Braden Arestides
Senior Design
Capstone Self-Assessment Essay

Our project revolves around algorithmically assisted music composition. Specifically, we want to create some sort of tool that song composers, or prospective composers, can use to help them create real-sounding compositions. From an academic perspective, this project will require practical knowledge of algorithms and software engineering practices. Also, effectively collaborating and working together in an organized manner will be very important. Past group class assignments don't fairly compare to the scale of this final project. Academically, outside of computer science, the project will require some music theory understanding to properly execute.

The knowledge I have gained from the classes of my curriculum will undoubtedly be useful during the planning and execution of this project. Design and Analysis of Algorithms will surely help in coming up with complex music generation algorithms. The practical development skills from Software Engineering will also be applied. Specifically, I will make use of the part of the class that focused on code review. I am sure that the experience I gained from group projects in Programming Languages and Software engineering will be useful. I would like to be able to apply what I learned in Database Theory and Design, but unfortunately, I don't think that will come up much at all for this project.

While class knowledge will surely be indispensable, I think that the experience gained from my co-ops will be even more useful. In my co-ops at Projetech, Inc, I gained real world experience with developing and maintaining a production application. The level of responsibility I was given means that I was expected to architect substantial feature implementations. I don't think this would happen at many other company co-ops. I've also gained significant experience with working with people, both technically and non-technically, at my co-ops. Sometimes this means collaboratively designing a solution, other times there are conflicts that need to be resolved. One of the most fun things I get to do at my co-ops is what I call "requirement-vetting". Many times, stakeholders will come to me with a proposed solution to a problem, and it will be my responsibility to implement it. Before I was experienced, I would just blindly implement this solution. Now, however, I realize that people are busy and can make mistakes. This means that the proposed solution might not actually be the best one. Sometimes there is a better solution that doesn't even require development! This saves me time, and saves the company money. Unfortunately, with a lack of stakeholders, I don't think I'll get to exercise this skill during this project.

This concept of algorithmically generated music is something that has interested me ever since I started learning about programming. So many aspects of music can be represented mathematically, which makes it such an apt domain for programming. I'm actually surprised that this concept isn't already more established. Or maybe it is, but it's all kept secret by the record companies. Some artists, like Aphex Twin, have experimented with algorithmic composition, and

I think it shows when you listen to him. But I'm more interested in creating a program capable of pumping out chart-topping hits. I think it would be really cool to completely fabricate an artist, and attract a fanbase, and then just say "Ha! It was a computer all along!"

I've already done some noodling around with this in my spare time, before this became our senior project. The code I've written works by sending midi signals so some software instrument. A lot of the code is actually machinery to get this working, but I have a few basic melody generators and a lot of ideas around analysis of existing music. I'm not sure sending MIDI signals is the best approach, though. The main upside is that the user gains complete control over the receiver. It can be a simple synthesizer, or a full-fledged DAW, or a music-notation software. The biggest downside is that in order to send the generated notes to an external receiver, we have to wait for the generator actually "play" them, the same way someone would play notes into a DAW using a physical MIDI keyboard. This is bad because if a user wants to make tweaks to the generation parameters, each tweak requires a re-recording of the output. The forerunning alternative approach is to design the program as a MuseScore plugin. MuseScore is a music notation software primarily used to compose music. There are two advantages to this approach: We can send updated melodies basically instantly by using the API, and we get to take advantage of MuseScore's existing abstraction for musical concepts (like notes, chords, measures). The main downsides are that we are limited to the MuseScore plugin development environment, and that the user's choices of sounds are significantly reduced.

I will consider the senior design project done once it can convincingly generate at least *some* realistic-sounding portions of a song. Long term, however, I would like to be able to generate full-length compositions that are indistinguishable from compositions written by humans.