Y. Ikeda, C. M. Herr, D. Holzer, S. Kaijima, M. J. Kim. M, A, Schnabel (eds.), *Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA* 2015, 829–838.

THE OTHER DIGITAL

A study between algorithmic design and glitch aesthetics in digital architecture

MATTHEW AUSTIN[1] and GAVIN PERIN[2]
[1 2] *University of Technology Sydney, Sydney, Australia*
{matthew.austin, gavin.perin}@uts.edu.au

Abstract. The paper compares the implications of glitch aesthetics as an alternative digital design process to the more the commonly used algorithmic processes. It will argue the synthetic nature of architectural production in the digital age is used typically to privilege the representation of *form* through lines and curves, while the production of glitches rely on the image. This reliance on the image means that the pixel comes to the forefront as a possible new medium of architectural drawing. This paper therefore aims to outline the advantages and problems with using 'glitches' within architectural production.

Keywords. Glitch aesthetics; Processing; theory; algorithmic design; process.

1. Introduction

This paper offers an introductory comparative discussion of the architectural opportunities opened by two different digital-art movements that arose around the turn of the century. The more established algorithmic movement turned to 'Processing'; an open source software developed in the early 2000s by Casey Reas and Ben Fry. The other contemporaneous, but far more marginal practice is the glitch aesthetics movement. There are two important benefits of such a discussion. First, digital architecture has become increasingly instrumentalised by privileging 'Processing' and algorithmic design over glitch aesthetics. The second conclusion is that glitch aesthetics offers an alternative mode of architectural production that expands the representational and formal limits that currently define and constrain contemporary digital-architecture practice.

It is hardly surprising that glitch aesthetics remains a marginal art and design practice given that the glitch is seen as a system fault (Davis, 2011, p.212). In contrast, the algorithmic design process forms a type of knowledge that converges towards known and controllable conditions. The architectural value of the algorithmic process is that it easily meets the accepted definition of research. This is because the algorithmic process intrinsically forms demonstrable and duplicable knowledge. In contrast, the glitch has been neglected as a field of research because its output cannot be easily instrumentalised. The inability to predict and control the process means it circumvents known formal outcomes. It is worth remembering that Greg Lynn's (1999, p.39) promotion of the animated diagram was based on the idea that it was the process and not its formal outcome that was to be "instrumental before it is representational." In this sense the rhetoric surrounding the early development of digital architecture was pursued *precisely* because the process avoided known disciplinary forms.

2. Convergent and Divergent Coded Processes[1]

The algorithmic tool, 'Processing', enables the author to choose specific representational forms with a high degree of certainty. In comparison glitch aesthetics is an unpredictable mode of production. Whereas the algorithm leads to a convergence between authorship, drawing and image, the glitch is less predictable and involves a process of error and trial. Irrespective of these differences, the important similarity between these two movements is their shared dependence on images. However, 'Processing' translates a series of drawing commands into images, while glitch aesthetics transforms the formal arrangement of images by manipulating its binary structure.

'Processing' translates the abstract nature of coding into an analogy of drawing. Rather than physically drawing a line from (x1,y1) to (x2,y2) the user achieves the same result by writing 'line(x1,y1,x2,y2)'. The designer writes a chain of commands and functions to create a direct equivalency between gestural movements of the hand and programming. To the novice, the advantage of 'Processing' is that it easily emulates pre-existing drawing techniques (Carpo, 2011, p.36). This capacity to emulate established drawing techniques resulted in the application of 'Processing' to replicate traditional architectural modes of representation. With reference to computer science, architecture's representational palette was soon extended to include concepts like logical structures, decision trees, loops, recursion, behaviours, functions, animation, and three-dimensionality. 'Processing's' interface helped embed algorithmic design as a procedural basis underpinning digital architecture. Importantly, the new drawing types depended on an algorithmic logic where the direct analogy to past methods of drawing was superseded with an entire-

ly new type of algorithmic drawing. The algorithmic drawing toolset, which includes 'Processing', quickly executes textual code to produce a visual output. This visual output allows the designer to make decisions that converges to an ideation of a particular outcome. This convergent process allows the designer to use each procedural step to come closer to a desired outcome. The possibilities of this new approach to architecture has introduced and supported an interest in parametricism, complexity, emergence and other such related behaviours.

The impact of 'Processing' is reflected in the range of techniques now used within digital architecture. From the development of Grasshopper and the accessibility of VB, Max Script and MEL in pre-existing modelling software,' Processing' has forged on a vast array of new disciplinary methods and processes. The big difference is that unlike 2001, where the designer would have had little mastery over the complexity of the code and the process, the formal consequences of today's methods and processes are well established. In the early years of the century the outcomes of the technique were explorative and unknown, while today digital design is now an established mode of production operating within a known family of outcomes.

It is an interesting historical fact that the development of glitch aesthetics paralleled 'Processing' but had its roots as a sort of digital 'counter-culture'[2]. Instead of looking for mastery and control over digital processes, glitching searched for its 'failures'. Although in a technical sense a glitch represents a real failure of a system, glitch aesthetics has 'institutionalised' these failures through the use of specific techniques and software. Techniques– such as 'data bending'(Figure 1) – and software - such as 'hex editors' and 'rom corruptors'[3] – force 'failures' by giving rise to a series of synthetic glitches coined as 'glitch-alike's' by Moradi (2004, pp.8-10). Both glitches and their synthetic counterparts share a series of visual characteristics that make them recognisable as an aesthetic, namely their fragmentation, repetition, linearity and complexity (Moradi, 2004, pp.28-33). The most important difference between the glitches and the 'glitch-alike' is that the latter is institutionalised to glitch on command, thus letting it become a viable design technique.

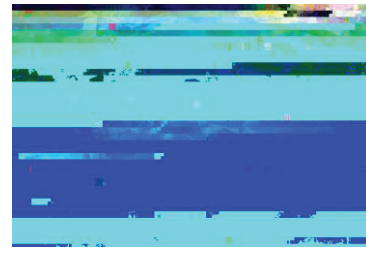

Figure 1. Example of data bending being used to create a glitch-alike.

An important similarity between glitch aesthetics and 'Processing' is that they both involve a digital process with a feedback loop between text and image. Glitching, however, takes small variations within the ASCII, hex or binary textual representation of an image to create vastly divergent formal arrangements. Every step in the glitching process generates an uncontrollable and unpredictable result, forcing the designer to procedurally move sideways as they evaluate unexpected outcomes. Glitch aesthetics demands the designer adopts new design reasoning. The convergence of algorithmic processes involves an active interplay between code and designer, whilst the glitch-alike requires the designer to be reactive. Unlike 'Processing's' 'convergent' outcomes, the core difference and, therefore, potential of glitch aesthetics lies with its capacity to generate 'divergent outcomes'.

3. The synthetic within digital production

Architectural production is now digital – albeit not in the sense that it is 'digital architecture' but that it is now more likely to be produced through pixel arrays on monitors and printers than by pen and paper. All modes of architectural production are transformed from a continuous mode of representation to discrete and pixelised analogies of desired objects. The move away from pen and paper is important because it severs the embodied intimacy of the manually drawn line, while the digital line is heavily mediated, both in its mode of production and in its mode of representation. The digital line not only involves a disembodied movement where gesture is forever separated from the appearance of what is, ultimately, a virtual, immaterial object. It also synthetically re-presents the geometry of the line through an arrangement of discrete pixels. The structuring of the digital environment as a pixel array or grid means it cannot truthfully produce a mathematical rendition of a line (Figure 2). It only produces the approximation of a line. This is a profoundly synthetic mode of production as the designer only ever sees an analogy generated through binary logics and algorithms that process the mathematical line into an image. All visual-digital acts are synthesised by the computer into 0s and 1s, these are then transformed into RGB or CMYK pixel arrays. While the digital interface draws objects, it does so by organising the pixel grid to simulate the constituent lines of the object. (Klette& Rosenfeld, 2004, p.15) Like the digital image, the resolution of the media, or its pixel density, serves only to give the appearance of the object. The digital environment is deceptive inasmuch as it provides the designer with the illusion of a line. Every time the designer draws a line, he or she is asking the computer to synthesise it. Consequently, the geometry of the pixel grid means digitally produced lines, curves, and forms are 'synthetic' simulations of their analogue counterparts.

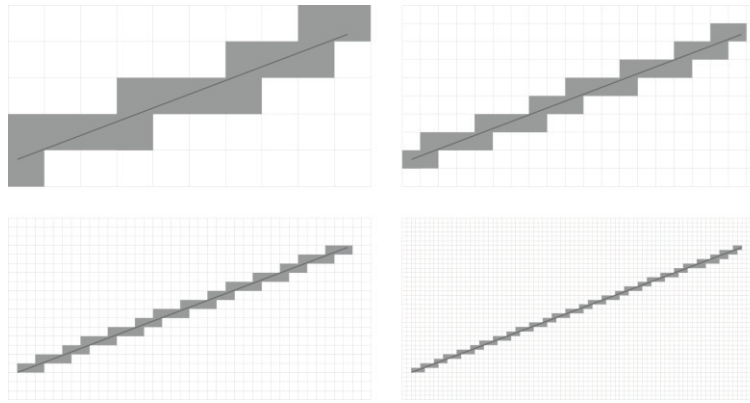

Figure 2. Example of how a line can be pixelated using a box counting method.

All acts within the computer are algorithmic, whether they involve 3D modelling, computer programming or glitching. All digital techniques manipulate binary digits in order to create an outcome. The critical difference between glitching and more commonplace algorithmic design processes is that glitching acknowledges the direct relationship between the pixel and its binary representation. The manipulation of an image's binary code requires that a direct transformation of a pixel grid is a valid method of production. While the computer privileges the more instrumental algorithmic design processes, the glitch adds to the repertoire of digital techniques without need of the illusory representational mediation of the object as delineated by lines. As long as digital architecture is concerned with images, glitching's capacity to operate on the digital image in a non-synthetic way gives the technique validation within the production of digital images.

There is another important implication of a glitch design process; the glitch's inability to be predicted or controlled by the designer challenges the traditional concept of authorship. This altered circumstance requires a rethinking of what constitutes authorship within the digital design process. This lack of controlled authorship suggests that glitching can offer a new array of digital techniques that sit outside of architecture's canon.

4. The role of families within digital-design processes.

Jeffrey Kipnis's (2008, pp.197-201) short essay, 'A Family Affair' outlines how the benefit of the digital design process is that it shifts the issue of authorship from the production of novel form to the production of 'families'. The family operates through the capacity of an algorithm to produce variation within a defined set of possible outcomes. Formal variation, as a product of a shared algorithmic genealogy, generates meaningful differences by edit-

ing the inputs of the algorithm. Within this framework the role of the designer is to select the member of the family that is the ‘best’ suited to selected architectural criteria (aesthetics, performance, etc.).

Stephen Harfield(2014, p.124-131) argues that the notion of families aims to impede the capacity for authorship. The benefit of the family is that it allows the designer to explore both a series of solutions and 'breed' formal outcomes. The attraction of this approach is that designing is not limited to a single outcome or weighted down by the ideological issues surrounding authorship. The algorithmic process understands form as the exploration and understanding of family resemblances. For example, a swarming algorithm will always generate outcomes that resemble ‘swarms’ and a branching algorithm will always generate outcomes that resemble ‘trees’. In both instances the designer generates a ‘good outcome’ by using the algorithm to control and manipulate these family resemblances. As Patrik Schumacher's (2012, p.1) advocacy of formal order demonstrates, the parametric process (which is here understood as a subset of algorithmic design) is profoundly strategic. This is why Schumacher’s ‘parametricism’ enforces a formal homogeneity that prioritises topological deformation rather than privileging the design of specific objects. Significantly, this example demonstrates how mastery of algorithmic design is measured through the control of generative processes rather than the direct authorship of a specific form. In this framework ideas of predictability, knowledge and foresight are paramount in the production of architectural form.

It is worth noting that glitching and algorithmic design work in the same way. Both have inputs, both transform these inputs, and both generate an outcome. Glitching is also able to generate families, inasmuch that outcomes formally resemble others from the same technique. Furthermore, different techniques can generate different sets of families. However, as already mentioned, the unpredictable nature of the glitch makes it less likely to be instrumental, thus the concept of families lacks practical application. If algorithms generate forms of synthetic-architectural representation, glitches produce unpredictable-direct representations of ‘intent-less’ binary transformations. Glitching differs from the algorithm by not generating ‘workable’ families that can be assessed to ‘solve’ a design problem. While the algorithm requires itself and a development environment to run, the glitch hijacks and re-appropriates an external file. This means the glitch’s chaotic nature and immediate relationship to the pixel array requires a rethinking of what is being transformed and what its formal implications are.

5. A 'glitching' process for formal design production

What then is required for the glitch to become capable of architectural production? Glitching would require that binary transformations to be capable of creating synthetic-architectural representation. Moradi brings to our attention this important point:

> The visual glitch is an artifact resulting from an error. It is neither the cause, nore the error itself, it is simply the product of an error and more specifically its visual manifestation. It is a significant slip that marks a departure from our expected result. (2009, p. 8)

Thus the use of glitching for formal design production would generate artefacts that have no explicit strategic-formal ambition. The glitch produces unusable material that requires further processes to produce three-dimensional form. The development of three-dimensional form (Figure 1) comes from processing a glitch through an image analysis algorithm. An obvious extension of this investigation would examine a glitching of the coded representation of a 3D modelling file. In this case the increased complexity of 3D modelling software makes it more difficult to open a glitched binary file. Furthermore, as the work of Mark Klink[5] shows, the successful data bending of ASCII files results in predictable geometric transformations. Klink shows how these types of ASCII transformations only edit the vertices of mesh geometry. Effectively this 'glitching' of the figured form transforms the model by replacing particular values in a text editor. This 'glitch' can be synthesised with a 'for loop' and series of 'if statements' within 3D modelling software, thus transforming the 'glitch' into an algorithmic process. This makes its glitched nature trivial[4]; it is a redundant and predictable process. This is in contrast to the outcome in Figure 1, which has no relationship to the figure in the image. The unique aspect of the glitched image is that, like any image, a second, extrinsic process is needed to move from two to three-dimensions.

Clearly, the comparison between the glitching of two and three-dimensional files demonstrates how the image file requires another process to give the glitch form. The question then is whether it is possible to glitch the glitch's own representational form. Can a glitch be glitched? Such a process must start with an image for the glitch to hijack. This leaves three choices; either to continue glitching, to 'save'[6] and normalise the error into a file which is glitched again, or to interpret the glitch. These choices condemn the image glitch to a chosen plane, allowing the designer to choose what, architecturally, the 'drawing' represents. The process of generating form is therefore a matter of selecting a glitch, choosing its plane and synthesising it through an algorithm. Once this process converts the glitch into three-

dimensions, the designer has the option of beginning the process again. This time they have more choices in the experimental parameters. For example the designer can control colour-spaces (e.g. CMYK, RGB), file formats (e.g. .JPG, .PSD), and resolutions (e.g. 1K, 4K). Although the design process has been expanded, at no stage does the designer have any more or less control over the outcome. The capacity to explore the glitch cannot be used to hone form to achieve a 'desirable' outcome. Glitching glitch-architecture gives rise to a much larger and more difficult design space.

It is important to add that although glitching is bound by the pixel grid, the introduction of rotation and scale means architectural form need not conform to the pixel grid. For example, the conversion of a glitch into a plan could be glitched again by taking sections in a direction that does not directly relate to the planar directions of the original image (Figure 3). This gives rise to a form that has multiple spatial axes (Figure 4). Of course, this procedural step is merely one of a vast array of possible techniques that can be used to generate unpredictable formal results.

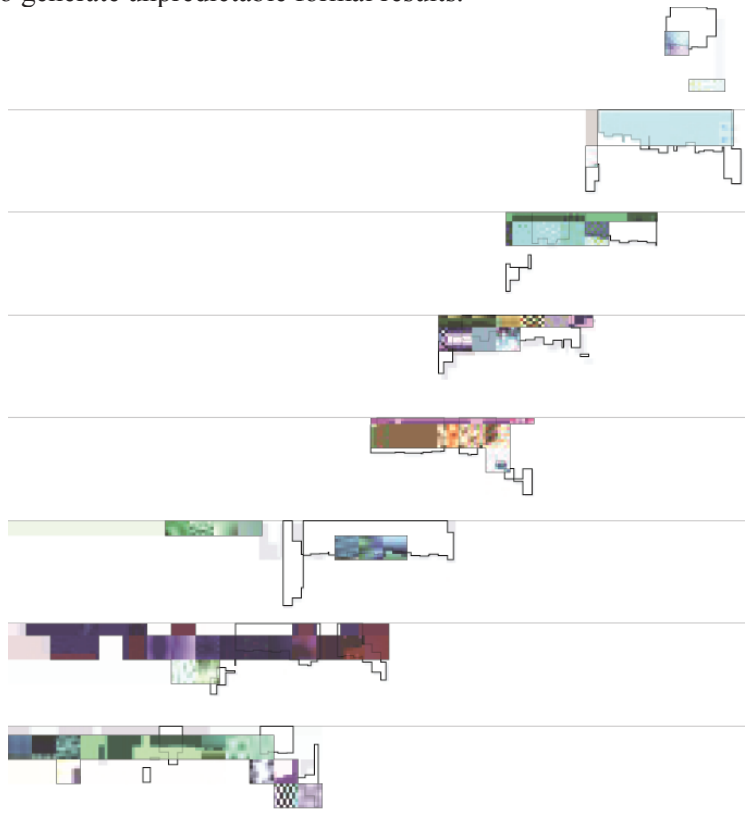

Figure 3. Example of the result of glitching sections of an interpreted glitched plan into form.

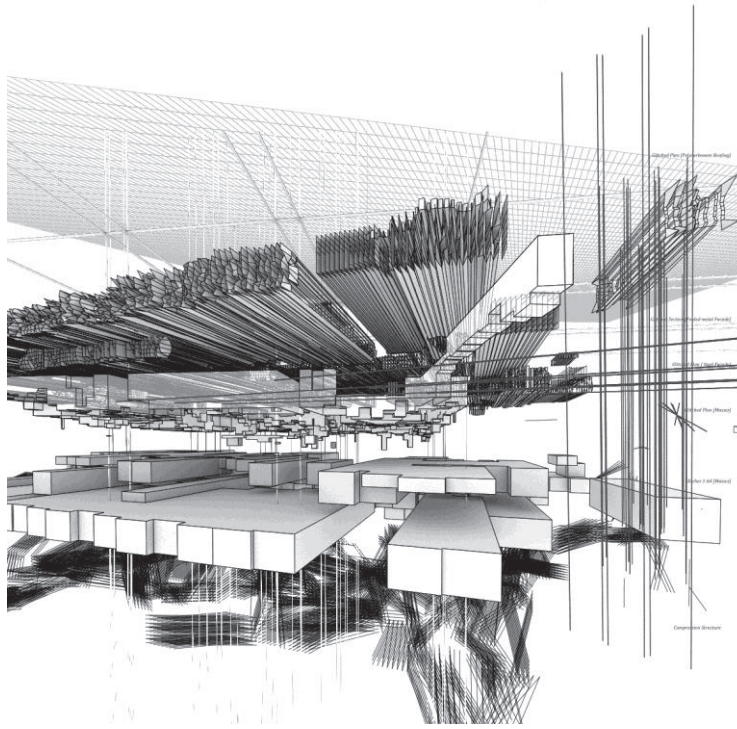

Figure 2. The result of varying algorithm, scale and rotation between glitches.

6. Conclusion

'Processing' and algorithmic design have given the designer the ability to predict and control a design process. The agency of the algorithm is that one can refine it to converge upon a desired design solution whilst working with the computer on the production of synthetic representational forms. It is clear that the maturity of algorithmic processes has produced a vast array of rich and complex images, however, this process always operates on a figured form, which here is understood as an object and a form of drawing. Glitch aesthetics instead is an underexplored process that exploits immediate transformations of binary information outside that of the figured form. In this way glitch aesthetics is a new procedural avenue of formal exploration within digital architecture. This new avenue is related to the algorithmic, however, the vital difference being that the designer cannot hone in on a particular solution. By extension, the glitch requires a rethinking of 'control', 'choice' and 'chance' within the production of digital architecture. Consequently, further research into glitch aesthetics would need to look at the issues around

the synthetic nature of pre-existing modes of architectural production. The glitch does not dissolve the validity of authorship and control, but it does introduce a new framework for critical thinking about the embedded, but all too often sublimated, binary logic of digitally generated architecture.

Endnotes

1. In this instance, the phrases 'convergence' and 'divergence' take as their source an analogy to a strict mathematical definition. For more information consider the properties of mathematical sequences and series.
2. This counter-culture status is based on the peripheral nature of the research. There are very few *academic* sources on the topic within the context of design. Further the primary source Moradi (2004) is an honours thesis, which outlines some important broad concepts but doesn't delve far enough into the topic to generate a rigorous academic discussion.
3. See <http://vinesauce.com/>.
4. The design interest in glitch aesthetics is its divergent and unpredictable nature and algorithmic design's strength is its convergence. If there exists an algorithmic-design process that is logically equivalent to a glitching process that glitch is convergent and therefore trivial.
5. See Klink's AASCI glitches of 3D modelling files at <http://www.srcxor.org/blog/> a good example can be found at <http://www.srcxor.org/blog/wp-content/uploads/photo-gallery/thumb/GlitchHead12small.jpg>.
6. Saving or exporting the file requires to a reprocessing of the files information content, thus changing the glitch from a representation of an error to the actual expected stored information within the image file.

References

Carpo, M.: 2011, *The Alphabet and the Algorithm*, MIT Press, Cambridge.

Davis, T.: 2011, Precise Mishandling of the Digital Image Structure, *Design, User, Experience, and Usability. Theory, Methods, Tools and Practice: First International Conference, DUXU 2011*, Orlando.

Harfield, S.: 2014, Cage, Chance and Architecture: Distancing the Formalizing Agent, *Centre 18: Music in Architecture-Architecture in Music*, Oregon, Quality Books, 124-131.

Kipnis, J.: 2008. 'A Family Affair', in *Greg Lynn FORM*, Rappolt, M. (ed.), Rizzoli, New York.

Klette, R. & Rosenfeld, A.: 2004. *Digital Geometry: Geometric Methods for Digital Picture Analysis*: Elsevier, Amsterdam, Boston.

Klink, M: "srcXor – Art and Computers". Available from <http://www.srcxor.org/blog/> accessed (1 December 2014).

Lynn, G.: 1999, *Animate Form,* Princeton Architectural Press, New York.

Moradi, I.: 2004, 'Glitch Aesthetics', Honours thesis, University of Huddersfield, Huddersfield.

Moradi, I.: 2009, Introduction *in* Moradi. I. Scott, A. Gilmore, J. and Murphy. C. *Glitch: Designing Imperfection*, Mark Batty Publishing, London, 8-9.

Schumacher, P.: 2012, 'Parametric Semiology – The Design of Information Rich Environments.' (Online copy of text published in *Architecture In Formation – On the Nature of Information in Architecture,* ed. Lorenzo-Eiroa, P. and A. Sprecher (Routledge, Taylor and Francis, New York, 2012) Available from: <http://patrikschumacher.com/Texts/Design%20of%20Information%20Rich%20Environments.html> (accessed 15 September, 2014).