or achieve clever perceptual effects, as with the artist Petros Vrellis[65] recreating famous artworks by threading string around a circular configuration of hooks (discussed in chapter 5), establishing a powerful sublime connection between a medium and the image it is used to create.

Interactive evolutionary search introduces a very different form of user interaction. Here the components of the evolutionary algorithm are distributed between software and human user. The human does the selection and the software produces new variants based on heredity, mutation, and genetic crossover. At first glance, interactive evolution is simply a variant of target-based evolution, except that in this case the target is not written in an algorithm but encoded in some human user's taste. In reality, however, there are issues with thinking of interactive evolution in this way. This should be fairly obvious from our discussion of creative processes from chapter 3. It was discussed in depth by Stanley and Lehman, the creators of the distributed interactive evolutionary art system Picbreeder, and authors of the book *Why Greatness Cannot be Planned: The Myth of the Objective*.[66] They observed that users of their Picbreeder system engaged in exactly the kind of open-ended exploration that we have seen elsewhere in creative practice, allowing the material itself to guide their search. They did not set fixed goals for their search but followed its evolutionary pathways by being prompted by things that appeared along the way. They were interested in the journey as much as in the end product.

Nevertheless, this does not by any means rule out the idea that interactive evolutionary search might achieve outcomes that are valuable and could not have been achieved by other means. The electronic media artist Jon McCormack produced his series of 3D generated plant and animal forms through this kind of interactive evolutionary search of L-system structures.[67] Even as the software developer himself, with direct access to the code describing these forms, it was still an effective practice for McCormack to set up an interactive search strategy to help him navigate the vast space of possible designs. The search experience becomes very different from setting up a parametric system and then tweaking each of the parameters in turn, or even tweaking the code by hand. Interactive search is radically different.

From a survey of the uses of interactive evolutionary computation,[68] it seems that evolutionary computation is particularly commonly applied

to creative domains that are not already well established as human creative activities. A possible theory, then, is that if we are practiced at creating something, or it is cognitively manageable, then there is no need for evolutionary computing, but if the form is novel or cognitively demanding or cumbersome to produce (such as McCormack's 3D plant forms[69]), then evolutionary computing is useful. Rather than evolving melodies or visual artworks, we find interactive evolutionary computing used for complex 3D structures, obscure electronic sounds, or abstract animations, as in Scott Draves's evolvable screensaver Electric Sheep.[70] This is a speculative observation, but there are clearly some domain specific areas of application that are better suited to this mode of exploration, at least because they are innately computational in their production.

One vision of the application context for interactive evolutionary computation is that it is hidden, embedded in everyday interaction without you knowing it, conforming to the ambient paradigm described above. When you visit a webpage, it is now common that a team of designers, developers, and content creators is busy behind the scenes tracking your experience and making improvements to the site in response to this data. The time you spend looking for a link, whether you choose to purchase the premium product, whether you agree to sign up to the mailing list, or the success with which an ad distracts you, are all factors that can be measured and that the site's creators have some potential control over through their interface design and content. The powerful thing about websites (and any form of cloud-delivered software-as-a-service), though, is that you can serve a slightly different version of the site to each visitor, manipulating any feature you like and see what effect it has on thousands of users, conducting real-time statistically significant user tests. Such methods are now common, although only relatively recently has this intensive real-time adaptation been put into effect. This strategy was most famously popularized during the 2008 US election campaign that brought Barack Obama to the presidency, where different stories, images, messages, and user-interface elements were trialed in front of a live audience from his campaign homepage.[71]

The same ideas can be applied to subtle aesthetic factors like the appeal of the logo, the color scheme, or the font, and attempts have already been made to create real-time evolvable websites. Various forms of implicit positive or negative feedback can be used to determine the evolutionary selection pressures on these components. The results of such a process, if

successful, might then be used to personalize your experience or might be tried out on other users. As our environment becomes increasingly hyper-networked, we can see how the same logic is applied easily to many other forms of artifact. Software components, entire computer programs, digital assets, and robotic behaviors can be made mildly adaptive through distributed evolutionary processes. With modern digital fabrication, the physical world becomes subject to the same potential.

Likewise, in the music, film, and television industries, there has always been a demand to speed up as best as possible the response to feedback and to trial variations in order to gain a comparative understanding of what works. DarwinTunes[72] was an intriguing experiment in creating generative music that evolved in real time according to user feedback. The creators of the system set up an online radio station that played loops of simple techno-music in four-bar blocks. At the end of each block, the music would shift to a new candidate—not exactly the most realistic listening experience, but one not entirely dissimilar to some dance music structures. Listeners could then give feedback via a web interface in real time. Over time, the music evolved and according to the system's authors increased in complexity.

Similarly, in Picbreeder,[73] while the evolutionary nature of the system is most explicit for those visiting the website, the option is also there to go onto the site only to browse, to look at one's friends' creations, or see the most rated images. In other words, in the massively distributed online form of interactive evolution, there is a seamless interplay between evolution as a form of interaction and more familiar forms of interaction related to selection, such as choosing from a list of presets, seeing what is popular, or seeking outputs by their properties.

As an electronic musician, my own interest in evolutionary computation has been not in the direct production of music using evolution but in the evolution of complex software objects that can be used to create music in live improvised performance. Evolved neural networks have been used to develop sophisticated and efficient robot behaviors, and by implication can also be used to produce interactive musical software agents that map inputs to outputs in interesting ways. In the case of audio, such agents are simply digital audio effects, and the space of possible audio effect processors is yet another creative space that is clearly defined and clearly constrained, yet full of rich unexplored possibilities. With PhD student Steffan Ianigro this

work has developed toward an online distributed evolutionary computing interface and repository for evolved audio effects.[74]

Here again, the evolutionary component of such an interface can be either operation-based—something the user might manipulate, configure, and control—or ambient, in which case users might download and try out different modules, without being necessarily aware that they are feeding back into a distributed, interactive evolutionary algorithm.

Distributed Evolutionary Systems

Further to the algorithmic specifics defined above, there are various ways in which systems can be enhanced by making them distributed. This includes using a distributed user base (as in distributed interactive genetic algorithms) or creating virtual populations of agents that interact with each other. It also includes the development of modular systems that are created by multiple developers, as in the FloWr framework for computational creativity.[75] When considering these systems, we come up against the distributed view of creativity discussed in chapter 3 and find that the generative form of creativity is much more explicitly expressed in these circumstances.

Distributed interactive genetic algorithms are interactive genetic algorithms in which multiple users have access to and influence over the evolving population. The primary aim of adding a multiuser element to such systems, usually stated, is to attempt to solve the problem of user fatigue. Evolution of complex structures requires many generations of mutation, genetic crossover, and selection, and it is too much to expect one person to sit and manage this process for too long, especially if we are expecting this to be an intrinsically motivated experience. Distributed interactive evolution purportedly address this problem by spreading the work across multiple users, lightening each user's workload. However, it remains to be conclusively demonstrated that the fatigue issue really gives an accurate portrayal of the problem with interactive genetic algorithms.

As we saw Lehman and Stanley demonstrating with their system Picbreeder, users don't tend to have clear objectives when evolving images but instead engage with the system through the type of interaction that Malafouris[76] described as material engagement, allowing their search to be directed by what is thrown back at them. The achievement of a concrete objective, then, is replaced by a more fluid notion of creative achievement.

At the extremes either everything or nothing produced by the system is an exciting discovery for a given user. More likely, the user experiences fluctuations in their level of engagement and excitement. The experience might well be an end in itself, of which the resulting images and their artistic merits are not fully representative. In addition, applying notions of perceived originality and identity, users may *wish* to put in effort as a way to carve out something that is uniquely theirs.

Applications like Picbreeder are compelling but are yet to make professional grade tools, although they may not be far off. Many visitors to the site are there out of curiosity rather than with a creative aim in mind. Without a clear model of what users are there for we should be wary of projections such as the assumption that fatigue is holding them back from achieving goals.

There is another reason why distributed interactive genetic algorithms might be more effective than single user algorithms. This is not to do with the efficiency of finding universally agreed "good" outcomes, but the efficiency with which the algorithm is able to match solutions with uses—something the Internet, with its many-to-many structure, is of course excellent at doing. Creative practitioners frequently go online to obtain resources produced by third parties that they can use in their own work, such as clip art, copyright-free audio samples, virtual studio plug-ins, and creative coding libraries. For the most part these resources are produced by people, but it would make little difference if they were produced automatically, as long as they serve the purpose we seek (caveat: again, it is worth noting how the social value of authorship varies from one domain to another). A distributed interactive genetic algorithm not only distributes the selective pressures imposed by different individuals but also distributes the results; all of the outputs generated so far by earlier evolution by users. Someone can go onto a site such as Picbreeder and interact with it not via selective evolution but in the more familiar way of scrolling through a list of objects that have been favorited or otherwise tagged as valuable.

By placing many evolved objects in front of many users, a distributed interactive genetic algorithm completes the Gibsonian (or equally we could say "niche constructionist") vision of ecological complexity accelerating evolutionary discovery. Since affordances describe relationships between pairs of entities, increasing the combinatorial mix increases the discovery of useful interactions. Just as the users may find objects they need, *objects*

may find users they need. To state that an evolved aesthetic object is actually in search of a user may be considered a flippant anthropomorphism, but in fact it is perfectly suited to interpretation by way of cultural teleofunction. By displaying the object in front of many users we are increasing the chance that it finds a use, just as Post-It note glue needed to find its use.

Performance and Interaction

Performative domains such as live music or dance can suggest very different algorithmic requirements and somewhat different aims within the space of computational creativity practice and its motivation, particularly when there is a real-time interactive element to the performance. There are several areas where this may be the case: live music performance, in particular improvised music; games and other forms of interactive entertainment such as interactive storytelling or interactive TV; and other forms of dramatic performance including interactive gestural experiences and robotics.

In one respect, the creation of a performance can be considered to have exactly the same expectations and goals associated with it as any other creative production, and in some cases the difference is more of a technicality. For example, a music performance system that interprets a score of solo piano music could plausibly be run offline, resulting in note data that could then be rendered on a digitally controlled piano. The system could run back and forth through its rendered output making modifications or trialing different candidates, just as a system generating a visual artwork might. Both systems may be subject to time constraints, depending on the use context, but not the constraint of real-time performance.

Introducing interaction, however, introduces a strict real-time requirement; events simply can't be mapped out in advance. Consider the same interpretation system with the only difference being that it must follow a conductor. The expressive performance would need to change depending on the input given by the conductor. The system can no longer simply perform its computation up front, because it needs to have knowledge about the changing tempo in order to make decisions about the placement or accentuation of notes.

Now imagine that the system also has to interact with other musicians reading a score, or further still, switch this performance context for one in which the music is improvised around an agreed structure, perhaps without a score at all. There are many forms of improvised performance with varying

types of organizing structure. In traditional jazz, one or more prior chord sequences and melodic lines are used, usually with a series of solos over the main chord sequence, topped and tailed by the lead melody. In free improvisation, a common theme is the aspiration toward "non-idiomatic" performance, where stylistic premeditation or organizing structures are eschewed as best as possible. Performers working in this domain inevitably abound with idioms, but they nevertheless often take the approach of minimizing the pre-agreement of elements.

In all of these cases, details aside, the basic constraint is the same, that the creative work unfolds in time rather than being created in what may be thought of as an out-of-time process. The work is live and cannot be started over, events that have occurred cannot be edited in light of what has later transpired, and choices about current actions can only be made in light of uncertain expectations about what is to come later.[77]

The real-time domain may therefore be described as having all of the demands of a non-real-time computational creativity scenario, with the additional demands of having to deal with this irreversibility with respect to past events and uncertainty about future events. In practice, though, there is also a prominent body of work in improvised systems that doesn't burden itself too much with the traditional issues of creative generation and instead focuses on the interaction experience and the potential to stimulate co-creative activity with one or more human partners. One burden, to generate original content, is lightened in place of another, to operate under real-time demands. Thus in a common free improvised music scenario, a solo musician will perform in a duet with a software system and the question is not so much whether the system generates original and pleasing music each time it runs but how it is experienced as an interactive agent expressing forms of musical intelligence and a sense of agency. For researchers such as Pachet (whose experiments with the Continuator[78] are described in chapter 1 and are discussed further below) a core question was how such a system stimulated new creative ideas in the performer and enabled them to reflect on their own performance. That is not to say that the system shouldn't aspire to the wider goal of producing a varied and original output, but that there is an intermediate and distinct area of interest in this particular case. It is also important to note that in the minutiae of the system's output (constrained though it may be), and also in the

co-creative interaction between performers, most systems designed to focus on performer engagement do still somehow contribute to the production of original and compelling creative events.

There are several candidate examples of system behaviors that might successfully give the partner musician this experience. Being able to play in the same key as the musician and follow the beat are basic examples, but they don't speak much to the system's ability to engage musically or creatively. A common elementary approach for system developers is to get the system to perform in a copycat manner, such as playing back reinterpreted fragments of the human player's performance (an example is the work on the Omax system[79]). This communicates that the machine is listening and responding, and establishes some connection between the two agents that is not just down to the human's capability. Although this sounds like the kind of thing a human improviser might do, some musicians playing with such systems report finding it disconcerting to have their own musical performance thrown back at them.[80] This depends on how literally the material is treated. More generally, of course, this is an example of a creative musical decision that is very much a matter of taste and compositional intention, as varied as musical styles are across the board.

More impressively, the system might attempt to find "continuations" of what the musician has played. In other words, rather than recycle the material, it might follow what is being played and come up with expectations about what should come next. This means that a flowing call-and-response interaction between a human and machine musician could be set up or the machine could play alongside the human with a sense of appropriate expectation. Couple this with a strong ability to make rhythmic predictions, and a strong sense of interaction can be set up. François Pachet, whose Continuator[81] is an early success story in this field, refers to such novel interaction scenarios as facilitating *flow*—the state of immersion that comes when experiences are suitably stimulating, which we encountered in chapter 2. He also refers to them as *mirroring* systems in that they mirror the performer's style.

Other ways to establish a sense of interaction might be more obscure. When improvisers play they don't necessarily perform continuations of what each other is playing, and in some styles of free improvisation there is little even that the audience might pick up about the interaction. In one experiment by Young et al.,[82] audiences were played improvised recordings

where an ensemble of free improvising musicians played with each other but may or may not have been able to hear each other (the performers, in separate rooms, were given headphones playing either a live feed of the other musicians, or white noise intended to block out the other musicians). The results showed that from the audience's perspective it wasn't always easy to tell whether the musicians could hear each other. This is a concept that was further corroborated by a formal study of causality in a standard jazz improvisation session, where musicians were seen not to have as much influence on each other as the score had on the musicians.[83] While the musicians may be responding to what they hear, they are not doing so in explicitly obvious ways like copycat playing or tight synchrony. Of course, in other instances they may be very explicitly doing this, so such results may be interpreted as being highly case specific.

Key concepts here that are closely related to autonomy and agency are *causality* and *coupling*. Causality is the influence one system has on another, and in performative contexts such as those we have just considered above, it is important to ask which systems influence which others. One way we can think of causality (referred to as Granger causality after its creator) is in terms of whether any given system's future state is best predicted by its own recent history or the recent history of things external to it. If the better predictor is the external factor, then there is a causal relationship involved. Autonomy can be thought of as the greater influence of the system's internal state on its future state.[84]

From cybernetic thinking, and commonly used in artificial life, we have the idea of coupling. Coupling is simply the linking of two or more elements in a cycle of cause and effect. Loose coupling refers to the situation in which elements are coupled but also strongly self-determined or influenced by other outside factors. They can be decoupled and carry on largely as usual. Despite the looseness of the coupling, however, loosely coupled systems can have strong interaction effects. Loosely coupled oscillators may, under the right circumstances, synchronize, as is the case when two pendulum clocks are standing on the same surface, such as a mantelpiece, and start to tick in perfect phase. Even though there is only a very weak force transmitted through the surface, it slowly takes effect, and once the oscillators are aligned the force acts to stabilize the synchronization and keep them from drifting. Under other circumstances, however, such as when

the natural oscillating frequencies of the coupled systems are different, the result of the interaction might be more unstable and lead to complexity or punctuated moments of stability.

This thinking can be applied to the way that improvising musicians interact, and a number of experimental machine improvisation systems have worked at this level of abstraction and dynamic attention.[85] In general this type of work is less about simulating human musicality and more about the creation of dynamic computer music performances that exhibit simple forms of machine agency. Works of this type have been placed under various banners from algorithmic composition to musical metacreation. The agency may not be focused on explicitly musical objectives such as achieving a harmonic outcome or exhibiting a musical understanding of the fellow performer. It may be responding to simpler conditions imposed by the composer and teased out by evolutionary selection, but in such a way that also drives the sense of flow described by Pachet.

In my own work,[86] I have used types of oscillatory systems that are capable of complex behavior. The actual behavior of the oscillatory system is shaped by an evolutionary algorithm that selects for abstract properties such as complexity, repetition, and responsiveness. These different goals are combined and experimented with manually. The resulting dynamic system can then be used in an interactive composition, responding to input from a live musician in the form of low-level features such as loudness, pitch, and spectral and timbral features, and driving the behavior of a separate digital music system which may itself contain generative elements. This is a naive system from the point of view of what musical AI is capable of but has been proven effective at exactly that issue of engaging musicians and creating a powerful sense of musical interaction and flow.

As with the lofty-versus-lowly computational creativity distinction in offline generative tasks, the pinnacle of such real-time co-creative systems is that the system should have a model of what it is involved in creating and can adapt what it is doing based on that model and the observed outcomes as they unfold. Because this is done in real time and involves other co-creators, this must in some sense take the form of a negotiation. In the lofty form of real-time creativity in improvisation, then, the system is sharing the creative task with other complex creative agents and the target is moving. The system must have an overall expectation of the outcome, an

idea of how its output contributes to that outcome, and must also be prepared to adapt both of these based on its experience, else there will be a permanent gap between the system's expectation and the outcome.

As with other art forms, the creation of music is never simply a matter of the production of abstract artifacts but is also guided by factors relating to display, whether competitive or cooperative. In the words of Philip Auslander,[87] "Musical performance is not just about achieving certain sonic ends—it is crucially also about perceptibly overcoming challenges presented by the means used to achieve those ends."

This observation is context for Auslander's discussion of issues of liveness and agency in the presentation of machine generated music. Interviewed by Auslander, Mari Kimura, a violinist who has performed extensively with the robot system, the GuitarBot, reveals a level of implicit misrepresentation in the GuitarBot performances:

> My compensating for the robot's or computer's lack of musical "integrity" as the performance goes along should be hidden from, or unnoticeable to, the audience. In short, my aim is that the performance as a whole come across to the audience as if the robot or computer is thinking, feeling, and being sensitive; that it possesses the "rights and responsibilities" of a true musician.[88]

Yu and Blackwell[89] elaborate on audience perceptions of agency drawing more squarely on psychological principles, namely the apparent mental causation model and the expectancy violation theory. The apparent mental causation model dictates how people identify causal relation; causation of event B by event A is perceived when event A precedes event B, event B occurs as expected, and event A is the only plausible source of the outcome. Expectancy violation theory claims that a person becomes more alert to a system when it violates expectations.

Relatedly, Banerji[90] contends that there are significant issues surrounding expectations; he proposes that technologies such as machine learning can "legitimize agency" in automated music systems, more specifically that "ML constitutes an effective means of encoding the socio-political ideals at the center of this musical practice." Such systems lay claim to a form of autonomy in that the programmer wasn't involved in specifying by hand the system's style, but such systems are not necessarily by virtue of this better creators of interactive experience. Models and studies of audience and performer perception also play a role in building this understanding.

> By contrast, tests of agents not based in ML reveal that human beings experience illusions of "adaptation" in interactions with systems which lack any adaptive capacity. Such results suggest that HCI research with artificial social interactants may be used to raise new questions about the nature of human interaction and interpersonal adaptation in the formation of relationships over time.[91]

Perceptions of agency and autonomy also relate to perceptions of humanness or lifelikeness, and this is an area that can be explored to dramatic effect. Several artists have been interested in creating forms of agencies that are original, not imitations of human behavior, as a creative objective in their own right, as sources of contemplation. Simon Penny, a leading figure in cybernetic-inspired interactive art, explains:[92] "An artwork, in my analysis, does not didactically supply information, it invites the public to consider a range of possibilities, it encourages independent thinking. So building an interactive artwork requires more subtle interaction design than does a system whose output is entirely pragmatic, such as a bank automat."

He says of his celebrated interactive robotic artwork *Petit Mal*:

> It was not my intention to build an artificially intelligent device, but to build a device which gave the impression of being sentient, while employing the absolute minimum of mechanical hardware, sensors, code and computational power. The research emerged from artistic practice and was thus concerned with subtle and evocative modes of communication rather than pragmatic goal based functions. My focus was on the robot as an actor in social space … Every attempt was made to avoid anthropomorphism, zoomorphism or biomorphism. It seemed all too easy to imply sentience by capitalising on the suggestive potential of biomorphic elements.[93]

In the domain of performance and interactivity, perhaps, then, we find the most explicit conceptual studies of the agency of digital systems.

Complete Creative Production Cycles and the Machine-as-Artist

Lastly we come to the lofty computational creativity scenario where a machine is embedded in a recognizable human role: a simulated, virtual, or machine artist. Everything discussed so far in this chapter can easily be understood as a continuation of human machine use in the arts, with new technology applied to existing human goals, realized in the hands of human artists. I have considered how such applications may have radical impacts on the production of creative outputs as well as on the related social configurations and cultures of production and consumption, but

none of these scenarios comes with the philosophical baggage that we see in the machine-as-artist scenario. This philosophical bent comes in turn with much confusion about what purpose such a scenario would serve. Indeed, for many the idea of creating a machine artist has little to do with practical use-cases and is instead a matter of philosophical enquiry, a theme explored throughout this book.

Where exactly does the distinction lie with the previously discussed scenarios? In many of those scenarios, machines perform some active role in a creative production task, and I believe they should largely be rightly considered computationally creative systems—that is, systems that in part automate creative production—even if they do so in relatively nonaware and nonautonomous ways such as in a structure-and-generate design pattern or an interactive evolutionary computing scenario. In all of these cases, machines are still active makers of artistic outputs in some capacity.

In a machine-as-artist scenario, the machine is not simply a production tool but an agent that performs a recognizable pattern of art creation that we would associate with a human artist, which may include social and cultural relations with other individuals, such as audiences and co-producers.

Take the full bells-and-whistles version of this scenario. You step into your city's modern art museum, and while browsing the permanent collection, you find an abstract expressionist painting created by a machine artist. The machine has a name and an identity, a recognizable style nurtured over several years of development. It can in some capacity discuss or explain what it is doing and even justify or "sell" the value of what it is doing through channels other than the painting itself. On further investigation, you find a list of programmers and creatives credited for creating and managing this machine artist, but you are nevertheless satisfied by the breadth and sophistication of its output, supported by documentary evidence, that it is a legitimate creative author, not simply a trick or form of industrialized art production.

As early computational creativity figures such as Cohen and Cope presented their work, there was some degree of willingness by audiences and commentators to view their systems in this light, and as we have seen these systems took on the role of philosophical probes, looking into both machine creativity as an abstract phenomenon, and the potential for machines to perform tasks and fulfill roles that humans do.

In fact, we face a more nuanced breakdown of categories that inform the machine-as-artist scenario, as follows:

1. Systems that allow the application of AI algorithms to art generation, typically as mere generators, but potentially including simple forms of evaluation and feedback.
2. Systems that perform generate-and-test cycles, involving various forms of adaptive and reflective behavior.
3. Systems that perform forms of engagement with audiences and/or grounding of their creative output, such as presenting an "identity," reflecting on current affairs, explaining themselves, responding to feedback, and having grounded motivations such as winning prizes.
4. Systems that convincingly replicate human traits and successfully assume human creative roles.

Here, Category 2 systems can be seen as extensions of Category 1 systems that advance the capacity and autonomy of these systems by introducing algorithms that enable the integration of generation and evaluation into a single adaptive system. Consider taking a successful machine-learning-based generative algorithm, having online users rank its outputs, and then feeding those rankings back into the training of the generation system through reinforcement learning.

Note, however, that embedding systems successfully into cycles of feedback as required by Category 2 would be likely to also involve aspects of Category 3: the system gets embedded in a sociocultural creative context, as a byproduct of extending its creative capacity to involve feedback.

At the same time, Category 3 aspects can be explored independently of whether the system exhibits Category 2 generate-and-test capabilities. Some of the most in-depth work in this area is by Colton and his team, working with the Painting Fool system,[94] situated in gallery contexts. One issue Colton has studied is the way in which a system can not only create an aesthetic artifact but also communicate its process and perhaps even convince an audience of the value of the work.[95] Along with each painting, the system generates descriptive text explaining what the motivation of the work was and what mood was applied. This establishes a multifaceted representation of the work, which may achieve some degree of social grounding.

Interest in the fourth category was an original motivator for much work in computational creativity, but this has significantly dwindled as we know

more about the technologies and applications of computational creativity. Distributed agency and distributed creativity perspectives have grown sufficiently in the time since Cohen and Cope that we are less inclined to frame computational creativity as a quest to create virtual artists. We are more familiar with AI algorithms that perform sophisticated tasks without being anthropomorphized, instead occupying a more amorphous expanse between designed and intentional stances. Kurzweil's framing, where machine artists overtake human artists, seems no longer to frame the problem in an appropriate way.

The open access edition of this book was made possible by generous funding from University of New South Wales.

This book was set in Stone Serif and Stone Sans by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Bown, Ollie, author.
Title: Beyond the creative species : making machines that make art and music / Oliver Bown.
Description: Cambridge, Massachusetts : The MIT Press, [2021] | Includes bibliographical references and index.
Identifiers: LCCN 2020017931 | ISBN 9780262045018 (hardcover)
Subjects: LCSH: Technology and the arts. | Arts—Technological innovations.
Classification: LCC NX180.T4 B68 2021 | DDC 700.1/05—dc23
LC record available at https://lccn.loc.gov/2020017931

10 9 8 7 6 5 4 3 2 1