The Spaces, Mobilities and Soundings of Coding

to appear in Hartmann-Petersen, K, Perez Fjalland, E.L., and Freudendal-Pedersen, M (eds) *Experiencing Urban Networked Mobilities*, Routledge.

Sung-Yueh Perng
NIRSA, National University of Ireland Maynooth, County Kildare, Ireland.
Email: sung-yueh.perng@nuim.ie

The Programmable City Working Paper 18
http://progcity.maynoothuniversity.ie/
31st May, 2016

From coded spaces to spaces of coding

The mobilities of people, technologies, resources and waste that constitute urban spaces are enacted and governed by complex, diverse and often hidden paths, rules and regulations (Urry, 2007, 2014). Increasingly, the interconnectedness of these mobilities is enhanced by computational intelligence to monitor, measure and preempt these flows at fine levels, creating new hopes, monsters and fears when engineering networked cities (Büscher et al., 2016). In this process, software and its data processing capabilities have become pervasively embedded in urban sociotechnical systems, which generate insights about cities from knowledge practices that are shifting as a result of the 'data revolution' (Kitchin, 2014) and preconfigures our experiences of living, working and travelling in coded space, without conscious reflection and sometimes creating new uncertainties (Graham, 2005; Kitchin and Dodge, 2011; Perng and Büscher, 2015). Despite its influences on urban spaces, software has largely remained a black-box for critical inspection, comprising specialist knowledge and reasoning and guarded against closer examination

through compilation into executable files and by legal and intellectual property rights. The practices of compiling code often escape the public (and academic) gaze because the spaces where coding takes places are behind the badge-protected gates of corporate buildings. Alternatively, these spaces can be distributed online, or located in programmers' private rooms, and are difficult to access.

However, there has been an energetic mobilisation of code writing beyond these legal, commercial, professional and knowledge confines. The mobilisation of software code for civic purposes, or civic hacking, has emerged as important everyday urban experiences. Regular events are organised by voluntary organisations such as Code for Ireland and Code for Boston, where people with programming skills can meet with community members or government officials to find out how their skills of building mobile phone applications (or apps), creating websites or generating insights from open data can be appropriated to address local issues, such as estimating wait time at an immigration office in Dublin (https://myq.ie) or alarming delays of public transportation services in Boston (https://twitter.com/mbta_alerts).

Opening up the black-box of software writing, however, leads to more issues that require further examination. At civic hacking events, the articulations and discussions about local problems can influence the design and focus of the apps or websites to be built. The knowledge and experiences that the participants at these events have about urban living, can engender immediate effects on how the problems are perceived and analysed when considering whether any solution can be developed for them. With these issues surfacing when coding processes become more open and accessible than they used to be, 'spaces of coding' also require critical attention to analyse the social, spatial and technical processes through which different kinds of urban experiences and knowledges become related to and translated into software solutions (for example, Lodato and DiSalvo, 2016; Maalsen and Perng , 2016). To do so, the chapter focuses on the fundamental process of software writing: transforming everyday experiences into sequences of code by observing the social and sonic activities happening at a workshop where participants were learning a new programming language. The focus on the introductory workshop also sheds light on the complexity and difficulty of interacting with software code that were already there at this early stage of the interaction. Accordingly, future research has to attend to much wider, and

complicated, social, embodied and collaborative practices that reshape the relationships between code, spaces of coding, civic hacking and future cities.

‘Code-alongs’, coding and soundscape

The chapter draws on my research ‘code-alongs’: participation in coding sessions to learn how to code or contribute to civic hacking initiatives, during 2014 and 2015 in Dublin. The discussion focuses on an introductory workshop organised by Coding Grace (https://codinggrace.com/) on ‘Processing’, a programming language for the visual arts, in which I participated as a programming beginner. At the workshop, I followed the tutor to write code to create canvases and simple shapes before adventuring into the visualisation and animation of 2D and 3D objects.

These coding sessions, including the civic hacking events I attended, can be quiet. At the workshop, the tutor explained the way Processing works, reasons and acts, and the participants replicated the tutor’s code as projected onto the screen at the front of the room. Particularly in the morning sessions, the delivery of these instructions dominated the space. While the tutor was experienced and confirmed with participants if his pace was appropriate, the responses were short. Instead, the prevailing sounds coming from the participants were the tying at varying speeds for code writing and note taking. Participants did ask questions, but in a careful manner, fearing of disrupting other participants and the flow of tuition.

The tutor adopted live coding during the workshop, which created a mixture of wonder and fear. Participants were amazed when experiencing the immediate effect that these code produced. But to live code well, the tutor had to have new lines of code keep appearing on the screen, while simultaneously providing enough context to understand and appreciate them (see Figure 1). He needed to explain the structure of the code, but more importantly the reasoning and appropriate amount of knowledge behind it, while avoiding delving in too deep to confuse tutees. The success of a living coding session also required considerable effort from the participants, because keeping up with the pace is no less a demanding task. Participants had to follow the code exactly as the tutor types on the screen and with a high degree of precision, otherwise they would be returned with nothing but error messages. Furthermore, neither the tutor nor the participants wanted the workshop to be a copy and

paste exercise, and both aimed at becoming acquainted with the way in which this particular language reasons, acts and often times confuses its users.

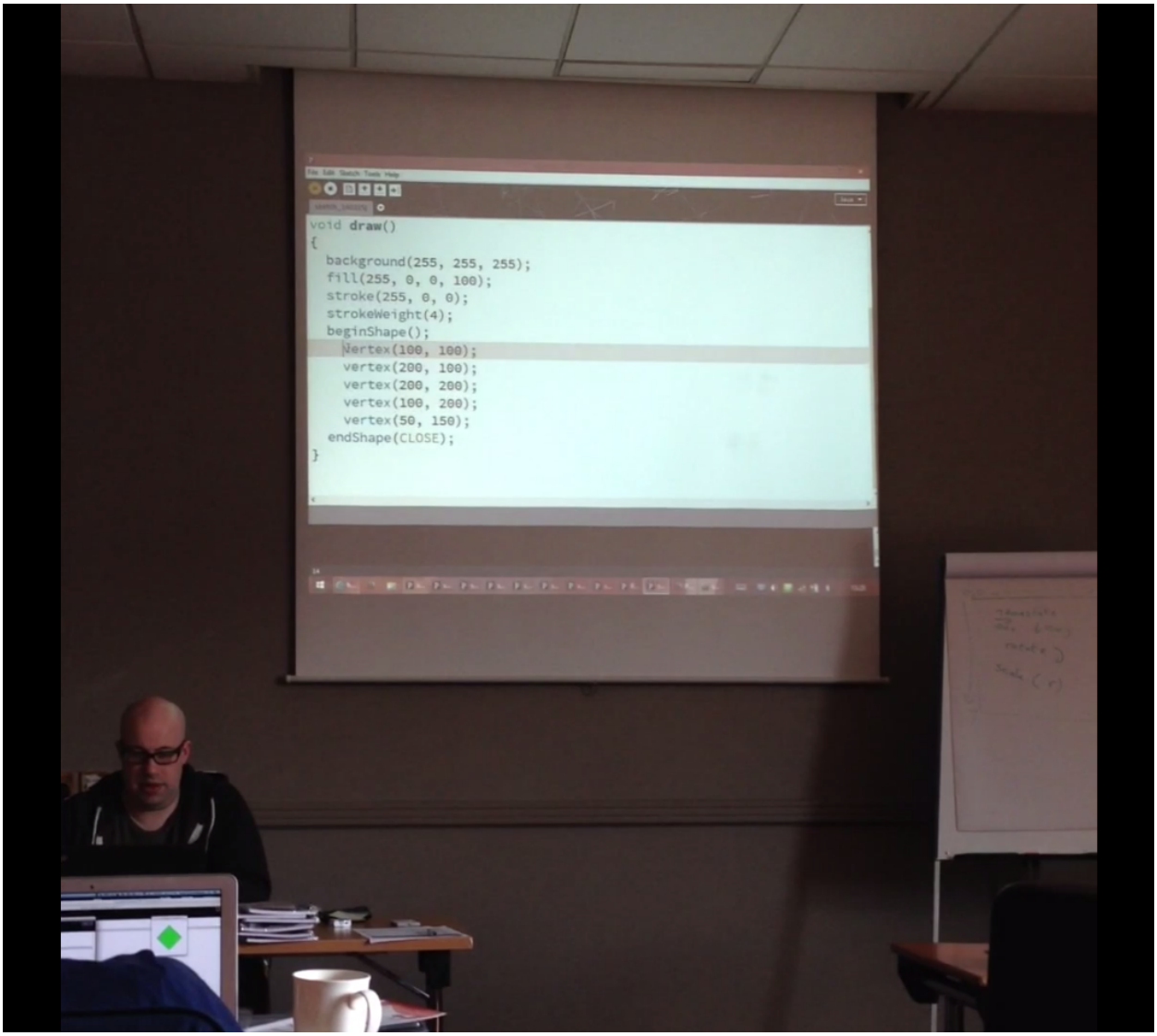

Figure 1: Live coding: writing and explaining code simultaneously

To do so, the tutor pointed out the difference between our normal day-to-day expressions and reasoning and how programming languages work. This meant for the participants that, when writing code, they were also labouring at converting everyday life experiences into highly stylised, regulated ways of expression. In the example below, when drawing a rotating diamond, the participants had to translate the mental image of the object into particular statements and numeric expressions of the position of the image on the screen (‘`rectMode(CENTER)`’), the size of the image and the colours of the rectangle and the borders (‘`size(200,200; fill(0, 255, 0); stroke(0,0,25)`’), the size of the

diamond (‘translate(width/2, height/2)’), and the amount it rotates each time (‘(rotate(rotateAmount))’ and ‘rotateAmount = rotateAmount + 1’), according to the specific order and style required by Processing.[1] All the participants got a revolving diamond on the screen, even with different colours and light effects. But it was after hard work of refashioning human experiences and reasoning according to that of the machine. Accordingly, in the process, concentration was paramount and chatter had to wait until the final successful run of the code excited acclamation.

```
float rotateAmount = 0;
void setup()
{
  size(200, 200);
  fill(0, 255, 0);
  stroke(0, 0, 25);
  rectMode(CENTER);
}
void draw()
{
  background(255, 255, 255);
  translate(width/2, height/2);
  rotate(radians(rotateAmount));
  rect(0, 0, width / 2, height / 2);
  rotateAmount = rotateAmount + 1;
}
```

Discussions and reflections

Sound-making is only one mechanism through which different forms of knowledge, reasoning and practices become engaged in collaboration or contestation, and wider explorations on all possible mechanisms of creating spaces of coding can provide more insights into and the potentials for making alternative future cities. By describing the interconnectedness between the sounds and activities happening at the coding workshop, I

[1] Extracted from the organiser’s workshop note, available at https://hackpad.com/Introduction-to-Processing-YKTwteQ7PrV [1 September 2015]

begin to depict participants' difficult encounters with software. The lack of interaction observed in the workshop was not a symptom of the format of the tutoring. Instead, it was caused by the constant oscillation between familiar and highly formalised streams of reasoning, and the associated systems of knowledge, styles of articulation and coding practices. When participants concentrated on repeating the tutor's code and using it for subsequent exercises, they were also rehearsing how they could reformulate their understanding of an object from their everyday experiences and points of view into series of expressions about the object according to the view of software. Furthermore, for beginners and experienced programmers alike, little slips can put carefully compiled code out of action because of different naming conventions and syntax requirements when switching between similar programming languages or frameworks. That is, programmers also have to acquire highly specific, language-dependent and contingent ways of articulating the relationships between themselves, the object they create and the context to situate the object. If a coding workshop quietens down as it goes on, it is because such reconfiguration of reasoning and articulation is complex and demanding, and participants were all committed to working through these difficulties for themselves.

At a coding workshop, the scope of a project, the goal, the exercises and the programming language to use are all preconceived, and participants have step-by-step instructions planned by the tutor to learn. However, these parameters are often more dynamic, fluid and contingent, and have to be negotiated when a project is pursued under the context of civic hacking. Added complexities emerge from having to define affected communities, the project's intended benefits and costs, required hardware and software skills and resources, and possible solutions; and negotiations occur among individuals, governmental agencies and civic organisations of differing motivations and concerns (Perng and Kitchin, 2015). As a result, the quiet and peaceful coding time has to be reconsidered and situated in the wider embodied and sociotechnical interactions that bring in contact different forms of reasoning, articulations, rules, regulations, knowledges and everyday practices. These interactions often unfold in situ, in short or sustained conversations to draw plans and sociotechnical imaginaries about urban futures, and in contestation when different motivations and interpretations about these futures confront one another. Accordingly, the ways the interactions cast shadow on code writing and solution development can change constantly and it remains uncertain how the hopes for more open and transparent spaces

of coding can be materialised. To create better networked urban experiences and futures, continued research is thus required to make explicit the tricky processes of opening up the spaces of coding and the unanticipated problems emerging from them.

Acknowledgement

The research for this paper was conducted under the Programmable City project, funded by a European Research Council Advanced Investigator award (ERC-2012-AdG-323636-SOFTCITY). I am grateful for those who contributed to make coding more accessible and diversify coding population and cultures.

References

Büscher M, Kerasidou X, Liegl M, et al. (2016) Digital urbanism in crises. In: Kitchin R and Perng S-Y (eds), *Code and the City*, London: Routledge.

Graham SDN (2005) Software-sorted geographies. *Progress in Human Geography*, 29(5), 562–580.

Kitchin R (2014) *The Data Revolution: Big Data, Open Data, Data Infrastructures and Their Consequences*. London: Sage.

Kitchin R and Dodge M (2011) *Code/space: software and everyday life*. Cambridge, Mass: MIT Press.

Lodato TJ and DiSalvo C (2016) Issue-oriented hackathons as material participation. *New Media & Society*, doi: 10.1177/1461444816629467.

Maalsen S and Perng S-Y (2016) Encountering the city at hacking events. In: Kitchin R and Perng S-Y (eds), *Code and the City*, London: Routledge.

Perng S-Y and Büscher M (2015) Uncertainty and transparency: Augmenting modelling and prediction for crisis response. In: *Proceedings of the 12th International Conference on Information Systems for Crisis Response and Management*, Kristiansand, Norway.

Perng S-Y and Kitchin R (2015) Solutions, strategies and frictions in civic hacking. In: *MEDIACITY 5: Reflecting on Social Smart Cities*, Plymouth, UK.

Urry J (2007) *Mobilities*. Cambridge: Polity Press.

Urry J (2014) *Offshoring*. Cambridge: Polity Press.