Tutorial 12: Using Mod music / sound

http://www.codemii.com/2009/04/21/tutorial-12-using-mod-music-sound/

In this tutorial you'll learn about adding music/sounds to the Wii. It's a fairly simple thing to do and we'll use Mod files. I assume everyone knows MP3s but you might not know about Mod files which are a little bit like midi files. Mod files in a sense have "samples" which can be used in different ways to produce sounds.

There are two ways which you can load music, either from the SD card or from your source. If you aren't going to allow users to change your music or sounds, I'd recommend loading from your source, it just saves time and it's nicer not have lots of files on the SD card.

Let's get started. You'll need to add –lmodplay to LIBS: in your makefile and also add #include in your main.c file.

So now you have a standard main.c file and your modified makefile like this zip file: tutorial12-makefile.zip. Copy this loop.mod file to your SD card's root or use ftpii to transfer it across.

Loading mods from the SD card

Before we begin, we need to also add libfat to our libraries in the makefile as below.

```
LIBS := -lwiiuse -lbte -lfat -logc -lmodplay –lm
```

We also need to include fat.h in our source code.

As well as the standard initialisation of the SD card:

```
If(!fatInitDefault()) {
  printf("Unable to initialise FAT subsystem, exiting");
  exit(0);
}
```

We have to initialise modplay, so we add the below in our initialise function.

```
Void Initialise() {
  ….
  AUDIO_INIT(NULL);
  MODPlay_Init(&play);
}
```

Now we can move on to the source code which will load one MOD file from the SD card as the background music.

```
// Modplay
static MODPlay play;
long mod_size;
char* buffer;
size_t result;
```

Above we just assign some globals. Play is the object which we will reference when wanting to play our mod file. mod_size will store the file size of our mod file, buffer will point to where the file is stored in memory and result is sort of just like mod_size except it will check to make sure the whole file is copied successfully to memory.

```
FILE *f = fopen("/loop.mod", "rb"); // change to "SD:/loop.mod"
                                    // if using r16
if (f == NULL) {
  fclose(f);
} else {
  fseek(f, 0, SEEK_END);
  mod_size = ftell(f);
  rewind(f);
```

So firstly we open the file that we have in our SD card's root directory. We do the usual check to make sure the file exists and then use ftell to give us the file size of the loop.mod file.

```
  // Allocate memory to contain the whole file:
  buffer = (char*) malloc(sizeof(char)*mod_size);
  if (buffer == NULL) {
    perror("Memory error\n");
  }
```

As you might recall buffer is the pointer to where the file will be stored in memory. We use malloc to assign a block of memory the size of our mod file and do a check to make sure the memory was assigned properly.

```
  // Copy the file into the buffer:
  result = fread(buffer, 1, mod_size,f);
  if (result != mod_size)  {
    perror("Reading error\n");
  }
  fclose(f)
```

If everything was fine, we copy the entire mod file into memory using fread to store it to the buffer. Result stores the number of bytes read. If result matches mod_size this means that it successfully read the entire file. We can then close the file.

```
  MODPlay_SetMOD(&play, buffer);
  MODPlay_Start(&play);
}
```

Now the important part, we have our file loaded in buffer and we use MODPlay_SetMOD to make reference to our play global variable that contains which mod file is loaded to it which we set our buffer to. MODPlay_Start is then used to start playing our mod file; again we make reference to the play variable.

```
While(1) {
  …
  if(pressed & WPAD_BUTTON_HOME) {
    MODPlay_Stop(&play);
    exit(0);
  }
  …
}
```

Our mod file will keep on playing in a loop until we specific to stop playing the mod file with MODPlay_Stop(&play) and that's that. You can view the complete source and try out tutorial12-mod-sdcard.zip

Loading Mods from source

Now we can move onto loading mod files from our source which I find easier.

You need to modify your makefile to add a rule which will convert all your .mod files in a specific folder to a 'compiled' state which is usable; it also generates a header file (an include file) which we will use.

Below is what you need to add to your makefile after "# main targets" as below.

```
#---------------------
# main targets
#---------------------
$(OUTPUT).dol: $(OUTPUT).elf
$(OUTPUT).elf: $(OFILES)

%.mod.o : %.mod
@echo $(notdir $<)
@$(bin2o)

-include $(DEPENDS)
```

You now need to specify the folder in which your mods will be stored, so modify or add the below line after "SOURCES :=" and make sure you create the /data subdirectory where your makefile is located.

```
DATA := data
```

So now you have our standard main.c file again and your newly modified makefile like this zip file: tutorial12-makefile2.zip. Once extracted, go ahead and compile the project. You'll see there is a loop.mod file in the /data directory. Once compiled, go to the build folder and you can see loop_mod.o and loop_mod.h, which shows the mod file converted to a 'compiled' state and the header file.

Again, we have to initialise modplay, so we add the below in our initialise function.

```
Void Initialise () {
  …
  AUDIO_Init(NULL);
  MODPlay_Init(&play);
}
```

We can now edit our source code as below.

```
#include “loop_mod.h”
```

Since we have the header file loop_mod.h, we just need to include that in our source code.

```
//Modplay
static MODPlay play;
```

We add the play variable as a global like last time.

```
MODPlay_SetMOD(&play, loop_mod)
MODPlay_Start(&play);
```

Then simply we can play the file as we did previously, this time having the loop variable when calling SetMOD. We don’t need any more code as this mod file is already in memory.

The next thing you can do is add more .mod files to your /data directory and re-compile. When you wish to change the mod file being played all you need to do is run MODPlay_Stop(&play); and then run MODPlay_SetMOD(&play, different-sound_mod); and MODPlay_Start(&play); as usual. You can view the complete source and try out tutorial12-mod-embedded.zip

And this concludes our tutorial about using Mod files, easy wasn’t it?

Others things you can try out is and make a different sound play every time a button is pressed on the Dpad, playing a mod file and pressing a button to stop and if you combine these tutorials making a sound when a button is pressed.