

Integrated Formative Assessment

(Trimester 2)

Riviera Taylor

Tuesday, April 2, 2024

Contents

Presentation Transcript	4
Trimester One	9
Prototyping	10
(Re)Prerecording a tape	10
Go Fish	12
Podcasts Continued	13
Mixing A Tape	14
Broadcast Seven	19
Editorial Team Meeting Recap	24
Saturday 18th: Extratonal Infrastructure 9	25
Special Issue #22	26
Listen Closely	26
Technical and Spiritual Manual for Post-apocalyptic Radio Making	30
Reading, Writing and Research Methods	34
Selected Wordhole Entries	34
Trimester Two	36
Prototyping	37
January 16th 2024	37
Crop Tool	39
Fix The Wi-fi	47
Side Project: Chess Diary	51
Reading, Writing and Research Methods	55
Selected Wordquilt Entries	55
Bibliography	56
Appendix One: Presentation Slides	57

Presentation Transcript

Over the past several months I have sought to produce practice-based research of my own accord which is both reflexive and astute. I strove to find out more about and put into practice strategies and procedures pertinent to the making of **performative technological artworks**. This presentation intends to isolate and make connections between various lines of enquiry, concepts and problems that I perceive are at stake in this activity. Throughout my time here so far, I have acquired knowledge in classes and public moments which have shaped the arc of my endeavours. During this presentation, I plot these pivotal findings in relation to the contemporary aspects of practice, thought and research that have and continue to inspire me. **This takes place with reference to the methodical way I have made prototypes and publications** whilst reading and making notes on texts. Things have not always worked out entirely as expected. I have allowed errors which have arisen to motivate my reflective process. Accordingly, I focused on deepening my perspective of pertinent thoughts and resources. As such, I am in the process of cultivating what I hope is a thoughtful and thought-provoking portfolio. One that both reflects and contributes to the field of experimental publishing.

A note about archival documentation: The document which accompanies this presentation is composed of, with tiny alterations, key contributions I have made to the wiki since September. The slides are for visual support. I will indicate when points in the slides are not covered in the wiki documentation. Likewise, there is material in the wiki documentation which I have not had the opportunity to discuss here. Moreover, rather than present with the Wiki, I preferred to typeset PDF documents using wiki documentation as content. There were several reasons for this. I plan to use ConTeXt to typeset my thesis and like to practice working with the software on formal occasions. Secondly, I wanted the opportunity to collect my writings for analytical purposes. I sensed that re-transcluding wiki pages was likely to generate a messy and time consuming editorial process. I have been selective in what I have included. For legibility I kept the amount of code to a minimum. To heighten engagement I added some additional photographs. Finally, I retained links to various wiki pages in the form of interactive images, texts and captions.

Now I would like to discuss **prototyping**. In prototyping, I have produced several standalone pieces of software that do something useful. These designs have sparked interesting conversations and acted as an impetus for further enquiry and reflection. For example...

Fix the Wifi, a non-linear text adventure game, was written in Python during the second trimester by myself, Mania and Anita. We talked and made sketches together

to establish the world of the game. After figuring this out we wrote code in an Etherpad. We made a cohesive effort to create the game. Our endeavours fit with Simon Yuill's writing from 2008 on what constitutes collaborative practice. I took it upon myself to improve the code and the experience of playing the game. The documentation below retains sample output from the script. However, I cut out the code in order to foreground a key discussion about the objective of the game (which is to fix the wifi). The conversation takes place in relation to Augusto Boal's writing in his book *Games for Actors and Non-actors* (2002). In this book Boal documents lots of games to warm people up for making *Forum Theatre*, a term which he came up with. *Forum Theatre* strives to carve out avenues for liberation in view of oppression, such as disenfranchisement or gender inequality. The public, in the capacity of what Boal calls *spect-actors*, engage in action and dialogue with a view to resolving social problems together. I found that people working together on issues they are concerned with resonated with the format of a text adventure game. In such scenarios, an interface is presented, but what to do next is puzzling. There are many possible ways to proceed with the game, but what are they? In my wiki documentation, I discuss whether and how other concepts of importance to Boal could be instrumental to the design of text games.

There's also the **Crop Tool**, which was made using Jupyter. It's a script which produces ConTeXt output via a Command Line Interface (CLI). ConTeXt is a microtypographical typesetting system, an alternative to LaTeX, also based on Donald Knuth's TeX. I made use of mathematical and scientific python libraries (Pandas, Numpy and SciKit Learn) to implement a scale-by-percentage argument in the Crop Tool's CLI. The scale feature became a case in point of how to apply interpolation to a real-world typesetting situation. On this basis, I would describe the crop tool as an example of distributive practice. According to Yuill a practice is distributive when it includes within itself 'the knowledge of how to produce that which it produces' (Yuill, 2008, p. 69). To me this implies distribution is an inherent aspect of algorithms which articulate how their own output will be made. The process responsible for generating the output of the Crop Tool is notated in Python.

I have been a full time Linux user since the pandemic. My familiarity with the Linux command line led me to write and share various shell scripts during the first trimester. **The podcast generator** is a `fish` script which works on our collective server, chopchop. Fish, the Friendly Interactive Shell, is an alternative to Bash. The code was inspired by a brief discussion about podcasts as an archival format for radio in the first trimester. I published various drafts of the script on the Wiki. Taking ownership of infrastructure in contrast to relying on cloud services was a motivating factor in my decision to write the podcast generator script. It implicitly sought to raise awareness about the need to self-host and rely less on cloud solutions. It was good to have the opportunity to look

further into this subject during the second trimester. Ultimately, the script allowed me to archive the broadcasts from Special Issue 22 in a podcast on chopchop.

Speaking of sound, I would like to thank Joseph for the chance to perform at Varia as part of **Extratonal Infrastructure 9** in November. I also performed live coding on air during one of our broadcasts. The practice featured in my Wordhole entry on the term Performative (not reproduced here). Moreover, live coding offered a reference point in my discussion of mixing a 1/4-inch tape for another broadcast. There is much research about live coding available online. The practice is one way amongst many in which FLOSS and performance overlap. The Human Computing Git exercise which Manetta guided us through at the beginning of the second trimester is similar and different to live coding. Further examples of practices similar to this exercise exist. For example, the one hour long performative exercise *Home is a Server* by the collective bolwerK. Marthe Van Dessel describes the work as follows: 'In Home is a Server the actors execute in a very precise manner the different tasks a server is given, when one would like to publish the recipe for pancakes on a wiki' (2013, p. 164). Research into this area of performance making is more sparse than research into live coding.

Digital **typesetting** is an integral component of my publishing practice particularly when it comes to printed matter. In the second trimester I used SILE, another piece of typesetting software, to create *A Chess Diary*. This was printed and bound in collaboration with a bookmaker who attends the chess club I go to. I put my knowledge of ConTeXt into action when making *Platform is the Problem* with Senka in advance of Zine Camp.

Several weeks after Zine Camp, also at WORM, the launch of **Special Issue 22** took place. At this event there was a mismatch between my expectations for the event and my experience of it. Overall I was happier with the outcome of our more recent Special Issue 23 launch event. I allowed what did not work in the first case to guide my preparations for the following one. Accordingly, my documentation is more focussed on my contributions to the first Special Issue launch event.

For this event, I typeset the **Technical and Spiritual Manual for Post-apocalyptic Radio Making**. It was licensed under the Collective Conditions for (Re-)Use license (CC4r). The publication was controversial and I would like to revisit some implications of the license in light of that. In its questioning of authorship the CC4r highlights problems endemic to the practice of licensing. It illuminates this both within and outside of FLOSS licensing contexts. During the Laurence Rassel Show (2007) it is argued that creative commons licenses re-enforce the concept of individual authorship. The conclusion is reached that these licenses fail to deconstruct the copyright regime. At the same time they exert pressure over public perception of what is held to be common. For myself the CC4r reveals how licenses such as the General Public Licence (GPL)

have held considerable sway over the discourse of copyright-antagonism. Applying CC4r to publications such as the *Manual* encouraged me to think through ethical implications of re-use from a material perspective. Ethical concerns were raised about the publication and challenges were made during the reflective session after the launch event. Considering these issues adjacent to the license influenced how I spoke about the *Manual* when re-using pages from it in wiki documentation. I was forced to think with due diligence to clauses which I stood by, such as providing a link to an alternative version of the text.

Besides this, I worked with Rosa on **Listen Closely**, a post-digital, audio-spatial installation. The installation featured two instances of a raspberry pi connected, via a USB-audio interface, to several FM radio transmitters. The intention was to send archival content from weekly broadcasts over FM frequencies. To achieve this, I wrote a SuperCollider script which communicated with a python application via the Open Sound Control (OSC) protocol. The software was working effectively on my computer and I was confident it would run on the pis also. However, this was not the case. The piece failed insofar that I made promises which could not be consistently delivered. We planned to greet survivors of the apocalypse with an emergency welcome message. I suggested we could do this over the radios. It was memorable when the emergency message came through loud and clear over the airwaves several hours before the doors were due to open. The system's inability to perform at the right moment had an impact on the experience which the public had of the launch event. People arrived, but it was not clear what the radios were there for.

A shortcoming of my archival PDF document is that it does not feature a section on Special Issue 23. Nevertheless, I collaborated with Michel and Wang on `/var/kitchen` (AKA **The Greasy Chip**) for the *Peripheral Centers and Feminist Servers Webquilt*. The challenges were different this time but our project came together well. I sketched out the content page and used Xterm.js to run a python process in a terminal in the browser. The running process was a game based on a recipe for chickpea curry which I found online. The game was non-linear, and took account of deviations from the exact recipe in more or less subtle ways. It was written in Python, like Fix The Wifi, and it would have benefited from a help feature. The public were invited to try and figure out how to play the game on the basis of the description of ingredients and the cooking method. This information was available in a zine along with the secret steps for cooking chickpea curry. Michel worked on the Zine, which was also available digitally in the web interface. They worked on the CSS also. We all wrote about ways in which various ingredients were analogous to components in servers, specifically feminist servers.

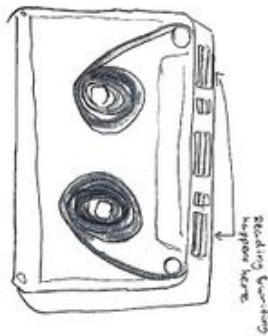
In summary, I have distilled the self-directed research which I have conducted since September into a fifteen-minute presentation. Throughout the presentation I have

frequently returned to the subject of performance as it intersects with artistic FLOSS practices. I discussed live coding but also touched on where, how and why this field offers ripening opportunities for further research in other directions. My interest in this topic is accompanied by digital typesetting practices. Lastly, for example through discussing python scripts, I have evidenced new developments in my technical abilities in collaborative settings. Topics discussed during this presentation are accompanied by documentation which I have aimed to produce with consistency over the past two trimesters.

Trimester One

Prototyping

(Re)Prerecording a tape



Anatomy of a Cassette Tape, illustration by Victor Utne-Stiberg

In preparation for broadcasting on Tuesday 19 September, Maria suggested producing pre-recorded, taped material to share on air. In response to this, Riviera and Victor experimented with a Sanyo Stereo Radio Double Cassette Recorder and a collection of cassette tapes.

Ideas

- Record the sounds in between different radio stations (Maria)
- Layer the sounds by covering the erase head (Joseph)

Experimenting with the Capstan and Pinch Roller Openings

In a conversation with Victor, Joseph highlighted that it is possible to 'layer' recordings onto a cassette tape. This is achieved by covering the Pinch and Capstan Roller Openings on a given cassette. Whilst playing around with the Sanyo machine Victor and Riviera figured out how to do this using masking tape.

We found out that the right-hand opening corresponds to the erase head, whilst the left hand opening is where reading takes place. By covering the right-hand opening, we were able to layer the recording. The effect of layering sounds raised questions about composition and the relationship between *noise* and *background*.

Composition

We first tried recording an amalgamation of sounds: classical music, radio signals, interference and noise. The tape became a document of what we did throughout the recording session and as such it bore a particular relationship to time. Parts of the tape

feature voices, quietly talking underneath classical music. Elsewhere there are blends of classical music. Other parts feature classical music alongside radio signals. Overall the quality of the composition is a mixture of experimentation, learning how to use the technology and various effects. This also led towards questioning authorship with Victor pointing out that the sounds one receives between radio stations are multiply authored.

Hardware Effects

The Sanyo device has a built in radio capable to picking up long-wave, medium-wave, short-wave and FM signals. It also has a button which switches between 'dub speed' and 'beat cancel'. Pressing this button changes the speed at which the tape plays. Thirdly, it has a dial which filters out high and low frequency sounds. The machine does not display the amount of time which has elapsed since the tape began playing. Due to this, it is necessary to estimate amounts of time when recording.



Hack a Tape, collage by Victor

Generating an echo/offset effect

1. record recorded material onto a second tape
2. record that recording onto the first tape, offset by some time.

Go Fish

In the shell I find a marvelous mess of constellations, nebulae, interstellar gaps, awesome gullies, that provokes in me an indescribable sense of vertigo, as if I am hanging from earth upside down on the brink of infinite space, with terrestrial gravity still holding me by the heels but about to release me any moment.

Nancy Mauro-Flude, 2008

The shell is a computer programme which launches other programmes. This efficient, textual method of interacting with a computer raises practical and ethical questions. On a practical level the question arises as to which shell to choose from. bash is the default login shell on Linux distributions such as Debian and Rasbian. However, it is possible to choose a different shell. This choice leads to a discussion of the ethical issue of accessibility. A feature of this discussion involves unpacking the relationship between norms and defaults. In this wiki page I outline some implications of switching to a shell other than bash by way of several examples. I describe why I have chosen to use one shell, rather than another shell and argue in favour of fish as a default shell. Overall I argue that shell scripting (with fish or bash) could be an effective way in which to activate Radio Worm's sonic archive.

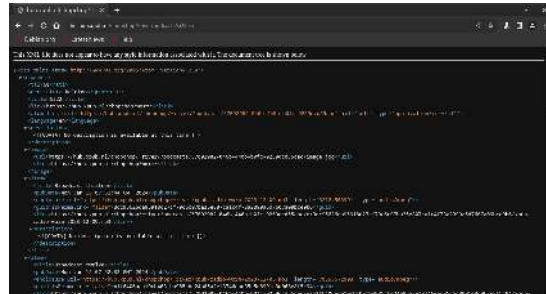
fish in practice

fish (<https://fishshell.com/>) has many features that other shells do not have. This includes enhanced autocompletion, a different way of scripting, improved syntax highlighting and idiosyncratic configuration. Three quarters of these features make it easier to get started with using the command line to interface with the computer. In implementing a different scripting language to bash, fish scripts pose a hurdle relating to interoperability. One must have access to a working installation of the software to run the script and often this is not the case by default.

fish departs from it's normative counterpart bash in both helpful and obscure ways. Firstly, there is no file such as .fishrc. Instead fish keeps configuration data in various user-specific and site-wide configuration files. Extending the functional capabilities of the shell involves adding scripts under `~/.config/fish/functions/`. Command expansion is written differently. For more examples, see [fish for bash users](#).

Podcasts Continued

The other week I started writing a shell script to generate podcasts in an RSS format. I first outlined the connection between podcasts and rss feeds following a discussion we had in class. Then I wrote about how I utilised `grep`, `cat` and `sed`. I combined these commands in a shell script.

A screenshot of a terminal window displaying a shell script. The script uses various shell constructs and commands like `grep`, `cat`, and `sed` to generate an RSS feed. The code is wrapped in a `#!/bin/sh` shebang and includes several `if` statements and `echo` commands for logging and output.

The aim was to write a file containing the tag-structure of an XML document whilst removing content. This outline structure became the basis of the podcast generator script, `skeleton`. Along the way I analysed and improved the first version of the script. I am summarising what I did the other week here because the commentary on the code is somewhat inaccurate. I utilised a less than consistent document production workflow to write the wiki page. However, I have since decided upon a more consistent way of writing wiki pages (using <https://pandoc.org>) and so pages will be better maintained starting from now.

What I worked on was quite relevant to our discussion of regular expressions on Monday 9th. It could furthermore bear relevance to CSS and paged media if an HTML page was made for the channel. I envision a publication made with `weasyprint`, perhaps using the technique of `imposition`. Each page might contain a link to a radio broadcast from the past, along with some text.

Discussion of the script

The script now features several additional flags. The "verbose" flag sends to stdout descriptive information about what the script is doing whilst it does it. This information is useful for debugging. It is also a more convenient means of documenting the software than writing about it. The "add" flag allows the user to add an mp3 file to a podcast. If the podcast does not exist, `skelegen` creates the channel. If the channel exists, `skelegen` adds the item to the channel. This is a flexible way of creating podcasts. Lastly, the "auto" flag automatically inserts a title and description for the item(s) / channel.

```
skelegen --generate /media/worm/radio/ --channel ~/public_html/podcast.xml
```

I turned the script into a zine using free and open source software. In particular I used Emacs, pandoc, ConTeXt and pdfcpu.

Mixing A Tape



Recording session in the studio on Thursday October 26th

This wiki post was made through a combination of writing and voice recording techniques. What I am going to do is read what I have written and expand, where appropriate, upon what I have written before returning to the main flow of the text. The text itself is fairly short. However, I feel that by expanding on what I have written by speaking, it will become longer and more detailed. I intend to then transcribe what I have said and edit the material. Then post the edited transcription on the wiki. So, without further ado...

On the afternoon of October 26, and during most of Friday 27th I made tape recordings, I am not referring to cassette tapes, but long reels of tape. The tape I was recording on to was approximately 500 meters in length, I made a recording which was the complete length of the tape on the Friday. In what follows I discuss my experiences of recording onto tape

The most effective setup

I sent audio from my laptop to channels one and two on a 16 channel mixing desk. As we can see in the photo taken on the Thursday I had several machines connected to each another. For the purposes of recording audio, connecting my laptop to the mixing

desk was one aspect of this setup. On the Friday, I also plugged a radio into channels three and four. Working with the 16 channel mixing desk was far more effective than working with the eight channel mixing desk. I did not take a photograph of my setup on a Friday. However, I will elaborate on two reasons as to why it was more effective to use the setup with the 16 channel mixing desk. Firstly, the effects panel is not working on the eight channel mixing desk, but it is working on the 16 channel mixing desk. Secondly, the eight channel mixing desk picks up a lot of noise and the 16 channel desk does not do this. I connected the 16 channel mixing desk to an Akai tape recording and playback machine. This machine belongs to one of Joseph's friends, and I made sure to be careful with it. It was According to Joseph the best tape machine out of the three which are available. Indeed, it was. I had difficulty getting the results I wanted on the pieces of hardware which I was using on the Thursday. In short, I was recording sounds from two devices on to tape and using the mixing desk as a way to get these machines to communicate.

Drawing on the prototyping classes, which we had on the Monday and the Tuesday was key to the activities I engaged in later in the week. In these prototyping classes, we looked at how to record sounds in analog and digital ways. We also created HTML audio mixers with functions which enabled looping the audio file and adjusting the playback rate with greater granularity. I built upon this HTML soundboard by adding eight separate audio tracks and adding a volume level slider. Each audio recording was a file in a series of recordings in Worm's radio archive. There were many variables which allowed me to alter the sound. This came from all of the hardware that I was using. I had plugged the laptop with this setup into the mixing desk, so that was one set of variables I could control (the playback volume and the playback speed). As I could also choose particular files and the time at which to play them from it was possible to layer these audio recordings. However, all the tracks were one hour in length. It was simpler to simply reduce the playback rate, so that the tracks continued to play for a very long time. In part, I was inspired by my previous collaboration with Victor in which we made a tape recording and overlaid sounds onto the tape. I placed this idea in a digital context and played multiple tracks simultaneously for an overlaid sound effect. These signals went directly into the mixer, the analog mixer, the physical piece of hardware made by Behringer. Specifically, the signals went into channels one and two which were panned to left and right respectively. The signal was then sent to the first and second output bus channels. The hardware has four output bus channels which meant that I was able to set four output levels individually.

Moreover, there were two pieces of hardware that I connected to the mixer. one of the pieces of hardware was my laptop, the other piece of hardware was a radio. I added the latter into the mix on the Friday in order to get radio signals. Partly because it was mentioned that radio was effective back in week one. I wanted to retrieve similar sounds

and combine with other sounds on the tape. And sometimes it works sometimes it didn't work. A time when it was effective was when I took out the connection between the radio and the mixing desk in order to tune it to a signal which I thought sounded right before reintroducing the sound. I was able to do this because of the four output bus channels. The radio was plugged into input channels three and four, which were panned also to left and right initially. As before, channels three and four were being sent to output bus channels three and four. I could set the levels of each of these output bus channels individually so what I could do was fade the radio all the way down, take out the connection, find a signal that I liked, remake the connection and then reintroduce the sound of the radio.

Likewise, there were physical buttons on the mixer, which controlled which output bus channel the input signal was to go to. It was possible to press eight of these buttons simultaneously with eight fingers and switch all the channels going to buses one and two to buses three and four and vice versa. This had a really pleasant effect. If the levels of output bus channels two, one, three and four were set to different levels, then it was possible to get a really nice effect by redirecting signals to the different levels of the master faders. It was also good that the effects panel was working, because I was able to apply effects such as reverb, flange and delay to the signal. For example, I applied a flange effect to voices speaking live on the radio. Because the effects panel was broken on the eight channel mixing desk, which I was using on a Thursday, it wasn't possible to apply these effects during that recording session.

The following table lists the variables which I was in charge of during the recording session.

Laptop	Radio	Mixing Desk	Tape Machine
Playback Rate	Volume	Pan	Line levels
Volume	(Fine) Tuning	Gain	
Loop function	Tone	EQ	
		Send buttons	
		Effects	
		Levels	

Whilst experimenting with these machines it was important, above all, to monitor the levels of the lines into the tape machine. I wanted to be as careful as possible about ensuring that the sound was not clipping or peaking and putting too much pressure on the tape machine. Conveniently, the tape machine has physical, analog indicators which show the amount of volume which is going through the left and right channels at any given time. These indicators move independently of one another. There's a lot

of manual engagement with the technology which lends itself to subtle differences in volume or subtle differences in pan and gain. This entails different experiences in the left and right channels. Furthermore the sounds I was recording were stereo signals. For these reasons it's very important to monitor the levels going into channels one and two because they can vary sometimes considerably.

Perhaps I will record more onto the tape which I was using to make it different. Currently, it's quite lengthy. It lasts for approximately an hour and a half I imagine. It would be very straightforward to plug more devices into this setup. I think there's an opportunity to add to the content of the tape, to change it up, to introduce other tapes halfway through at different times. This would add in more discontinuity. But returning to the table, which I was speaking about. On the mixing desk, there was as I mentioned, a pan as well as the equalizer. And the equalizer was nice. Sometimes if it got too bassy, you could turn the bass down. Or if I wanted to emphasise the treble, I could increase the treble level. In general, one could record a variety of sounds into the tape using the mixing desk as a intermediary.

The influence of live coding

I have titled this section 'The influence of live coding' because of the performative quality of the experience of experimenting as I did. In short, I regard performance as the connective tissue which binds live coding to recording onto tape. In my previous broadcast with Senka and Lorenzo I attempted to live code with sounds taken from Worm's radio archive. That was an exciting and yet challenging experience for several reasons. Firstly, it was quite audible when I was typing. That was disrupting the experience of listening to the algorithms because there was this thumping noise in the background. Secondly, live coding was tricky because the samples were of a very variable length. Normally I live code with percussive samples and semi-percussive tones. Working with samples of various lengths was challenging. However, I believe this was something which this experiment with the tape recorder could have benefited from. I selected audio recordings from Worm's archive using broadcasts which were from the same folder. Presumably they were all part of the same series of broadcasts. All were exactly an hour in length and I hoped their similarities would produce an effect that was aesthetically consistent.

However, the tracks were somewhat dissimilar in content. There were times when you heard people shouting. There were times when there was heavy club music. There were other times when there were bird noises, for example. So there was a lot of variety in these shows. But primarily, I used instrumental sounds. If I had used shorter sounds that might have been effective because of the loop functionality of the HTML mixer which I had made. The ability to loop could have been used to a rhythmic effect. But

that's not something which I attempted. Nevertheless, I imagined using the browser-based mixer as a live interface with which to create sounds on the recording. It's not live coding, just mixing sounds in the browser. Nevertheless, I've been reflecting on my live coding practice having spent some time in a wood workshop and in a ceramics workshop. It was interesting to work with these materials. I had worked with them in a limited capacity when I was a pre-teenager but not since. And it's interesting to come back to these materials after so many years. They never really featured previously in my practice, in any way whatsoever. So, spending time in the Wood workshop, one afternoon, turning wood, just to see what it was like. And to create various shapes with curves out of a square block of wood was extremely satisfying. And got me thinking about crafting materials. With the ceramics, the plasticity of the material, the ability to transform it into so many different shapes, made it difficult to work with. On the whole, cutting wood with machines is more mathematical than preparing and shaping clay.

I started to think about these experiences in terms of my own practice. For me it's about shaping something. I was thinking about the materiality of code and I was thinking about shaping code in the same way that one shapes a piece of wood, or a block of clay. With live coding, one can start with an algorithm which is so simple and expand it into a different shape. Or apply other variables and pass different keywords to the algorithm. Or construct it differently. It's possible to shape visual and sonic output in a way analogous to shaping wood and clay. I hadn't thought about live coding in that way before. I thought about it, in part, as a demonstration of how I was using a computer, because I felt like it was quite different to the way many other people use computers. Also, live coding is performative and I was trying to make performance art. So, live coding seemed like a practical solution. And I didn't think about it in relation to workshop practices involving materials such as clay and wood.

In a sense, the experiment required me to turn from software to hardware. I was using buttons and dials to shape the sound and experiment with different effects. What happens if I switch the button to a different position? How do I ensure that the signal coming through is loud and clear? What happens if the bass is too high and needs to be reduced? It was deliberate experimentation with particular constrictions except that the constrictions are there in front of you. They're not concealed within the source code, even open source code. Because there is a physical interface to work with.

Broadcast Seven

On the morning of Halloween 2023, me, Senka and Victor met at Worm to broadcast a radio show. The title of the broadcast was 'Personal Accounts of Irreplaceable Lace: A 200 year long history of lost and found'. This wiki post discusses that broadcast. I focus on the discussions which were had in the run up to the show, the broadcast itself and the feedback which was given in the Aquarium during the afternoon.

Making the Broadcast

In preparation for the broadcast I recorded a 45 minute long 1/4" tape. That process is discussed [here](#). Initially, there was a plan to transport the tape machine to the Radio Booth at Worm. In the end, I decided against this idea as the machine belonged to one of Joseph's friends and the bag for transporting the equipment did not seem very sturdy. It would have been necessary to carry the machine to Worm (it's heavy) and bring it back again. To reduce the chances of potentially damaging the machine, I instead plugged the Akai GX-4400D into a Sony IC Recorder ICD-PX470. Conveniently, the handheld recorder has a line input which made making a digital copy quite simple. The digital file was then broadcast on Tuesday.



Sony IC Recorder ICD-PX470

Me, Senka and Victor had three meetings in advance of the broadcast. During these meetings we shared ideas, produced material and rehearsed what we would perform. Our first meeting was mostly conversational and sought to demarcate the direction in which we would take the broadcast. There was discussion of aliens and zombies, resources were shared and the decision was made to craft a narrative. Our next meeting was both longer and more productive. We figured out that the narrative would be split into three sections: past, present and future. The narrative would span 200 years from 1928 to 2128. We came up with names for the archivists who would be the protagonists of the narrative: Erhka and Akhre. These names are anagrams of Arkhe, the Greek word from which the English term "archive" is derived. Erhka, it was decided, would be an archivist from the past whilst Akhre would be the archivist from the future.

Victor took the future, Senka took the past and that left me with the present. The middle section of the broadcast was structured a bit differently to the opening and closing sections. Rather than writing character-driven, fictional voice memos, I spoke as a narrator and discussed the 'history' of the archive between 1998 and 2057. This was with the agreement of Senka and Victor.

During our second meeting, we worked with three pieces of paper, one for each time period. We decided an 'exquisite corpse' would be an effective way to generate material, and introduce a diversity of ideas into the narrative. Each of us had a say in each section of the storyline, providing ideas, keywords and possible plot points. We then transcribed the paper based notes onto an Etherpad so that each of us could be reminded of the content of the discussion and what had been written down. Because it was Halloween we were interested to make a story with elements of Gothic horror, but also speculative fiction. We explored dichotomies and themes which would characterise each time period. The following table illustrates these themes.

Past	Present	Future
Birth of Modernism	Liminal	Solar Punk
Industrial Revolution	Translation	Alive (Undead)
Consumer Society	Passion (Passive + Active)	Hopeful
European politics	Change (Climate (/) Politics)	Repurposed
disposable things		

To forge a coherent narrative, I built on what had been discussed in our meetings and copy which had been written in the Etherpad. For example, Senka wrote about the quality of the material of Lace — a fictional audio recording material which we invented and around which the narrative unfolded. There was a practical reason for inventing the material; at first we thought simply to go with tape. However, Senka, whilst researching the historical context of the history of tape discovered something surprising. Tape was invented in Germany in the 1920s, however it was kept a secret and was not widely used until the end of WWII. After finding this out Senka wrote about it in the Etherpad. They made a few suggestions about how we might proceed:

1. Adjust the dates of the fictional narrative to fit with the history of tape
2. Write an alternative history in which WWII never happened

We agreed that the first option was inconvenient; beginning in the 1950s would have been disruptive to what Victor and I had written. The second option was inspired by a podcast Senka had heard of called 'Within the wires'. This podcast was a fictional narrative which imagined a 20th Century without the concept of a nuclear family. The

podcast, to paraphrase Senka, elaborated on an alternative history in which WWII never happened. This was partly because nation states looked very different because nuclear families were considered barbaric. I suggested that whilst that might work in that podcast, it might not work in our broadcast. An alternative, I suggested, was to imagine a fictional audio recording medium. Doing so would allow us to keep the dates the same without ending in the murkiest territories of alternative histories about WWII. Senka responded with an enthusiastic 'YES' in the Etherpad and so we went ahead with the history of lace. I decided to call the material lace because I thought that flat shoelaces looked a bit like magnetic tape, but were sufficiently different.

Senka went on to describe the qualities of this fictional material. I expanded on Senka's discussions of the medium, referring to it's 'gummy' quality and the optimal temperature at which it should be stored. This provided consistency in the narrative. As the author of most of the material in the present moment, I also endeavoured to leave the plot open such that Victor could pickup on narrative strands in the future. Likewise I was inspired by what he had written and sought to write copy which would bridge the lives of Erhka and Akhre.

At our third meeting, I played Victor and Senka the tape I had made. Whilst doing so, we read through the material we had produced to check how much time it would take to read aloud. Following the read-through, we fed back to each other, highlighting plot-points and discussing opportunities to make the broadcast more coherent. We next collaborated on developing the written material through another exquisite corpse exercise. With regard to the structure of the broadcast, we decided to intersperse the sounds of the tape and the spoken narrative elements. This was both a practical — it was difficult to speak over the noisy sounds — and aesthetic — to suggest the passing of time — decision. At this meeting, I also recorded the sound of the tape being rewound from the end to the beginning. Vice versa, I recorded the tape being played on fast forward from the start to the end. This gave us an additional ten minutes of audio material.

Broadcasting on Tuesday



Victor, Senka and Riviera in Worm's Broadcasting Studio

We met around 9:45 at Worm to set up for the broadcast. I had borrowed a mini-jack to phono cable and two phono to jack adaptors from the XML lab to connect my laptop to the mixing desk. Generating pre-recorded material and playing this on air was pragmatic for several reasons. Firstly, we knew how much material we would have. Secondly, and more importantly, it allowed us to mute the microphones in the radio booth and confer with each other about our next steps during the broadcast. This was in contrast to the first broadcast I made with Lorenzo and Senka in week three. Then, the mics were live for much of the broadcast which gave us little opportunity to discuss with each other and cue various moments.

Unbeknown to we caretakers, the streaming software which normally runs in the background on the computer in the radio booth was not running. As such, it only transpired that we were not broadcasting what we were performing after forty minutes. Senka's partner messaged Senka and asked when we were going to start speaking. It turned out that ska music and string instruments were being broadcast for a third of the time that we were supposed to be on air. Fortunately, we were recording everything and there was an opportunity to replay the 'lost' audio to the group on Tuesday afternoon. Doing so, the group agreed, made the narrative make more sense.

We encountered a second technical hitch during the broadcast. Fortunately, this arose just after we fixed the streaming issue so we were not confronted with two issues simultaneously. In short, Audacity stopped working. Why? Because the file had not been saved to the correct location and the computer hard drive rapidly became full. Thankfully Florian was there and he helped us fix both issues as they arose. We lost a couple of minutes of the tape recording being broadcast in the recording, but this was a relatively inconspicuous error. It was a good stroke of luck that the recording did not stop whilst we were talking / telling the narrative.

Feedback following the broadcast

Due to the technical difficulties which we encountered in the morning part of the afternoon was spent listening to the part which was missed. This was important, because everything which Senka had written and said was not broadcast, which made the narrative difficult to understand. It was good, therefore, to have time to listen to this section of the broadcast in the afternoon.

As Thijs pointed out, we caretakers had messaged the group encouraging everyone to stay at home, rather than meet in the Aquarium for a more comfortable listening experience. We also did not share information about what it was we would be broadcasting but preferred to keep this a secret/surprise. During the feedback session in the afternoon, I contended that it would be better if, moving forwards, we were more open with each with regards to the content of the broadcast. There were two reasons for this. Firstly, it was a practical matter. If, as it happened, something were to go wrong, keeping each other in the loop about what to expect could ensure that some technical issues could be detected quicker and resolved sooner. Secondly, it was a matter of audience. We are aiming to reach audiences broader than our own group. Thus, there are opportunities to input on, feedback about and shape each others practices prior to moments of publication. We are valuable resources for one another and we can afford to take advantage of that. I emphasised that the broadcasts, the moments at which things are made public, can still be novel and exciting for wider audiences.

Another key takeaway from the feedback session was as follows. It would have been interesting to talk about the methods we utilised in making the broadcast within the broadcast itself. This had occurred to me in a slightly different guise during the week. The reason we decided not to do this was because we wanted to immerse the audience in the narrative. However, half of the broadcast was full of prerecorded audio. It would have been possible to cut this down somewhat to allow for a discussion of the methods which we utilised to make the narrative.

Editorial Team Meeting Recap

I took the bakfiets from WORM to Karel Doormanhof this afternoon. When I arrived back at the Wijnhaven building I found that two groups had formed: the editorial team and the installation team. I joined the editorial team and spoke with Senka, Mania and Michael about what had been going on. I asked for the readers digest of the conversation so far. It had been decided to hold a series of interviews for the special issue. Connecting with people at WORM, such as Ash, Lieuwe and Lukas would be the starting point.

Possible Questions for Structured Interviews with Radio Makers

We spoke about interviewing radio makers in a structured way. That is, putting the same questions to each interviewee to collect data which could be better tabulated and drawn upon in a structured way.

Here are the questions which were written in the Etherpad. Perhaps we could vote on these to decide what to ask radio makers?

- Who are your listeners?
- Who are you listening to?
- Do you have an archive?
- Is the element of liveness important to you (does “archiving” matter)?
- How do your listeners use it?
- How did you start with radio worm?
- Why/What is important to make public / in making a public?
- What does (Radio) Worm mean to them (two questions ;)?
- What do you not choose to broadcast? (public / private) What are the errors? (Ash's example of turning on the microphones to hear the way decisions get made)
- What languages do you use in your radio? (access and who can understand these languages)
- Would you like to be include in the archive and if so how? (Are there “other” materials/recordings that could represent your broadcast / audience)
- How do you share the (radio) air?
- What do you find important to transmit and broadcast and why?

Saturday 18th: Extratonal Infrastructure 9



On Saturday 18th I had a gig at Varia. I spent some time preparing for the performance in the run up to it. At the event I performed for 20 minutes with Fluxus and Tidal Cycles, both of which are free / open source pieces of software. Here's some of the feedback I received from the audience:

'Congratulations'

'You're so fast at coding'

'It's very interesting what you are doing, I've never seen this before'

'You created a universe and I cannot do that'

'You should create more compositions'

The audience were quiet throughout the performance. Perhaps it would have been better if more of them had been seated during the performance. However, I chose not to force the audience sit down, some voluntarily sat on the floor. In any case I was averagely satisfied with the performance. The performance hadn't changed much since I last performed it in March. If anything I was a bit rusty. I had a good conversation with the violinist after the concert about music theory. In particular, I found out about the tonic, subdominant and dominant pattern within scales.

Now that I have done my performance at Varia, I am left thinking about performance practices involving free software which do not feature live coding. I have amassed a considerable amount of material in preparation for the performance. I have at least 41 practice sessions from November 2023. I could execute shell commands on these files to gain statistical data about the files. Perhaps there's a way I can incorporate this into the .fm (fragmented media) file type.

Special Issue #22

Listen Closely

OSC is a communication protocol for many pieces of audio software and hardware. During week nine and ten, Rosa and Riviera began developing a prototype for the FM radio transmitters. There are several of these transmitters and an equal number of receivers. On these transmitters we intend to broadcast excerpts from Worm's Radio archive, including the contributions we have made as part of SI#22. To achieve this, Riviera and Rosa realised it would be necessary to develop some software. Figure 2 illustrates the concept we have in mind for the hardware. We believe this setup is more pragmatic than connecting a raspberry pi to each radio transmitter, although this would be possible. Figure 3 illustrates how the software works.

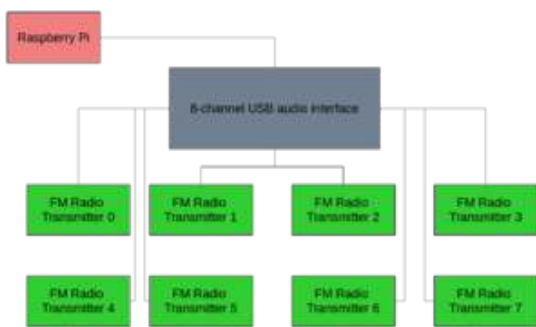


Figure 2: Hardware connections diagram

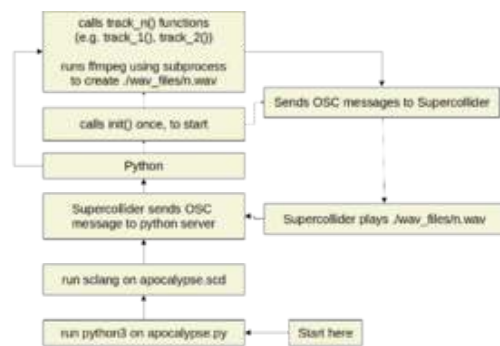
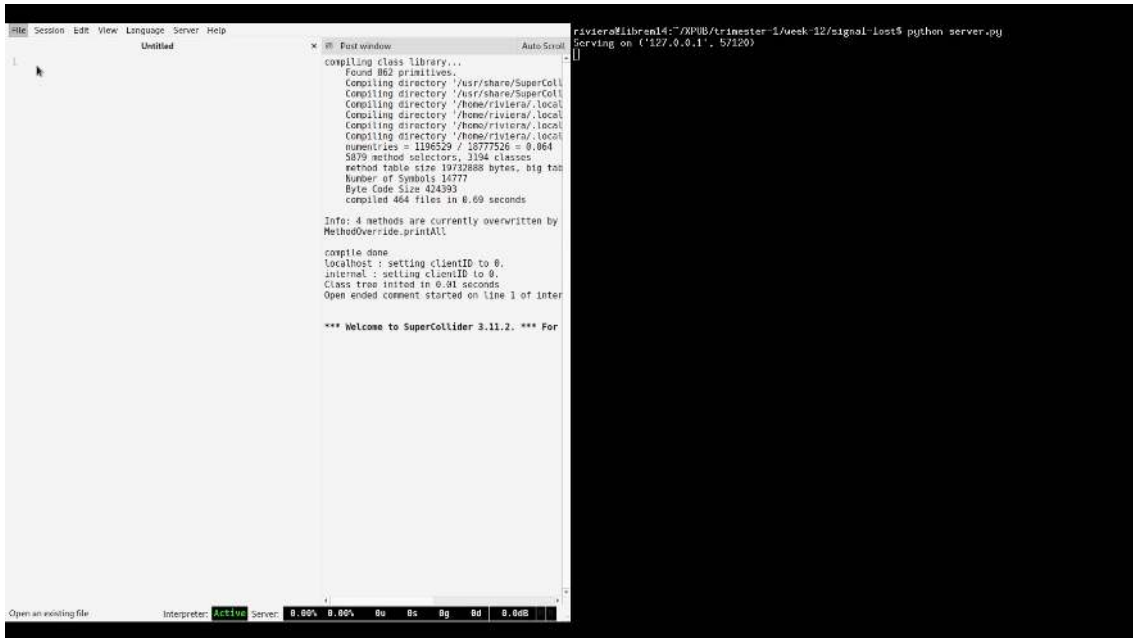


Figure 3: Python / Supercollider OSC Communication

A supercollider server running on the raspberry pi will expect the pi to be connected to a USB audio interface with eight outputs. The supercollider server listens for OSC messages on port 58110. Simultaneously running on the Pi is another OSC application based on the python-osc python library. The python application is comprised of a server, dispatcher and client. The dispatcher is essential because it is capable of calling the client when the server receives a message. The python client sends OSC messages to the Supercollider server. Supercollider sends messages back to the python server. The OSC messages contain a 'channel' and a path to a WAV file. To begin with, start the python server. Then execute the supercollider document with sclang. Supercollider will introduce itself to the python server, and then the process of playing audio files enters into a loop.

The following images illustrate the software side of *Listen Closely*. Specifically the interplay between supercollider and a python OSC application.



```
File Session Edit View Language Server Help
mixer.scd
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.

compiling class library...
Found 862 primitives.
Compiling directory /usr/share/SuperColl
Compiling directory /usr/share/SuperColl
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
numentries = 1196529 / 1877526 = 0.064
5879 method selectors, 3194 classes
method table size 19732888 bytes, big tab
Number of Symbols 14777
Byte Code Size 424393
compiled 464 files in 8.69 seconds

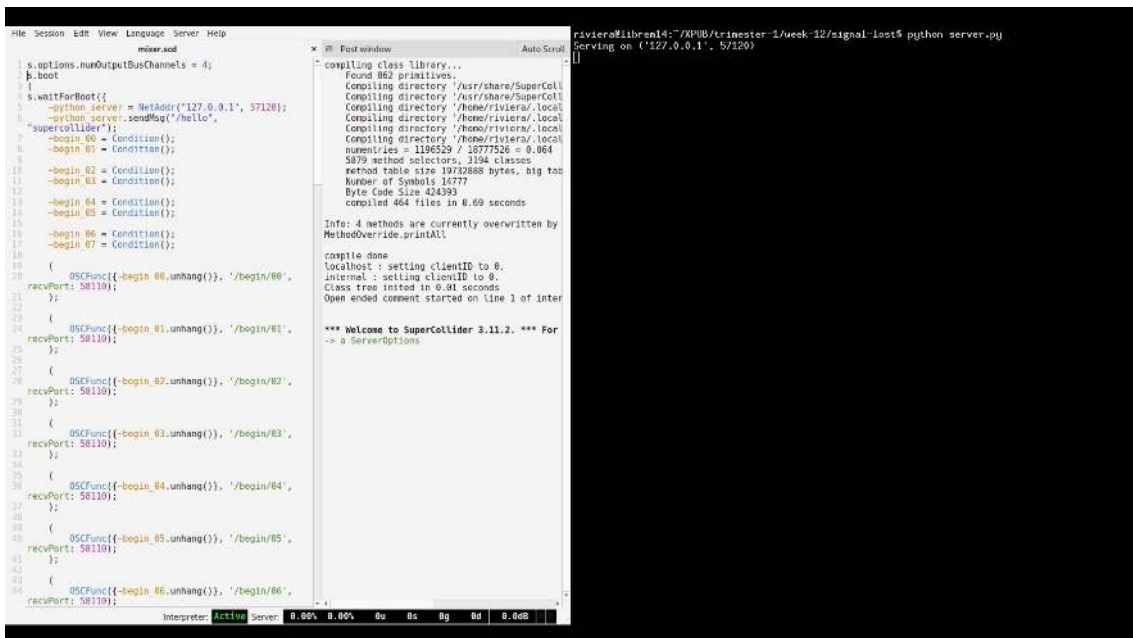
Info: 4 methods are currently overwritten by
MethodOverride.printAll

compile done
localhost : setting clientID to 0.
internet : setting clientID to 0.
Class tree init'd in 0.91 seconds
Open ended comment started on line 1 of inter

*** Welcome to SuperCollider 3.11.2. *** For
-> a ServerOptions

Interpreter: Active Server 0.00% 0.00% 0u 0s 0g 0d 0.00B
```

Supercollider and XTerm



```
File Session Edit View Language Server Help
mixer.scd
1. s.options.numOutputBusChannels = 4;
2. boot
3. {
4. s.waitForBoot{
5. -python server = Method('127.0.0.1', 57120);
6. -python server.sendMsg('hello');
7. 'supercollider';
8. -begin_00 = Condition();
9. -begin_01 = Condition();
10. -begin_02 = Condition();
11. -begin_03 = Condition();
12. -begin_04 = Condition();
13. -begin_05 = Condition();
14. -begin_06 = Condition();
15. -begin_07 = Condition();
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.

OSCFunc{(-begin_00.unhang()), '/begin/00',
recvPort: 58110;
};
OSCFunc{(-begin_01.unhang()), '/begin/01',
recvPort: 58110;
};
OSCFunc{(-begin_02.unhang()), '/begin/02',
recvPort: 58110;
};
OSCFunc{(-begin_03.unhang()), '/begin/03',
recvPort: 58110;
};
OSCFunc{(-begin_04.unhang()), '/begin/04',
recvPort: 58110;
};
OSCFunc{(-begin_05.unhang()), '/begin/05',
recvPort: 58110;
};
OSCFunc{(-begin_06.unhang()), '/begin/06',
recvPort: 58110;
};

compiling class library...
Found 862 primitives.
Compiling directory /usr/share/SuperColl
Compiling directory /usr/share/SuperColl
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
numentries = 1196529 / 1877526 = 0.064
5879 method selectors, 3194 classes
method table size 19732888 bytes, big tab
Number of Symbols 14777
Byte Code Size 424393
compiled 464 files in 8.69 seconds

Info: 4 methods are currently overwritten by
MethodOverride.printAll

compile done
localhost : setting clientID to 0.
internet : setting clientID to 0.
Class tree init'd in 0.91 seconds
Open ended comment started on line 1 of inter

*** Welcome to SuperCollider 3.11.2. *** For
-> a ServerOptions

Interpreter: Active Server 0.00% 0.00% 0u 0s 0g 0d 0.00B
```

The code for Listen Closely is loaded in SuperCollider

```

File Session Edit View Language Server Help
mixer.aod
1 s.options.numOutputBusChannels = 4;
2 s.boot
3
4 s.waitForBoot{
5   ~python_server = NetAddr("127.0.0.1", 57120);
6   ~python_server.sendMsg("hello");
7   ~superCollider();
8   ~begin_00 = Condition();
9   ~begin_01 = Condition();
10  ~begin_02 = Condition();
11  ~begin_03 = Condition();
12  ~begin_04 = Condition();
13  ~begin_05 = Condition();
14  ~begin_06 = Condition();
15  ~begin_07 = Condition();
16
17 (
18   OSCFunc{~begin_00.unhang(); "/begin/00",
19   recvPort: 58119;
20   };
21 );
22
23 (
24   OSCFunc{~begin_01.unhang(); "/begin/01",
25   };
26 );
27 (
28   OSCFunc{~begin_02.unhang(); "/begin/02",
29   recvPort: 58119;
30   };
31 );
32 (
33   OSCFunc{~begin_03.unhang(); "/begin/03",
34   };
35 );
36 (
37   OSCFunc{~begin_04.unhang(); "/begin/04",
38   recvPort: 58119;
39   };
40 );
41 (
42   OSCFunc{~begin_05.unhang(); "/begin/05",
43   };
44 );
45 (
46   OSCFunc{~begin_06.unhang(); "/begin/06",
47   recvPort: 58119;
48   };
49 );
50
Interpreter: Active Server: 0.52% 0.52% 0h 0s 2g 107k 0.00s

```

```

rivierra@libren14:~/XPUB/trimester-1/week-12/signal_lost$ python server.py
Serving on ('127.0.0.1', 57120)

compiling class library...
Found 862 primitives.
Compiling directory /usr/share/SuperColl
Compiling directory /usr/share/SuperColl
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
Compiling directory /home/riviera/.local
numentries = 1196529 / 1877526 = 0.064
5879 method selectors, 3194 classes
method table size 1973288 bytes, big too
Number of Symbols 14777
Byte Code Size 424393
compiled 464 files in 0.69 seconds

Info: 4 methods are currently overwritten by
MethodOverride.printAll

compile done
localhost : setting clientID to 0.
internet : setting clientID to 0.
Class tree inited in 0.01 seconds
Open ended comment started on line 1 of Inter

*** Welcome to SuperCollider 3.11.2. *** For
-> a ServerOptions
-> localhost
Donting server 'localhost' on address 127.0.0
Found 74 LADSPA plugins
JackDriver: client name is 'SuperCollider'
SC_AudioDriver: sample rate = 48000.000000, d
JackDriver: connected system:capture 1 to Su
JackDriver: connected system:capture 2 to Su
JackDriver: connected SuperCollider:out 1 to
JackDriver: connected SuperCollider:out 2 to
SuperCollider 3 server ready.
JackDriver: max output latency 42.7 ms
Requested notification messages from server
localhost: server process's maxLogins [1] net
localhost: keeping clientID (0) as confirmed
Shared memory server interface initialized

-> localhost
/home/riviera/Music/signal_lost/wav_files/00
/home/riviera/Music/signal_lost/wav_files/02
/home/riviera/Music/signal_lost/wav_files/04
/home/riviera/Music/signal_lost/wav_files/06

```

The supercollider server is booted up

```

File Session Edit View Language Server Help
mixer.aod
1 s.options.numOutputBusChannels = 4;
2 s.boot
3
4 s.waitForBoot{
5   ~python_server = NetAddr("127.0.0.1", 57120);
6   ~python_server.sendMsg("hello");
7   ~superCollider();
8   ~begin_00 = Condition();
9   ~begin_01 = Condition();
10  ~begin_02 = Condition();
11  ~begin_03 = Condition();
12  ~begin_04 = Condition();
13  ~begin_05 = Condition();
14  ~begin_06 = Condition();
15  ~begin_07 = Condition();
16
17 (
18   OSCFunc{~begin_00.unhang(); "/begin/00",
19   recvPort: 58119;
20   };
21 );
22
23 (
24   OSCFunc{~begin_01.unhang(); "/begin/01",
25   };
26 );
27 (
28   OSCFunc{~begin_02.unhang(); "/begin/02",
29   recvPort: 58119;
30   };
31 );
32 (
33   OSCFunc{~begin_03.unhang(); "/begin/03",
34   };
35 );
36 (
37   OSCFunc{~begin_04.unhang(); "/begin/04",
38   recvPort: 58119;
39   };
40 );
41 (
42   OSCFunc{~begin_05.unhang(); "/begin/05",
43   };
44 );
45 (
46   OSCFunc{~begin_06.unhang(); "/begin/06",
47   recvPort: 58119;
48   };
49 );
50
Interpreter: Active Server: 0.95% 0.95% 16s 4s 2g 111k 0.00s

```

```

rivierra@libren14:~/XPUB/trimester-1/week-12/signal_lost$ python server.py
Serving on ('127.0.0.1', 57120)

will send a message /begin/00 /home/riviera/Music/signal_lost/wav_files/00.wav
will send a message /begin/02 /home/riviera/Music/signal_lost/wav_files/02.wav
will send a message /begin/04 /home/riviera/Music/signal_lost/wav_files/04.wav
will send a message /begin/06 /home/riviera/Music/signal_lost/wav_files/06.wav

```

Supercollider requests four tracks from the python application, which the python application sends across

```
File Session Edit View Language Server Help
Post window
Found 862 primitives.
Compiling directory '/usr/share/SuperCollider/SCclasslibrary'
Compiling directory '/usr/share/SuperCollider/Extensions'
Compiling directory '/home/riviera/.local/share/SuperCollider/Extensions'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/Vowel'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/Birt-S'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/SuperD'
nentries = 1196529 / 1877526 = 0.664
5879 method selectors, 2194 classes
method table size 1932888 bytes, big table size 158270208
Number of Symbols 14777
Byte Code Size 42493
compiled 464 files in 8.69 seconds

Info: 4 methods are currently overwritten by extensions. To see which, execute:
MethodOverride.printAll

compile done
localhost : setting clientID to 8.
internal : setting clientID to 0.
Class tree init'd in 6.01 seconds
Open ended context started on line 1 of interpreted text

*** Welcome to SuperCollider 3.11.2. *** For help press Ctrl-D.
-> a ServerOptions
-> localhost
Booting server 'localhost' on address 127.0.0.1:57110.
Found 74 LADSPA plugins
JackDriver: client name is 'SuperCollider'
SC AudioDriver: sample rate = 48000.000000, driver's block size = 1024
JackDriver: connected system:capture_1 in SuperCollider:in_1
JackDriver: connected system:capture_2 to SuperCollider:in_2
JackDriver: connected SuperCollider:out_1 to system:playback_1
JackDriver: connected SuperCollider:out_2 to system:playback_2
SuperCollider: s server ready.
JackDriver: max output latency 42.7 ms
Requested notification messages from server 'localhost'
localhost: server process's maxoutages (1) matches with my options.
localhost: keeping clientID (8) as confirmed by server process.
Shared memory server interface initialized
-> localhost
/home/riviera/Music/signal_lost/wav_files/00.wav
/home/riviera/Music/signal_lost/wav_files/02.wav
/home/riviera/Music/signal_lost/wav_files/04.wav
/home/riviera/Music/signal_lost/wav_files/06.wav
/home/riviera/Music/signal_lost/wav_files/03.wav
/home/riviera/Music/signal_lost/wav_files/02.wav
/home/riviera/Music/signal_lost/wav_files/01.wav
/home/riviera/Music/signal_lost/wav_files/05.wav

Interpreter: Active Server: 1.14% 1.15% 16s 4s 2g 113s 0.66s
```

When the tracks conclude, supercollider requests more audio via OSC messages sent to the python application

```
File Session Edit View Language Server Help
Post window
Found 862 primitives.
Compiling directory '/usr/share/SuperCollider/SCclasslibrary'
Compiling directory '/usr/share/SuperCollider/Extensions'
Compiling directory '/home/riviera/.local/share/SuperCollider/Extensions'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/Vowel'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/Birt-S'
Compiling directory '/home/riviera/.local/share/SuperCollider/downloaded-quarks/SuperD'
nentries = 1196529 / 1877526 = 0.664
5879 method selectors, 2194 classes
method table size 1932888 bytes, big table size 158270208
Number of Symbols 14777
Byte Code Size 42493
compiled 464 files in 8.69 seconds

Info: 4 methods are currently overwritten by extensions. To see which, execute:
MethodOverride.printAll

compile done
localhost : setting clientID to 8.
internal : setting clientID to 0.
Class tree init'd in 6.01 seconds
Open ended context started on line 1 of interpreted text

*** Welcome to SuperCollider 3.11.2. *** For help press Ctrl-D.
-> a ServerOptions
-> localhost
Booting server 'localhost' on address 127.0.0.1:57110.
Found 74 LADSPA plugins
JackDriver: client name is 'SuperCollider'
SC AudioDriver: sample rate = 48000.000000, driver's block size = 1024
JackDriver: connected system:capture_1 in SuperCollider:in_1
JackDriver: connected system:capture_2 to SuperCollider:in_2
JackDriver: connected SuperCollider:out_1 to system:playback_1
JackDriver: connected SuperCollider:out_2 to system:playback_2
SuperCollider: s server ready.
JackDriver: max output latency 42.7 ms
Requested notification messages from server 'localhost'
localhost: server process's maxoutages (1) matches with my options.
localhost: keeping clientID (8) as confirmed by server process.
Shared memory server interface initialized
-> localhost
/home/riviera/Music/signal_lost/wav_files/00.wav
/home/riviera/Music/signal_lost/wav_files/02.wav
/home/riviera/Music/signal_lost/wav_files/04.wav
/home/riviera/Music/signal_lost/wav_files/06.wav
/home/riviera/Music/signal_lost/wav_files/03.wav
/home/riviera/Music/signal_lost/wav_files/02.wav
/home/riviera/Music/signal_lost/wav_files/01.wav
/home/riviera/Music/signal_lost/wav_files/05.wav
/home/riviera/Music/signal_lost/wav_files/07.wav

Interpreter: Active Server: 1.18% 1.15% 16s 4s 2g 115s 0.66s
```

This continues forever

Technical and Spiritual Manual for Post-apocalyptic Radio Making

The *Technical and Spiritual Manual for Postapocalyptic Radio Making* was produced as part of SI22: Signal Lost, Archive Unzipped. The text was composed of two parts which concerned the how and why of postapocalyptic radio making. Part one was comprised of several diagrams illustrating how to make various pieces of FM radio equipment; transmitters and receivers for example. The second part included interviews with radio makers at Radio Worm. The *Manual* was licensed under [Collective Conditions for Re-use](#). This license was developed by Constant and informs the following discussion of the text.

In short, the interviews had many transcription errors. Concerns were raised about the fact of publishing a text which did not accurately represent the people directly concerned with the material. There was a sense that interviewees were disappointed by the way the transcripts had mangled their spoken words. There was apprehension about the possibility of interviewees having words attributed to them which they never said. Furthermore, these apprehensions were raised by an interviewee after the event in the context of a reflective discussion. Circumspect of these concerns, I have been selective about the material presented in the slideshow and have not included excerpts from the interviews.

My questions are as follows: Does the CC4r provide a framework that assists with navigating through the ethical concerns outlined above? Or do the ethical concerns override? The manual was contentious. My contribution to the text was primarily in typesetting it with ConTeXt. As Maria has pointed out, however, I also rendered the transcripts which were, with minor changes here and there, placed in the *Manual*.

Perhaps I could have been more careful about which pages from the text to upload to the Wiki. I was clear with myself that I would not upload pages from the interviews. However, I did not fully take into account other concerns raised by interviewees which I have tangentially touched on above. Instead, what I uploaded sought to demonstrate capabilities afforded by the ConTeXt typesetting software. I also wanted to share some tricks I had picked up relating to vertical spacing. In view of the likelihood and location of someone encountering documentation of the manual, I have removed images from the slideshow which may be considered contentious.

I believe that scope for making such mistakes is built into the CC4r. On the one hand the license 'favours re-use and generous access conditions'(Constant, 2023). On the other hand, it notes 'that there may be reasons to refrain from release and re-use' (Ibid.). Suppose I were to frame an act of release or re-use in terms of an antonymous or pseudonymous term. Is this a way to sidestep or shift the terms of what is at stake

in the CC4r, namely re-use? For example, if I spoke of redistributing these parts of the *Manual* rather than re-using them. Would that be coherent with the CC4r? What does re-use become in such contexts? Is such a version of events a satisfying outcome of taking the 'implications of (re-)use into account' (Ibid.)?

The screenshot shows a web browser window with the URL `http://www.wiki.wdha.nl/mw-media/design/images/4/43/FM_transmitter_circuit.png`. The page content includes:

How to make a transmitter

Necessary components

- Power supply:** Provides the necessary electrical power to operate the transmitter.
- Oscillator:** Creates alternating current at the frequency on which the transmitter will transmit (the carrier wave).
- Modulator:** Adds useful information to the carrier wave. If you use frequency modulation to do so (FM) you make slight increases or decreases in the frequency of the carrier wave.
- Amplifier:** Amplifies the modulated carrier wave to increase the power of the broadcast.
- Antenna:** Converts the amplified signal to radio waves.

Diagram: FM transmitter circuit.

The diagram shows a detailed electronic circuit for an FM transmitter. It includes a power supply section with a transformer, a variable capacitor (VC1), a coil (L1), and a variable capacitor (VC2). The circuit also features a diode (D1), a resistor (R1), and an antenna connected to a speaker.

Diagram illustrating how to make a transmitter

The screenshot shows a web browser window with the URL `http://www.wiki.wdha.nl/mw-media/design/images/4/43/FM_receiver_circuit.png`. The page content includes:

How to make a receiver

Necessary components

- Antenna:** (eg. a piece of wire) to capture the radio waves.
- RF amplifier:** Amplifies the weak radio frequency (RF) signal from the antenna in order for the tuner to process it.
- Tuner:** A circuit (coil + capacitor) that can extract signals of a particular frequency (resonant frequency). Without it, the antenna captures all radio waves, which are then collectively amplified by the amplifier.
- Detector:** Responsible for separating the audio information from the carrier wave.
- Audio amplifier:** A circuit that amplifies the signal coming from the detector.

Happy radio wave hunting!

The diagram is a block diagram showing the flow of a signal from an antenna through an RF amplifier, a tuner, a detector, and an audio amplifier, finally reaching a speaker.

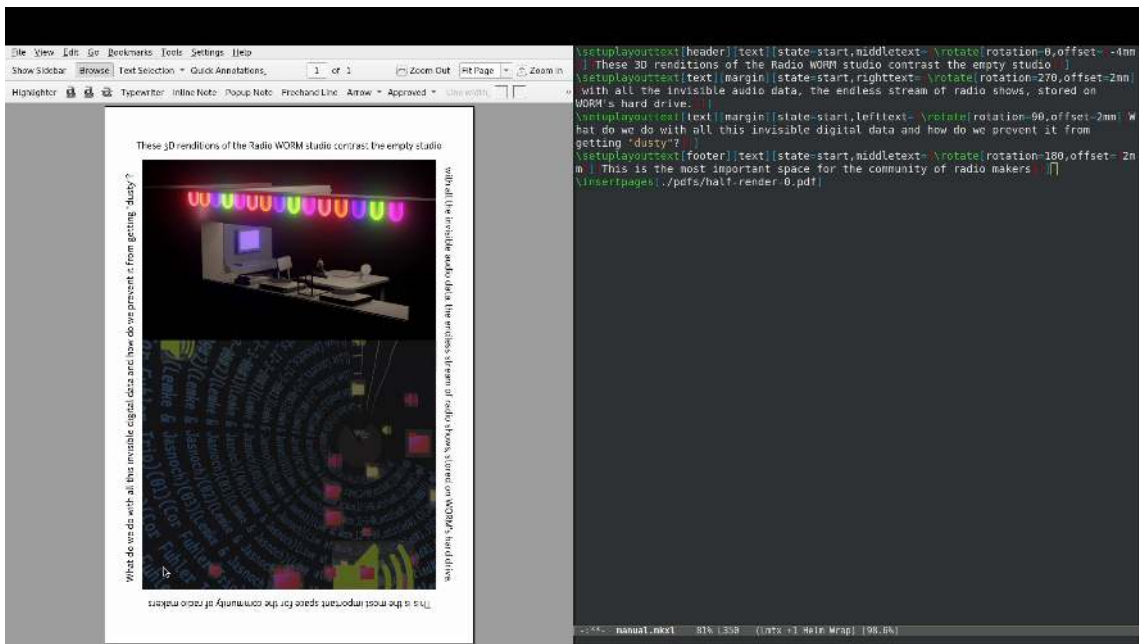
Diagram illustrating how to make a receiver

This screenshot shows a PDF viewer interface. The main content area displays a page with dense Chinese text, likely a technical document or research paper. The sidebar on the right contains a table of contents with entries such as 'Rain Received and Rain Decoded by Wang and Zuzu'. The interface includes standard PDF viewer controls like 'File', 'View', 'Edit', 'Go', 'Bookmarks', 'Tools', 'Settings', and 'Help' at the top.

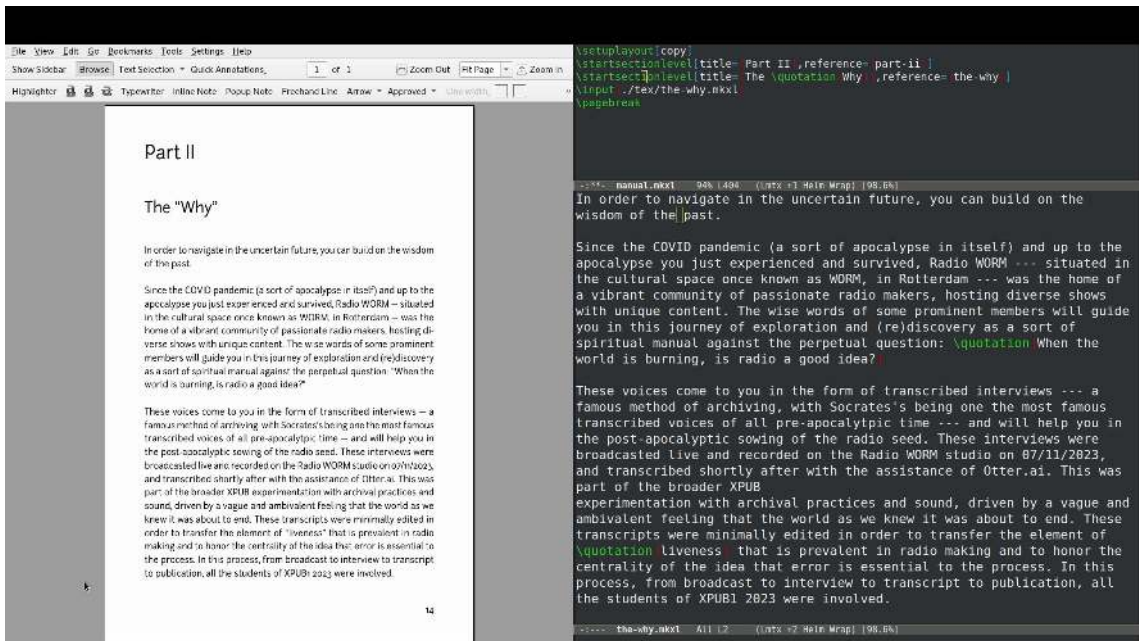
Rain Received and Rain Decoded, by Wang and Zuzu

This screenshot shows a PDF viewer displaying a page with a photograph of a device labeled 'worm belt'. The device is a rectangular control panel with several buttons and a small display. The text 'worm belt' is written in red above the device. The PDF viewer interface is visible at the top and bottom of the page.

Image page, the Worm Bel



Images of *Sound you see me, sound you don't* by Lorenzo and Senka with layout text



Preamble to Part II: The Why

Reading, Writing and Research Methods

Selected Wordhole Entries

Shell Scripts

Throughout SI #22: Radio Worm - Protocols for an Active Archive, several shell scripts have been made. These are small computer programmes which enhance productivity and make tasks less repetitive. On several occasions we have implemented shell scripts to get things done and probably we will use shell scripts again in the near future. In 'Tools to Fight Boredom: FLOSS and GNU/Linux for artists working in the field of generative music and software art' Marloes de Valk likens shell scripts to a sort of 'glue' capable of binding things together. Throughout SI #22 shell scripts have been used to diverse ends. This ranges from making a podcast generator to downloading files from the internet. From compiling pdf documents with pandoc and weasyprint to manipulating texts with grep, awk and sed. These scripts can be found on the internet via Gitea, on the Wiki and in etherpads. Overall, shell scripting has been more of a distributive practice than a collaborative practice. We have shared algorithms with each other without necessarily writing those algorithms together (i.e. in an Etherpad). Depending on how the shell script is written, a specific shell may or may not be required to execute the code. Primarily we have been using bash, the Bourne Again Shell, and fish, the Friendly Interactive Shell. These shells have subtle differences in syntax; bash often cannot execute fish scripts and vice-versa. Below is an example of a fish script which, when executed, downloads all episodes of the Laurence Rassel Show.

```
#!/usr/bin/fish
for n in 01 02 03 04 05 06 07 08 09 (seq 10 17);
  wget (printf \
    'http://www.publicrec.org/archive/2-01/2-01-014/2-01-014-%s.mp3' \
    $n);
end
```

Control Societies

Gilles Deleuze discusses the concept of a society of control in his 1992 text '[Postscript on the Societies of Control](#)'. Societies of control are governed by codes, which give access or bar individuals from flows of information, at "informational intersections" and, as a result, the subject flows "in a continuous network." Deleuze contrasts control societies with disciplinary societies, as theorised by Foucault, which are governed by the

control of discreetly defined spaces through the execution of social protocols. However, control societies, Deleuze argues, are distinct from [disciplinary societies \(as theorised by Michel Foucault\)](#) but in connection to them, as their immediate development after WWII. Control societies are reforming the institutions of disciplinary societies, including prisons, hospitals and schools, but one can go further and think of how eg. genetic engineering or pharmaceutical research is conducted. Reflecting on Foucault's analysis of disciplinary societies, Deleuze reminds us that subjects in disciplinary societies are constituted on the basis of two axes: 'number' and 'signature', and are tangible and clearly witnesses in our society. These organising principles are of less importance within control societies and instead codes are implemented. Such codes govern who can access what. This is especially evident not only in the end-barrier but in the systems of tracking and actual control that defines whether one qualifies to cross a barrier. Some methods of control, that could have been borrowed from previous societies of sovereignty, are becoming interchangeable. At the time the essay was written, Deleuze could already perceive a crisis of the institutions as new mechanism of controls were being installed and implemented in different types of systems: school, prison, hospital and corporate. The question that arose was how unions would survive the advent of new, harmful forms of societies of control.

Application (as used by us)

Collective Annotation of Postscript on the Societies of Control: <https://pad.xpub.nl/p/PostscriptControlSocieties>

Subgroup Annotation of Postscript on the Societies of Control: https://pad.xpub.nl/p/Deleuze_Control_Group

Trimester Two

Prototyping

January 16th 2024

Manetta stated that whilst taking part in [algolit](#) she encountered a [video](#) illustrating the workings of a 'quick sort' algorithm through the medium of an Hungarian folk dance. Inspired by this dance, Manetta guided us through a performative exercise in which we had to make pull and push requests to a git repository. Two actors were the git repository, the remaining actors were git users who were able to make push and pull requests to the git repository. I have chosen deliberately the term "user". As Shusha Niederberger's points out, '[d]espite their central position in data, users are considered only at the margins of the current critical discourses about the implications of data-driven environments' (Niederberger 2023). Each git user was equipped with several, blank, paper cards approximately the size of an ISO/IEC 7810 ID-1 card. These were for writing git commit messages corresponding to individual changes made to "files". These files were pieces of A4 paper with grid patterns printed on them. There were several aims to the game. The people playing the git repository had to keep track of the order in which the commits were received whilst responding to pull and push requests from the users. Users were able to write information in the files and commit their changes to the repository by sending push requests.

We engaged in two rounds with subtle changes to the rules in each round. The exercise presented an opportunity to think through software and imagine how software could be otherwise. During the exercise I had the chance to think like a git repository in the first round and like the user of a git repository in the second round. The movement from the former to the latter was interesting in itself. It was interesting to see how the git repositories implemented the logic of push and pull requests differently across the rounds.

Round One: Being a git repository with MariaX

In round one there were three files which the users were writing. These were called `index.html`, `banana.jpeg` and `veryimportant.txt`. I suggested to Maria that we keep track of the order in which we received the commits by adding timestamps to the commit messages. We did so, and when pull requests were made, one of us gave the cards we had available to the user making the pull request. With this system, the information about recent commits was always dispersed around the room. It was not possible to get a complete overview of all the commits. It was implicit that taking part in the performance effectively benefited from knowledge of how git works. However,

it was simultaneously a learning opportunity and, as I mentioned earlier, the playful dimension encouraged imaginative thinking about how software such as git might function differently. Ultimately, the results spoke for themselves. There were some consistencies in the files, but there were also many discrepancies. Memorably, the question arose as to the issue of writing over one another. For example, during the exercise, one user drew a sheep in box 1,1 whilst another drew a cross in the same box. The effect of this was striking out the drawing of the sheep. Perhaps one reason for the differences between the files was partly attributable to the wide scope of possible drawings which could be placed in each box. These ranged from sheep to flowers, to crosses to block colours and there was some confusion amongst users about how to interpret the commit messages. However, I consider that much of the discrepancies were attributable to the way in which Maria and I served the git repository.

Round Two: Being the user of a git repository served by Senka and Victor

Senka and Victor handled pull and push requests in the second round. Simultaneously the rules were tightened up a bit such that only block colours could be placed in a given square on the file grid. Also, the users worked collectively on only one file.

Crop Tool

Sketch

These notes are concerned with a bash script I wrote recently, applied to ConTeXt and then translated to Python. The output of the script is a list of dimensions corresponding to a particular ratio. I would like it to extend it to a pdf-output. Interaction with the command line interface could be like this: `crop --ratio=2:3 --paper-size=A4 portrait.pdf`

First two empty lists are declared

```
x = []
y = []
```

Subsequently, the variables of `paper_height` and `paper_width` are given.

```
paper_height = 297
paper_width = 210
```

The code could be improved by allowing the user to provide an argument. For example `--papersize=A4`, whereupon the dimensions of the A-series paper size could be retrieved from somewhere for use in the calculations. I'll think about implementing this feature at a later point. Next up are two variables, `step_x` and `step_y`.

```
step_x = 3
step_y = 2
```

Let's fill the list with some values

```
for n in range(step_x, paper_height, step_x):
    x += [n]
```

```
for n in range(step_y, paper_width, step_y):
    y += [n]
```

The lists, though of different lengths, are now full of numbers. These amount to two sequences of numbers within a particular range. When aligned, the relationship between the numbers in the lists are always in proportion to the ratio given by the user. Here is how the lists can be aligned:

```
i = 0
for n in range(0, len(x)):
    print(y[i], x[i])
    i += 1
```

Prototype

```
import argparse
import math
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

To set up a command line application the argparse module is used. Argparse allows for flags and positional arguments to be given to the script when executed at the command line. The values passed in by the user are stored in variables. Argparse also implements a help flag which offers information about available flags.

```
parser = argparse.ArgumentParser(description='Crop typesetting areas.')
```

Arguments are added. outfile is a positional argument whereas the remaining arguments are flags. The flags will be looked at in more detail later on.

```
parser.add_argument('outfile',
                    metavar='OUTFILE',
                    nargs=1,
                    help="Write to a file")
parser.add_argument('--papersize',
                    metavar='PAPERSIZE',
                    nargs=1,
                    default='A4',
                    help="Provide a standard papersize")
parser.add_argument('--ratio',
                    metavar='RATIO',
                    nargs=1,
                    default='2:3',
                    help="Crop the paper to this proportion")
parser.add_argument('--orientation',
                    metavar='ORIENTATION',
                    nargs=1,
                    default='portrait',
                    help="Switch between portrait and landscape.")
parser.add_argument('--scale',
                    metavar='SCALE',
                    nargs=1,
                    default=[90.0],
                    help="Scale the size of the cropped page.")
```



```
_StoreAction(option_strings=['--scale'], dest='scale', nargs=1, const=None,
default=[90.0], type=None, choices=None, required=False, help='Scale
the size of the cropped page.', metavar='SCALE')
```

For the sake of example, let's pass the following arguments to the script.

```
args = parser.parse_args(args=['--scale', '90',
                              '--ratio', '5:3',
                              '--papersize', 'A3',
                              '--orientation', 'landscape',
                              'main.tex'])
```

Wishlist

It would be interesting to add a `--page-on-page` flag which introduces variation in the output. When active, this flag would print the cropped page on the given page size at the given scale and ratio. This is the default behaviour at the moment. Implementing this flag would result in an alternative default behavior where the output is a page already cropped to size.

Papersize Dictionary

I drew up a dictionary of A-series papersizes based on information at papersizes.io. This way paper dimensions can be referenced by name.

```
portrait_paper_sizes = {
    # size width height (mm)
    "A0" : [841, 1189],
    "A1" : [594, 841],
    "A2" : [420, 594],
    "A3" : [297, 420],
    "A4" : [210, 297],
    "A5" : [148, 210],
    "A6" : [105, 148],
    "A7" : [74, 105],
    "A8" : [52, 74],
    "A9" : [37, 52],
    "A10": [26, 37],
    "A11": [18, 26],
    "A12": [13, 18],
    "A13": [9, 13],
    "2A0": [1189, 1682],
    "4A0": [1682, 2378],
```

```

    "A0+": [914, 1292],
    "A1+": [609, 914],
    "A3+": [329, 483]
}

```

Ratio

```

ratio = args.ratio[0].split(":")
ratio_x = int(ratio[0])
ratio_y = int(ratio[1])
print(f"Crop ratio: {ratio_x}:{ratio_y}")

```

Crop ratio: 5:3

The ratio is provided to the script with the `--ratio` flag. By default the ratio is 2:3. Some calculations need to be done so let's initialise some variables.

```

possible_widths_list = []
possible_heights_list = []
w = ratio_x
h = ratio_y

```

In order to ascertain the size of the cropped page, I'm calculating a list of measurements. These measurements indicate towards the 2D area of the cropped page. The values are later used in the context of the scale feature. The following calculation checks the ratio against the dimensions of the page. A for loop is used to provide a limit to the length of the list which contains the measurements described above.

```

if (math.floor(paper_width / ratio_y)) > (math.floor(paper_height / ratio_x)):
    # If the paper is landscape
    for dimension in range(math.floor(paper_width / ratio_x)):
        possible_widths_list += [w]
        possible_heights_list += [h]
        w += ratio_x
        h += ratio_y
else:
    for dimension in range(math.floor(paper_height / ratio_y)):
        possible_widths_list += [w]
        possible_heights_list += [h]
        w += ratio_x
        h += ratio_y

```

Pandas, Numpy and SciKit Learn

At the beginning of the script, I imported (parts of) these modules into the python script. This was to enable python to make use of different mathematical functions. In particular, I'm going to use a pandas DataFrame, SciKit Learn's MinMaxScaler and Numpy's interp function. The purpose is to provide the user with the ability to scale the size of the cropped page in the output. In short, the values in `possible_widths_list` and `possible_heights_list` are adjusted to a percentage scale. That there can be more or less than 100 values in the `possible_widths_list` and `possible_heights_list` means that the value of the length of the list needs to represent 100%. To begin with, let's create a DataFrame and a scaler. The code which appears below was adapted from [this website](#).

```
df = pd.DataFrame({"widths": possible_widths_list, "heights": possible_heights_list})
scaler = MinMaxScaler()
```

Visualising the Dataframe

The dataframe resembles a table of widths and heights spanning a range of values.

```
print(df)
```

	widths	heights
0	5	3
1	10	6
2	15	9
3	20	12
4	25	15
..
79	400	240
80	405	243
81	410	246
82	415	249
83	420	252

```
[84 rows x 2 columns]
```

Adding scaled values to the dataframe

This code assigns a percentage-based value to each possible width and height.

```
tmp_widths = df.widths - df.widths.min()
```

```

tmp_heights = df.heights - df.heights.min()
scaled_widths = tmp_widths / tmp_widths.max() * 100
scaled_heights = tmp_heights / tmp_heights.max() * 100

df["scaled_widths"] = scaled_widths
df["scaled_heights"] = scaled_heights

print(df)

```

	widths	heights	scaled_widths	scaled_heights
0	5	3	0.000000	0.000000
1	10	6	1.204819	1.204819
2	15	9	2.409639	2.409639
3	20	12	3.614458	3.614458
4	25	15	4.819277	4.819277
..
79	400	240	95.180723	95.180723
80	405	243	96.385542	96.385542
81	410	246	97.590361	97.590361
82	415	249	98.795181	98.795181
83	420	252	100.000000	100.000000

```
[84 rows x 4 columns]
```

Interpolating the values

Next, the values are interpolated. To my understanding, this is like cross-referencing the values in one list against the values in another. It's like creating an array with floating-point indexes. The values in between are interpolated and rounded to the nearest mm. The resulting values are consistently approximate.

```

scaled_paper_height = math.floor(np.interp(95.2, scaled_heights, possible_heights_list))
scaled_paper_width = math.floor(np.interp(95.2, scaled_widths, possible_widths_list))

```

```

print(scaled_paper_width)
print(scaled_paper_height)

```

```

400
240

```

Notice that the printed values correspond to the scaled values in the DataFrame. It's best if the user can determine the scale to crop the paper to. So, the first argument to `np.interp` is replaced with `args.scale[0]`.

```
scaled_paper_height = math.floor(np.interp(args.scale[0], scaled_heights, possible_heights_list))
scaled_paper_width = math.floor(np.interp(args.scale[0], scaled_widths, possible_widths_list))
```

Writing to a file

The output of the script is code which can be understood by the ConTeXt typesetting software. F-strings containing the values calculated by or provided to the script are used. The variables feature at key points in the ConTeXt code. The file is created. Then, a blank layout is defined and setup.

```
f = open(args.outfile[0], "w")
f.write("""\definelayout[blank] [
topspace=0mm,
backspace=0mm,
bottomspace=0mm,
width=fit,
height=fit,
header=0mm,
footer=0mm,
leftmargin=0mm,
rightmargin=0mm,
leftmargindistance=0mm,
rightmargindistance=0mm]
\setuplayout[blank] """)
```

Then, having turned off page numbering, the f-string containing the values of `scaled_paper_width` and `scaled_paper_height` are passed to `\definepapersize`.

```
f.write(f"""\definepapersize[scaled] [width={scaled_paper_width}mm, height={scaled_paper_height}mm]
\setuppapersize[scaled] """)
```

The code takes landscape mode into account using an if statement

```
if "portrait" in args.orientation:
    f.write(f"[{args.papersize[0]}]")
else:
    f.write(f"[{args.papersize[0]}, landscape]")
```

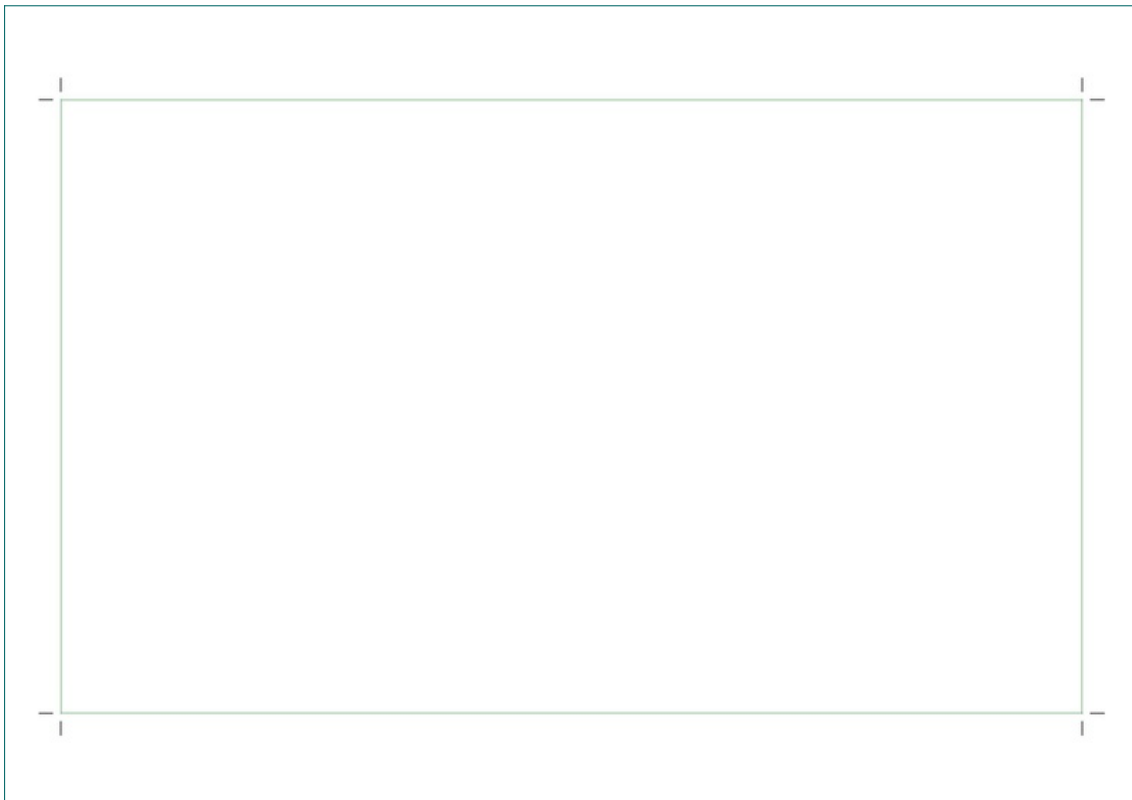
Finally, the layout is setup, the frame is switched on and the text environment is invoked. Inside the text environment, a frame which fills the typesetting area is included to ensure there is content in the document.

```
f.write("""\setuplayout[location="" "{middle,middle}" "",mark-
ing=empty]
    \showframe
    \starttext
    \startframedtext[width=\textwidth,height=\textheight]

    \stopframedtext
    \stoptext
    """)
f.close()
```

PDF Output

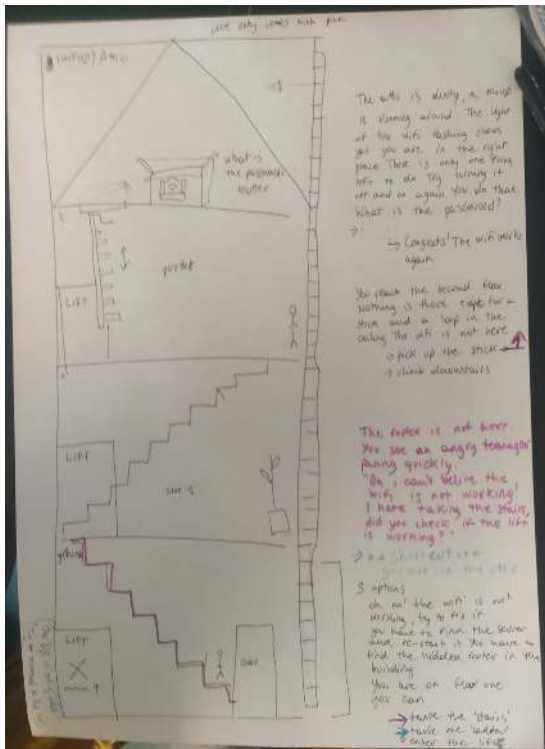
ConTeXt can be run on the output file, in this case `main.tex`, to produce a pdf.



The consequence of running ConTeXt on the output of the script.

Fix The Wi-fi

In "Games for Actors and Non-Actors" Augusto Boal opens his discussion of the 'dialectical structure of the actor's role' by stating that:



A map of doors, rooms and objects in Fix the Wifi

The fundamental concept for the actor is not the 'being' of the character, but the 'will'. One should not ask 'who is this?', but rather 'what does he want?' The first question can lead to the formation of static pools of emotion, while the second is essentially dynamic, dialectical, conflictual, and consequently theatrical.

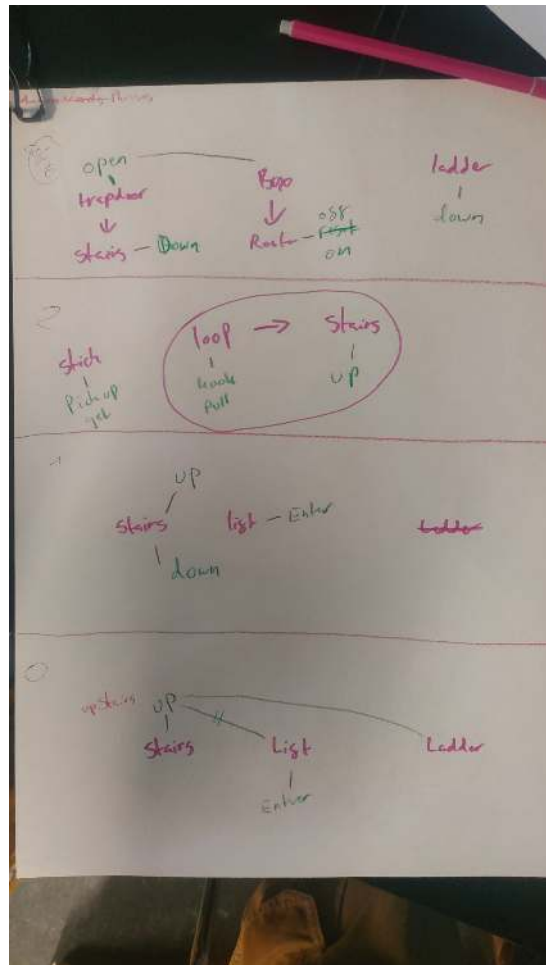
According to Boal what determines the will is bound up with the core of the performance; what it endeavours to communicate to audiences. Boal offers several examples from Shakespearean tragedies to argue that '[o]nce chosen, the central idea of the work must be respected at all costs' (p.41). From this perspective, it is obligatory that every will contributes to fleshing out the thing the play is about.

In text adventure games you are the actor. What does it mean to transpose Boal's thinking to text adventure games? How might this be done?

Fix the Wi-fi is a text adventure game which was written in collaboration with Anita and Mania. In Fix the Wi-fi the aim of the game is to turn the wi-fi on. The code for the game was based on the XPUB with rooms script which is (not publicly available) on chopchop. To make the game, maps and diagrams were drawn up to illustrate the world in which the game would take place. This was a constructive approach and working in a non-linear way enabled us to think through aspects of the game in tandem with one another. As we were unable to install the game on chopchop before the end of the prototyping session we met up the following day to finish it. This text discusses the code which features in the game, the process of making the game and offers a provocation for SI23. I hope that the changes I have made to the script may be useful in the context of SI23 if we decide to make a python text adventure.

I speculate, in relation to Fix the Wi-fi and with Boal in mind, that the will of the actor is limited by the role played. Alternative modes of engagement with the software can therefore be ruled out insofar that these do not reflect the will of the actor. This is in part because it is FLOSS and modes of engagement are broad in scope. For example, it is not the will of the actor to extend the game through contributions to the code which makes it work. Nor is the actors' will to read the code which makes the game function to understand how to proceed with the game. In the context of Fix the Wi-Fi, the will of the actor is to fix the wifi. There are several reasons to support the conclusion that the actor wants to fix the wifi. During our conversations, Me, Mania and Anita spoke about the lift only working when the wi-fi is enabled.

I have not implemented a working lift as doing so is relatively challenging. Nevertheless, the actor may strive to fix the wi-fi because the lift is broken and needs to be fixed.



A simplified map of doors, rooms and objects

Furthermore, residents are not angry when the wifi is working and they are able to connect to the internet. In other words, the aim of fixing the wi-fi can be understood as the crux of the game and this necessarily shapes the will of the actor. This will can be instrumentalised through code in game-world mechanisms which consolidate and reify the will of the actor. However, it is simultaneously the case that the creation of in-game mechanisms is a result of the actors' will. That is, it is a meshwork which supports the actor to realise the object of their will, because that is what is wanted *by the actor*. There is tension here. On the one hand, a will is imposed on the actor by code. On the other hand the will of the actor determines the code. Preserving agency is of importance in this dynamic.

Sample Output

Oh no! The wifi is not working, try to fix it
You have to find the hidden router and re-start it in order to play
more text-adventure games.
You are on the ground floor and look around, but the router is no where
to be seen. You are in
a tall building, with three floors and an attic. You see some steep
stairs leading to the
floor above, a closed lift and an emergency ladder out of the window.

```
Do you want to:  
--> take the stairs and 'go upstairs'  
--> climb the ladder 'climb ladder'  
--> enter the lift 'enter lift'
```

go upstairs

You opened the door to Floor One...

look

```
'-- .  
/ \ | .---. ---. .--- /|  
|_  | . ' \ . ' \ / \ |  
|   | |   | |   | |   ' |  
|   /\_  `...' `...' /   _|  
/
```

The router is not here. You see an angry teenager
pacing quickly. Oh I can't believe the wifi is not working! I hate
taking
the stairs, did you check if the lift is working?

go upstairs

You opened the door to Floor Two...

go upstairs

You opened the door to the Attic...

```
disable router
```

You switch the router off

```
enable router
```

You switch the router on

It worked! The wifi works again, congrats!!

```
climb ladder
```

You opened the door to the Ground Floor...

```
go upstairs
```

You opened the door to Floor One...

```
look
```

```
,-- .  
/ \ | .---. ---. .--- /|  
|_ | . ' \ . ' \ / \ |  
| | | | | | | | ' |  
| ^\__ \_.' \_.' / _|  
/
```

The router is not here. Nobody is here. Did you check if the lift is working?

Side Project: Chess Diary

February 5th 2024

Chess is a hobby of mine. I attend a chess club every week here in Rotterdam. In competitive chess games it is common to record the moves made during the game on a scoresheet. The other week I designed a chess diary using Simon's Improved Layout Engine ([SILE](#)). The layout engine itself is TeX-inspired. It borrows some algorithms from TeX and uses a similar syntax. I'm familiar with ConTeXt which is a variant of TeX. I have a love-hate relationship with ConTeXt. On the one hand, it offers extreme microtypographical precision. On the other hand, the software is idiosyncratic and is not conducive to collaborative practice. Due to this ambivalence which I have about ConTeXt, I wanted to produce a publication using an alternative layout engine. In this case I chose SILE.



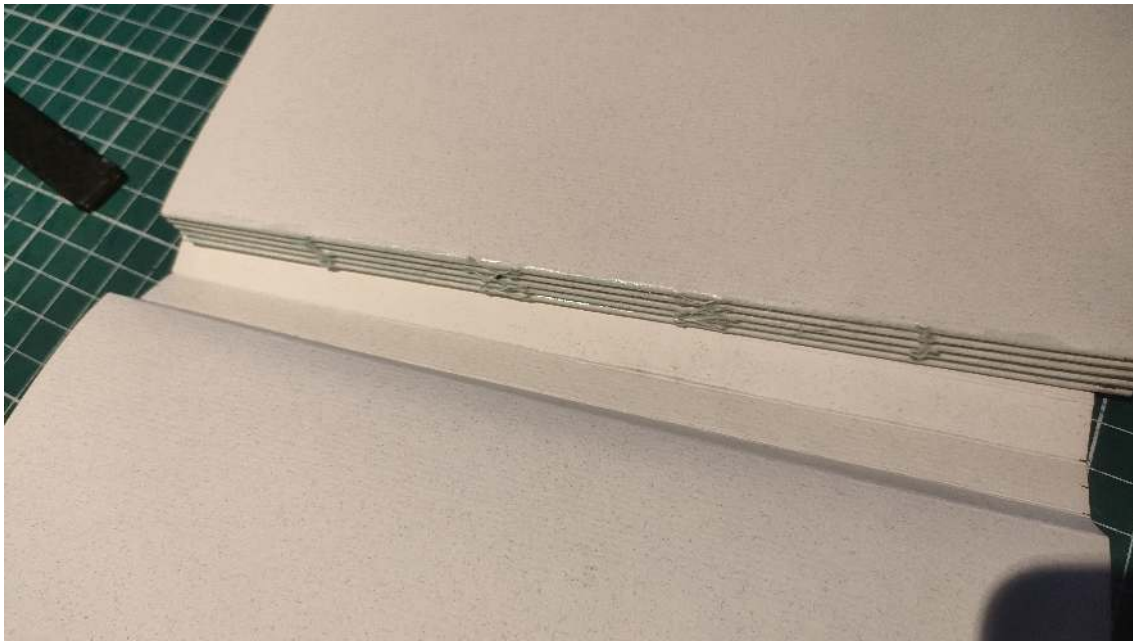
Initial Impressions of SILE

A blank cover for the chess diary

SILE is a layout engine for typesetting PDF documents. It reads code written in a format specific to SILE and generates PDF output. SILE can read XML files, though the manual advises against hand-coding such documents because the TeX-like syntax is simpler to write by hand. SILE is comprised of a selection of packages which are written in Lua. One adds directives in ones SILE code to load particular packages. This is necessary in order to utilise the macros which are defined in the package. For instance it is not possible to call the `\lorem` macro unless one loads the lorem package beforehand. Thus the following code will produce 20 words of lorem ipsum when processed by SILE.

```
\begin{sile}  
\use [module=packages.lorem]  
\lorem [words=20]  
\end{sile}
```

In contrast to ConTeXt, SILE is refreshing because it has only one manual which covers much of what the software can do. However, unlike ConTeXt, which offers many cohesive features, I found that SILE had limited support for two-column layouts. Yes, the SILE manual offers an example of how to create a page with a two-column layout. However, I did not want the columns to fill the entire text. Rather, I wanted the two-column layout to begin after a five lines of single-column (full width) text. I could not figure out a simple solution to this problem. Therefore, it was necessary to emulate a two column layout using the `\hfill` and `\hrulefill` commands. To align the text in the columns it was a matter of figuring out the ratio of `\hfill` to `\hrulefill`. These commands insert stretchy horizontal whitespace and stretchy horizontal rules respectively. I designed the layout for two lines and then utilised a for-loop to generate the same two lines 16 times. I repeated this for-loop on the next page with 20 iterations. Once I was satisfied with the layout of the pages, I utilised a third for-loop to generate fifty game-sheets, complete with page numbering. Estimating the vertical whitespace between each line was a key aspect of ensuring the fifty pages would render as intended.



The cover of the Chess Diary is not yet attached to the spine

An Afternoon with Roderick

Roderick also attends the Chess club. He trained as a graphic designer and has various pieces of book-binding equipment in his possession including a guillotine, a printer and a foil press. He suggested it might be nice to produce each 100-page chess diary in five signatures which could then be bound together. Each signature would be comprised of five A4 sheets folded in half to produce A5 pages. We also discussed the dimensions of the publication, given that it would probably be used by chess players whilst sitting at a table with a chess board between them. Roderick suggested that it might make sense to resize the pages so that they were of a non-standard dimension. However, I thought this would be impractical. In any case, we attempted to print the prototype I had designed with SILE using his laptop and printer. This resulted in a print error, however. Whereas we made little progress in producing a physical outcome together on Sunday, we intend to meet up again and make more headway soon. We are optimistic that the chess diary could be a meaningful piece of merchandise for club members.

February 17th 2024



Sewn together signatures



The uncropped content of the chess diary

Roderick and I met up on the afternoon of February 17th. Roderick had been able to fix the printing issue we were having last time. Since our previous meeting some minor edits were made to the pdf. Roderick had printed five copies comprised of five signatures each. Changes to the PDF included the addition of a blank page at the beginning, a colophon at the end and some adjustments to the page numbering. The pages were printed with crop and fold marks because the printing space had been resized to smaller dimensions. The edits to the pdf were made using Adobe software by someone Roderick knows.

We proceeded to fold the signatures. This was tricky as we were going against the natural grain of the paper. Roderick demonstrated a technique for folding. One creates an L-shape with one hand. The thumb is placed at the outer, bottom corner of the paper whilst the index finger is placed somewhere along the outside edge. Then one

uses a runs a folding tool up and down the inside edge at a 45 degree angle to create a sharp fold. Having folded each of the signatures, it's necessary to punch holes along the inside edge. Again, Roderick demonstrated to me a technique for this. First, a punching template is designed on a separate piece of paper using a compass. Sewing stations are indicated to by short lines placed at regularly patterned intervals along the inside edge on the outside of the paper. Next, the folded edge of a signature is placed over the edge of a table. One takes the punching template in one hand and a punching needle in the other. The punching template is placed face-up halfway through the signature. The hand which holds the punching needle presses down on the paper to prevent it from moving. Aiming towards the body, holes are carefully punched through the paper at a 45 degree angle. These holes are made at the intervals indicated to by the punching template. Having punched holes in the inside edges of the signatures it's time to sew the signatures tightly together. Roderick showed me some tricks for threading the sewing needle to prevent the string from becoming disconnected from the needle. We measured out a length of string six times the length of the spine, because we had five signatures. Then, starting from the outside the signatures were sewn together. Roderick emphasised the importance of keeping the thread tight but not so tight that the paper rips. He also stated it was important to stitch the signatures together by making loops which go from below to above.

We sewed the signatures together and then set about gluing them together. We ran PVA glue up and down the spine and placed weight on the books to press them shut. Roderick spoke a bit about fly-sheets. He also spoke about the ways in which different bindings can impact how flat the pages lie when the book is open. I decided to make a prototype without fly sheets and Roderick made his with these included. To create covers for the book, we moved on to discuss the cover design. We chose some linen and cut this to size based on the sum of the width of the pages and the width of the spine. Despite that we have plans to use a foil press to design a pattern on the spine, we decided to add creases to the spines. Initially, we glued the linen to pieces of paper and used a press to flatten them whilst the glue dried. We took the paper-linen out of the press and cut it down to the correct dimensions for the book. To illustrate where the spine would be and where we would glue it the collection of signatures, we creased the cover using a metal creasing tool.

Next time we meet we are going to create covers using the foil press and hopefully attach the covers to the books.

Reading, Writing and Research Methods

Selected Wordquilt Entries

Infrastructure

What is Infrastructure? According to Martin Brauch (2017) '*Infrastructure* is the underlying system of structures, facilities and services that are essential to the functioning of an economy'. Brauch's definition is commonsense following Susan Leigh Star's statement that '[p]eople commonly envision infrastructure as a system of substrates' (Star, 1999, p. 380). Such a perspective resonates with the definition of 'critical infrastructure' detailed in the Sendai Framework Terminology on Disaster Risk Reduction (UNDRR, 2017). Nevertheless, Star casts doubt on the usefulness of an everyday understanding of infrastructure in the context of the production of 'large-scale technical systems' (Star, 1999). Instead, she foregrounds the relational quality of infrastructure; that it appears differently to different people based on their relationship to it. In concordance with Star, the adequacy of the commonsense understanding of infrastructure is reiterated by Lauren Berlant. Berlant writes, '[i]nfrastructure is not identical to a system or structure It is the living mediation of what organizes life: the lifeworld of structure' (2016).

Membership

In Star's *Misplaced Concretism*, the discussion of 'membership' is sandwiched between writing on objects/communities of practice and borderlands/boundary objects. Membership is integral to Star's discussion of marginality - understood in the technical, sociological sense. But what is membership? Star reflects on Jean Lave and Etienne Wenger's discussion of membership which they take to be the polar opposite of 'illegitimate, peripheral participation'. However, for Star, membership would not exist without the processual naturalisation of objects. This differs to Lave and Wenger's concept and contributes to Star's argument that membership has both individual and collective dimensions. '[I]ndividually', membership can be understood 'as the experience of encountering objects and increasingly being in a naturalised relationship with them'. 'Collectively membership can be described as the processes of managing the tension between naturalisation..., on the one hand, and the degree of openness to immigration on the other'.

Bibliography

- Berlant, L. (2016) 'The commons: Infrastructures for troubling times*', *Environment and Planning D: Society and Space*, vol. 34, no. 3, pp. 393–419 [Online]. DOI: [10.1177/0263775816645989](https://doi.org/10.1177/0263775816645989) (Accessed 14 March 2024).
- Boal, A. (2002) *Games for actors and non-actors*, Second Edition., New York, Routledge.
- Brauch, M. D. (2017) *Sustainable Infrastructure: Definition and Co-Benefits*, International Institute for Sustainable Development (IISD) [Online]. Available at <https://www.jstor.org/stable/resrep14773.4> (Accessed 14 March 2024).
- Constant (2023) CC4r * COLLECTIVE CONDITIONS FOR RE-USE [Online]. Available at <https://constantvzw.org/wefts/cc4r.en.html> (Accessed 28 September 2023).
- Deleuze, G. (1992) 'Postscript on the Societies of Control', October, The MIT Press, vol. 59, pp. 3–7.
- Mauro-Flude, N. (2008) 'Linux for Theatre Makers: Embodiment & Nix Modus Operandi', in Mansoux, A. and Valk, M. de (eds), *Floss + art*, Poitiers, France, GOTO 10, in association with OpenMute, pp. 206–223.
- Niederberger, S. (2023) 'Calling the User: Interpellation and Narration of User Subjectivity in Mastodon and Trans*Feminist Servers', *A Peer Reviewed Journal About*, vol. 12, no. 1.
- Rassel, L. and Thaemlitz, T. (2007) Laurence Rassel Show, Public Record Archive [Online]. Available at <http://www.publicrec.org/archive/2-01/2-01-014/2-01-014.html> (Accessed 27 March 2024).
- Star, S. L. (1999) 'The Ethnography of Infrastructure', *American Behavioral Scientist*, vol. 43, no. 3, pp. 377–391 [Online]. DOI: [10.1177/00027649921955326](https://doi.org/10.1177/00027649921955326) (Accessed 13 March 2024).
- UNDRR (2017) 'Critical infrastructure', <http://www.undrr.org/terminology/critical-infrastructure> [Online]. (Accessed 14 March 2024).
- Van Dessel, M. (2013) 'Home is a Server', in Laforet, A., de Valk, M., Aktypi, M., Mertens, A., Snelting, F., Lakova, M., and Höfmüller, R. (eds), *"Are You Being Served?" (Notebooks)*, Constant, pp. 164–166 [Online]. (Accessed 30 April 2023).
- Yuill, S. (2008) 'All Problems of Notation Will be Solved by the Masses: Free Open Form Performance, Free/Libre Open Source Software, and Distributive Practice', in Mansoux, A. and Valk, M. de (eds), *Floss + art*, Poitiers, France, GOTO 10, in association with OpenMute, pp. 64–91.

Appendix One: Presentation Slides

Integrated Formative Assessment (Trimester 2)

Riviera Taylor

XPUB

2 April 2024

Introduction



- Making performative technological artworks
- Producing prototypes and publications
- A note about archival documentation

Documentation



- Promotes analysis
- Based on wiki contributions
- Transclusion can become messy
- Minimal code

Prototyping

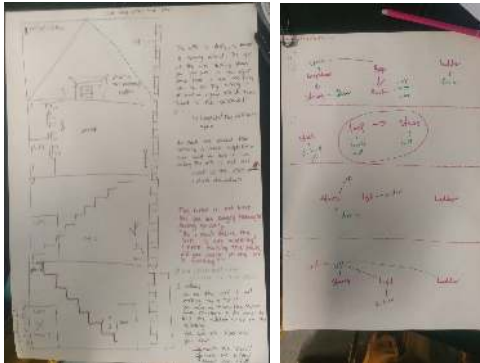
Trimester 2

- Fix the Wifi
- Crop Tool

Trimester 1

- Podcast Generator

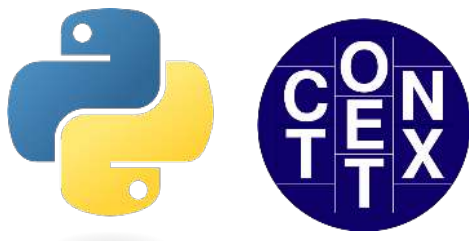
Fix the Wifi



Non-linear, Python text adventure game

- Collaborators: Mania, Anita
- *Games for actors and non-actors* (Boal, 2002)
- What to do next?

Crop Tool



Produces cropped pages

- Command line interface
- Applied interpolation
- Distributive practice

Podcast Generator

Fish, the Friendly Interactive Shell

```
#!/usr/bin/fish
for n in 01 02 03 04 05 06 07 08 09 (seq 10 17);
  wget (printf \
    'http://www.publicrec.org/archive/2-01/2-01-014/2-01-014-%s.mp3' \
    $n);
end
```

- Taking ownership of infrastructure
- Podcasts as an archival format for radio

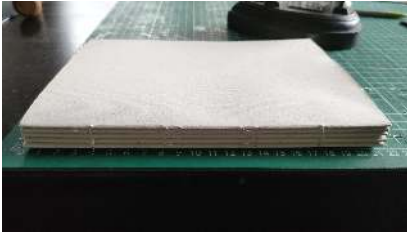
Extratonal Infrastructure 9



Live coding with Fluxus and Tidal Cycles

Typesetting

A Chess Diary



Saddle Stitching

- Collaborator: Roderick Asgar
- Typeset with SILE
- Scorebook for chess games

Platform is the Problem



Photograph by Senka

- Collaborators: Senka, Victor
- Typeset with ConTeXt
- Made for WORM's Zine Camp 2023

Special Issues

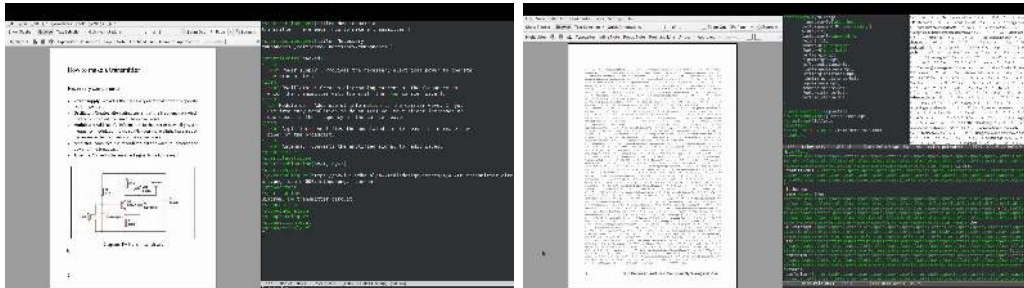
Special Issue 22

- Technical and Spiritual Manual for Post-apocalyptic Radio Making
- Listen Closely

Special Issue 23

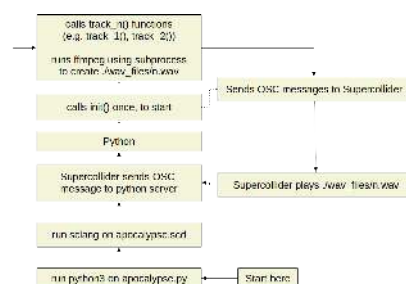
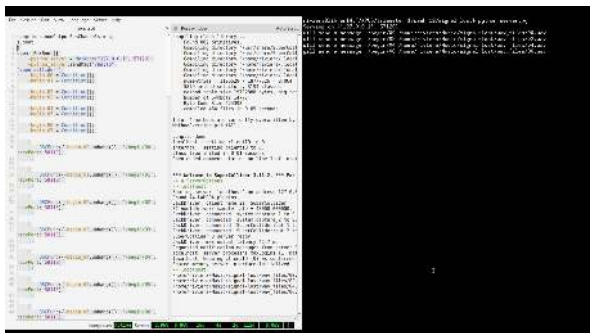
- /var/kitchen

Technical and Spiritual Manual for Post-apocalyptic Radio Making



- Collaborators: Maria, Alessia
- Typeset with ConTeXt
- CC4r affected my documentation and (re-)use

Listen Closely



Audio-Spatial Installation

- Collaborators: Rosa, Thijs, Victor
- SuperCollider and Python via Open Sound Control

/var/kitchen



- Collaborators: Michel, Wang
- Python text game, playable in the browser

Conclusion

- Performance and FLOSS
- Digital typesetting
- Python in collaborative settings

Bibliography

Boal, A. (2002) *Games for actors and non-actors*, Second Edition., New York, Routledge.

