**Follow these instructions to configure your GBA makefile to be able to use sound.**

1. **Open the makefile with WordPad (NOT Word)**, as this preserves the formatting best. In the first couple of paragraphs you will see a list of directory names, such as TARGET, BUILD, SOURCES etc. Basically, the words in all-caps are like shortcut variable names, each containing a list of directories that the makefile will consider when the variable name is used. Each item in a list should be separated with a space. **Add a new list using the variable name MUSIC, and add the folder “music” to this list.** This makes sure our music will be included when compiling.
2. **A couple of paragraphs later, in the list of extra included libraries (libs), add:**

-lmm

This is the name of the Maxmod library of functions. Remember when we did PS3 exercises last year we had to link padutil.a if we wanted to use the controller? This is essentially doing the same thing. If you use any more libraries you will need to put them here too.

1. A bit later, where it “automatically builds a list of object files for the project”, **change the BINFILES list so it only includes the file soundbank.bin.** This file hasn’t been created yet, but we are just telling the computer that it will belong to **BINFILES** once it has been created.
2. **Now add the following code after the BINFILES line.** Let the text wrap naturally and be careful of spaces – there is a space after the word “wildcard”!

export AUDIOFILES := $(foreach dir,$(notdir $(wildcard $(MUSIC)/\*.\*)),$(CURDIR)/$(MUSIC)/$(dir))

This seems to get everything in the MUSIC directory list (i.e. everything in the “music” folder, see step 1) and label it as AUDIOFILES. Presumably this is because there might be more than one folder of audio stuff.

Still with me? Good. Here’s where it gets confusing.

1. Further down, **under the section on “main targets”, but after the gba and elf OUTPUT lines, add this code exactly as shown. You might want to copy and paste as it is quite delicate**:

soundbank.bin : $(AUDIOFILES)

@mmutil $^ -osoundbank.bin -hsoundbank.h

%.o : %.bin

@echo $(notdir $<)

@$(bin2o)

This seems to compile everything in the AUDIOFILES list (see step 4) into a single file called “soundbank.bin”. It creates some header files, and then uses “soundbank.bin” as a template to create a file called “soundbank.o” using a utility program called “bin2o”, which is located somewhere inside DevKitPro. This doesn’t complete the process though, as you will see in the next step.

1. We know from step 5 that we created a file called “soundbank.bin”, and from step 3 that “soundbank.bin” was to be included in the BINFILES directory list**. Move up a bit in the makefile, to where there is an instruction to export OFILES, and find the BINFILES bit** that says:

$(addsuffix .o,$(BINFILES))

At the moment it is **adding** the suffix of “.o” onto anything in the BINFILES list, so we would end up with “soundbank.bin.o”, which is no good. We need to make sure it takes everything in the BINFILES list and **replaces** the suffix “.bin” with “.o”, so “soundbank.bin” becomes “soundbank.o”. **Change this part of the code so it says:**

$(BINFILES:.bin=.o)

1. That should be it! For the love of god, save and close the makefile!
2. Add these headers to your C source code:

#include <maxmod.h> // maxmod library

#include "soundbank.h" // created by building project

#include "soundbank\_bin.h" // created by building project

maxmod.h should be found automatically as it is part of DevKitPro. If not, ask for help. soundbank.h and soundbank\_bin.h will be created when you run the makefile. They will appear in the ‘build’ folder rather than an include folder, but it is okay to leave them there.

1. After the line:

irqInit();

Add this:

irqSet( IRQ\_VBLANK, mmVBlank );

This appears to set a callback for the GBA to retrieve Maxmod sound engine updates whenever the display reaches a vertical blank (i.e. when VBlankIntrWait(); finishes at the end of each frame). However, we still need to get Maxmod to actually update itself, so after the line:

VBlankIntrWait();

You need to call Maxmod’s per-frame update function. Use the Maxmod documentation to look up the name of this function!

1. Now, nothing will happen if we don’t initialise the Maxmod engine, so after the line:

consoleDemoInit();

We need to initialise Maxmod, and point it initially to our soundbank memory area, the handle to which is provided in the soundbank\_bin.h header file (check the “build” folder).

mmInitDefault( (mm\_addr)soundbank\_bin, 8 );

Then we start playing and looping the song:

mmStart( MOD\_XXXXXXXXXX, MM\_PLAY\_LOOP );

But replace “MOD\_XXXXXXXXXX” with the correct name, using the list in soundbank.h for help (check the build folder). If you have more than one mod file, they should all be compiled into the same soundbank by the makefile, so you should be able use the mmStart command later to switch between songs (providing they are both in the same soundbank).