Interview with Alyssa Rosenzweig conducted by the Polytechnic Library of the UAM

1. **You started working on free software projects at a very young age. What attracted you to that world, and what was it like finding your place in a technical community as a teenager?**

I grew up in front of a text editor and a C compiler. Programming was my hobby from my earliest years of primary school until it became my job in university.

My teenage years added social and political dimensions to my beloved craft. In school, I learned about famous activists for social progress: Dr. King, Gandhi, Cesar Chavez, and so on. At home, I identified these figures with heroes in the free software movement. I was already putting my code on GitHub under a permissive license; it was a small leap to understand that code as a vehicle for social progress.

As an adolescent with ample free time and the zeal of the newly converted, I dedicated myself to the advancement of the cause. I believed that, whatever the technical question, the answer was software freedom. That belief drove me to advocate in prose and in person, emulating the techniques I studied in school, while building free software replacements for the proprietary software around me. My early projects shared that common thread:

My classes required proprietary learning management systems to access course materials and check grades? I didn't want to run the proprietary JavaScript anymore, so I reverse-engineered the REST APIs, wrote my own clients, and adapted an open source project (Canvas) for the user interface.

My friends wanted to chat on Discord, a proprietary messaging platform? I contributed to the open source client purple-discord.

My new laptop had an Arm Mali GPU requiring proprietary graphics drivers? Well, we know how to handle that.

An adult in the community once told me that reverse engineering a GPU was too challenging for anyone but an expert. With all the hubris of a 15-year-old, I ignored his advice and did it anyway.

The technical skills of a seasoned adult with the posture and boundless energy of a teenager: a terrifying combination for proprietary software.

No meritocracy exists, but technical communities online get closer than any company or academic institution. In some ways I acted my age, but people respected my code and treated me as a peer. In a real sense, I grew up in the free

software community. Now we have come full circle: I regularly receive email from students asking advice. I do my best to respond, because I can't pay back the people who supported me throughout my journey, but I can pay it forward for the next generation.

2. **For those who might not be familiar with it, how would you explain what Asahi Linux is and why it's such an important project?**

Asahi Linux is a project to run Linux (and other open source operating systems) bare metal on Apple Silicon Macs. Prior to 2020, Macs used Intel chips, so Intel's open source contributions enabled Linux on Macs (with some rough edges). Since 2020, new Macs replace the Intel processor with Apple's own in-house designs, beginning with the Apple M1. Unlike Intel, Apple neither releases documentation for its chips nor contributes open source software for the chips. If you don't want macOS, you're out of luck.

Software freedom is strongest when users can run whatever software they want on whatever hardware they want. The switch to Apple Silicon threatened to "lose" Macs forever in the non-macOS world. Asahi Linux enables Linux as an option for Mac users and Macs as an option for Linux users. Although I may not recommend Macs, it is important that the option exists.

3. **Much of your work has focused on running Linux on Mac devices. Why did you decide to start this process on Macs, such a closed and proprietary system?**

Precisely *because* they are closed and proprietary systems. Through reverse engineering, we turn closed systems into *de facto* open ones. Armed with the knowledge we gain from reverse engineering, we may write our own drivers for the hardware, turning proprietary systems into open source ones respecting software freedom.

There is comfort working on open systems, tending to flowers along a well-trodden trail. But first, someone needs to blaze that trail, hacking through weeds and planting seeds that may take years to bloom. In my life, I find there are many valuable projects I could work on and causes I could advocate for, but the highest return-on-investment comes from performing the tasks that nobody else will.

In 2018, the closed nature of Arm's Mali GPUs and its proprietary drivers threatened the frontier of software freedom. Nobody else was coming to save that, so despite my age, I drew some triangles.

In 2020, the release of the Apple M1 as a general-purpose computing platform with a closed in-house GPU again threatened the open ecosystem our community has

built. With my Mali experience, I was equipped for the challenge. So I got to work – because who else would?

4. **Do you think free technologies could someday become the main solution for our devices? How do you think that could be achieved?**

My views on software freedom have evolved over time. The world has changed and so have I. When the free software movement started in the 80s, proprietary licenses were the problem and writing free replacements the solution. Modern concerns are more complex: surveillance capitalism, centralized social media, and so on. In a world where proprietary software from the 80s respects its users more than some open source programs today, writing code is not enough. The irony of "open source" winning and "free software" losing is that open source software runs the entire world today, yet digital rights continue to backslide. Nevertheless, the decentralized user-centric nature of free software facilitates grassroots resistance to any digital threat. It is a tool in the modern activist's toolbox, alongside traditional techniques like labour organizing and letter-writing campaigns. On its own, free software will not save us, but we cannot succeed without it.

5. **In one of your posts, you wrote: "In 3D graphics, once you can draw a triangle, you can do anything." Could you tell us a bit more about that? Could you also explain, in simple terms, how graphics work?**

The goal of computer graphics is to transform a scene into image, consisting of millions of pixels with particular colours. How?

One idea is to model the physical world, applying the laws of physics to simulate rays of light. We describe the scene mathematically, modelling it as primitive shapes like spheres and boxes, plus light sources like the sun. We then trace individual rays of light from each light as they bounce off reflective surfaces like water and eventually hit the viewers' eye. As the ray bounces, we calculate its colour based on the materials of each object it hits, and whatever colour it has in the viewer's eye is the colour of the corresponding pixel in the image.

While this is how light behaves in the real world, it's inefficient in software since most of those light rays never reach the viewer. Tracing rays from the viewer backwards to the light source is faster. This algorithm is known as ray tracing, and its spectacular results have powered 3D animated films for years. Unfortunately, until recently ray tracing was still too slow for real time.

We need a new approach: rasterization.

Rather than calculate where rays of light intersect objects in a scene, we instead calculate where each object ends up in the scene and draw that directly. To pick which colours to use, we will take inspiration from the physical laws and attempt to approximate the behaviour of light. To simulate reflections, we might draw each object a second time and its reflected location. In rasterization, we're not physicists, we're painters. The goal is to simply draw something that looks good, and draw it fast.

Rasterization's efficiency makes it the traditional technique of choice for real-time rendering, aided by dedicated rasterization silicon: the GPU. The goal of GPU drivers is producing the necessary incantations to drive that hardware.

Hardware needs to be simple, as it is limited by the number of transistors on the chip. Rather than rasterize complex shapes, we break objects into simple shapes implemented by the hardware. The simplest approach is breaking everything into triangles, because anything can be described with triangles. A rectangle is two right triangles split along the diagonal. A cube is twelve triangles: two for each of the six faces. Complex shapes like trees and people are approximated by thousands of triangles. Curves theoretically require infinitely many triangles, but we approximate with just enough that the viewer won't notice.

Putting it together, a 3D video game will produce many thousands of triangles each frame, for the hardware to rasterize and display. Our job writing a driver is to facilitate communication between the video game and the hardware. And once we can draw a triangle, we can draw anything.

6. **What does your reverse engineering process look like?**

Reverse engineering is science, the pursuit of knowledge via the scientific method. Writing drivers is engineering, applying the knowledge from science to solve problems.

How do I reverse engineer? The same way any scientist works: forming hypotheses, collecting data, and revising those hypotheses.

Usually, we have a proprietary driver for the hardware we are reversing. We treat that driver together with the hardware as a black box that we cannot directly inspect, just as the world itself is a black box in physical science. Instead, we perform experiments to infer how the black box works. What does that look like concretely?

Graphics drivers sit between graphics applications and the hardware. We first need a tool to intercept the communications between the graphics driver and the hardware. This tool will inject itself between the driver and the hardware, inspecting the giant array of bytes the driver prepares for the hardware. Due to

software and hardware differences, we need to build this tool ourselves for each new reversing project, but the general idea stays the same.

Our goal is to decode the meaning of that pile of bytes, to eventually build our own software that produces an equivalent pile on its own. This tool is our lab instrument collecting data. What we need next is an experiment.

The proprietary graphics driver should work with any graphics application, so it's time to write some graphic software. We start by running a simple program our tool, recording what the proprietary driver sent the hardware. Next we make a small change to the program, run it again, and compare the data. If we are lucky, there will be a single small change in the output of the driver: we have found the hardware bit corresponding to the graphics state we modified. Success.

"All" that's left is repeating that process thousands of times until we have figured out most of the bits.

Most. Our model of the hardware will never be perfect. Without documentation from the vendor, there will always be unknown dark corners. But knowledge in science does not need to be perfect to be useful to engineering. There are unsolved problems in physics, gaps in our understanding of the physical world, but we build trains and planes anyway. And there are unsolved mysteries in the Apple M1, but I wrote a graphics driver for it anyway.

7. **Beyond the projects you've already developed, are there any others you'd like to work on in the future?**

Who knows? I choose projects that solve problems faced in our community. As new problems appear, new opportunities for projects follow. Before the Apple M1, there was no Asahi Linux. Despite regularly giving technical talks with a cape, witch hat, and magic wand, I have no special talent for prophecy. We'll have to learn what the future holds the old-fashioned way.

8. **Is there anyone —inside or outside the tech world— who has particularly inspired you? What advice would you give to someone who wants to start contributing to free technologies? Where should they begin?**

So many people have inspired me. How could I hope to list them all here?

Inside tech, earlier reverse engineers paved the way for me to emulate their craft. Take Rob Clark, who kicked off the reverse-engineering of Qualcomm's Adreno GPU. Thanks to his efforts reversing the hardware and building the free software "Freedreno" driver, we now have mature free software support for Adreno shipped in products by Google and Valve. As for Rob, he never quit, and he is now employed by Qualcomm to continue his work from the inside. That's a success: he freed

Adreno, changed the course of the industry, and built a great driver. Without fanfare he puts in the work, day in and day out, to make the world a little better, and the payoff is better than anyone imagined. When I got my first Mali device, I did my best to follow Rob's footsteps. It worked: there has not been reverse engineering of Arm's Mali in years, because there is now a team at Arm finishing the driver I started. Both Qualcomm and Arm turned the page on open source graphics, and it would not have happened without Rob.

Beyond reverse engineering, many in the free software graphics community have inspired me to grow as a driver developer and compiler author. Faith Ekstrand, Connor Abbott, Daniel Schürmann, Kenneth Graunke – these people have taught me more about how to build graphics drivers than any class could.

Beyond tech, many social activists inspire me, both for what they stand for and for their foundational belief that progress is possible and within reach if we work together. That includes the stories from history textbooks, but it doesn't stop there. You don't have to be a household name to make a difference locally. Far from tech, far from the limelight, and far from the past, I have close friends organizing for transgender rights in the US south, in an era when those rights are under attack. And it's working. Huntsville, Alabama is a little bit safer thanks to their tireless work. If they can turn hearts, correct narratives, and defeat unjust bills in their sphere, I can do the same in mine.

Whether it's graphics drivers or human rights, the idea is the same: work together, work on what nobody else will, and work to make your local slice of the world a little better.

Positive change often has a domino effect. Rob didn't know in 2012 his project would lead to mine, I didn't know in 2018 that my project would lead Arm to support free graphics drivers, and nobody knows what Arm's pivot might mean down the line. We can't outsmart or outwit the monoliths, but we can stay nimble and improve what we can, where we can, and share a drink when the chips fall into place.

Nobody can single-handedly make change, and there is more injustice in the world than there is time in any single person's day. Your challenge – should you choose to accept it – is to find your niche, find some time, and find some friends. If you make a small difference, that is enough. You will have succeeded.