Alex Ding
Commonwealth School
Junior

Personal Statement

When I grew up in China,[1] I always enjoyed problem solving and exploring wherever my curiosity took me. But every "why" directed to the knowledge recited from my teachers' lecture notes was met with a "you'll learn it later." Being a gamer from an early age, computers fascinated me: how, for example, do the movements on my mouse translate into pixel changes on the screen. When I asked my father, all he could tell me was that "there're a lot of complicated things going on. Just wait till college." The lack of access to more advanced knowledge was intellectually stifling. Following the curriculum designed to march everyone to graduation at the same, steady pace was painful. I had the time and the desire to learn more, but my youth seemed to be an impossible limiting factor reigning me in. The best (and only) CS course my middle school offered at ninth grade barely taught me anything, and due to the inter-institution rivalries in China, colleges competed by excluding outsiders and hoarding their lessons and resources. When too much became too much, I moved to the U.S. for my current high school, one that promised the release from this age limit. To my school and I, this age limit never made much sense anyway: courses should organize their students based on their level of knowledge, not age.

It was liberating. The desire to learn more was rewarded, not punished. I set out on learning spree for the pure goal of acquiring more knowledge, signing up for online courses and cruising through the materials in every spare moment. I held that the more facts someone knows, the better that person is as a computer scientist. I was wrong, of course, but no one ever corrected me, for I was on the journey alone, until one day, I got The Structure and Interpretations of Computer Programs (SICP) as a gift from my advisor. It was mind-blowing, especially for someone who has only taken introductory-level programming courses. I had a bias towards powerful languages: I thought a heavy-lifting languages like C++ could do more than a simpler languages could ever do. SICP, however, was a treatise on just how much a limited and rigid language like Lisp can do. Of course, the book taught me interesting concepts, but more importantly, it taught me that transcending all the facts and concepts was the way of thinking. It's representative of the whole field of computer science: combining elementary components in amazing ways to create interesting systems. From Nand gates to computers, from 0s and 1s to meaningful representations, from basic objects to powerful classes, what is amazing isn't just the end product, but rather the mean to the end, the abstraction on each level to effect the seemingly magical machines that we use and love today.

The summer of my freshman year I went to NuVu Studio, being assigned into a group applying AR to health. I knew next to nothing about the topic at hand: I've never done

---

[1] Up to the end of Middle School, which unlike the U.S., is 9th grade.

anything on Unity, nor have I worked on AR, nor did I know C# at all. Yet I learned that, with just the right techniques of googling, I could pick up any skill that I might need. Trained with the art of problem solving and thinking process of a computer scientist, I was able to hack through whatever challenge I met. I rapidly picked up C#, realizing that it is trivial to do so with the design concepts (such as OOP) already understood. Each of the problem solved advanced the skill set I had. That was when I realized the power of project-based learning. It enabled me to get things done, to rapidly apply the knowledge learned systematically for interesting purposes. It made me realize that there was nothing stopping me from the ends I wanted to achieve. Later, when I wanted to take Theoretical Calculus, I just self-studied Calculus over the summer to skip it in school. A year later, when I needed knowledge on multivariable calculus for helping out on a ML package, I simply picked it up over the summer. It was empowering.

I love exploring new tangents and concepts, and PRIMES offers me exactly such an opportunity. The lack of guidance in my exploration earlier caused me to waste a lot of time learning for the sake of gathering more knowledge, but with a set goal and a helpful mentor to direct my efforts, my passion and curiosity would yield much greater results. This is also why I think I am a good candidate. I may not be the best qualified mathematician or programmer, but I have seen drastic developments as a computer scientist over the last two years. I sought out every opportunity to engage in projects and learning. A rare research opportunity would be an invaluable experience and challenge that I would rise up to and tackle.

In the future, I hope to work as a researcher in the field of CS, perhaps with a specialization in AI and ML. With that goal in mind, I hope to actively seek out opportunities to challenge myself as an independent researcher. After all, what distinguishes CS from other fields of research is that I can just sit down with a laptop and no other special equipments and do something amazing.

I enjoy working on problems that force me to derive my own approaches and/or reuse preexisting approaches in new and exciting ways. The music genre transform project (described in the short answers to earlier questions) is one example of it: it gives a new take on ML by back propagating on the input data, not the model parameters.

Thank you for taking the time to review my application. I spent a great deal of time working on the problem sets. And thank you for providing these problem sets. They motivated me to read up a huge assortment of texts and to practice program design. I learned a lot working on them.