Calvin McCormack
October 18, 2024
Music 256A
Stanford University

Reading Response #4

There is a sentence on page 196 that really resonated with me: “Aesthetics is not something you add to a nearly finished product, but an active force an **intentionality** that shape design from the **start**. It takes the form of **creative constrains** and **articulations of preference** that push a design forward in a sea of **petrifying possibilities**, or **guiding principles** through which decisions are considered through a **human lens**.” When I first began delving into digital music making, computer music, and programming, I found this ocean of possibilities to be “petrifying”. In each of these spheres, the rabbit holes are many and deep. In music production, there are several “major” daw’s, within which there are thousands of software instruments, plugins, and then very deep rabbit holes such as another whole bottomless programming environment existing *inside* of Ableton. In computer music and programming, there are so many languages and further frameworks within each language, which are constantly being developed and redeveloped and deprecated and replaced, that it often feels like wading into quickly shifting sands, and the search for the “correct” option feels fruitless and discouraging. If I could give advice to myself back then, (while I could just tell myself what my favorite daw and plugins end up being…) I would say to just pick one and dive in as far as you can. Some things work better or are easier here vs there, but really going deep and discovering your own workflow will help you learn better than obediently following youtube tutorials, and you will create your own idiosyncratic method for making something (maybe discover/develop something new!) and will be more rewarding overall.

“The trick, I think, is to use it freely but wisely” (page 202) sounds like another advocation for Buddha’s middle path. But for principle 4.10 “Programmability is both blessing and curse”, I see more blessing than curse; in this example even the “curse” is Chuck! I think also that this paradigm applies to any design or artistic related endeavor, it’s a spin on the eternal battle between fine details and the big picture. I think both paths have their potential pitfalls, but what seems most important to me is to remain open to unexpected happy accidents (like Chuck?) that arise from deep dives, procrastination, perfectionism, or trying to use a program in a weird “wrong” way, and I think Chuck, and its many Ch- prefixed children, is a great example. They are all deep rabbit holes inside another rabbit whole. And as much as I love (and need) many commercial software, particularly when not-crashing is a high priority, I think this is where open-source shines as a development philosophy.