http://dx.doi.org/10.14236/ewic/EVA2019.59

A Framework for Future Paintings

Stuart Faromarz Batchelor
Artist
Flat 6, 16 Bramah Road
London, SW9 6FJ, UK
sfbatchelor@gmail.com

Painting is a physical practice that engages enactive modes of understanding as well as requiring a heightened visual perception and observational skills. This paper argues that by bringing technology into the painting process, tools can be developed that seek to empower this latent intellect and ability that is normally lost when using a computer in conventional ways. This paper seeks to outline principles necessary to build with software in a dynamic way – leaving space for creative observation, discovery and evolution. Specifically by implementing a Digital Painting Interface and software framework to allow for the manipulation of computer graphics through painting and exploring the differences between these two historically antithetical creative processes. From this, an environment can be developed which allows for the freedom to create using either the artist's hand or the computer's generative algorithms in addition to producing a range of artistic works neither painting nor graphics programming alone would be able to create.

Painting. Creative coding. Computer interaction. Graphics programming. Algorithmic art.

1. INTRODUCTION

While working in the CG Industry as an Interactive Developer making real-time digital installations Stuart Faromoarz Batchelor observed that the space for creative experimentation with software was virtually non-existent. Working with the computer typically involves an execution of an already devised idea, not a sandbox to discover new ones – it is a matter of building on older, well known algorithms and frameworks. This process is great for building reliable and functional software and as a pathway to cost-effective production and commercial viability, but not so good for discovering new ideas.

This is not an issue with the industry's infrastructure but in the way we are conditioned to interact with computers and code. With programming restricted to text files hidden behind glass and by the act of creation being disconnected from the final product, many of our innate human modes of understanding are wasted – severely stifling our range of intellect and therefore the kinds of things we can think about.

This paper explores how we can make our software tools humane and unrestricted in this way by combining traditional painting and programmatic art techniques, amplifying and nourishing our modes of understanding.

2. A DIGITAL PAINTING

Creative programming and painting are two apparently polar opposite artistic practices. One uses direct, gestural movement, material senses and expressive motor skills while the other relies on the conceptual design through symbol manipulation of computational aesthetics; repetition, transformation, parameterisation, visualisation and simulation (Reas 2010).

This presents an unexplored opportunity to progress the painting and algorithmic practice. Just as how artists and scientists combined in the 19th century to expand the painter's palette with new pigments (Gurney 2010), so too could the addition of digital aesthetics expand the current possibilities of the painted image.

Most historically pioneering art was an examination based on the materials used to push logical understanding or overcome the substantiality of the medium. Artists such as Turner (Figure 1), Blake (Figure 2*)* and Rembrandt (Figure 3) communicate metaphysical symbols not just by their static depiction but also by the manner in which they are painted optically, the light behaving on the pigment, derived from their "deep insight into the meaning and limitations of the tradition" (Berger 1973).

By bringing programming into the painting environment the intention is that tools will be developed to facilitate the creation of paintings enhanced by digital capabilities; continuing the painting practice's use of material, it's interactive behaviour and the meaning that makes. Utilising the directness of physical mark making and dynamisms of computation, mitigating the trade-offs when using only a single one of these mediums (Victor 2013).

Figure 1: *J.M.W Turner Rain, Steam and Speed – The Great Western Railway (1844, Wikipedia).*

Figure 2: *William Blake, 'The Inscription over the Gate' (1824–1827, from Illustrations to Dante's Divine Comedy).*

Figure 3: *Rembrandt Harmenszoon van Rijn Self Portrait with Two Circles (1665–1669, Wikipedia).*

2.1 Representations and their media: tools for thinking

Tools are extensions of our intentions. They amplify our capabilities, the things we can already do, and amplify our possibilities, things we could not do before.

Representations, the way we externalise thought through using tools, have been responsible for the intellectual progress of humanity. Enabling us to think new things.

Representations are expressions of ideas; many of the great ideas in history can be traced back to their representations. William Playfair's graphs for data (Figure 4), Gottfried Liebnitz's calculus notation (Figure 5); both new ways of thinking through representation. The ideas live in the representations, and representations live in a medium.

The most widespread media of thought, enabling and then disseminating ideas such as Playfair's and Liebnitz's, was print. In this way the media, the tools we use, dictate the kinds of things we are capable of thinking of. A powerful medium is one that allows for powerful representations (Victor 2015).

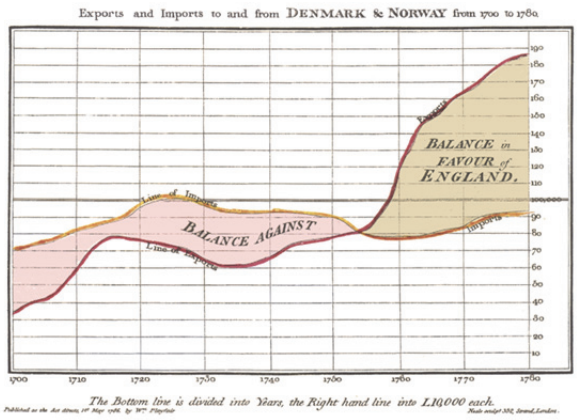

Figure 4: *William Playfair's trade-balance time-series chart, published in his Commercial and Political Atlas, 1786 (Wikipedia).*

$$\int f(x)\,dx$$

Figure 5: *Gottfried Wilhelm von Liebnitz's calculus notation, first published in Acta Eruditorum, June 1686, (Wikipedia).*

However, using any technology is a double-edged sword. While representations draw on certain human capabilities to be effective, they also neglect others. If we fully pivot our representations around a small gamut of capabilities, like we have done with print, the others atrophy, and in a real sense so do we.

> [Physicists] found they needed imagery… a style of thinking based on a kind of seeing and feeling. That was what physical intuition meant.
>
> Feynman said to Dyson, and Dyson agreed, that Einstein's great work had spung from physical intuition and that when Einstein stopped creating it was because he stopped thinking in concrete thoughts and became a manipulator of equations.
>
> Intuition was not just visual but also auditory and kinesthetic. Those who watched Feynman in moments of intense concentration came away with strong, even disturbing sense of the physicality of the process as though his brain did not stop with the gray matter but extended through every muscle in his body (Gleick 1992).

2.2 Optimised paper thoughts: computers as means of production

This kind of knowledge work, of sitting at a desk and statically working with the media, has not changed in form for 200 years, yet has evolved into a lifestyle in the 21st century.

With the advent of the digital age we found it possible to compound the quantity and optimise the quality of what we were already capable of – a brilliant use of computers as a means of production. Efficiency through digitisation develops creativity by allowing more time for exploration and less on the final outcome (Reas 2010).

However this limitless variation of similar artistic artefacts can be confused with limitless creative possibilities. This confusion constrains the creativity that the computer affords, a creativity that comes from a syntactic understanding of the medium, derived from its limitations. We are not free to invent the language of a medium. It is innate and there for us to explore (Youngblood 1989)

If we continue to emulate older media, we continue to think in old ways, we continue to think highly optimised, but essentially paper-based thoughts (Victor 2015).

2.3 Procedural literacy: computers as means of conception

For artists, designers and researchers, occupations pivoting around unknown results and risky processes, using the computer as a means of production, using predefined pipelines and ways of thinking, are less useful. To realise a new or unique vision requires the artist to exceed the limitation of the existing tools (Reas 2010).

The transformation and participation the computer enables is where it's true potential lies; procedural literacy is the utilisation of this potential (Mateas 2008).

Procedural literacy specifies a set of rules for a system, to generate a representation rather than authoring the representation itself (Bogost 2007). It is an indirect, mutable and distinctive material for thinking.

Each programming language is a different weaving of this material. Syntax and grammar define what is possible within that language and the programmer's choice of language is dictated by operating system, budget, aesthetics and writing speed (Reas, 2010). In this way, choice of language and environment encourages a certain, unique and new, way of thinking.

2.4 Pictures behind glass

Though code gives artists access to the technology, allowing its potential for expression to be fully understood (Malina 1990), our current screen based interfaces obfuscate this understanding.

To see a picture is to see our relationship with it in space (Berger 1973). It is a very real and tangible relationship with our sense of touch, movement and volume.

2d is an abstract description of paintings, a way to categorise rather than a reality (Ikeda 2018). They exist in 3d space; they reflect photons from their surface to your eye, meaning they are interactive.

By transforming images into true 2d on our monitors, being neither interactive nor 3d, we destroy this relationship, and with it, fundamental sensory abilities to understand – our tactile, kinaesthetic and spatial intellect are amputated (Victor 2015).

This is a moral argument as the process means we are denying much of what makes us human. It is also a practical one, as we are operating sub-optimally – as if we were an inefficient, incompletely utilised, program. Who knows what we could be capable of if we used all our cores – our wasted senses (Victor 2015)?

2.5 Painting as a tactile distributed system

Panting, in opposition to programming, utilises spatial, kinetic and tactile senses. It does not use abstract equation manipulation but concrete thoughts. It is a muscular and mental process, vividly imagining your outcome, formulating and then finally executing. A process all but physically shared with that of computer algorithm artists.

Understanding painting and understanding algorithms at the component level gives little insight into their effect as a whole. It is as difficult to understand the millions of interactions in a boids system as it is to understand the infinite nuances of painting technique.

> The aggregate motion of the simulated flock is the result of the dense interaction of the relatively simple behaviors of the individual simulated birds (Reynolds 1987).

They are a complex aggregate effect of their components. Both are understood from the direct manipulation of their parameters, simplified abstractions. In painting these are the drawing, the edges, the values and the colours (Schmid 2013). In a simulation they are user defined and dictate the creative potential and the degree of direct authoring the user has on the final outcome. From the emergent interaction with these parameters an understanding is built from the act of doing, the systems are understood not from symbolic, language based, intellect but rather enactive, action based, intellect.

Through parameterisation the processes of programming and painting share similarities, facilitating the creative feedback loop, immediate observation between one's actions and their consequences.

By connecting the parameterisation of physical material to the parameterisation of digital mutability it is hoped that computing may be separated from it's 2d interface; combining procedural literacy and abstraction with the muscular and concrete, the digital expansion of the painter's palette and the physical expansion of computing.

3. IMPLEMENTATION

3.1 Outline specification

A system to create new digital paintings must be; computational, allowing procedural literacy and aesthetics, and direct, through the composition of physical, authored marks and the responsiveness of changes; to the code, the paint and the parameters.

The initial conditions for this are ones that facilitate the participation of the computer in the development of the artistic form as well as involving the physical act of painting to engage spatial and material senses.

The metrics of success of this kind of system are still largely unknown. However, the process made should allow for the creation of works neither painting nor programming alone could create. By exhibiting aesthetics from both, the aim is to develop technology that aids and amplifies the modes of thinking required to make such works. In what follows, I present some preliminary ideas.

3.2 Alla Prima: direct painting and direct programming

Alla Prima is the most interactive form of painting (Schmid 2013), it focuses on finishing the work in one sitting and thus forces control through intuition and reaction. It is spontaneous and emergent, the antithesis of the traditional coding process.

The generative works made within this system will be coded and painted *alla prima*. By aiming to finish a generative work in one session, this impromptu and immediate environment is used as the conditions for the software tools coded; a framework to create *ad-hoc* generative effects – simple, mutable, and expressive.

In this way, the artist codes similar to how they mix paint within the session, reacting and changing based on their previous results, participating with the media. Enabling algorithmic effects to not only

be spatially manipulated through paint but developed and coded alongside the work, mixing the virtual palette and letting new effects emerge from each session.

3.3 Software system

Using C++ code as a programming language allows for a flexible range of digital creative choices, permitting the deep control of CPU and GPU functionality. The open source library openFrameworks will be the graphics server code in order to focus primarily on the usage of graphics algorithms and not on their development.

A custom C++ dynamic compilation framework is used to remedy the creative disconnect suffered from long compilation times and to let the artist have a direct and immediate connection to the digital artwork.

Using this model (Figure 6) for the software allows the program, Application.exe, to continuously run, and be added to simultaneously through the dynamically loaded Content.dll. For example, the artist may want to tweak a value, expose a parameter as a slider or develop an entirely new generative effect and C++ class; these are coded and then compiled into the running program. If compilation fails the program continues and gives the appropriate error messages. If compilation works then the old Content.dll is swapped out for the new – allowing changes and iteration to occur spontaneously and as immediately as possible.

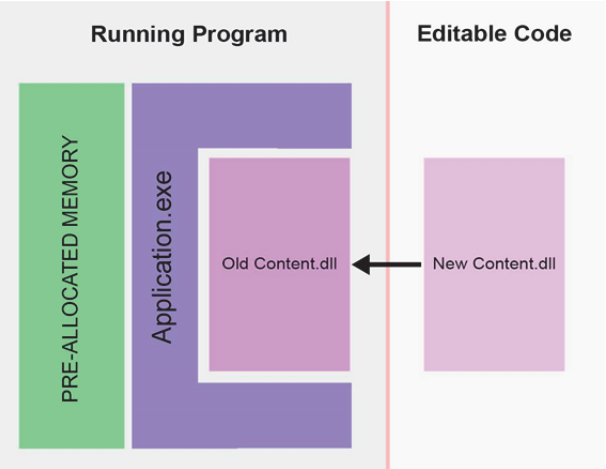

Figure 6: *Live Painting Framework Program Layout. Allowing dynamic loading of code edited and changed by the artist. Image by author, 2019.*

Similar mechanisms are also in place for the dynamic changing of GLSL shaders; threaded file-watchers recompile these elements as they are changed to allow for emergent rendering.

A memory pool preallocated by the main program feeds uninitialised memory into the Content.dll code to minimise memory allocation times and avoid allocation on the heap.

3.4 Painting interface

To connect the physical paint with the virtual code a live camera feed is used with OpenCV to isolate individual paint strokes for use within the system as they are painted (Figure 7). Paint is captured in this way and saved to disk. These images are then used in each graphics layer in the final generative panting – allowing algorithms to be driven by the paint data as well as a means of spatially composing within the virtual image.

Images are captured by calculating the difference between the current camera frame and the previous one – it is up to the artist to decide when to recapture the background plate and start a new brushstroke image and how many images they wish to capture. Parameters are exposed for the RGB channels respectively to allow fine-tuning of the captured region's thresholds. Once the artist is happy with the captured brushstroke they can save it to disk and integrate it digitally or continue to paint.

3.5 Generative effects

From the captured images, generative layers are composed together in a 3d scene. These effects accept either the live image feed or one of the saved images as input as default. The artist is allowed to develop new effects as the session continues within the C++ framework (Figure 8). This allows for creative experimentation and failure, balancing between authored marks and generative effects. The computer participates in the creation, affecting the artist's next decision, to paint more, to code another effect, to experiment with the parameters and images currently available.

This linking of physical interface and virtual possibilities brings an enactive, physical kind of thinking to the programming practice and algorithmic design. The artist does not need to remain in the realm of abstract algorithms nor is bound by the limitations of physical media.

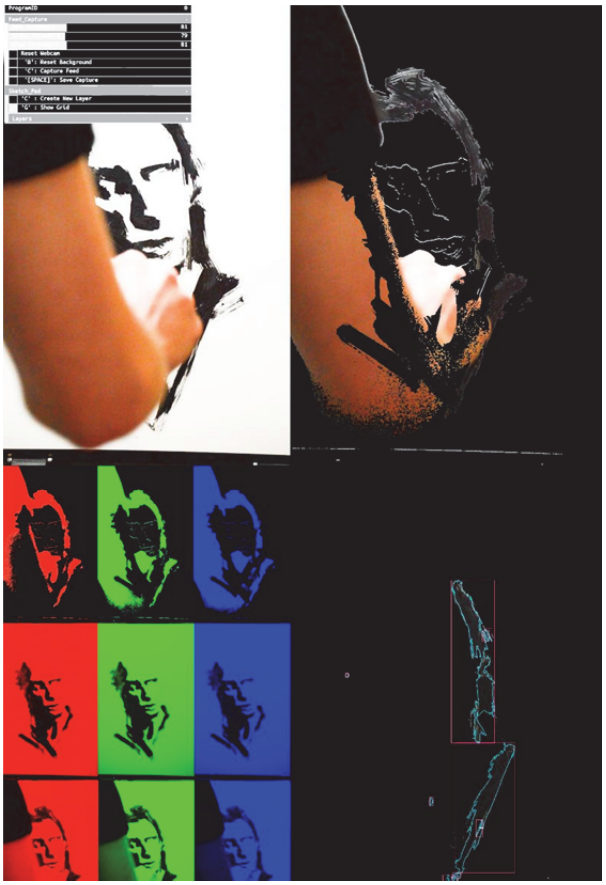

Figure 7: *The Capture Feed Viewer. Top Left: Live Video. Top Right: Live Difference Image. Bottom Left: RGB Channel Difference Images and the Backplate the live view is differenced against. Bottom Right: the current captured difference image. The user may go back and forth between this view and the Composer Viewer (Figure 8) anytime. Image by Author.*

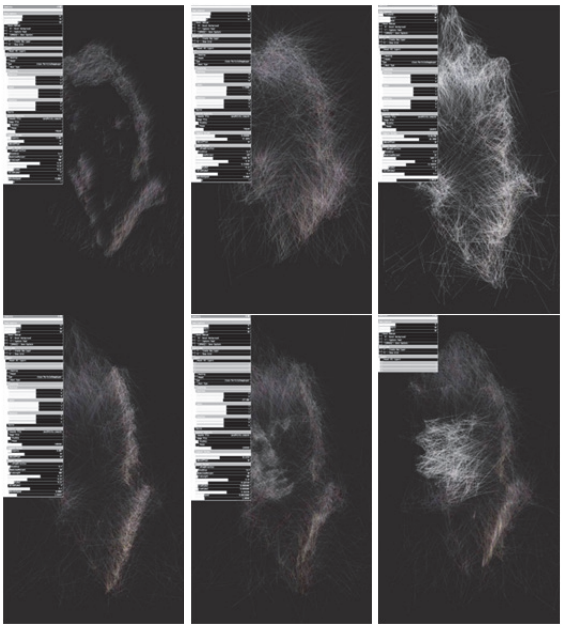

Figure 8: *The Composer Viewer, creative process from top to bottom, left to right. Using an image to drive a generative effect, tweaking parameters, adding more paint tweaking those and so on. Images by Author.*

3.6 Final outcome

The final pieces are generative works with various permutations. They can be presented as stills or time-based works (Figure 9).

The piece(s) created with this process only use 2 types of animated graphics classes driven by the live painted feed – a point cloud simulation with connecting lines and a warping slit scan of the paint. This is used to demonstrate the creative depth and variation – artists author an effect and through the creative session discover unexpected results. This is a known quality of generative practices, however the dichotomy between this and authored brushwork creates a fascinating relationship to be explored.

The specific implementation of this framework is used as a proof of concept; can the use of programming under painting conditions yield new insights. It is a framework, an environment, conditions within which to develop new technology, new understandings of hybrid aesthetics, through creating artwork.

From this, issues with latency, CPU to GPU data transfer, the physical-digital interface, the creative feedback loop, the mixing of code and image, can be addressed, optimised and formalised.

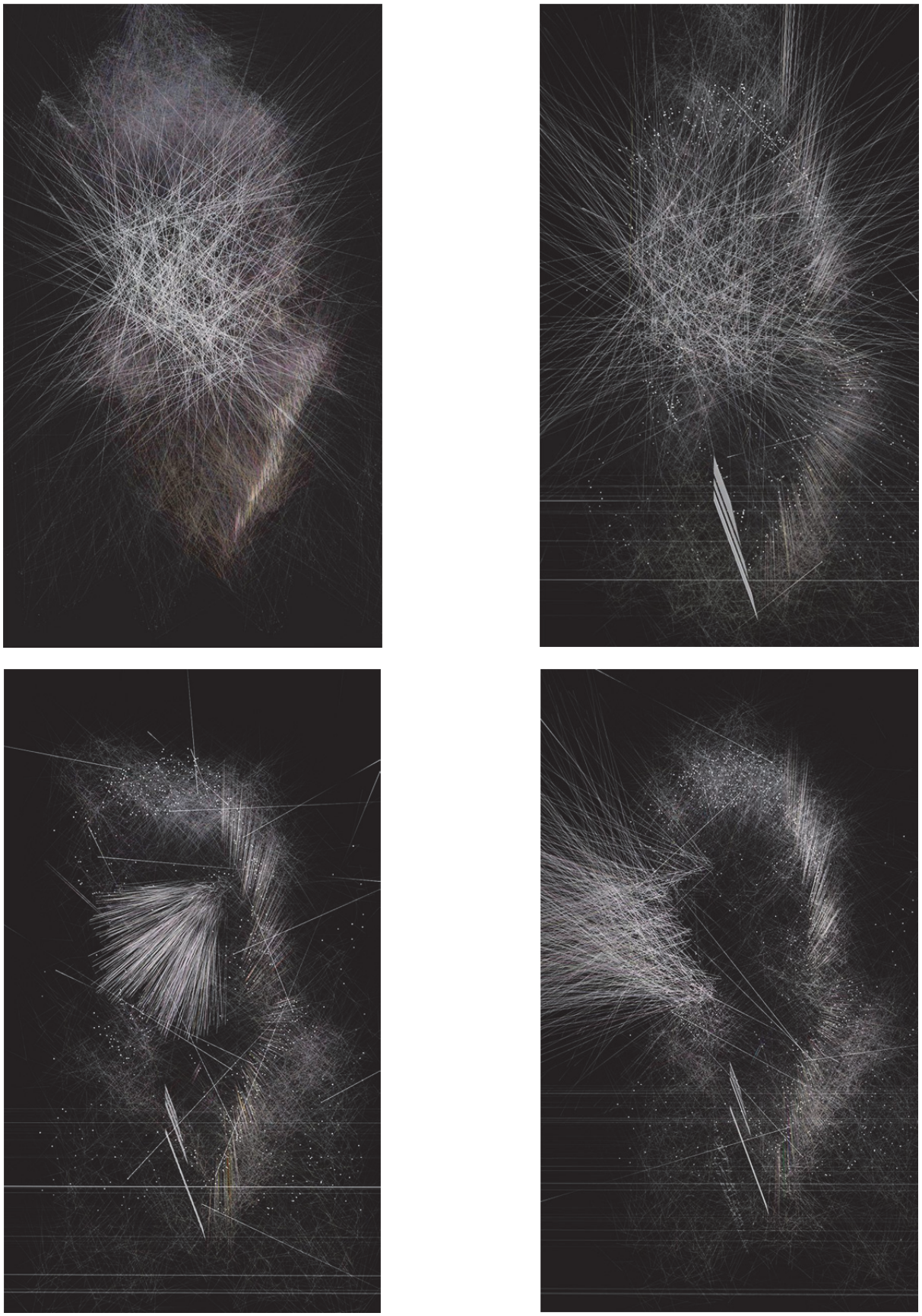

Figure 9: *Stills of the final generative work. That can either be viewed as a still or video piece. With the possibility of adding audience interaction. Images by author, 2019.*

4. CONCLUSION

4.1 Evaluation

The pieces created, though visually and conceptually interesting, are different to the art and controlled creativity such as the original craftspeople made. Despite being systematic in its actions, this process does not have the intention normally observed in *alla prima* and other artistic practices.

Indirection at some point during creative processes to explore is common, but the level seen within this work prevents it from containing authored expression. The pieces do evoke emotions, making them analogous to shapes in clouds and other phenomena – this evaluation as art is open for discussion.

4.2 Future ideas

However the focus of this paper is to develop an evolving process that allows for such chaos, emergence and unintended results. It will only be through future work that control and intention are formalised.

Further research in this area will have benefits towards emancipating our ways of thinking while using the computer by demonstrating that the computer as a medium isn't rigidly connected to it's interface. Instead, computers are still a largely unknown medium of thought, one without a defined form yet, that we must be careful of unconsciously marrying our assumptions with, less it burden the development of future tools and limit our human abilities.

Through the artistic works created, the artist seeks to educate not just specialists in this field but also the general audience, a platform for this conversation to be brought to the public's attention.

4.3 Conclusion

This prototype and its accompanying research is one vantage point to get a glimpse of the larger problems we face with computer interaction. Not only is our present state limited through hardware and software's standard uses, but without placing a great demand on our technology, our ideas and ways of representing them will stagnate; we will become blissfully unaware of our own dogmas.

5. REFERENCES

Berger, J. (1973) *Ways Of Seeing*. British Broadcasting Corp.

Bogost, I. (2007) *Persuasive Games: The Expressive power of Videogames*. MIT Press, Cambridge, MA, p.4.

Casey R. (2010) *Form + Code*. Princeton Architectural Press, p.25.

Gleick, J. (1993) *Genius: The Life And Science Of Richard Feynman*. Vintage.

Gurney, J. (2010) *Color And Light.* Andrews McMeel Publishing.

Ikeda, R. (2018) *Continuum*. Éditions Xavier Barral, p.158.

Malina, R. F. (1990) *Digital Image: Digital Cinema: The Work Of Art In The Age Of Post-Mechanical Reproduction*. *Leonardo* Supplemental Issue 3, p.33.

Mateas, M. (2008) Procedural Literacy: Educating the New Media Practitioner. In Davidson, D. (ed.) *Beyond Fun*. ETC Press, Pittsburgh, PA, p.67.

Schmid, R. (2013) *Alla Prima 2: Everything I know about Painting.* Stove Prairie Press.

Reynolds, C. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model*, SIGGRAPH '87 Proceedings*, p.25.

Victor, B. (2013) *Drawing Dynamic Visualizations* https://vimeo.com/66085662 (retrieved 27 February 2019).

Victor, B. (2015) *The Humane Representation Of Thought.* https://vimeo.com/115154289 (retrieved 5 December 2018).

Youngblood, G. (1989) *Cinema And The Code*. Leonardo Supplemental Issue 2, p.27.