I was initially intrigued by programming because of boredom. My school used a website called Khan Academy to teach kids math and such. But KA also had a robust coding curriculum and a project board with coding projects created by students of the KA coding course. Most of the projects on the board were either simple or low-effort recreations of popular games, such as Five Nights at Freddies and a faithful 1-to-1 recreation of the first level of Super Mario Bros. However, the more I played these games during class time (when I should have been doing the math work we actually used KA for), the more and more I wanted to make a game like these. So I started the Khan Academy basic programming course. It was entirely focused on Javascript, and you were given a window that would push your code in real time as you pressed run, similar to Scratch (which I had also grown familiar with at the time). The lessons consisted of using draw functions to draw simple shapes and create images or animations with them. I was also taught how to debug my code with their cheery syntax error “character”, Oh Noes. Over time I learned how to use simple Javascript, and started trying to make games myself. They were incredibly rudimentary with messy code and hundreds of if statements made up most of the code. But the point was made clear to me: I liked this. I loved creating something interactive and the challenge of creating it was what drove me to learn more. I have continued to learn and pick up small bits of programming knowledge since then, like scratch visual coding and extremely basic C# in order to make my own games in the Unity engine. I also took a course in my senior year to learn Python at a basic level, and I understand the basic structure of most object-based languages.

Video games are my passion, so it’s only natural I turn to them as pinnacle examples of perfect creative programming. They are the perfect unity of computing and creativity, and each aspect of this unification can have different levels of complexity. My favorite example of the harmony between programming and creativity is a game called Minit. Minit is an 8-bit adventure game made in 2018, and it revolves around one simple concept: limited time. As soon as you start moving in the game world, a timer of one minute starts to tick down. Once the clock hits zero, everything on the map including your location is restored to their original states. Using your one minute sparingly, you can discover different parts of the map and use it to permanently aid your exploration in the next run through. Graphically, Minit is a very simple game. Its black and white pixel art color scheme is unobstructing yet detailed enough to add charm to the world. The genius in this game is its incredibly simple timer mechanic. By giving the player a limited amount of time to do their actions, it forces exploration and gives urgency of action to the player. Minit is a joyful experience despite being so rudimentary in its mechanics and creative scope, and that’s the beauty of it. Games do not have to be huge and complex to be enjoyable and memorable. They just have to provide a fun and engaging challenge. However, that’s not to say complexity can’t also be amazing. Take a game such as Dishonored 2. Dishonored is a game centered mostly around assassinating and killing bad guys as stylishly and silently as possible. However, the game gives subtle hints throughout your playthrough to how the fictional world around you is reacting to your actions. There are hundreds of different ways to approach each situation, and each one has a different outcome (that usually boils down to “did you do a good thing, or an evil thing?”, but I digress). The genius within this choice that most players would not think twice about, is that the game is coded with a complex morality system that reacts to the moral decisions of the player. You can hear terrified guards talking about an enraged and silent killer

that’s been cutting through the police force like a scythe, just before descending upon them to gut them like fish. This doesn't only affect the atmosphere and worldbuilding, but the gameplay as well. If you have been killing rather than sneaking past or safely incapacitating combatants, the enemies will begin to number their forces, making each encounter after a brutal killing spree more difficult than the last. Furthermore, the story, while complicated and too exhaustive to explain, also is determined by your actions. Most importantly the ending, which can be different depending on your mercy towards enemies and characters throughout the game. The morality system allows for this level of complexity and immersion, and is an amazing example of how robust programming and mechanics can enhance a creative experience. Finally, taking a detour from games, one of my favorite YouTubers is a charming dude named Acerola. He’s a graphics programmer who is super into video games and the computer science that goes into making them. He makes a lot of videos regarding post-processing effects and explanations of the rendering pipeline in most video games. But most interestingly to me, he makes tutorials of sorts for many different interesting and seemingly complex visual effects and shaders using programming and math. He explains how games use simple ideas such as sine waves and noise functions to create realistic looking ocean water. He also explains the career path of a graphics programmer, and how it may not be what many of the people who watch his videos (such as myself) expected it to be.

A dream project is a hard concept for me. I have multiple ideas such as games and interactive art projects I have been conceptualizing for years but can’t make, mostly due to a lack of programming knowledge. I think to simplify it for myself, my dream project would probably be a game of incredible proportions, that allows for an incredible amount of player self-expression, a detailed and immersive world with engaging mechanics, and clever systems and challenges. I don’t know for sure what kind of game that would be, be it an RPG or an open-world game or even a boring first person shooter. But I just know that the thing holding me back from making such a project is my inexperience with programming. The programming provides the structure upon which I can expand my creative scope. That’s what a good sense of programming would be able to get me.