Keith Thompson

Mike Forsyth

Chapter 3: Game Audio

**Section 3.1 WAV files and the AC97**

**Subsection 3.1.a: WAV File Conversion**

We made a python script that parsed the wav files for us. It was really easy because python had a built in library that can parse wav files. The only trouble that we had was getting the program to parse the file correctly. We found out that the python library doesn’t universally parse all wav files without more work. So for our script we made sure to grab wav files that were specifically to the spec that we needed, as in they had a sample frequency and mono sound that was required for our script for parse it correctly.

Our generated C files included the sampling rate, number of samples, and an array of the samples of each .wav file. These values were necessary when working with our sound interrupt handler. The number of samples made it possible to stop outputting data to the FIFO when a certain sound finished. The sampling rate was needed when deciding the frequency for the DAC. The array of samples was needed as output to the FIFO. The FIFO would then output the data to the DAC producing the sounds for the game.

**Subsection 3.1.b: AC97 Operation**

The AC97 initially needed to be hard reset and have all of the FIFOs cleared. It then needed to have its initial volume set using the auxiliary output register. Next variable rate audio (VRA) needs to be enabled which allows DAC and ADC sample rates to be programmed via registers 2Ch and 32h. Next the codec needs to be given time to initialize and be ready using the predefine wait function. After the codec is ready the DAC sampling rate needs to be set using the predefined macros. This sampling rate should match the sampling rates of the .wav files. Lastly, data can be placed in the pre-built FIFO which will autonomously and separately (does not use our Microblaze processor) empty itself, giving its contents to the DAC. The DAC then gives its signal to the speakers which will produce the desired sound.

**Subsection 3.1.c: Sound Triggering**

Our code chooses sounds based on the priorities we came up with in the tickHandler. We decided that the tank explosion takes the highest priority since it happens rarely and should grab the player’s attention. The mothership has the second highest priority since its appearance is a special event in the game and should be given greater notice than the normal aliens. The moving aliens have the lowest priorities since they move the most often. Tank bullets and mothership explosions are lower priority than tank death and higher than alien movement. The tank bullets are given higher priority than the mothership since they occur less frequently than the mothership.

**Bug Report**

* One bug was the python script for parsing wav files. We found out that we needed to either make the program a lot larger to be able to handle different kinds of wav files, or we needed to find wav files that fit our spec that we were parsing. After we figured this out (with the help of classmates) we just obtained and used wav files that we could parse and then it worked fine
* When doing a basic sound test using polling we did not loop the sound repeatedly which resulted in unknown data being placed in the FIFO. This resulted in very noisy results.
* In our final interrupt handler for sound we test to see when the sound has completed. Initially, however, we had an unnecessary “-1” in the check that prevented the sound from every being properly finished and resulted in a repeated stuttering sound.