Landon Judkins

MidiHero Postmortem

MidiHero's core mission is to be an entertaining game that develops musical skill. This is a primary innovation of MidiHero as well. Unlike previous music games like DDR and Guitar Hero, becoming an expert MidiHero player will actually hold some value because the player will have developed useful musical skills. This mission requires that MidiHero interface with real instruments, cultivate musical ability, and still be an entertaining game. Much of the overall game design was borrowed from existing music games, but a few key changes need to made. Whereas DDR and GuitarHero can present multiple difficulties for the same song easily, MidiHero has more restrictions on what it can have the player do since all of the input maps to MIDI output. In Guitar Hero for instance, an effort is made to make the notes seem to match the song but in reality there is no one-to-one correspondence between input and audio output, and the Guitar Hero controller is in fact incapable of representing enough notes to actually play the song. In MidiHero, the song can certainly be presented at different speeds and with different grading criteria, and the game may also isolate individual tracks, wait for player input, or any number of other training features, but the notes to be played need to map to real MIDI events. However, there isn't any reason that MidiHero can't provide choice for what notes to play. This will be the other main innovation in MidiHero. Instead of providing purely single-track linear game-play, MidiHero will allow for multiple paths through timing based pattern recognition. Amplitude, another simplified music scroller, allows the player to move between tracks, but this is done using track selecting input and does not allow divergence within the note playing interface. MidiHero relies solely on the MIDI Controller interface so the pattern based selection system will be necessary but also more appropriate because it will not break the flow of the music. This will not only allow the player to choose how the song will sound, but it will hopefully provide a MIDI game interface that could be easily used for other types of games like 2D platformers or even fighting games.

What Went Right

1. The introduction of choice in gameplay was in my opinion the best thing to come out of this semester. Originally I wanted to reproduce Guitar Hero for the guitar, but once I realized that Audio-to-MIDI wasn't going to happen (see “What went wrong”), the game lacked innovative aspects. Adding choice opened up a lot of new opportunities, and even if I never get dynamic music generation to sound great, choice will remain important even if only used for deterministic decision tree based songs.

2. When I ran into the Mac OS X performance bugs in Java 2D (see “What went wrong”), I switched to JOGL (Java OpenGL binding) which ended up being a really good idea. JOGL allowed me to bypass the Java 2D issues and transition to more sophisticated graphics. This also ended up being a great learning experience because I had never worked with any graphics libraries aside from Java 2D, but I was able to rework SGE to be based on JOGL, which will be much more useful for games.
3. The move to staff notation ended up being a really effective choice. The staff provided a simple, but not altogether foreign, way of prompting the player. Most musicians that tried the game were able to read the notation immediately and most beginners were able to pick it up in a few rounds. Part of the success of using just staff notation relied on using a small set of notes to author or choose beginner songs.
4. The particle system used for the graphics ended up being a last minute addition but it provided a very effective, good looking interface with very little art. Originally, I wanted the user interface to be simple and relatively free of extraneous glitter, but after the transition to OpenGL was complete, I decided that the game should have an immersive visualization which was still conceptually simple. A particle system made the most sense because it was something that I knew would not require a lot of prior graphics knowledge or custom art, both of which would have delayed the game even further.
5. The dynamically generated music in MidiHero has not sounded that good yet, but I'm confident that it will be one of the most important features of MidiHero in the future because it will give it another advantage over traditional music scrollers. Eventually, I think MidiHero will incorporate a distributed database component which tracks user preferences and uses that data to generate user-specific music at least as well as a mediocre musician, and possibly much better than that.

What Went Wrong

1. The use of Java 2D ended up being a huge mistake. Most of the time I was working on SGE and the early MidiHero prototypes, I assumed that processing speed would not limit my graphics because they were very simple. When I first started investigating why my graphics system was running so terribly slow, I naturally assumed that it was something that my code was doing wrong, but what I eventually discovered was that the Mac OS X (my development platform) implementation of Java 2D has a number of serious performance issues. As an example, it takes about 4 milliseconds just to draw one image. These issues were so severe that the entire SGE

graphics model had to be overhauled to use JOGL instead.

2. One of the biggest, mostly avoidable mistakes made during development was allowing the build to be broken a lot of the time. The problem originated with the Java 2D issues which pretty much rendered the engine and game broken. Once I had repaired the graphics system, I had decided to make a number of changes to the game, including choices, so the build remained broken while I added those features. Instead of getting something up and running quickly, I spent a lot of time making my game logic extremely generic and extensible, but I ended up not using most of the generic features which I built in like grading a number of MIDI messages besides ON/OFF, custom tolerances for every note, a generic pattern matching system, and a number of other features which just didn't end up being necessary in practice.
3. Another major problem with the MidiHero project is that it was constantly changing directions. As a prototype it was a very effective testing tool for game concepts and I was able to walk away with a lot of new innovative features which weren't originally part of the design. Unfortunately, this also made the game not end up being the polished user release that I had hoped it would be. The stricter requirements on MIDI data removed any chance of easily implementing MIDI import like I did in the early versions, but the staff notation prevents beginners from reading difficult songs anyway. A lot of the algorithms for manipulating music, such as translating notes across keys or shifting them in key never made it into use in the final version. Part of this problem was attributable to the fact that I could get away with all these sudden changes since MidiHero was not a group project.
4. The dynamic music generation didn't end up producing something playable yet. Although I personally learned the most about music from training mutating songs, it was definitely something that required a conscious commitment which I wouldn't expect from an uninvested player. The algorithms did not always produce variants that sounded good which meant the player really had to make good choices or the song would start sounding worse and worse.
5. Implementing Audio-to-MIDI turned out to be unrealistic and already attempted. Early on in the research for Audio-to-MIDI it started to be pretty clear that no one had been able to do this well yet, even though a lot of qualified people had tried. To make matters worse, a commercial game “Guitar Rising” was already implementing what MidiHero was originally intended to be, Guitar Hero with a real guitar, but even their team could not get chord recognition to work. Eventually, I just had to abandon the idea for the time being.

Conclusion

Overall, I would say that the project was a success, as both a learning experience and an experimental game. There were a number of problems, but is some ways, the problems provided some of the most valuable lessons. I definitely started to see how constant demos and changes affect coding practices, and I think that future development of MidiHero will really benefit from the experience that I've gained. I also think that I took away a lot of new ideas that will be a big part of the game when finished. If I were to do the whole thing over again, knowing what I know now, I would definitely do a lot of things differently, but I'm pretty satisfied with the experience as a whole and what I got out of it.