Miguel Escobar Varela

Hacking and Rehearsing: Experiments in Creative Tinkering

Hacking – improving software through a process of trial and error – is a mode of rehearsal. Such is the claim made by Miguel Escobar Varela in this article, which he furthers by exploring the similarities between the ways theatre makers and software programmers speak about their crafts. Understanding software programming as an essentially creative process should be of interest for theatre scholars, who are constantly searching for modes of academic discourse that are sensitive to the specificity of theatre. By offering examples from interface design for the study of Javanese theatre, Escobar argues that creating software, through an iterative process of trial and error, can become part of the methodological palette of theatre scholars. Miguel Escobar Varela is Assistant Professor of Theatre Studies at the National University of Singapore, and has worked as a theatre researcher, computer programmer and translator in Mexico, the Netherlands, Indonesia, and Singapore. His articles on the intersection of digital technology and theatre studies have been published in *Digital Scholarship in the Humanities*, *Asian Theatre Journal*, *Performance Research*, *Contemporary Theatre Review* and are forthcoming in *TDR* and *Theatre Research International*.

Key terms: digital humanities, wayang kulit, computer programming, Indonesia, Peter Brook, Jerzy Grotowksi, Sigit Sukasman.

THE WAY to create something beautiful is often to make subtle tweaks to something that already exists, or to combine existing ideas in a slightly new way. This kind of work is hard to convey in a research paper.

Paul Graham, *Hackers and Painters*

In an article in the *Guardian* in 2012, Olivier Choinière's *Project Blanc* is described as 'theatre-hacking'.[1] In this Montreal-based performance, the director guides the audience into the performance of another theatre collective, the Théâtre du Nouveau Monde (TNM), and this experience is described as 'surreptitious infiltration of another artist's work'.[2] Leaving aside the merits of this performance, the usage of the word 'hacking' here echoes the common perception of hackers as crackers: people who infiltrate, steal, and disrupt. However, in the original conception, the word 'hacking' referred to an attitude of profound respect for the work of others, with material being reused and improved in other contexts. I want to suggest it is important for theatre people to understand the original meaning of hacking, since it has more in common with theatre practices than is conventionally believed.

Hacking here refers to a mode of working based on hands-on experimentation and sharing. It consists of a series of 'subtle tweaks to something that already exists', as the opening quote by Paul Graham reminds us.[3] Graham, who is both a software engineer and a painter, argues that his two professions have more in common than people usually assume: 'What hackers and painters have in common is that they're both makers.'[4] Makers, he suggests, proceed through trial and error. This is a mode of creative tinkering that also describes how performances are created during the rehearsal or laboratory work many performance makers undertake. It is especially true for devised performances or practice-based research projects, but I would argue that it describes the way many other theatre performances are created.

By suggesting that rehearsing and hacking share much in common, the objective of

  doi:10.1017/S0266464X15000871

this article is to describe a way in which we can contextualize the usage of computational tools in theatre studies. This is an important matter, as the rise of digital humanities (DH) methodologies is starting to affect how we conceptualize and carry out research in theatre. By rethinking hacking and rehearsing together, I aim to argue that hacking is not destroying and that computing is a creative and experimental process that can help us expand the vocabulary and range of DH methods in theatre studies without relinquishing creativity, criticality, and experimentation, but rather capitalizing on them. For this I offer a historical overview, and look at the work of prominent theatre directors from different traditions. I also refer to the reflections of the novelist and programmer Vikram Chandra, the theories of computer scientist and cultural critic Fox Harel, and my own experience doing theatre and writing code.

Ways of Understanding Hacking

The word 'hacking' is subject to a variety of interpretations but perhaps this is also true for the words 'art' or 'performance'. As Paul Graham, visual artist and software engineer notes: 'I think hackers just have to resign themselves to having a large random component in their reputations. In this they are no different from other makers.'[5]

For most lay people, hacking conjures up images of illegal transgression: the hacker is a criminal who steals data and breaks computer systems. However, originally the term referred to a counter-intuitive feat of creative problem-solving which might go against conventions (but not necessarily against the law). This interpretation of the word is now back on the rise in popular lore as well, with the rise of initiatives such as Hackerspaces, websites such as lifehacking.org, and even academic volumes such as *Hacking the Academy*.[6] The new, regained popularity of this word is important for the ways in which we might rethink the work of the theatre scholar in relation to technology.

Before examining those implications in detail, it is worth considering the origin of the word. The word 'hacking' originally meant coming up with an unexpected, quick solution to an engineering problem. According to Stephen Levy, it emerged as part of the in-group lingo of the Tech Model Railroad Club (TMRC). The club was an association of (mostly) electrical engineering students at MIT who became interested in using computers at a time before computer science was recognized as an official area of studies and before operating systems were in place.

Working with computers implied wiring circuits directly into the machine or writing code in assembly language. TMRC members often stayed up until late working with computers, endlessly tinkering with the unexplored possibilities of these devices. This tinkering came to be known as 'hacking'. Unlike the certified engineers who worked on officially sanctioned projects, hackers often worked on projects for simple enjoyment:

> A project undertaken or a product built not solely to fulfil some constructive goal, but with some wild pleasure taken in mere involvement, was called a 'hack'. . . . To qualify as a hack, the feat must be imbued with innovation, style, and technical virtuosity. Even though one might self-deprecatingly say he was 'hacking away at the system', the artistry with which one hacked was recognized to be considerable.[7]

Evidence to support Levy's claims can be found in the copious output of music projects and games that were not considered scientifically productive at the time. And even if this interpretation does not correspond to the reality of how those early hackers worked, it has come to define how a community of software programmers imagine themselves today, as heirs of the counter-culture movements of the 1960s. Levy is not alone in interpreting the working modes of the hackers as being the result of a particular way of understanding cultural production. According to Johnny Ryan, the origins of hip-hop music can also be placed in the same historical context, and they can serve for an illustrative comparison. He describes how Cool DJ Herc and other Bronx DJs

> further developed the Jamaican practice of toasting, or speaking over the record. This new way of remixing records at all-night parties in the basket-

https://doi.org/10.1017/S0266464X15000871 Published online by Cambridge University Press

ball and tennis courts of Bronx neighbourhoods in the early 1970s marked the birth of hip-hop. It was also an early example of popular user-dirven innovation. . . . The hip-hop DJs who adapted this into something new, though performers themselves, were at a remove from the original creation of the music. Rather than musicians in the traditional sense, they were a new type of empowered user. The turntables of the Bronx, like the hackers, the homebrew clubs, and the perpetual beta of Web 2.0, are an instance of an enabling technology and the human impulse to adapt and hack.[8]

Whether the stories of the early hackers are mythical or not, these ways of imagining their interaction with computers in a broader cultural background has led to the development of 'the hacker ethos', a belief that systems can be collectively improved by creative solutions. The prerequisite of such an ethos is the open sharing of information and a creative disposition. As Mark J. V. Olson notes:

A hacker ethos is a way of feeling your way forward, through trial and error, up to and perhaps beyond the limits of your expertise, in order to make something, perhaps even something new. It is provisional, sometimes ludic, and involves a willingness to transgress boundaries, to practise where you don't belong.[9]

As Ted Suiter asserts, this way of 'feeling your way forward' through trial and error, has gained cultural currency and is now applied to activities in many different areas:

The fact that this is about a relationship to knowledge systems means that the term has, over the last thirty years or so, come to be applied to an ever-growing assortment of activities: life hacking, game modding, phone phreaking, iPhone jailbreaking, and IKEA hacking, among others. In each of these activities, you can see the kernel of the same hacker ethos. Each of these activities is based on the use of playful creation to enrich knowledge of complex systems, whether you are making furniture from the complex system that is the IKEA catalogue, or learning how to game Ma Bell for free calls to Bangalore. This sort of playful creation should not be unfamiliar to academics. It is not dissimilar to the Situationist Internationals concept of detournement or Dick Hebdidge's notion of subcultural style systems. It is Lévi-Strauss's bricolage reimagined for a time when computers have replaced magic.[10]

To this list, we could certainly add theatre rehearsing.

Rehearsing and Tinkering

There are many different strategies for rehearsing, but a dominant approach in post-war contemporary western theatre is for directors to 'feel their way forward' through trial and error. An anecdote narrated by Peter Brook resonates with many contemporary theatre directors. When directing his first professional play, Brook recounts spending a sleepless night plotting every movement of his actors using paper cut-outs. But then on the day of the first rehearsal, he realized his preparations were useless.

As the actors began to move I knew it was no good. These were not remotely like my cardboard figures, fine large human beings thrusting themselves forward, some too fast with lively steps I had not foreseen, bringing them suddenly on top of me – not stopping, but wanting to go on, staring me in the face, or else fidgeting, pausing, even turning back with elegant affectations that took me by surprise. We had only done the first stage of the movement, letter A on my chart, but already everyone was wrongly placed and movement B could not follow. My heart sank and, despite all my preparation, I felt quite lost. Was I to start again, drilling these actors so that they conformed to my notes? . . . I stopped and walked away from my book, in amongst the actors, and I have never looked at a written plan since. . . . Of course, all work involves thinking: this means comparing, brooding, making mistakes, going back, hesitating, starting again. The painter naturally does this, so does the writer, but in secret. The theatre director has to expose his uncertainties to his cast but in return he has a medium which evolves as it responds: a sculptor says that the choice of material continually amends his creation: the living material of actors is talking, feeling, and exploring all the time – rehearsing is a visible thinking-aloud.[11]

This 'visible thinking-aloud', like Olson's 'feeling your way forward', is a mode of hacking. Or, put differently, both hacking and rehearsing share a belief in hands-on tinkering, a practice of 'hesitating, starting again' that cannot be substituted by theoretical machinations. The way Paul Graham talks about the creation of software is remarkably similar to Brook's account of the rehearsal process:

I was taught in college that one ought to figure out a program completely on paper before even going

near a computer. I found that I did not program this way. I found that I liked to program sitting in front of a computer, not a piece of paper. Worse still, instead of patiently writing out a complete program and assuring myself it was correct, I tended to just spew out code that was hopelessly broken, and gradually beat it into shape. Debugging, I was taught, was a kind of final pass where you caught typos and oversights. The way I worked, it seemed like programming consisted of debugging. . . . If I had only looked over at the other makers, the painters or the architects, I would have realized that there was a name for what I was doing: sketching. As far as I can tell, the way they taught me to program in college was all wrong. You should figure out programs as you're writing them, just as writers and painters and architects do.[12]

Both Brook and Graham describe their disdain for a plan on paper and express their support for creative tinkering: debugging or thinking-aloud are necessary for the creative process. Whether a maker is busy creating a theatre performance or a piece of software, they need to work directly on the problems they are being faced with. I would suggest that this playful tinkering has come to characterize many developments in software engineering but also many approaches to theatre: it has become the dominant way theatre directors work.

The Autodidactic Path to Discovery

Despite differences in aesthetics and approach, most contemporary theatre makers are people without a fully thought-out plan, they are tinkerers and risk-takers. The larger point made by Peter Brook in this comment is that he learned to direct theatre through practice. Paul Graham similarly claims that he learned more from failing and debugging than from the way software engineering was formally taught. In other words, both honed their crafts through self-directed learning. As Ted Suiter notes, hacking is an attitude to knowledge that values the autodidact's path:

> There are many definitions of 'hack', some of them seemingly deeply contradictory. Yet there is, in the final analysis, a unity to the term. Originally, the term was used to describe computer code. There were two opposing meanings to calling a piece of code a 'hack'. One: it is expertly written efficient, and does precisely what it is intended to do, with eloquence. The other was that the code was hastily written, sloppy, and essentially only just good enough. . . . As mutually exclusive as these two connotations of the term may seem, however, both the polished, impressive hack and the quick-and-dirty hack have a fundamental similarity. They are both born of a certain relationship to a certain type of knowledge. Hackers are autodidacts. From the earliest hackers working at large research universities on the first networks to anyone who deserves the term today, a hacker is a person who looks at systemic knowledge structures and learns about them from making or doing.[13]

The self-directed yet systematic approach to discovery through practice was also an inspiration for Jerzy Grotowsky. He famously claimed to have been inspired by the way the international physics laboratory of Neils Bohr worked, where top scientists from around the world would gather to work together.[14] However, Bohr's laboratory was one of the best funded scientific initiatives of the time, where innovative, cutting-edge instruments were developed.

This wealth of resources and tools is in contrast with the *via negativa* approach of Grotowski and his colleagues. Perhaps, their approach had more in common with the MIT hackers (who were active at roughly the same time as Grotowski). The machines that the MIT hackers had access to were of course also extremely expensive and innovative, but the hacker's approach to working with the machines was to work around limitations. The hackers would stay up at night trying to work with limitations of memory and processing capacity that required creative approaches to reduce the size of the programs. In hacker parlance, reducing the size of a program was called *bumming:*

> The practice of taking a computer program and trying to cut off instructions without affecting the outcome came to be called 'program bumming', and you would often hear people mumbling things like, 'Maybe I can bum a few instructions out and get the octal correction card loader down to three cards instead of four.'[15]

Although the objectives and materials of their respective quests for simplification were different, Grotowski and his colleagues shared

https://doi.org/10.1017/S0266464X15000871 Published online by Cambridge University Press

a fascination with subtracting rather than adding. Their essential strategy was the *via negativa*, subtracting limits from the actor rather than adding skills: 'To eliminate from the creative process the resistances and obstacles.'[16]

The emphasis on reduction does not describe the work of all hackers nor the work of all theatre directors. But there is an underlying trait that has more general applicability. The fascination with reduction was part of a larger philosophy where creative approaches were valued, and I would argue this was the case as much in theatre directing as it was in hacking.

Sukasman: Hacker of *Wayang*

My previous examples of how tinkering has guided the development of theatre have centred on European directors who became famous in the 1960s. But this approach has also come to characterize the work of theatre makers in other places and times. An example from Indonesia of a theatre maker whose mode of working was also defined by creative tinkering was Ki Sigit Sukasman (1936–2009). Sukasman was a visual artist from Yogyakarta who had the opportunity to study *wayang* (Javanese leather puppet theatre) in his youth and then travel to the US and the Netherlands and to spend more than a decade working as a design teacher in Germany.

Sukasman was a visual artist with a lifelong fascination with *wayang* carving. His quest to redefine the morphology of the puppets ushered in a new era of visual creativity to the world of *wayang*, one of the most important performing arts traditions in Indonesia. His international career began when he visited the World Fair in New York in 1964 and the Netherlands in 1965. Shortly after, he settled in Germany for several years before returning to Indonesia in 1974 to take care of his ailing mother. Although his obsession with *wayang* carving had accompanied him since childhood, it was only then, in his late thirties, that he began to perform *Wayang Ukur* ('Measured *Wayang*'). He was known for his obsessive attention to every detail of the *wayang* performances, which would be polished over extensive rehearsal processes.

He was never the *dalang* (puppeteer) himself; three *dalang* were responsible for puppet manipulation in his shows.[17] He was thus a meta-dalang, controlling the controllers. In his performances, he used specially crafted puppets, a front stage, and a great number of light fixtures with coloured gels. He was a master in creating three-dimensional illusions on the *wayang* screen. By instructing some *dalang* to sit behind the screen and others to sit in front of it, he explored different registers of visuality, urging the *dalang* to delve in the nuances of shadow, colours, and size afforded by the puppets. Sukasman would often include actors and dancers in the show, who would stand behind the screen or on a raised platform above it in order to suggest a wide array of visual effects. He devised his shows through an extensive, relentless process of creative tinkering where every detail was carefully tested and re-examined.

To give an example of his legendary perfectionism I will refer to an unfortunate performance that was offered in honour of Sukasman three years after he passed away and to the comments that were given by the spectators. In Java, *wayang* performances are customarily offered one thousand days after someone's death (*peringatan seribu hari*). In Sukasman's *peringatan,* which I attended in 2012, a performance inspired by *Wayang Ukur* was presented. However, the lights failed shortly after the performance began. An audience member said to me that this was the work of Sukasman himself, who was sabotaging the show from the other world for failing to adhere to his exquisite perfectionism. This comment was not meant as a joke. People believe today that no one can live up to the expectations of the late genius.

The memory of Sukasman lingers in the minds and the works of contemporary Javanese artists, many of whom worked with him or saw his shows (which were also transmitted by the national television station TVRI). One of his most famous disciples is Heri Dono, probably the most famous Indonesian visual artist alive, whose paintings, installations, and performance pieces are inspired by Sukasman's relentless tinkering with the meanings and shapes of *wayang*

puppets. Sukasman's attitude is summed up in the following passage by Hardja Susilo:

> Sukasman has had extensive training in traditional puppet making. He knows how to do it the 'right' way. However, his curiosity was aroused when one of his teachers told him that *wayang kulit* puppets were already perfect. There was no way in which anybody could improve them – altering the proportions by as much as one millimetre would flaw the characterization of a puppet. Sukasman tested the edict and found it to be an exaggeration. He challenged this unreasonable precision with uncompromising precision of his own, which was based on his personal interpretation of each *wayang* character. Spending his own money and recruiting several traditional artists, he proceeded to 'improve' *wayang* puppet proportions by exact quantification, that is, by exact measurement, or in Indonesian *ukuran* or *ukur*, hence the name *wayang ukur* or 'measured' puppets.[18]

A freer translation of *wayang ukur* could be *hacked wayang*. The *wayang* tradition is hierarchical and procedural, much as was work with mainframe computers in the sixties.

> The culture of the first people to use the term 'hack' produced a second-order meaning as well. A hack is a practical joke, a playful subversion or gaming of a system. . . . There is a sense of play in coding – it is not apparent to everyone, but it is there. The fundamental action here is the same: it is the clever gaming of complex systems to produce an unprecedented result.[19]

Wayang Ukur, as hacking, is the result of a playful subversion, derived from a process of continuous tinkering. Sukasman's approach meant taking apart this traditional knowledge and seeing what could be done with it, how it could be rearranged through thoughtful, continuous experimentation. Sukasman was one of the pioneers of *wayang kontemporer*, new versions of this tradition that currently take only a couple of hours and include elements from popular music, television narratives, and new media.

Digtial Tinkering and Theatre Scholarship

Thus far I have made a case that the way some theatre makers work is similar to the way some computer programmers work. Certainly, not all theatre directors work by feeling their way forward. In some cases, such as commercial musicals, planning is more important since time for experimentation is limited. Likewise, not all programmers hack their way through code. But I argue that there is a similarity in the way some people create theatre and the way some people create software. My personal experience in both areas suggests that both creating theatre and coding software are phenomenologically similar: that is, they are experienced in similar ways.

When I was in my teens, I became interested both in doing theatre and in making software. I felt that both theatre making and software programming were open playgrounds, where I could experiment and tinker my way through a creative process with nothing but simple materials. Developing projects in either domain was a matter of testing, rearranging, asking for comments. In both cases, it was also a social experience. I never took a software course. I learned to work with Linux and program in Python and JavaScript by thinking of ideas, making mistakes, and trying again. I asked questions in online discussion boards and asked people to look and review my code.

The same process describes my theatre experiments. I was part of a theatre collective that experimented with everyday objects and stories. We didn't have a clear idea of how to craft our theatre productions: we learned how to do this through a process of iterative tinkering, asking people for advice, taking distance, and trying to improve on our solutions. This was remarkably similar to the experience of making code.

I am not the only person to note this similarity between the creative process of making art and coding software. Vikram Chandra, renowned Indian novelist, worked for a long time as a software programmer. Without any formal training and learning by endless creative tinkering, he identifies the pleasure in experimentation as the driving force in his learning process:

> The work of making software gave me a little jolt of joy each time a piece of code worked; when something wasn't working, when the problem

https://doi.org/10.1017/S0266464X15000871 Published online by Cambridge University Press

resisted and made me rotate the contours of the conundrum in my mind, the world fell away, my body vanished, time receded. And three or five hours later, when the pieces of the problem came together just so and clicked into a solution, I surfed a swelling wave of endorphins. . . . Even after you are long past your first 'Hello, world!' there is an infinity of things to learn, you are still a child, and – if you aren't burned out by software delivery deadlines and management-mandated all-nighters – coding is still play. You can slam this pleasure spike into your veins again and again, and you want more, and more, and more.[20]

The connection to creativity has also been noted by programmers such as Paul Graham, who repeatedly asserts that painting and hacking share much in common. Considering this similarity is important for theatre researchers because it has implications for how software programming can be used as a part of their methodology.

The 'Hacking Approach' to Scholarship

In some circles, there is a resistance to using digital tools to conduct research in the humanities. This fear often assumes that using digital tools means substituting interpretation with statistical analysis or machine learning, a path that will not get us closer to the quest for meaning. Steven Marche expresses this fear with great eloquence:

Meaning is mushy. Meaning falls apart. Meaning is often ugly, stewed out of weakness and failure. It is as human as the body, full of crevices and prey to diseases. It requires courage and a certain comfort with impurity to live with. Retreat from the smoothness of technology is not an available option, even if it were desirable. . . . Through the perfection of our smooth machines, we will soon be able to read anything, anywhere, at any time. Insight remains hand made.[21]

Although I believe that statistical analysis and other techniques might prove to be useful tools, I do agree with Marche that insight is something that will remain hand made: intuition and nuance can only be provided by a human interpreter and not by a machine. However, in contrast to his views, I do not think that the use of machine techniques excludes this kind of hand-made insight. Vikram Chandra is vocal about this connection when he talks about his experience writing novels and coding software:

My writing life and my life with computers, in spite of their differences, seem mirrored, twinned. Both are explorations of process, of the unfolding of connections. Both reward curiosity, dogged patience.[22]

The hacking approach I have been suggesting is a way of retaining the hand-made approach to insight. But how exactly can this be translated into theatre scholarship? In what follows I will sketch out an example by referring to my work on the Contemporary Wayang Archive (CWA), an online collection of video recordings of non-conventional *wayang* shows made in Java between 2000 and 2004. As part of my work on the CWA, I added translation notes and commentary to twenty-four full-length recordings of Javanese innovative *wayang* performances.

My objective was to find a way to integrate critical commentary and translation notes in the graphic interface of the video player. For the first version of the website, I designed a note-viewer for display next to the video player. This note-viewer allowed users to scroll through translator notes and honorifics.[23] A user can pause the video and read the honorifics or translation notes. This was, at least, the idea on paper. But when the time came to test this design I realized it created an awkward viewing experience for users who wanted to watch the video full-screen, since they had to pause the video, exit the full-screen mode and scroll through the notes. (See Figure 1, opposite, top.)

Thus, I set about coding a different kind of interaction. When in full-screen mode, users can strike any key for the video to be paused and the translation notes to be displayed. This is a very simple and common solution, but testing it out gave me another idea. I enjoyed being able to jump in and out of the story, reading the notes I had made. This made me think of a *dalang*, who uses music in order to punctuate the story, creating pauses for reflection and explanation. The *dalang* do not play the music themselves, but use a wooden mallet called a *cempala* (pronounced 'chempolo') to this effect. By hitting

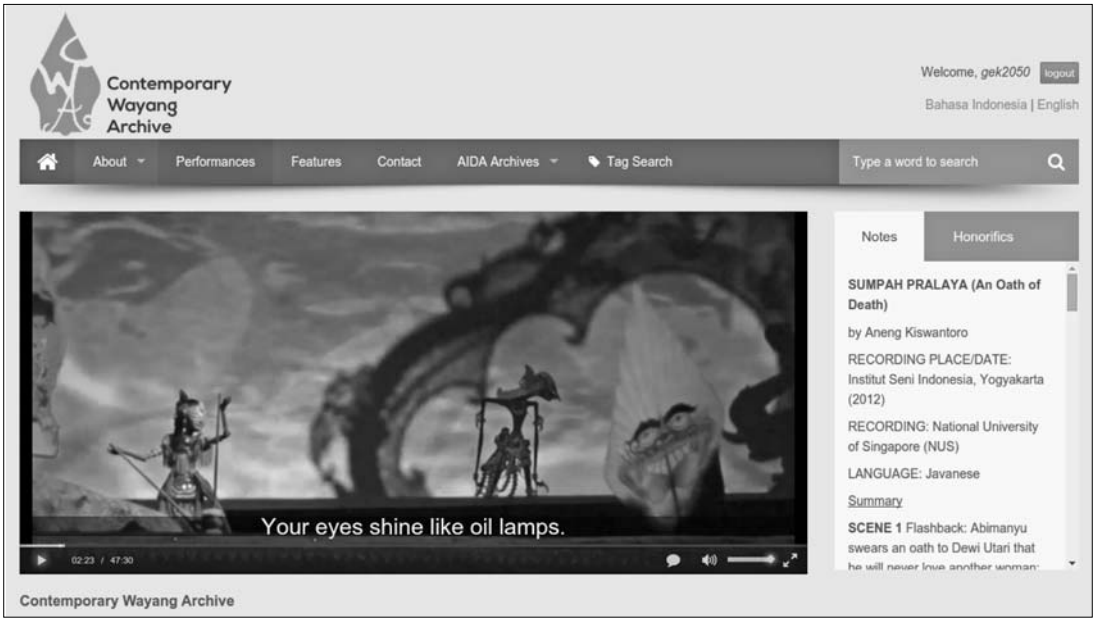

Fig. 1. A screenshot of the Contemporary Wayang Archive. The panel on the right includes notes and honorifics.

the *cempala* against a wooden box, they instruct the musicians to begin or stop specific pieces. I thought it would be wonderful if I could use a *cempala* to begin and stop the flow of the video. I happen to have a *cempala* next to my desk and glancing at it in the course of this meditation gave me another idea. I realized that the embodied experience I wanted to try was only a hack away.

So I started experimenting with ways to connect the *cempala* to my computer. I ended up wiring a button into my *cempala* and using an Arduino micro-controller (a cheap and easy to use piece of open-source hardware often used for artistic projects)[24] as an interface to my computer. In the new version, when the user hits on the cempala, the video starts and stops. (See Figure 2, below.)

This is not a complex project, but the expertise I need to wire and program the micro-controller surpassed what I knew about electronics before the beginning of the project. Nevertheless, I was able to learn what I needed by watching online tutorials, asking around, failing over and over again, and thinking-aloud through my mistakes.

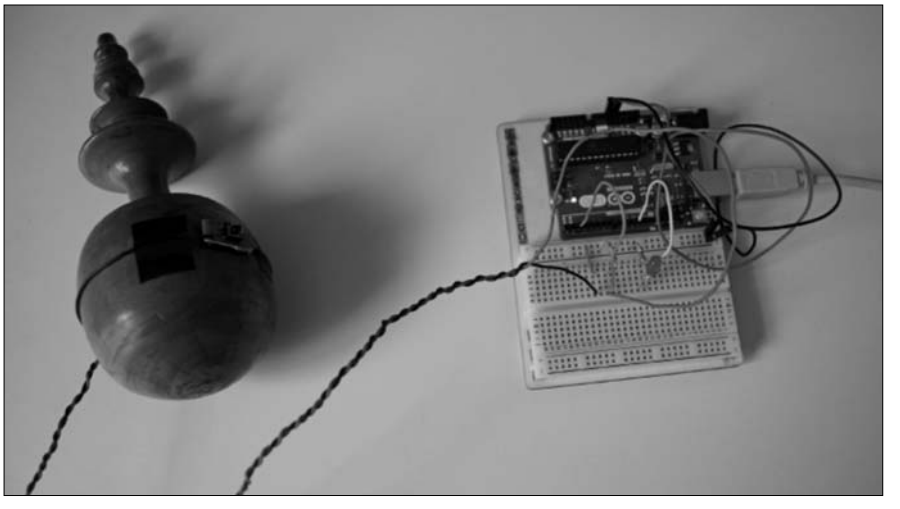

Fig 2. The *cempala*, connected to a computer via an Arduino interface.

https://doi.org/10.1017/S0266464X15000871 Published online by Cambridge University Press

This process of experimentation and trial is just a part of the process of tinkering through which I engaged my study of *wayang*.

Engaging in Theatre (Re)search

Perhaps not all theatre scholars have played with electronic circuits and micro-controllers, but I believe that the experience of figuring something out as you try it is a familiar one for theatre scholars. In theatre studies, as in much of the humanities, there is no predefined form through which to report our findings. The structure of a research paper in theatre and performance research is not as rigid as it is in the (social) sciences. It cannot be. Every theatre scholar is tasked with finding the appropriate way to convey an argument about performance. I believe most scholars search for this form through creative tinkering. As Joseph Roach notes:

> Conventional definitions of research . . . typically convey the idea of a systematic inquiry, governed by methods of empirical verification, into a previously unknown area of knowledge or unsolved problem. . . . Theatre and performance scholars, however, must search after their object, conducting (re)search into what was once, but at the moment of writing is no longer, an event.[25]

Thus, the hacking-based approach to the development of digital theatre scholarship is one way of engaging in theatre *re*(*search*). For theatre scholars, there is no fixed form in which to report our findings; we need to find the right words, concepts, allusions, circuits, and interfaces to allow us to search after something that is 'no longer an event'.

Digital scholarship in theatre can be a playful, experimental, and social process. Embracing this way of working requires us to think about programming as a tool for creative tinkering, for iterative re(research). Using computational tools and methods for the ends of theatre and performance research will not wrest creativity and intuition away from our methodological approaches, nor will it mask our discipline as a technical, pseudo-scientific objective domain. Rather, computational tools should enable us to re-imagine our relationship to technology.

As Wendy Chun notes, 'software offers us an imaginary relationship to our hardware'.[26] Though she is thinking of hardware in a technical way, her insight can be extended to other domains of culture if we think of hardware in a more general way. Software also offers us an increasingly imaginary relationship to our theatre 'hardware': the objects and bodies that populate our stages. Increasingly, technology is becoming central to how different aspects of theatre making are carried out: production notes, light design, texts, rehearsal, and publicity depend on a wide array of technologies. The same can be said of the activities of documentation and analysis that are central to theatre scholarship.

Software tools have become essential components in the theatre-maker's toolkit, even for people not ostensibly interested in working with technology as a subject-matter. The words from Kitchin and Dodge ring as true as in most domains of cultural production:

> It is very difficult to avoid the effects – the work – of software in the world . . . because of the difference it makes to the constitution and practices of everyday life. Indeed, to varying degrees, software conditions our very existence. Living beyond the mediation of software means being apart from collective life.[27]

However, this does not imply that we are subject to one way of using technology. The computational tools at our disposal can be modified by us through a process of experimentation and tinkering. This is important, as Douglas Rushkoff notes, because knowing how to work our tools can determine how much we are capable of affecting how they work:

> In the emerging, highly programmed landscape ahead, you will either create the software or you will be the software. It's really that simple: program, or be programmed. Choose the former, and you gain access to the control panel of civilization. Choose the later, and it could be the last real choice you get to make.[28]

Contrary to what some people think, learning to program does not require extensive technical or mathematical training. It requires

curiosity and an open mind. As Stephen Ramsay notes:

One thing is certain: Being good at mathematics in no way guarantees that one will be good at programming (or vice versa). My own (admittedly anecdotal) experience as a teacher suggests that being musical, enjoying games and puzzles, being a tinkerer, loving to cook, and being a good long-form writer are far better predictors of success. A very high tolerance for frustration helps as well.[29]

A similar point is echoed in the works of Paul Graham, who argues that originality and a desire for tinkering are essential for hackers:

The fact that hackers learn to hack by doing it is another sign of how different hacking is from the sciences. Scientists don't learn science by doing it, but by doing labs and problem sets. Scientists start out doing work that's perfect, in the sense that they're just trying to reproduce work someone else has already done for them. Eventually, they get to the point where they can do original work. Whereas hackers, from the start, are doing original work; it's just very bad. So hackers start original, and get good, and scientists start good, and get original.[30]

Programming is closer to the modes of working of theatre makers and researchers than people conventionally assume. And even learning a little bit of programming, learning to playfully hack and adapt our tools can help us re-imagine our relationship to our tools and to our theatre. By learning to program – or rather, by discovering programming through creative tinkering – we can also choose to represent things differently, away from hegemonic paradigms of representation.

Notes and References

1. Kelly Nestruk, 'Theatre Hacking: What's it All About?' 10 May 2012) retrieved from <www.theguardian.com/stage/theatreblog/2012/may/10/theatre-hacking-canada>, accessed 12 March 2015.
2. Ibid.
3. Paul Graham, 'Hackers and Painters' (May 2003), retrieved from <www.paulgraham.com/hp.html>, accessed 120 March 2015.
4. Ibid.
5. Ibid.
6. Dan Cohen and Tom Scheinfeldt, ed., *Hacking the Academy: New Approaches to Scholarship and Teaching from the Digital Humanities* (Baltimore, University of Michigan Press, 2013).
7. Stephen Levy, *Hackers: Heroes of the Computer Revolution* (Sebastopol: O'Reilly Media, 2010).
8. Johnny Ryan, *A History of the Internet* (London, Reaktion Books, 2010), p. 168.
9. Mark. J. V. Olson, 'Hacking the Humanities: 21st Century Literacies and the "Becoming-Other" of the Humanities', in E. Belfiore and A. Upchurch, ed., *Humanities in the Twenty-First Century: Beyond Utility and Markets* (New York: Palgrave Macmillan, 2013), p. 238.
10. Tad Suiter, 'Why Hacking?', in Dan Cohen and Tom Scheinfeldt, ed., *Hacking the Academy*, p. 7.
11. Peter Brook, *The Empty Space* (London, Penguin, 2008), p. 119–21.
12. Graham, 'Hackers and Painters'.
13. Suiter, 'Why Hacking?', p.7.
14. Jerzy Grotowski, *Towards a Poor Theatre* (London: Methuen, 1986), p. 18.
15. Levy, *Hackers,* p. 11
16. Grotowski, *Towards a Poor Theatre,* p. 96
17. In a conventional *wayang* show a single *dalang* controls all the puppets, commands the musicians, and speaks all the character parts.
18. Hardja Susilo, 'The Personalization of Tradition: the Case of Wayang Ukur', in Jan Mrázek, ed., *Puppet Theater in Contemporary Indonesia: New Approaches to Performance Events* (Ann Arbor, University of Michigan Press, 2002), p. 183.
19. Suiter, 'Why Hacking?', p. 7–8.
20. Vikram Chandra, *Geek Sublime: the Beauty of Code, the Code of Beauty* (New York, Faber and Faber, 2013), p.18–19.
21. Stephen Marche, 'Literature is Not Data: Against Digital Humanities', *Los Angeles Review of Books,* 28 October 2012, retrieved from <http://lareviewofbooks.org/essay/literature-is-not-data-against-digital-humanities>, accessed 12 March 2015.
22. Chandra, *Geek Sublime,* p. 230.
23. Honorifics, the words used to address others, are highly standardized in Javanese. In the performances, the characters use one of almost forty honorifics to address each other. The translation into English aimed to retain this aspect.
24. For more information and a project gallery, see <http://arduino.cc>.
25. Joseph Roach, 'Re:search', *Contemporary Theatre Review,* XXV, No. 1 (2015), p. 30.
26. Wendy Chun, 'On Software, or the Persistence of Visual Knowledge', *Grey Room,* 18 (Winter 2005), p. 43.
27. Rob Kitchin and Martin Dodge, *Code/Space: Software and Everyday Life* (Cambridge, Mass.: MIT Press, 2011), p.1.
28. Douglas Rushkoff, *Program or be Programmed: Ten Commands for the Digital Age* (Berkeley: Counterpoint, 2011), p. 2.
29. Stephen Ramsay, 'ENGL 4/878 FAQ', retrieved from http://jetson.unl.edu/syllabi/2014/fall/dh/html/01_overview.html>, accessed 12 March 2015.
30. Graham, *Hackers and Painters.*

https://doi.org/10.1017/S0266464X15000871 Published online by Cambridge University Press