Does AI Herald the End of Creative Coding?
Paper type: Debate Sparks

Jon McCormack
SensiLab, Monash University
Melbourne, Australia
jon.mccormack@monash.edu

Stephen James Krol
SensiLab, Monash University
Melbourne, Australia
stephen.krol@monash.edu

Abstract

Advances in AI assisted code authoring and development mean that it is now possible for anyone to create custom software without extensive programming knowledge or experience. In this short paper, we ask what this means specifically for those who use computer coding as a creative medium. We discuss issues of cognition, aesthetics, creativity and knowledge, speculating on how creative coding might change in both the short and long term future.

Introduction

The widespread industrialisation of AI technologies continues at a rapid pace, including the advancement of AI systems that can design, write and debug complex computer programs from high-level, natural language instructions. "Coding Agents" such as *Claude Code*[1], *OpenAI Codex*[2] and *OpenCode*[3] have found rapid adoption amongst both experienced programmers and novices alike. Many people have commented on the significant increase in capability of the most recent models, with Coding Agents now being viewed more as collaborators rather than basic assistants or tools. Platforms such as *Lovable*[4] promote the concept of "vibe coding", where "the people who've had ideas but lacked the technical skills to bring them to life" are now "empowered" to write apps by telling an AI their idea in plain language and letting the AI platform create it. Lovable claims over 100,000 new projects are built every day (Lovable 2025) and that everyone can now be a "builder".

This newfound ability of AI to code competently has also led to job losses in software development (Techcrunch 2026), a downturn in new university enrolments in computer science and software engineering (Braue 2026), and proliferation of code with significant security and privacy issues (Brewster 2026). Driven by narratives of productivity gains, along with a general "Fear of Missing Out" on the next revolution in AI, coding agents are now widely adopted by developers, or at least employers are requiring developers to use them.

[1] https://code.claude.com/docs/en/overview
[2] https://openai.com/codex/
[3] https://opencode.ai
[4] https://lovable.dev

In this short paper, we want to consider specifically how AI coding agents might impact the field of "Creative Coding", although some of our arguments could apply to programming more generally. Creative Coding is an artistic practice that uses computer code as its primary medium of creative expression. A "Creative Coder" is a person who writes computer code that generates a creative artifact or behaviour. Creative Coding encompasses practices such as Generative Art, Algorithmic Art and Computer Art (McCormack 2024). In contrast to related practices, such as "Software Art" (Arnes 2004), in Creative Coding it is what the code produces – not the code itself – that is considered the main object of art appreciation. Some well known artists include Frieder Nake, Vera Molnár, Casey Reas and Zach Lieberman (to name just a selective few). Creative Coding also extends beyond the world of fine art, into commercial fields such as motion graphics, design and commercial art, but the basic principle remains: practitioners expressing their creativity through the medium of code.

A Golden Age For Creators?

Having an AI system that helps you write code does initially sound appealing. Programming is a complex skill, difficult to master, requiring many thousands of hours of practice and experience to gain proficiency (Robins, Rountree, and Rountree 2003; Bennedsen and Caspersen 2007). Modern software development relies on convoluted development environments, arcane build chains and multiple APIs. The professional programming world is filled with acronyms and jargon, making it difficult for artists and designers to easily understand and gain entry. This frustration led to the development of specialised "Creative Coding Environments", such as *Processing* (Fry and Reas 2014), which simplified the development environment and incorporated a rich set of functions targeted towards creative applications, such as sound and visual processing, interactivity, and graphics. But even with these artist-friendly environments, understanding how to program – having knowledge of data structures, algorithms and complexity measures – remains an important aspect of creative coding (Luxton-Reilly et al. 2018).

So for the experienced creative coder, agentic AI is appealing. Tasks that previously took days or weeks of coding labour can be undertaken by an agent in just a few minutes. Using plain language to direct an agent mitigates the need

for artist-friendly IDEs, or programming language simplification, currently common in creative coding development environments. For example, one can prototype in *Processing* (simplified Java) and then ask the agent to convert the code to high-performance, optimised C. Need a support library that is missing in your current environment? Get a coding agent to download, configure and install it for you, or even write it for you from scratch. Past projects can be given new life: augmented, extended, and modernised. The artist is no longer limited to a particular platform, language, API or development environment. Coding agents potentially eliminate the need for emulators or hardware virtualisation because they can translate software from one environment to another. Responsibility for the preservation of creative code artworks shifts from the museum conservator to the AI agent.[5]

These advantages might mean a new "golden age" for creative coding, at least for those with sufficient prior programming experience to understand and manipulate the code generated by an AI agent. Agentic AI coding is like having a whole team of coders at one's disposal at a tiny fraction of the cost of human programmers. Limited only by token budget, this team can work anywhere, anytime, on any programming task.

But like other AI incursions into human creative making such as text-to-image (TTI) generators, we need to be wary of how these new-found abilities might play out at scale and in the longer term. When capable TTI systems first appeared several years ago, many rushed to evangelise a new, democratised creative era, where anyone can "create stunning art in seconds" (McCormack et al. 2023). The reality today is not that everyone is creating "stunning art". TTI systems have flooded the Internet with AI slop (Kommers et al. 2026; Mahdawi 2025), are being used to promote and reinforce extremist political narratives (Watkins 2025), to create sexually explicit images of people against their will (Gentleman and Horton 2026), or as a way to save money on making Christmas cards (McCormack et al. 2024).

Just as the ability to generate an image from a prompt does not make you an artist, the ability to generate software does not necessarily make you a creative coder. You need artistic knowledge and skills as well, along with an understanding of how coding and creativity are enmeshed. While experienced creative coders may benefit from coding agents in the short term, once anyone can become a "builder" we are also likely to see coding skills devalued and a significant increase in AI code slop. The long-term consequences might be that human authoring of software becomes completely unnecessary and redundant as a useful skill. Instead, people will direct AI systems who write, maintain and debug the code. The complexity of these systems might increase to the point where no human can understand them at a code level, forcing a reliance and trust onto AI systems completely.

[5]Long term preservation of code-based artworks remains challenging, due to the technology they were developed on becoming obsolete or inaccessible (García and Vilar 2010).

Cognitive Offloading and the Locus of Creativity

Using a coding agent to write code creatively offloads the complexity of designing and writing software from the artist to the machine. At first glance, this might seem beneficial, because it frees up time and cognitive capacity to focus on the creative ideas, which the machine then translates into code. But this benefit is based on a false separation of creative ideation and coding practice. As clay is to a potter, code is to the creative coder: it is the material that is fashioned and manipulated into the system that produces the creative work. This is what Malafouris calls *Material Agency* (Malafouris 2008) – the act of coding is not simply to execute an internal mental plan of code authorship. Rather, the final form *emerges* from the interaction between bodies, tools, materials, habits and environment. Creatively developing software involves feedback loops between people, materials, code and culture; the ideas are iterated, modified and developed until the final product emerges. One rarely begins with a finished idea that is simply realised transparently through code. The fashioning and forming of code is integral to the creative process and the development of creative intention. Here, we think of colloquial expressions of how the creative process works: "one thing led to another", "trial and error" and "it wasn't until I'd done X that I realised Y" (Dietrich 2004).

At a level of abstraction, coding creatively is no different than a sculptor fashioning clay at a potter's wheel: the (code's) material behaviour and resistance give rise to the form. The artist shapes the form, but the form responds during the fashioning process and the creative artefact emerges from the nexus of those interactions. If we change or remove the material conditions from the process, the creative practice is no longer the same.

Conceptual problems of exchanging an embodied creative practice, such as painting or photography, with a prompt-based specification have been raised previously (McCormack et al. 2023). Most obviously, exploring creative concepts through language is limited to what can be expressed linguistically, and in the case of prompt-based systems those limitations are compounded by requiring explicit directives that can be correctly interpreted by an AI system. To paraphrase the painter Edward Hopper, "If you could say it in words, there would be no reason to code". While multimodal AI systems mitigate linguistic limitations to some extent, they still rely on being able to effectively translate ideas in one medium (e.g. language, visuals) directly to another (code). This may not be a stumbling block for engineering applications, where clarity, robustness and directness are important. But for creative coding, where ambiguity, poetics, metaphor and aesthetics are primary, it is a major conceptual shift.

Of course, human language is eminently capable of ambiguity, poetics and metaphor. Programming also makes use of language (both human and machine), but a fundamental difference is that human language evolved for human communication and expression, whereas programming languages are interpreted by machines. Machines have no use

for poetics, ambiguity or metaphor – they simply interpret the compiled code directly as instructions. So the use of artistic concepts, such as aesthetics, poetics, metaphor, etc. are necessarily in code's semantics, not the machine's interpretation of them.

Coding agents shift the locus of creative authorship from code-as-material to prompt-as-specification, and this shift is incompatible with creative coding's tradition of treating code itself as the aesthetic object. Some form of "creative prompting", or more likely human-agent interaction may soon replace direct coding. This interaction might become recognised as a new creative practice, but it will no longer be creative coding, since the material conditions are incompatible.

Excess and Inevitability

Current developments in AI can be understood through the lens of *excess*. Model training requires excessive amounts of data, resources[6] and energy – making foundational model development only possible for a few select, highly-resourced corporations. AI can generate obsessively and faster than people, resulting in the slop that is now polluting our digital lives. With the advent of coding agents, slop is now permeating software[7], causing problems for cybersecurity, open source developers, software engineers and of course, end users. The training data – the code on the Internet – is of varying quality, but often tends to be "average". Like all complex human endeavours, writing high quality software is difficult and time consuming, learning to program competently takes thousands of hours of practice and experience, so it tends to be protected from wide public exposure due to this investment. But this "average" code is largely what AI systems are trained on, and while RLHF and curated post-training shift output away from the training-data mean (Niekerk et al. 2025), the quality and suitability of code generated can vary, particularly if the code-base exceeds the LLM's context window.

For creative coding in particular, artists see their carefully crafted code as the basis of their artistic practice – what defines their aesthetic and makes their practice unique or, what Dissanayake calls "making special" (Dissanayake 1995). So while many explanatory or educational examples of creative coding exist on-line, far fewer professional creative examples are available to be scraped.[8] So commercial AI's need for excessive training examples is limited by Creative Coding's cultural practices.

Another way AI is publicly promoted is through it's *inevitability*. A strong narrative from tech evangelists and tech companies with a vested commercial interest in AI is that the development of increasingly powerful AI systems are *inevitable*. This sense of inevitability helps to mitigate objections or constraints on AI's development: if AGI is inevitable, how can we act to stop it? If we don't develop AI someone else will, and they'll reap the benefits or behave irresponsibly. These narratives echo similar arguments around inaction on climate change that play to strong hedonic and egoistic values (Steg 2023): if it is inevitable, how or why *should* I stop it? But developments in AI are not inevitable. We all have the right and responsibility to help determine the future we want to live in.

Conclusion: the end of Art-making?

In a recent article, media theorist Lev Manovich questioned if art making is still meaningful or even necessary when AI can often do it as well as or even better than humans (Manovich 2026). Across all the previous historical transformations of art due to technological or cultural change, one assumption stayed fixed: art is something made by humans. Manovich suggests that the next transformation may break that continuity. If AI systems can generate paintings, images, films, music, writing, and design it at high quality, then the future may not simply involve new artistic tools. It may involve a weakening of the human-centred concept of art itself.

Unlike AI systems that directly generate images, films, music or writing, coding agents generate software, which then needs to be run on a computer to reveal what it does. In creative coding the expectation is that the software *produces* something creative, not that the software itself is creative. Coding AIs are potentially only limited by Turing-completeness in what they can do.[9] So there is the potential for this meta-system to produce work that has greater originality and artistic value over the slop-producing statistical mash-ups from AIs directly trained on current human creativity. However, this potential rests on the assumption that AIs can code *creatively*.

Creative Freedom

In his book, "Mindless: The Human Condition in the Age of Artificial Intelligence", the economic historian Robert Skidelsky warned that "Unless we understand technology as a system of ideas rather than as a necessity, we will be powerless to choose which technology is best suited to our needs and purposes." (Skidelsky 2024, p.269). He questions if new technologies increase user's freedom to choose their own plans for life (i.e. are liberating), or if they force people to work within the systems imposed on them (subjugation). At the heart of art-making is the idea of creative freedom: the ability to express yourself how you want, in your own way. The early days of computing were full of new creative possibilities, many we have seen materialise over the last 75 or so years. Amongst these possibilities was idea that AI could make Art (Cohen, Cohen, and Nii 1983), something that was only taken seriously outside the world of the technological arts in the last few years.

If machines diminish the meaning and significance of human art-making then the freedom to be creative – in terms

[6] Estimates of over $100M are often cited for foundation models.

[7] *Lovable* publicly claims 36 million apps having been built on its platform.

[8] While high-quality creative code being publicly available is relatively rare, the *results* of that code are much more widely distributed, making it ripe for generative AI image generators to train on.

[9] Although there are many more practical limitations.

of employment, livelihood and social recognition – is threatened. "Being creative" risks becoming a nostalgic, recreational activity: something we might occasionally do to pass the time once creativity has been automated out of human interest.

Coding agents have the potential for today's coding artists to mitigate the limitations of time and complexity when writing code. Provided the locus of creativity remains centred around the artist and their code, we should expect to see a proliferation of interesting new examples of creative coding. But if the human authorship of code becomes obsolete due to AI, creative coding will become a lost practice. As Manovich points out, the continued importance of artmaking, in whatever form, should not be treated as self-evident merely because the process feels meaningful to us now. So while the short term may be something of a renaissance for creative coding, its long term future is uncertain. At this critical point in AI's deployment, it is important to question how these technologies can bring a deeper and richer sense of creative life to us all, rather than forcing us to conform to a technologically imposed concept of creativity.

References

Arnes, I. 2004. Read_me, run_me, execute_me: Software and its discontents, or: It's the performativity of code, stupid! In Goriunova, O., and Shulgin, A., eds., *Read_me. Software art and cultures conference*. Aarhus: University of Århus. 176–193.

Bennedsen, J., and Caspersen, M. E. 2007. Failure rates in introductory programming. *SIGCSE Bull.* 39(2):32–36.

Braue, D. 2026. University ICT enrolments down in 2026.

Brewster, T. 2026. Anthropic's Claude Is Pumping Out Vulnerable Code, Cyber Experts Warn. Section: Newsletters.

Cohen, H.; Cohen, B.; and Nii, P. 1983. *The First Artificial Intelligence Coloring Book*. Art and Computers. Addison-Wesley.

Dietrich, A. 2004. The cognitive neuroscience of creativity. *Psychonomic Bulletin & Review* 11(6):1011–1026.

Dissanayake, E. 1995. *Homo aestheticus: where art comes from and why*. Seattle: University of Washington Press.

Fry, B., and Reas, C. 2014. *Processing: A Programming Handbook for Visual Designers and Artists*. Cambridge, MA, USA: MIT Press, 2 edition.

García, L., and Vilar, P. M. 2010. The challenges of digital art preservation. *e-conservation* 5:43–53.

Gentleman, A., and Horton, H. 2026. 'Add blood, forced smile': how Grok's nudification tool went viral. *The Guardian*.

Kommers, C.; Duede, E.; Gordon, J.; Holtzman, A.; McNulty, T.; Stewart, S.; Thomas, L.; Jean So, R.; and Long, H. 2026. Why Slop Matters. *ACM AI Lett.* 1(1):3:1–3:6.

Lovable. 2025. Lovable raises $330M to power the age of the builder.

Luxton-Reilly, A.; Becker, B. A.; Cao, Y.; McDermott, R.; Mirolo, C.; Mühling, A.; Petersen, A.; Sanders, K.; Simon; and Whalley, J. 2018. Developing Assessments to Determine Mastery of Programming Fundamentals. In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*, ITiCSE-WGR '17, 47–69. New York, NY, USA: Association for Computing Machinery.

Mahdawi, A. 2025. AI-generated 'slop' is slowly killing the internet, so why is nobody trying to stop it? *The Guardian*.

Malafouris, L. 2008. At the potter's wheel: An argument for material agency. In Knappett, C., and Malafouris, L., eds., *Material agency: Towards a non- anthropocentric approach*. New York, NY: Springer Science and Business Media, LLC. 19–36.

Manovich, L. 2026. The Future of Art? Technical report, Cultural Analytics Lab.

McCormack, J.; Cruz Gambardella, C.; Rajcic, N.; Krol, S. J.; Llano, M. T.; and Yang, M. 2023. Is Writing Prompts Really Making Art? In Johnson, C.; Rodríguez-Fernández, N.; and Rebelo, S. M., eds., *Artificial Intelligence in Music, Sound, Art and Design*, 196–211. Cham: Springer Nature Switzerland.

McCormack, J.; Llano, M. T.; Krol, S. J.; and Rajcic, N. 2024. No Longer Trending on Artstation: Prompt Analysis of Generative AI Art. In Johnson, C.; Rebelo, S. M.; and Santos, I., eds., *Artificial Intelligence in Music, Sound, Art and Design*, 279–295. Cham: Springer Nature Switzerland.

McCormack, J. 2024. Algorithmic art. In Thomas, P., ed., *Encyclopedia of new media art*, volume 2: Artists & Practice. London, England: Bloomsbury Publishing.

Niekerk, C. v.; Vukovic, R.; Ruppik, B. M.; Lin, H.-c.; and Gašić, M. 2025. Post-Training Large Language Models via Reinforcement Learning from Self-Feedback. arXiv:2507.21931 [cs].

Robins, A.; Rountree, J.; and Rountree, N. 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* 13(2):137–172. _eprint: https://doi.org/10.1076/csed.13.2.137.14200.

Skidelsky, R. 2024. *Mindless: The Human Condition in the Age of Artificial Intelligence*. New York NY: Other Press.

Steg, L. 2023. Psychology of Climate Change. *Annual Review of Psychology* 74(Volume 74, 2023):391–421.

Techcrunch. 2026. Layoffs.

Watkins, G. 2025. AI: The New Aesthetics of Fascism.