CS 2261 Lab 10:
Sound

Provided Files

- `main.c`
- `myLib.c`
- `space01.c`
- `space02.c`
- `space03.c`
- `space04.c`
- `space05.c`
- `space06.c`
- `space07.c`
- `text.c`
- `Originals folder containing`
    - `GameSong.wav`
    - `StartSFX.mp3`
    - `TitleSong.wav`

Files to Edit/Add

- `main.c`
- `sound.c`
- `GameSong.c`
- `StartSFX.c`
- `TitleSong.c`

Instructions

In this lab, you will be modifying the attached project to play sounds correctly. You will be editing the code that currently plays sound to pause, stop, and loop sounds properly.

- **TODO 1 - Convert sound files**
    - You will need to convert the attached sound files to be the correct .wav format using Audacity and then use the wav2c program (located in

Resources -> GBA Software Resources, called wav2c.exe for windows and WavToGBA.dmg for mac) to convert the .wav files into .c and .h files

- **Formating the .wav file**
  - **(***Please note that these steps may slightly vary for different versions of Audacity.)*
  - Open .wav audio files in Original folder in Audacity
  - Select the track
  - Select Tracks -> Stereo Track to Mono
  - Set the Project Rate (Hz) in the lower left corner of the screen to 11025
  - Go to Tracks -> Resample and choose 11025 as the new sample rate
  - Select File -> Export Audio
    - Be sure to save it in your lab folder where the rest of your .c and .h files are
  - Under the "Save as type" dropdown, select Other uncompressed files
  - If there is a button for "options," press the button
  - You should see dropdown menus for "Header" and "Encoding"
  - Set Header to "WAV (Microsoft)"
  - Set Encoding to "Unsigned 8-bit PCM"
  - Type ".wav" after your file name (for example, GameSong.wav)
  - When the Edit Metadata screen pops up, press the Clear button then hit OK.
- **Converting the .wav file**
  - **Windows**
    - Drag your newly saved wav file onto the wav2c.exe in the file viewer
    - It will automatically run and give you .c and .h files in the folder the .wav file is located
  - **Mac**
    - Download and install the wav2c app
    - After opening the app, select the .wav file
    - The app should generate the .c and .h files in the same folder the .wav file is located
    - If this does not work, use Wine with the following instructions
      - Type into terminal: wine
      - Drag wav2c.exe onto terminal window

        - Drag the audio file onto the terminal window
        - Press enter
        - .c and .h files will be outputted in the folder the .wav file is located
    - Convert all of the sounds in the Originals folder
    - TODO 1.1: Include the sound .h files at the top of main.c
- **TODO 2 - Complete the playSound functions**
    - TODO 2.0: Navigate to the playSoundA function in sound.c. This is the function we will use to tell our game to start playing a sound, so let's complete it.
    - TODO 2.1: Assign all of soundA's appropriate struct values
        - soundA is declared in sound.h. You can find the SOUND struct in myLib.h
        - Data, length, frequency, and loops are assigned values passed in through the function.
        - isPlaying should be equivalent to "true," since when we call this function we want a sound to start playing.
        - The duration formula is ((VBLANK_FREQ*length)/frequency)
        - We can ignore priority, as it is not necessary for this lab
        - vBlankCount should start at 0
    - TODO 2.2: Complete playSoundB with the same logic as playSoundA (these functions will be almost identical)
    - At this point your code should build, but you won't hear any sound.
- **TODO 3 - Complete the interrupt handler**
    - TODO 3.0 - Navigate to the setupInterrupts function
    - TODO 3.1 - Set up the interrupt handler register
        - The interrupt handler register is a macro that can be found in myLib.h and should point to the interruptHandler *function*
        - HINT: you will have to cast the interruptHandler function to something. Look closely at the register macro.
    - TODO 3.2 - Handle soundA playing in the interruptHandler function
        - This is where we want to determine if we need to stop playing soundA or not.
        - First, increment the sound's vBlankCount
        - Then, if the sound's vBlankCount is greater than the song's duration, we know we've reached the end of the song. You need to handle two cases here:
            - If the sound loops, restart the song

            - If the sound does not loop, set the sound playing to false, turn off the DMA channel the sound is using, and turn off the timer the sound is using. Looking at the playSoundA function will be helpful here.
    - TODO 3.3 - Handle soundB playing in the interruptHandler function
        - This will be extremely similar to TODO 3.2
    - TODO 3.4 - Call the two setup functions for sounds and interrupts in main.c
    - At this point your code should build and start playing and looping music on the title screen! If it does not, take a closer look at your sound and interrupt functions.
        - Wait for the title song to loop and make sure you do not hear "Cornerface screaming at you", aka your game playing random bits of memory that will make a screeching noise. This means you are not correctly stopping your sounds in the interruptHandler function.
- **TODO 4 - Pausing, unpausing, and stopping music**
    - We want to be able to control when our music plays and when it doesn't, so we have three functions to handle this.
    - TODO 4.1 - Complete the pauseSound function
        - To pause a sound, we want to set soundA and soundB playing to false and stop their timers
    - TODO 4.2 - Complete the unpauseSound function
        - To unpause a sound, we want to reverse the changes made in TODO 4.1 (HINT: check out the timer flags in myLib.h)
    - TODO 4.3 - Complete the stopSound function
        - Completely stop soundA and soundB. Should be very similar to some of the code you wrote in TODO 3.2 and TODO 3.3
    - At this point your code should build, but as we have not called any of these functions yet, you will not notice any changes.
- **TODO 5 - Set up more sounds!**
    - Now that we have everything in place to play sounds, we can add more to our game.
    - TODO 5.1: Play GameScreen music and StartSFX sound when the player presses START to transition from the start to game state.
        - Call stopSounds first, in case any songs are currently playing.
        - Use playSoundA and playSoundB to play both sounds at once. Check out the songs .h files to figure out what variables to pass into those functions.

- TODO 5.2: Pause the music when transitioning from game to pause screen.
    - HINT: We wrote a function that does exactly this.
- TODO 5.3: Stop sounds and restart the title song when going from game to splash screen.
- TODO 5.4: Play spaceSound here.
    - Frequency should be SPACE01FREQ
    - Make sure this does not interrupt the GameSong (should play on a different channel)
- TODO 5.5 - Unpause the music when transitioning from pause to game screen.
    - HINT: We wrote another function that also does exactly this.
- At this point, you should be done! Check that your lab meets all of the requirements below.

The final completed lab will have the following

STARTSCREEN:

- A looping title song (TitleSong)
- A sound that plays when going to the game screen (StartSFX)

GAMESCREEN:

- A looping game song (GameSong)
- A sound effect that plays on pressing the A Button
- Pausing the sound when going to pause screen
- Stopping sound when going back to the start screen and restarting the title song

PAUSESCREEN:

- Unpausing the sound when returning to the game screen.

Submission Instructions

Zip up your ENTIRE project folder and submit it. Make sure your submission includes a compiled .gba file.
Name your submission "Lab10_FirstLast.zip", e.g. "Lab10_GavinFree.zip"