1 Scripting cultures

Digital design is now fully assimilated into design practice, and we are moving rapidly from an era of being aspiring expert users to one of being adept digital toolmakers. This primer looks at this transition and acts as a first resource for all those curious about developing a higher-order engagement with the computer, but with an eye to critical enquiry rather than geekdom. *Scripting Cultures* considers the implications of lower-level computer programming (scripting) as it becomes more widely taken up and more confidently embedded into the 'design process'.

Scripting is a rather loose term by any definition and in this primer can be taken to mean computer programming at several levels. For the novice dabbling at the more accessible end of the user spectrum, scripting is the capability offered by almost all design software packages that allows the user to adapt, customise or completely reconfigure software around their own predilections and modes of working. At its most demanding for the emerging connoisseur, scripting can refer to higher-level computer programming where, in the 'open-source' environment, 'libraries' of functions can be combined with preconfigured routines (algorithms) as a means to produce manufacturer-independent digital design capability.[1] At its simplest, therefore, scripting affords a significantly deeper engagement between the computer and user by automating routine aspects and repetitive activities, thus facilitating a far greater range of potential outcomes for the same investment in time. Along with extending design experimentation, scripting can also be the antidote to standardisation forced by an ambition to lower production costs, rather than any more sophisticated motivation: the previously elusive opportunities for multiple versioning and bespoke production can now be considered more seriously through the use of scripting. This new territory combines with emerging affordable digital fabrication technologies taking advantage of the improving file-to-factory protocols. This has the potential to free up the designer to spend more time on design thinking. Authoritative customisation of the 'black box' affords the designer opportunities to escape the strictures inherent in any software – by definition in ways not thought of by the makers, otherwise it would be an existing capability.

Strictly speaking, to script is to write a screenplay or dialogue from which a play might be performed. Setting down the language from which others perform is presumably why the word 'scripting' has entered the lexicon of software users, and in computing, 'scripting language' is often synonymous with 'programming language': it is the means by which the user gives highly specific instructions to the computer with which they are interacting.[2] At a semantic level it is possible that the designer is less likely to flinch at the term scripting than they might at the term programming, for it is quite clear that most of the designers who use computers as a core part of their digital practice do not automatically turn to programming to form part of their repertoire. By not doing so users at once place their entire trust in the software engineers in the expectation that those anonymous collaborators have thought through all that might be wanted by the designers, just as they are conceding that what seems on occasion endless manual repetition is an acceptable use of their time when they could otherwise have been seeking some degree of automation. Software modified by the designer through scripting, however, provides a range of possibilities for creative speculation that is simply not possible using the software only as the manufacturers intended it to be used. Because scripting is effectively a computing program overlay, the tool user (designer) becomes the new toolmaker (software engineer).

Motivation to contribute to the scripting Zeitgeist

Why write a book on scripting in terms of culture when it may only be a passing 'style'?

Scripting is not new to design and was originally considered the task of a specialist; being taught to program computers in any way was not part of a design education. It is only recently that there has been a sufficient groundswell of interest to prompt change. Many designers are now aware of the potential of scripting, but it is still seen as a difficult arena to enter. This book joins the growing list of titles that have emerged over the last few years offering routes for designers into the world of scripting. This primer could treat scripting as a technical challenge requiring clear description, guidance and training, but instead leaves that task to others and focuses on motivation. Crucially, *Scripting Cultures* offers some answers to why the designer would script in the first place, and considers some of the cultural and theoretical implications along the way. Scripted code readily changes hands and, in terms of potential risks, it could become a cloning tool for less talented operators to mimic their masters. In contrast and in terms of opportunities scripting ought to be the opposite: a liberating design force unleashed by the Internet combining with the innate human desire to share knowledge; the *live hive* in which the collective critical mass is far greater than the sum of the individuals.

This book offers three important differences to other titles and seeks to provide complementary material rather than dig away at essential points of difference. Firstly, the predominant theme is 'cultural' rather than 'practical', 'computational', 'artistic' or 'generative'. It enquires into the cultural implications of scripting and asks what are the cultures of scripting as, emerging in myriad ways, they more conspicuously influence the designer's toolkit. Secondly, on an associated website hosted by Wiley, this book directs readers to substantial worked examples of code adopted in some of the most widely used modelling software for some of the project work described. In these samples, every parcel of code is provided for the reader with basic explanations as to what it does, and why it appears where it does in the script. Thirdly, for the designer who does not want to work on top of manufacturers' software packages, a worked example will be laid out as proof of concept using freely available open-source software, offering the experienced designer complete freedom in the way that they operate.

The book is organised in three sections. It commences by considering the fundamentals of computing and design as a means of capturing some of the spirit at the time in which I am writing. More critically still, the primer moves on to distil the thoughts of many of the current generation of key scripters into an action plan as first steps to deducting some kind of collective '*quo vadis?'*. As essential background for the aspiring novice, this is followed by a succinct consideration of what others have written on the subject and the principal choices of approach available to the scripter but not discussed in detail in this primer. The final section is a series of five commentaries around two decades of my own endeavours in the field. This is preceded by an account of a set of design preoccupations as the context for this personal line of enquiry. I hasten to add that I am not an expert scripter myself, but the longevity of my involvement has led to a series of insights shared here, to serve at least as provocations if not as an actual *modus operandi* for general adoption. The introduction to the five projects, 'Dimensions', covers my motivation to start scripting and commences the general theme around the primacy of first idea, then ideation, conceptual development and, ultimately, logical exposition. Technique, which could so easily have been foregrounded, has instead taken a back seat. Rapid changes in software and emerging alternative computational design approaches enforce this 'knowhow' reticence, not least to avoid the primer becoming obsolete soon after publication. The intellectual challenges of ideation and the logic of digital design discovery alone are sufficient motivation for an exposé of this kind, transcending any need for a developed discussion on the relative ease or difficulty of particular software or open-source opportunities in our current era.

Necessarily, I bring my own baggage to this account, but in conducting my research for the primer it became clear that my situation, that of an autodidact, is not the exception that I had perhaps assumed. This is one of two reasons why it is titled *Scripting Cultures* and not simply *Scripting Culture*; this is to say, innovative scripting designers do not want to be locked into a single defining culture. The other motivation to pluralise 'culture' is to reinforce the message that the book is not about identifying and tuning into the latest swarm phenomenon, and placing an umbrella over a 'new' design movement or style. Scripting, as an approach to computational design, offers access to whole new ways of exploring design, but design remains always at the core. It is clear to me that so long as coders follow their own leads, there will be many scripting cultures. Scripting is especially prominent now because of the difference between digital design pre scripting and digital design now, as scripting steps temporarily into the limelight as a burgeoning new creative force, as agents of change often are. Its assumed novelty will pass, no doubt, but we are still at the stage of largely uncritical engagement. This primer will play its part, I hope, in encouraging digital designers to take up scripting while still continuing to think for themselves as designers first, as they always have done.

References

1 At the highest level still (short of designing one's own computer) there is 'machine code', the actual machine operating language of a computer with which the user-engaged operating language negotiates. We will not be going anywhere near there.
2 Some software includes opportunities to engage through scripting at several levels of sophistication, from macro writing, scripting, and programming via a SDK (software development kit).