UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Rafał Gosiewski

# Portfolio

| | |
|---|---|
| **Course:** | Arquitectura y Entornos de Desarrollo para Videoconsolas (AEV) |
| **Year:** | 2018 |
| **Description:** | This document has been created as a portfolio for all practice/laboratory sessions. Here I placed all steps which you have to do to try finish all exercises and some solution for problems which I had and they have never been mentioned in exercise guide. |

## Table of contents

# Installation of devkitPro development environment for the architecture of the NDS and N3DS video game console

## Installation environment

First what I had to do, was configure DevkitPro environment. On SourceForge.net platform there is a prepared package with auto-installator. Unfortunately, at the moment when I tried to configure and run that script, one element in that package wasn't available.

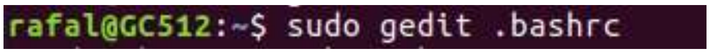
After some tries I finally decided configure it manually, downloading all necessary packages separately, and then run the automated script BUT modified (I removed whole section about checking for newest versions of packages). In my case, that was a solution.

Below, I am attaching list of all packages and their version, which I installed.

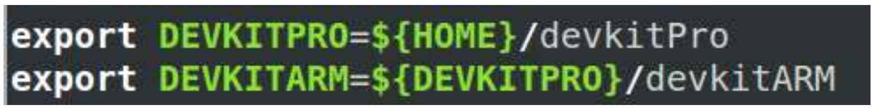
```
[libndsfat]
Version=1.1.2
File=libfat-nds-1.1.2.tar.bz2

[devkitARM]
Version=47
File=devkitARM_r47-i686-linux.tar.bz2

[libmirko]
Version=0.9.7
File=libmirko-0.9.7.tar.bz2

[libgbafat]
Version=1.1.2
File=libfat-gba-1.1.2.tar.bz2

[citro3d]
Version=1.3.1
File=citro3d-1.3.1.tar.bz2

[filesystem]
Version=0.9.13-1
File=libfilesystem-0.9.13-1.tar.bz2

[dswifi]
Version=0.4.2
File=dswifi-0.4.2.tar.bz2

[defaultarm7]
Version=0.7.3
File=default_arm7-0.7.3.tar.bz2

[maxmodgba]
Version=1.0.10
File=maxmod-gba-1.0.10.tar.bz2

[gp32examples]
Version=20051021
File=gp32-examples-20051021.tar.bz2

[gbaexamples]
Version=20170228
File=gba-examples-20170228.tar.bz2

[maxmodds]
Version=1.0.10
File=maxmod-nds-1.0.10.tar.bz2

[3dsexamples]
Version=20170714
File=3ds-examples-20170714.tar.bz2

[libgba]
Version=0.5.0
File=libgba-0.5.0.tar.bz2

[ndsexamples]
Version=20170915
File=nds-examples-20170915.tar.bz2

[libctru]
Version=1.4.0
File=libctru-1.4.0.tar.bz2

[libnds]
```

```
Version=1.7.1
File=libnds-1.7.1.tar.bz2
```

Then I had to set environment variables DEVKITARM and DEVKITPRO. To do that I edited file .bashrc and added two export commands, and also executed them in console.



```
rafal@GC512:~$ sudo gedit .bashrc
```



```
export DEVKITPRO=${HOME}/devkitPro
export DEVKITARM=${DEVKITPRO}/devkitARM
```

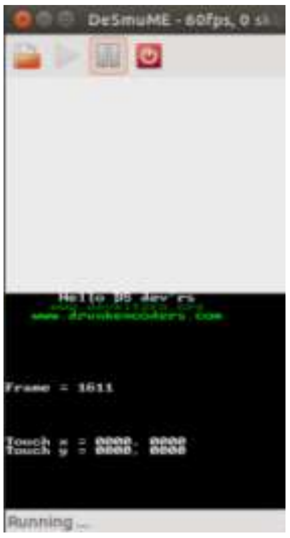
The last thing was to execute make for both collections with examples - for NDS and 3DS.

First I just execute make command and got some error information. Then I realized that i got them from examples which weren't inside package prepared for that exercise, so I had to remove "extra" examples, and then executed make again. That one was successful.

I didn't have any other problems with installing and configuring the devkitPro environment.

## Emulators

Firstly, I tried to run hello_word example for nds using inbuilt emulator DeSmuME. As you can see on image below, it worked correctly.



```
Hello DS dev'rs
[illegible]
www.drunkencoders.com

Frame = 1611

Touch x = 0000, 0000
Touch y = 0000, 0000
```

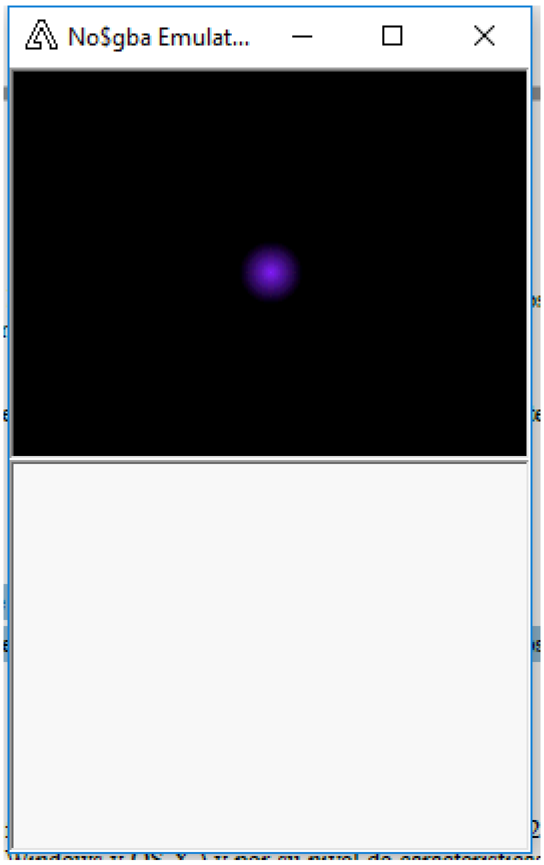
Then I tried install Citra emulator. For that, because my OS is Ubuntu 16.04 LTE I followed the instructions from the page: [https://github.com/citra-emu/citra/wiki/Building-for-Ubuntu-16.04](https://github.com/citra-emu/citra/wiki/Building-for-Ubuntu-16.04)

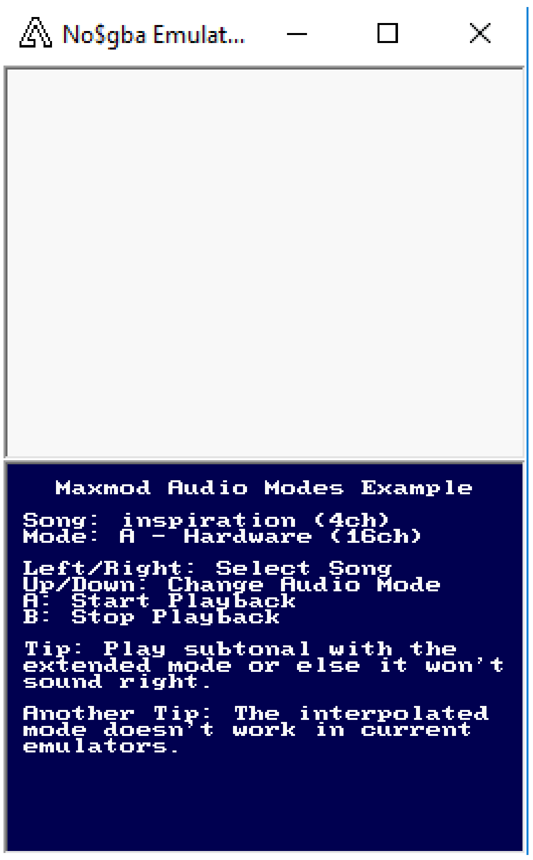
I had some problems, because at the day when I tried to do that, the make command after downloading didn't work properly, so it was impossible to install Citra easily. I looked at GitHub repository for more detailed instructions. Unfortunately Citra isn't final project, and still has different versions. I tried a lot things to install it on Ubuntu 16.04 LTE, and always got the same errors during making. In my last try I called make command with --ignore argument.
It also didn't work, so finally I decided to install and configurate whole environment once again at Windows 10. Success.

## Exercise 1.

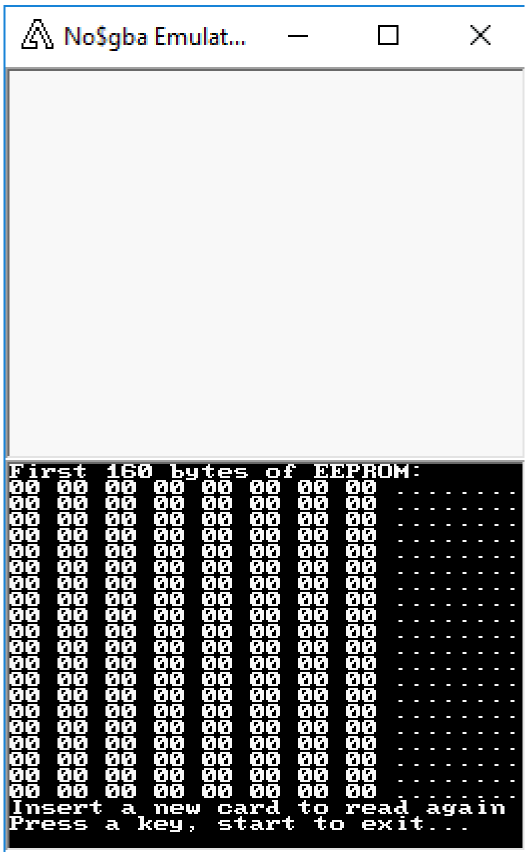
I installed 2 different emulators - Citra, for all 3ds projects, and no$gba for all nds projects. Below I attached screenshots from examples which I tried to run on each of the emulator.

***NO$GBA:***

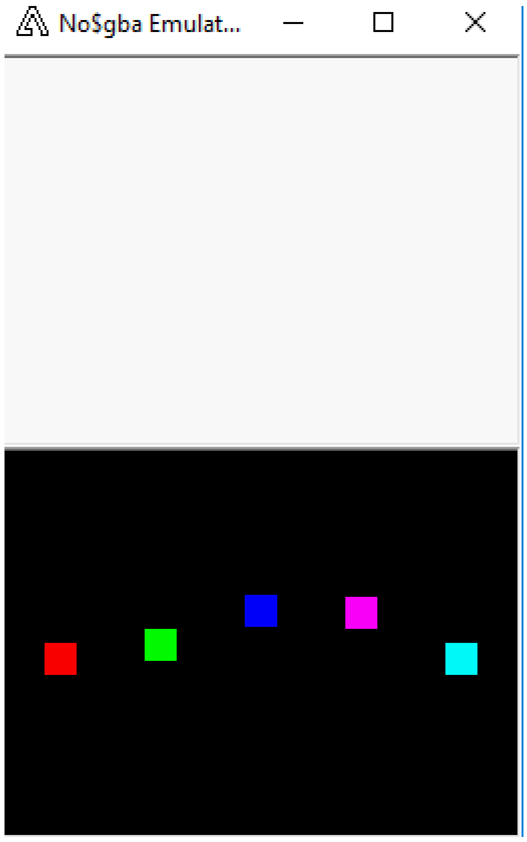


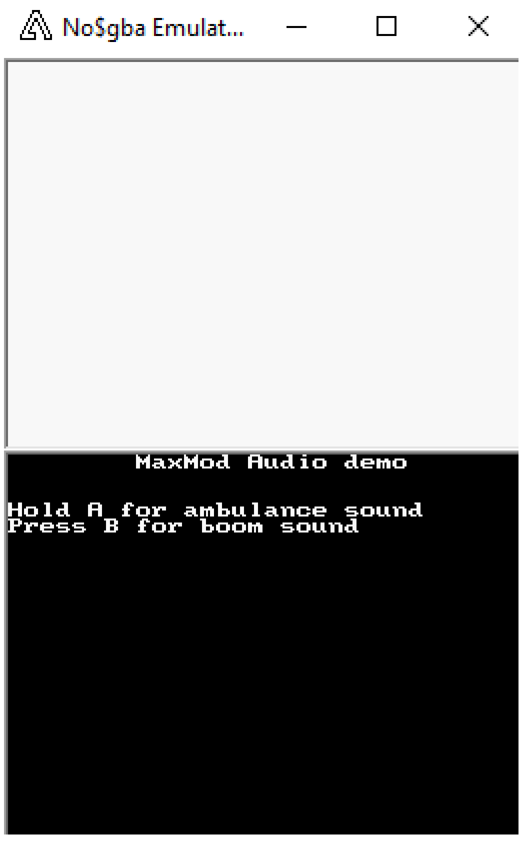


```
Maxmod Audio Modes Example

Song: inspiration (4ch)
Mode: A - Hardware (16ch)

Left/Right: Select Song
Up/Down: Change Audio Mode
A: Start Playback
B: Stop Playback

Tip: Play subtonal with the
extended mode or else it won't
sound right.

Another Tip: The interpolated
mode doesn't work in current
emulators.
```

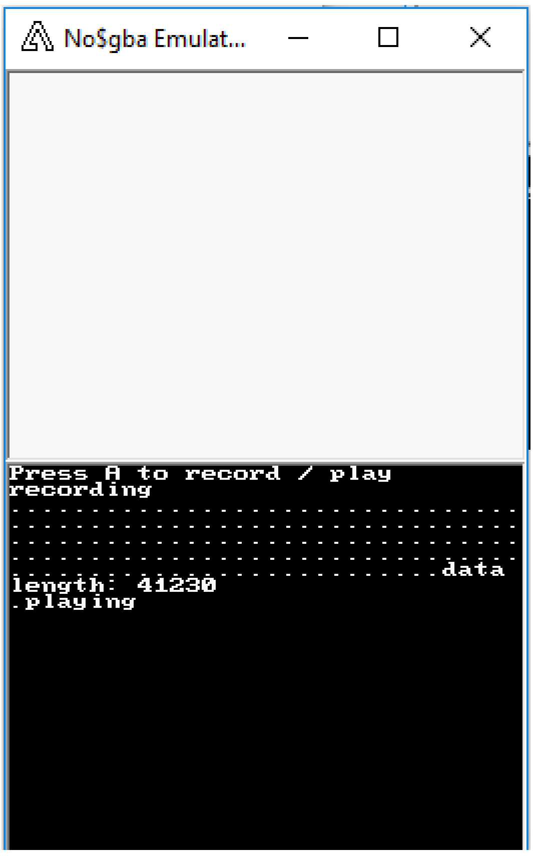


```
First 160 bytes of EEPROM:
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
Insert a new card to read again
Press a key, start to exit...
```

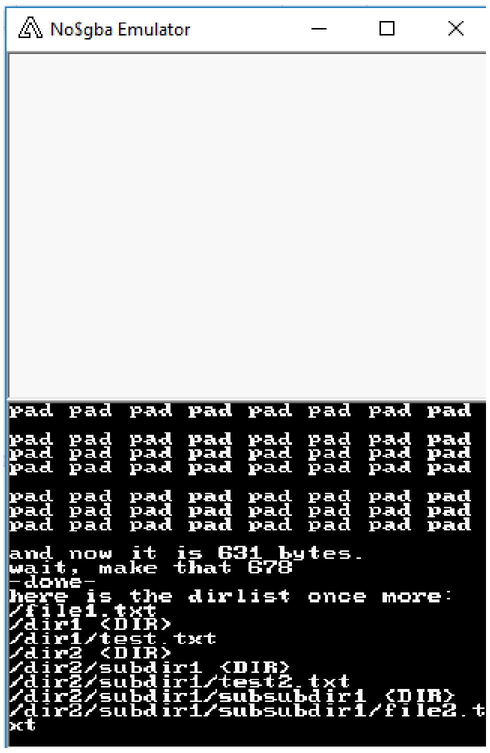




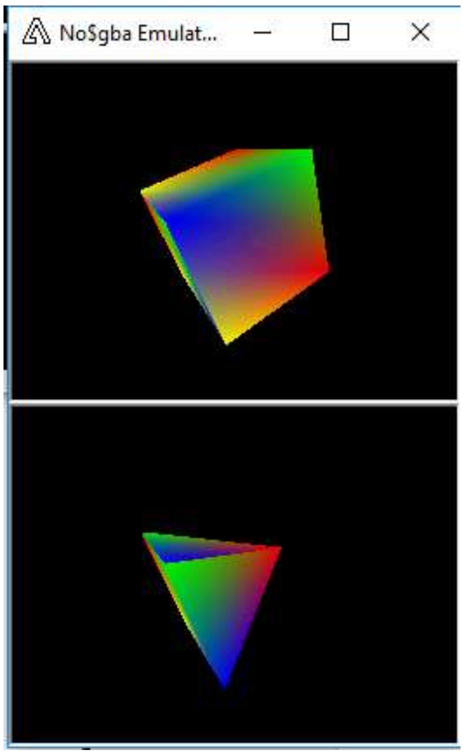
```
       MaxMod Audio demo

Hold A for ambulance sound
Press B for boom sound
```

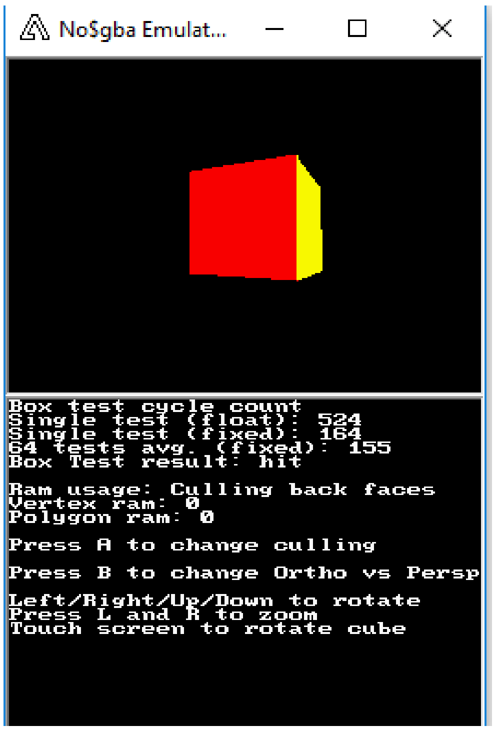


```
Press A to record / play
recording
........................................
........................................
........................................
...............................data
length: 41230
.playing
```



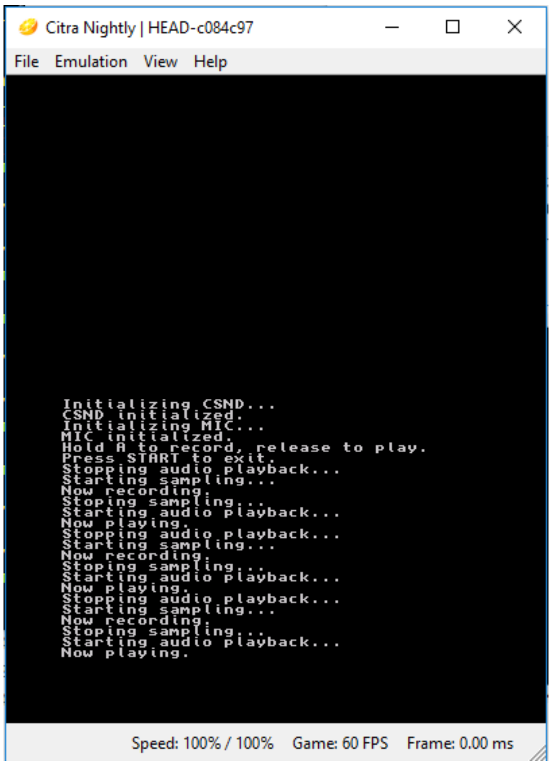
```
pad pad pad pad pad pad pad pad

pad pad pad pad pad pad pad pad
pad pad pad pad pad pad pad pad
pad pad pad pad pad pad pad pad

pad pad pad pad pad pad pad pad
pad pad pad pad pad pad pad pad
pad pad pad pad pad pad pad pad

and now it is 631 bytes.
wait, make that 678
-done-
here is the dirlist once more:
/file1.txt
/dir1 <DIR>
/dir1/test.txt
/dir2 <DIR>
/dir2/subdir1 <DIR>
/dir2/subdir1/test2.txt
/dir2/subdir1/subsubdir1 <DIR>
/dir2/subdir1/subsubdir1/file2.t
xt
```





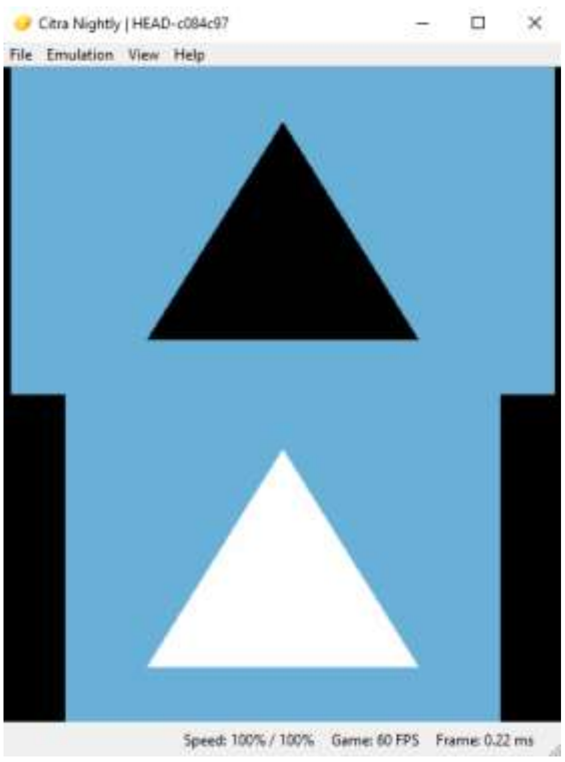
```
Box test cycle count
Single test (float): 524
Single test (fixed): 164
64 tests avg. (fixed): 155
Box Test result: hit

Ram usage: Culling back faces
Vertex ram: 0
Polygon ram: 0

Press A to change culling

Press B to change Ortho vs Persp

Left/Right/Up/Down to rotate
Press L and R to zoom
Touch screen to rotate cube
```

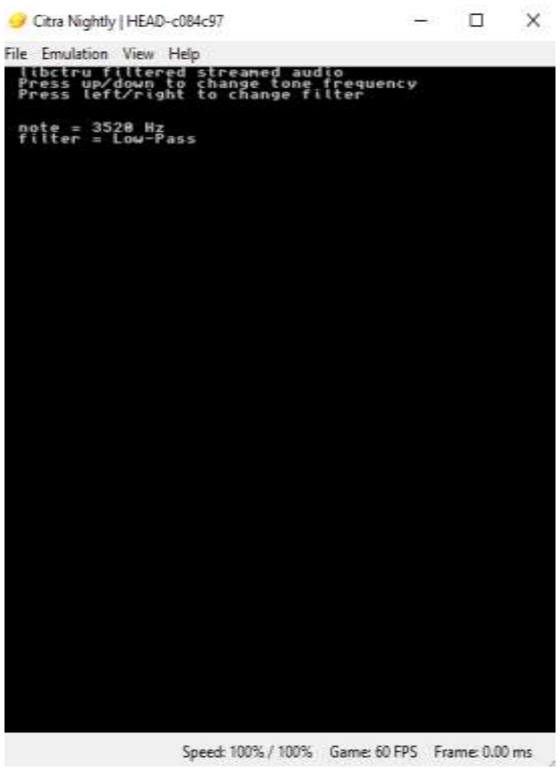
## Citra:



```
Initializing CSND...
CSND initialized.
Initializing MIC...
MIC initialized.
Hold A to record, release to play.
Press START to exit.
Stopping audio playback...
Starting sampling...
Now recording.
Stoping sampling...
Starting audio playback...
Now playing.
Stopping audio playback...
Starting sampling...
Now recording.
Stoping sampling...
Starting audio playback...
Now playing.
Stopping audio playback...
Starting sampling...
Now recording.
Stoping sampling...
Starting audio playback...
Now playing.
```

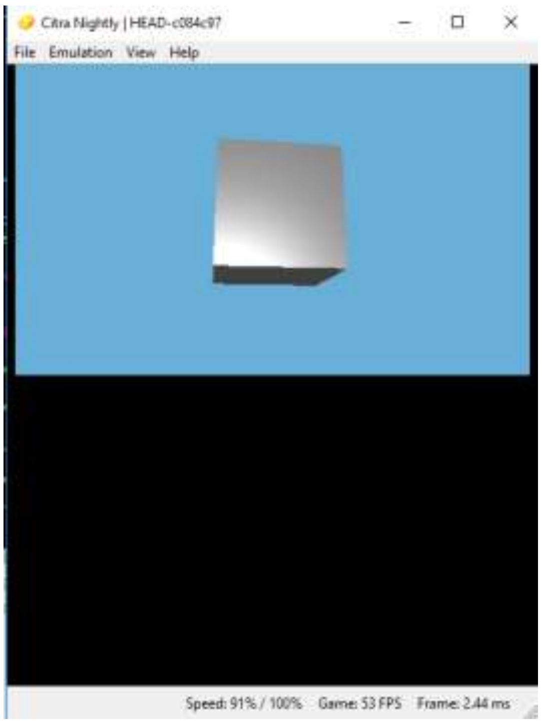
Speed: 100% / 100% Game: 60 FPS Frame: 0.00 ms

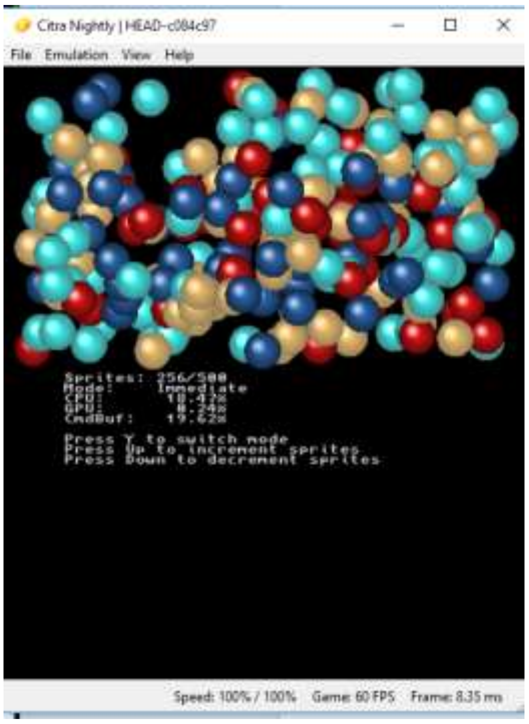

Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 100% / 100% Game: 60 FPS Frame: 0.22 ms


Citra Nightly | HEAD-c084c97
File Emulation View Help

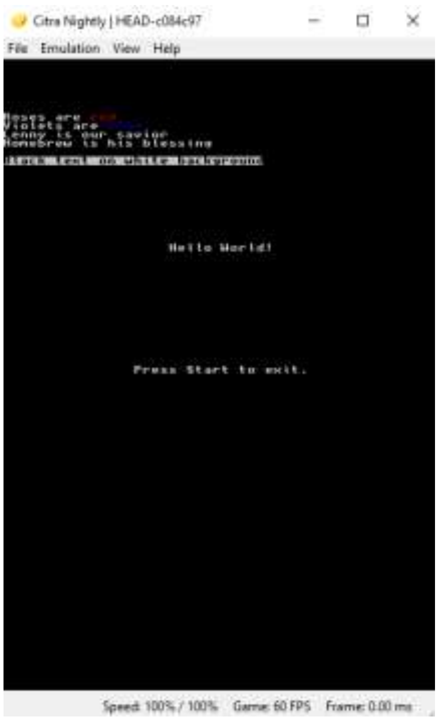
```
libctru filtered streamed audio
Press up/down to change tone frequency
Press left/right to change filter

note = 3520 Hz
filter = Low-Pass
```

Speed: 100% / 100% Game: 60 FPS Frame: 0.00 ms

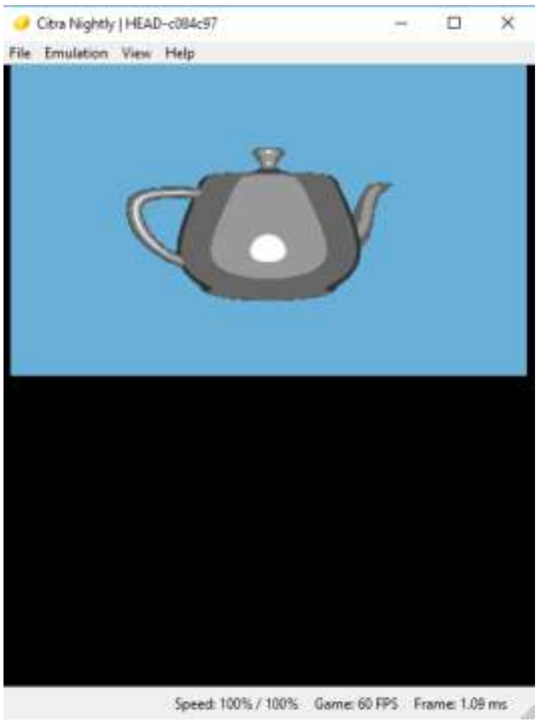

Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 91% / 100% Game: 53 FPS Frame: 2.44 ms

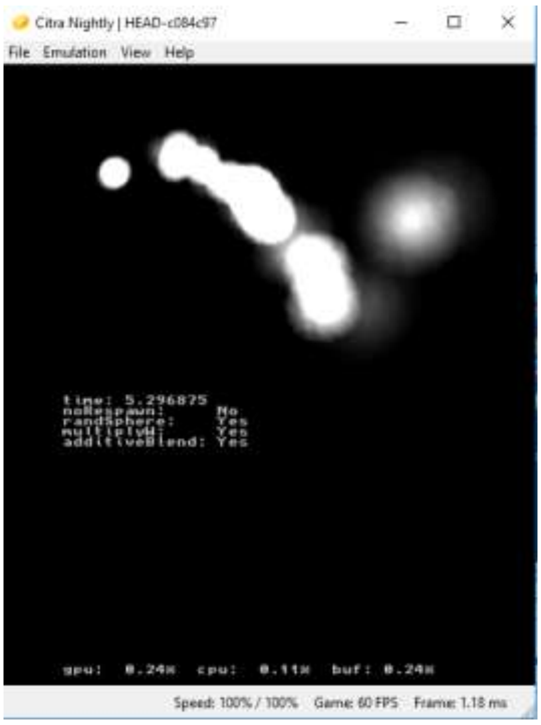

Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 100% / 100% Game: 60 FPS Frame: 8.35 ms

```
Sprites: 256/500
Mode:     Immediate
CPU:         18.42%
GPU:          8.24%
CmdBuf:      19.62%

Press Y to switch mode
Press Up to increment sprites
Press Down to decrement sprites
```


Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 100% / 100% Game: 60 FPS Frame: 0.00 ms

```
Roses are red
Violets are blue
Lenny is our savior
Homebrew is his blessing
Black text on white background


                Hello World!


           Press Start to exit.
```


Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 100% / 100% Game: 60 FPS Frame: 1.09 ms


Citra Nightly | HEAD-c084c97
File Emulation View Help
Speed: 100% / 100% Game: 60 FPS Frame: 1.18 ms

```
time: 5.296875
noRespawn:     No
randSphere:    Yes
multiplyW:     Yes
additiveBlend: Yes

gpu:  0.24%  cpu:  0.11%  buf: 0.24%
```

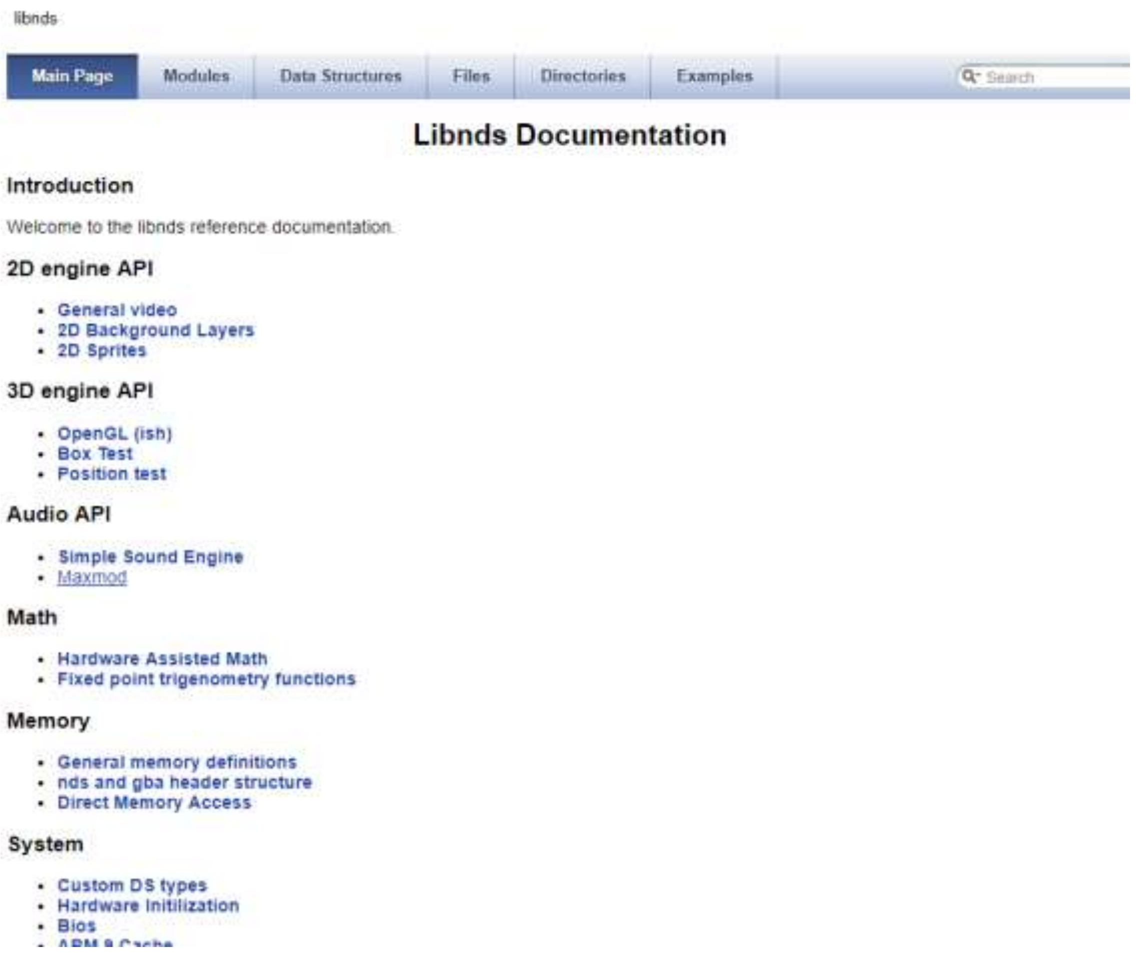
I only saw difference in record exercise - the difference was in way of printing information on the screen.

During that Exercise I run all examples from devkitPro for 3ds and nds collection and analyzed all source codes.

## Access to information on the NDS video game console hardware

### Documentation:

Firstly, I had to download whole documentation for libnds from SourceForge.net. I used version from 2011.02.14 - the same which has been mentioned in the laboratory guide. Below, screenshot from main page of that documentation.



Libnds Documentation

Introduction

Welcome to the libnds reference documentation.

2D engine API

- General video
- 2D Background Layers
- 2D Sprites

3D engine API

- OpenGL (ish)
- Box Test
- Position test

Audio API

- Simple Sound Engine
- Maxmod

Math

- Hardware Assisted Math
- Fixed point trigenometry functions

Memory

- General memory definitions
- nds and gba header structure
- Direct Memory Access

System

- Custom DS types
- Hardware Initilization
- Bios
- ARM 9 Cache

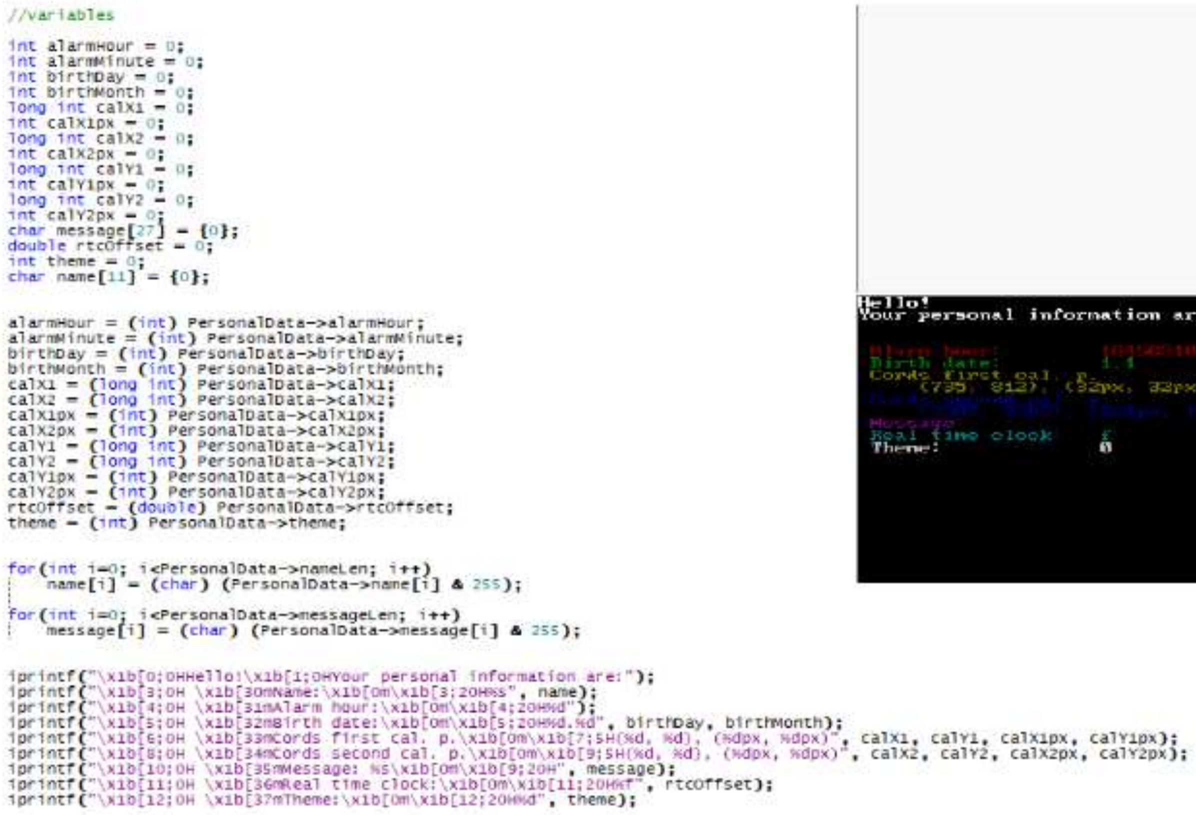
### Access to the system:

In that section I was learning about system functions and possibilities depends on hardware configuration. In screenshot below, you can see some information from Personal Data and NDSX from arm9_main.cpp example.

```
/*
    Initialization done. Print welcome message.
*/
char name[11] = {0};
for(int i=0; i<PersonalData->nameLen; i++)
    name[i] = (char)(PersonalData->name[i] & 255); // get ascii-bits from utf-16 name
printf("Hi %s,\n", name);
printf("Your DS is %s!\n", (NDSX_IsLite())?"a Lite":"a Phatty");
printf("Your battery is %s!\n\n", (NDSX_GetBattery_Status())?"almost empty":"well charged");
printf("TouchIcon = Change Brightness.\n");
printf("TouchElse = Toggle Backlights.\n");
printf("L = Toggle Blinking.\n");
printf("R = Toggle BlinkSpeed.\n");
printf("\x1b[11;0HBrightness: %u  \n", NDSX_GetBrightness());
printf("\x1b[12;0HBacklights: %u  \n", NDSX_GetBacklights());
printf("\x1b[13;0HBlinking:   %s  \n", NDSX_GetLedBlink_Status() ? "on":"off");
printf("\x1b[14;0HBlinkspeed: %s  \n", NDSX_GetLedBlink_Speed() ? "fast":"slow");
```

## Exercise 1:

In first exercise I had to print all information from PersonalData structure use ANSI escape characters for "styling" screen output. This is printscreen from emulator and sourcecode



```
//variables

int alarmHour = 0;
int alarmMinute = 0;
int birthDay = 0;
int birthMonth = 0;
long int calX1 = 0;
int calX1px = 0;
long int calX2 = 0;
int calX2px = 0;
long int calY1 = 0;
int calY1px = 0;
long int calY2 = 0;
int calY2px = 0;
char message[27] = {0};
double rtcOffset = 0;
int theme = 0;
char name[11] = {0};

alarmHour = (int) PersonalData->alarmHour;
alarmMinute = (int) PersonalData->alarmMinute;
birthDay = (int) PersonalData->birthDay;
birthMonth = (int) PersonalData->birthMonth;
calX1 = (long int) PersonalData->calX1;
calX2 = (long int) PersonalData->calX2;
calX1px = (int) PersonalData->calX1px;
calX2px = (int) PersonalData->calX2px;
calY1 = (long int) PersonalData->calY1;
calY2 = (long int) PersonalData->calY2;
calY1px = (int) PersonalData->calY1px;
calY2px = (int) PersonalData->calY2px;
rtcOffset = (double) PersonalData->rtcOffset;
theme = (int) PersonalData->theme;

for(int i=0; i<PersonalData->nameLen; i++)
    name[i] = (char) (PersonalData->name[i] & 255);

for(int i=0; i<PersonalData->messageLen; i++)
    message[i] = (char) (PersonalData->message[i] & 255);

iprintf("\x1b[0;0HHello!\x1b[1;0HYour personal information are:");
iprintf("\x1b[3;0H \x1b[30mName:\x1b[0m\x1b[3;20H%s", name);
iprintf("\x1b[4;0H \x1b[31mAlarm hour:\x1b[0m\x1b[4;20H%d");
iprintf("\x1b[5;0H \x1b[32mBirth date:\x1b[0m\x1b[5;20H%d.%d", birthDay, birthMonth);
iprintf("\x1b[6;0H \x1b[33mCords first cal. p.\x1b[0m\x1b[7;5H(%d, %d), (%dpx, %dpx)", calX1, calY1, calX1px, calY1px);
iprintf("\x1b[8;0H \x1b[34mCords second cal. p.\x1b[0m\x1b[9;5H(%d, %d), (%dpx, %dpx)", calX2, calY2, calX2px, calY2px);
iprintf("\x1b[10;0H \x1b[35mMessage: %s\x1b[0m\x1b[9;20H", message);
iprintf("\x1b[11;0H \x1b[36mReal time clock:\x1b[0m\x1b[11;20H%f", rtcOffset);
iprintf("\x1b[12;0H \x1b[37mTheme:\x1b[0m\x1b[12;20H%d", theme);
```

```
Hello!
Your personal information are:

 Alarm hour:           [illegible]
 Birth date:           1.1
 Cords first cal. p.
      (735, 812), (32px, 32px)
 [illegible]

 Message
 Real time clock       f
 Theme:                0
```

## Exercise 2:

timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5), timerCallBack);

function timerStart is defined in library timers.h
First argument correspond to “id” of hardware timer (0-3)

Second one describe the timer chanel clock divider - in another way, how many times timer clock will be divided. We can choose between 1, 64, 256, 1024 versions. Each of them give us different frequency.

Third argument depend on the frequency, describe how many times function passed as the last argument (callback) will be executed before clock everflowing.

As I mentioned before, the last argument is a callback function.

To collect some data from timer, we can use functions: timerPause(), timerElapsed(), timerTick(). Each one return us the number of ticks from the last timerElapsed() calling or the actual value of tick counter. Based on that, and knowing frequency, we can easily calculate the time and manage it.

## Exercise 3:

To collect date and time from device, we can use library <time.h> and  calculate the date exactly like it was done on example below. First we have to get reference to timeStruct using method gmtime(). Then we just need to extract each value from the object.

```
time_t unixTime = time(NULL);
struct tm* timeStruct = gmtime((const time_t *)&unixTime);

hours = timeStruct->tm_hour;
minutes = timeStruct->tm_min;
seconds = timeStruct->tm_sec;
day = timeStruct->tm_mday;
month = timeStruct->tm_mon;
year = timeStruct->tm_year +1900;
```

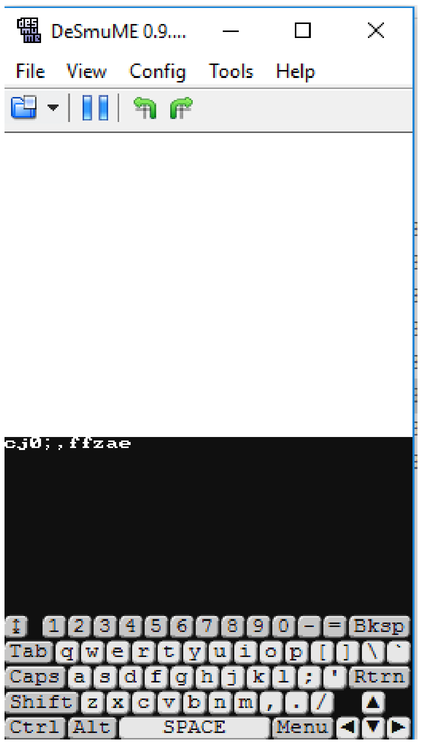
In this example, upper screen is used for displaying 3D elements. Whole screen is updated based on time extracted from the code previously desciribed. Inside the main loop, after each time updating we call method update3D(), but this method based on object which were created once before main loop, and it was task for method init3D(), where was told that 3D objects will be inside the upper screen.

```
void init3D()
{
    //put 3D on top
    lcdMainOnTop();

    // Setup the Main screen for 3D
    videoSetMode(MODE_0_3D);
```

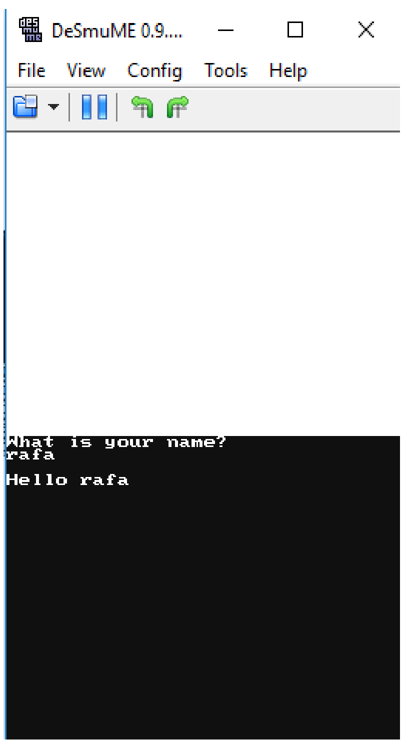
## Management of user input

### Exercise 1:

Below I attached screenshots of emulator, from each of exercises from /examples/input package. From that exercise I started using desmume emulator instead of no$gba.


DeSmuME 0.9....
File View Config Tools Help
1 2 3 4 5 6 7 8 9 0 - = Bksp
Tab q w e r t y u i o p [ ] \ `
Caps a s d f g h j k l ; ' Rtrn
Shift z x c v b n m , . /
Ctrl Alt SPACE Menu

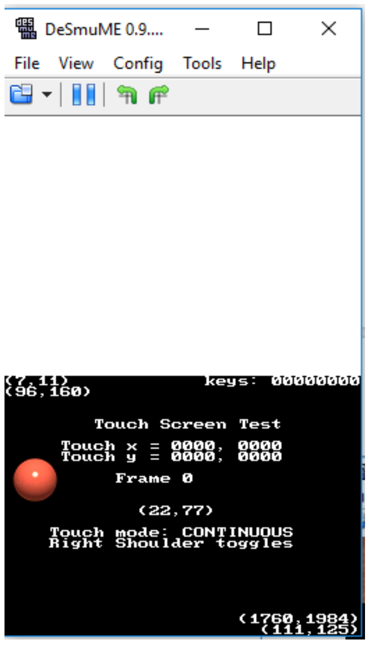
```
cj0;,ffzae
```



```
What is your name?
rafa
Hello rafa
```

DeSmuME 0.9...
File View Config Tools Help

```
Touch x = 1872, 0118
Touch y = 0592, 0038
Touch Area (pressure) 0136
```



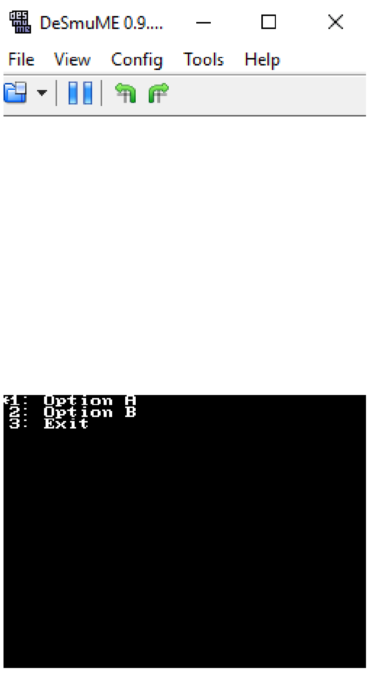
```
(7,11)                 keys: 00000000
(96,160)

        Touch Screen Test

      Touch x = 0000, 0000
      Touch y = 0000, 0000

           Frame 0

           (22,77)

     Touch mode: CONTINUOUS
     Right Shoulder toggles



                        (1760,1984)
                          (111,125)
```

## Exercise: text menu

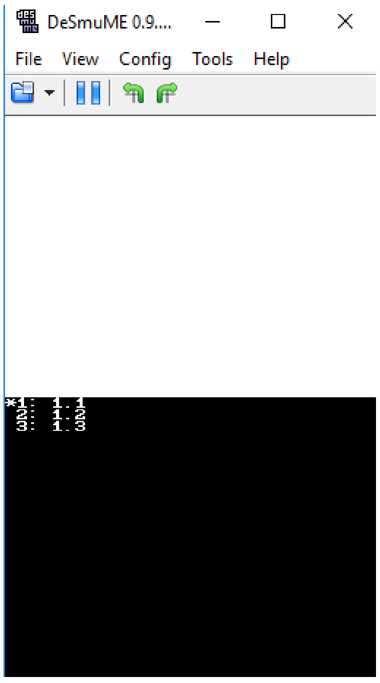
*( portfolio/projects/lab2 )*

At the moment, when I was finishing that exercise, I was working on Windows 10 (every week have different problems with libraries and environments, and some things I finish on Ubuntu and some of them on Windows) and it was impossible to set the debugger adequate for the ARM architecture so I just write the short program and builded executable app. Below I attached the screenshots from the app, which represent the changing menu, and the source code.

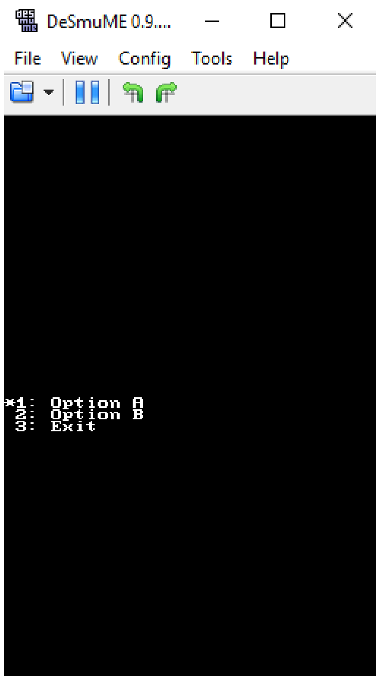
Selected option Option A -> 1.1 activate the upper screen and set the background to black.



```
*1:  Option A
 2:  Option B
 3:  Exit
```



```
*1:  1.1
 2:  1.2
 3:  1.3
```



```
*1:  Option A
 2:  Option B
 3:  Exit
```

```
#include <nds.h>
#include <stdio.h>
#include <stdlib.h>
//-----------------------------------------------
void action1_1() {
        videoSetMode(MODE_5_2D);
        videoSetModeSub(MODE_0_2D);
        vramSetBankA(VRAM_A_MAIN_BG);
}
void action1_2() {
        videoSetMode(MODE_5_2D);
        videoSetModeSub(MODE_0_2D);
        vramSetBankA(VRAM_A_MAIN_BG);
}
void action1_3() {
        videoSetMode(MODE_5_2D);
        videoSetModeSub(MODE_0_2D);
        vramSetBankA(VRAM_A_MAIN_BG);
}
void action2_1() {
        videoSetMode(MODE_5_2D);
        videoSetModeSub(MODE_0_2D);
        vramSetBankA(VRAM_A_MAIN_BG);

}

void action2_2() {
        videoSetMode(MODE_5_2D);
        videoSetModeSub(MODE_0_2D);
        vramSetBankA(VRAM_A_MAIN_BG);
}
//-----------------------------------------------
struct Demo
{
        fp go;
        const char* name;
        const char* description;
};

struct Category
{
        const char* name;
        Demo *demos;
        int count;
};

struct Demo submenuA[] =
{
        {action1_1, "1.1", "Option 1.1"},
        {action1_2, "1.2", "Option 1.2"},
        {action1_3, "1.3", "Option 1.3"}

};

struct Demo submenuB[] =
{
        {action2_1, "2.1", "Option 2.1"},
        {action2_2, "2.2", "Option 2.2"}

};


struct Category categories[] =
{
        {"Option A", submenuA, sizeof(submenuA) / sizeof(Demo)},
        {"Option B", submenuB, sizeof(submenuB) / sizeof(Demo)},
        {"Exit", 0, 0},
};

//---------------------------------------------------------------------------------
int main(void) {
//---------------------------------------------------------------------------------
        int keys;

        while(1) {
                int selectedCategory = 0;
                int selectedDemo = 0;

                bool selected = false;

                int catCount = sizeof(categories) / sizeof(Category);
                int demoCount = 0;

                videoSetModeSub(MODE_0_2D);
                consoleDemoInit();

                while(!selected) {
                        scanKeys();
                        keys = keysDown();
                        if(keys & KEY_UP) selectedCategory--;
                        if(keys & KEY_DOWN) selectedCategory++;
                        if(keys & KEY_A) selected = true;
                        if(selectedCategory < 0) selectedCategory = catCount - 1;
                        if(selectedCategory >= catCount) selectedCategory = 0;
                        swiWaitForVBlank();
```

```
				consoleClear();
				for(int ci = 0; ci < catCount; ci++) {
					iprintf("%c%d: %s\n", ci == selectedCategory ? '*' : ' ', ci + 1, 
categories[ci].name);
				}
			}

			selected = false;
			demoCount = categories[selectedCategory].count;
			if ( 0 == demoCount ) exit(0);

			while(!selected) {
				scanKeys();
				keys = keysDown();
				if(keys & KEY_UP) selectedDemo--;
				if(keys & KEY_DOWN) selectedDemo++;
				if(keys & KEY_A) selected = true;
				if(keys & KEY_B) break;

				if(selectedDemo < 0) selectedDemo = demoCount - 1;
				if(selectedDemo >= demoCount) selectedDemo = 0;
				swiWaitForVBlank();
				consoleClear();

				for(int di = 0; di < demoCount; di++) {
					iprintf("%c%d: %s\n", di == selectedDemo ? '*' : ' ', di + 1, 
categories[selectedCategory].demos[di].name);
				}
			}

			if(selected) {
				consoleClear();
				iprintf("Use arrow keys to scroll\nPress 'B' to exit");
				categories[selectedCategory].demos[selectedDemo].go();
			}
		}

}
```

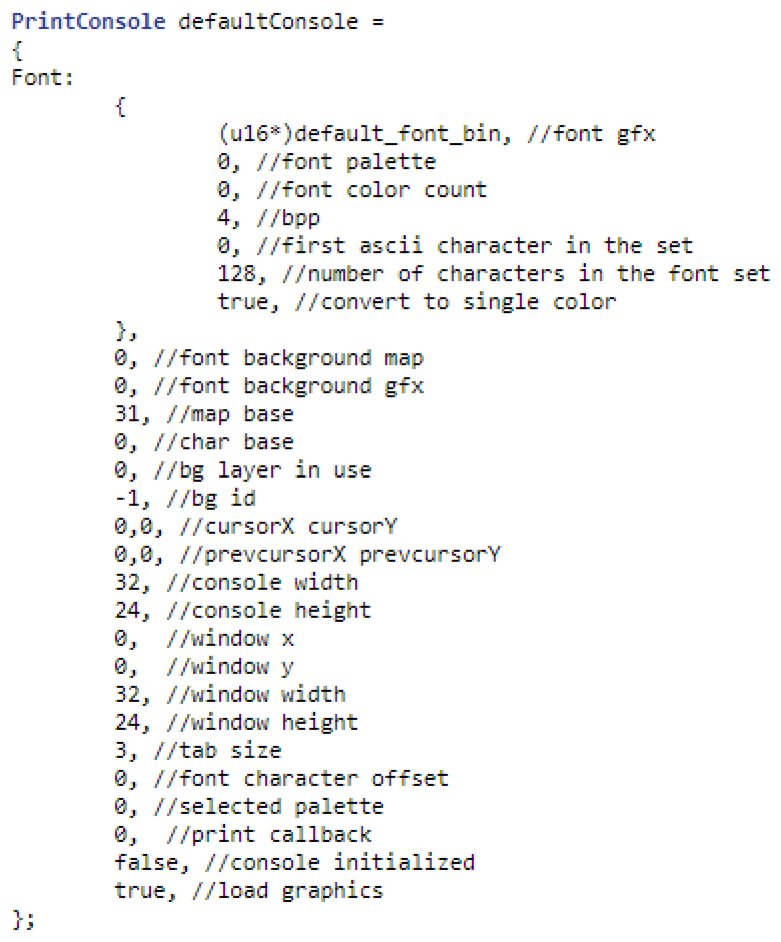
## Use of the image on the NDS video game console

### Printing and drawing text

**Exercise 1:**

```
PrintConsole* consoleDemoInit(void) {
        videoSetModeSub(MODE_0_2D);
        vramSetBankC(VRAM_C_SUB_BG);
        return consoleInit(NULL,defaultConsole.bgLayer, BgType_Text4bpp,
                                BgSize_T_256x256, defaultConsole.mapBase,
                                defaultConsole.gfxBase, false, true);
}
```

PrintConsole is object for storing configuration for console render context. It can sotre information as:



```
PrintConsole defaultConsole =
{
Font:
        {
                (u16*)default_font_bin, //font gfx
                0, //font palette
                0, //font color count
                4, //bpp
                0, //first ascii character in the set
                128, //number of characters in the font set
                true, //convert to single color
        },
        0, //font background map
        0, //font background gfx
        31, //map base
        0, //char base
        0, //bg layer in use
        -1, //bg id
        0,0, //cursorX cursorY
        0,0, //prevcursorX prevcursorY
        32, //console width
        24, //console height
        0,  //window x
        0,  //window y
        32, //window width
        24, //window height
        3, //tab size
        0, //font character offset
        0, //selected palette
        0,  //print callback
        false, //console initialized
        true, //load graphics
};
```

In the method consoleDemoInit is called method consoleInit which returns reference to PrintConsole object with the configuration based on the passed parameters. This parameters are:

| | |
|---|---|
| **console** | A pointer to the console data to initialze (if it's NULL, the default console will be used) |
| **layer** | background layer to use |
| **type** | the type of the background |
| **size** | the size of the background |
| **mapBase** | the map base |
| **tileBase** | the tile graphics base |
| **mainDisplay** | if true main engine is used, otherwise false |
| **loadGraphics** | if true the default font graphics will be loaded into the layer |

## Exercise 2:

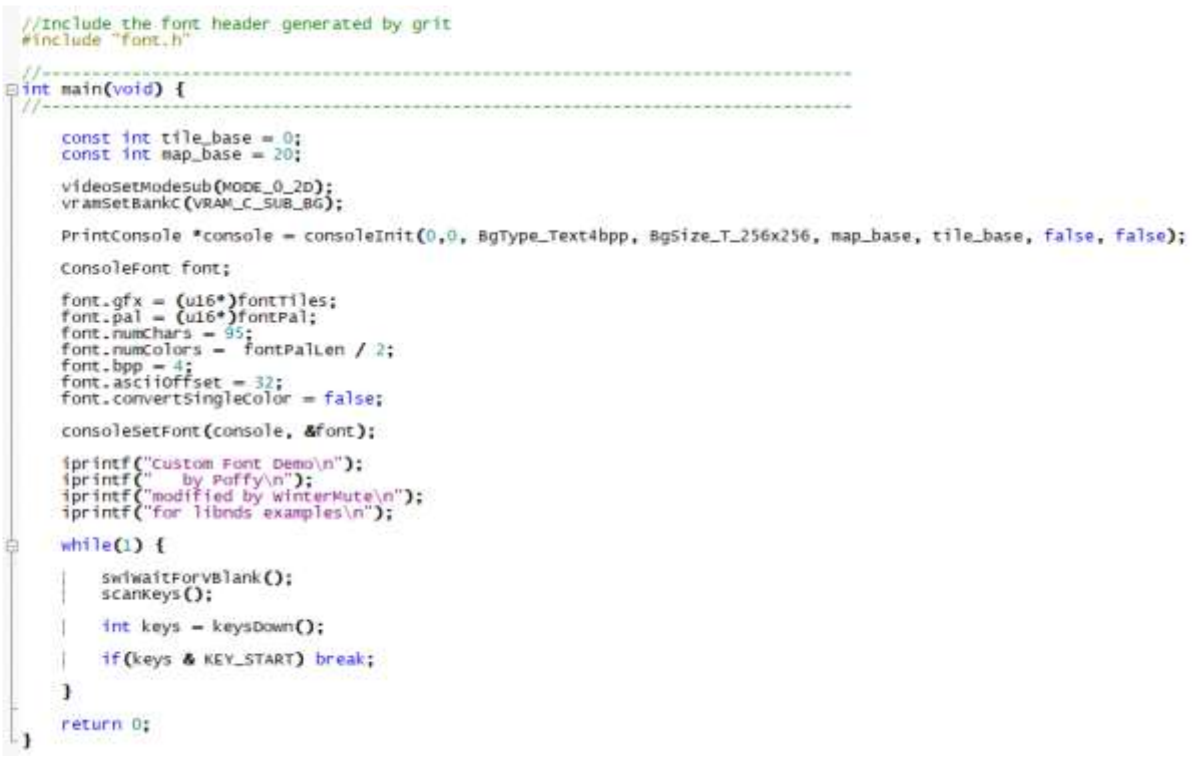
```c
//---------------------------------------------------------------------------------
int main(void) {
//---------------------------------------------------------------------------------
	touchPosition touch;

	PrintConsole topScreen;
	PrintConsole bottomScreen;

	videoSetMode(MODE_0_2D);
	videoSetModeSub(MODE_0_2D);

	vramSetBankA(VRAM_A_MAIN_BG);
	vramSetBankC(VRAM_C_SUB_BG);

	consoleInit(&topScreen, 3,BgType_Text4bpp, BgSize_T_256x256, 31, 0, true, true);
	consoleInit(&bottomScreen, 3,BgType_Text4bpp, BgSize_T_256x256, 31, 0, false, true);


	consoleSelect(&topScreen);
	iprintf("\n\n\tHello DS dev'rs\n");
	iprintf("\twww.drunkencoders.com\n");
	iprintf("\twww.devkitpro.org");

	consoleSelect(&bottomScreen);

	while(1) {

		touchRead(&touch);

		iprintf("\x1b[10;0HTouch x = %04i, %04i\n", touch.rawx, touch.px);
		iprintf("Touch y = %04i, %04i\n", touch.rawy, touch.py);

		swiWaitForVBlank();
		scanKeys();

		int keys = keysDown();

		if(keys & KEY_START) break;

	}

	return 0;
}
```

The video mode for that example is defined in methods videoSetMode and videoSetModeSub which both, set the mode as MODE_0_2D which give us 4 2D backgrounds.

The memory space is defined in methods vramSetBankA and vramSetBankC. This example is using bank A for the main background, and bank C for the second background.

Text attributes have been set by using BgType_Text4bpp and BgSize_T_256x256. The first one means that tiled background with 15 bit tile indexes without rotation or scaling possibility,, and the second sets pixel text background size as 256x256.

## Exercise 3:



```
//Include the font header generated by grit
#include "font.h"

//---------------------------------------------------------------------------------
int main(void) {
//---------------------------------------------------------------------------------

    const int tile_base = 0;
    const int map_base = 20;

    videoSetModeSub(MODE_0_2D);
    vramSetBankC(VRAM_C_SUB_BG);

    PrintConsole *console = consoleInit(0,0, BgType_Text4bpp, BgSize_T_256x256, map_base, tile_base, false, false);

    ConsoleFont font;

    font.gfx = (u16*)fontTiles;
    font.pal = (u16*)fontPal;
    font.numChars = 95;
    font.numColors =  fontPalLen / 2;
    font.bpp = 4;
    font.asciiOffset = 32;
    font.convertSingleColor = false;

    consoleSetFont(console, &font);

    iprintf("Custom Font Demo\n");
    iprintf("   by Poffy\n");
    iprintf("modified by WinterMute\n");
    iprintf("for libnds examples\n");

    while(1) {

        swiWaitForVBlank();
        scanKeys();

        int keys = keysDown();

        if(keys & KEY_START) break;

    }

    return 0;
}
```

All characters are save in the bmp file in director gfx with the name font.bmp. Below I attached part of that file. The console is using special font pallet. This information is set in consoleInit method passing false as the

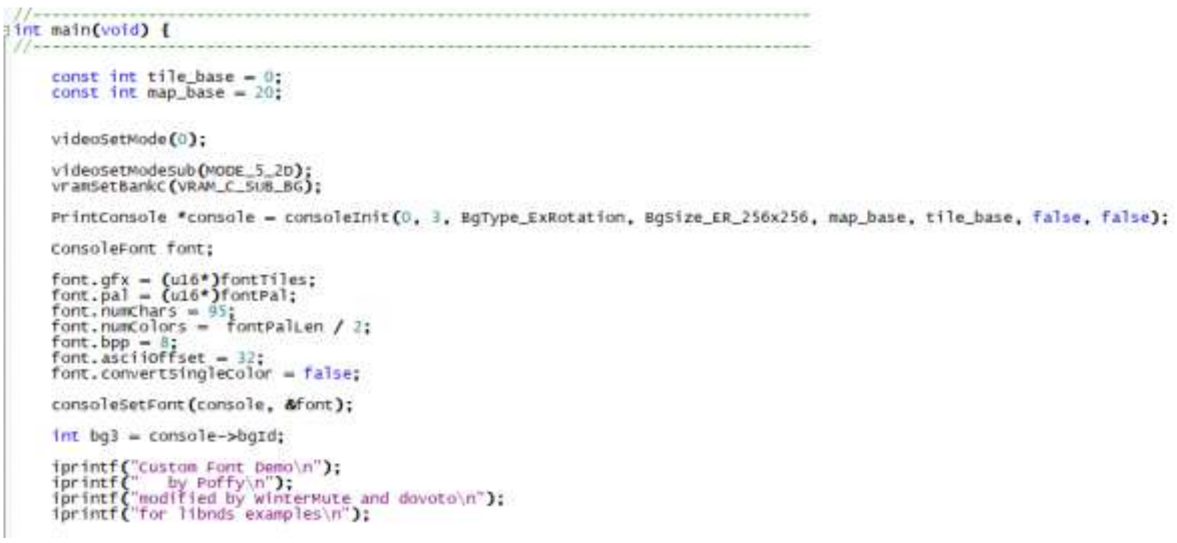
Whole file has size 8x768 px, which means that each character has size 8x8 px, and there are 96 characters.

The action to assign the image file as a font source has been done by font.gfx = (u16*)fontTiles. The fontTile variable has been defined in the header file generated by grit.

The letters starts from 32 character. In the code we can see that this information is set in font.asciiOffset.

Font also has set information about color depth in bits per pixel and in that case this value is equal 4 so it means 16-bit color depth.

## Exercise 4:



```
//---------------------------------------------------------------------------------
int main(void) {
//---------------------------------------------------------------------------------

    const int tile_base = 0;
    const int map_base = 20;


    videoSetMode(0);

    videoSetModeSub(MODE_5_2D);
    vramSetBankC(VRAM_C_SUB_BG);

    PrintConsole *console = consoleInit(0, 3, BgType_ExRotation, BgSize_ER_256x256, map_base, tile_base, false, false);

    ConsoleFont font;

    font.gfx = (u16*)fontTiles;
    font.pal = (u16*)fontPal;
    font.numChars = 95;
    font.numColors =  fontPalLen / 2;
    font.bpp = 8;
    font.asciiOffset = 32;
    font.convertSingleColor = false;

    consoleSetFont(console, &font);

    int bg3 = console->bgId;

    iprintf("Custom Font Demo\n");
    iprintf("   by Poffy\n");
    iprintf("modified by WinterMute and dovoto\n");
    iprintf("for libnds examples\n");
```

In that example the video mod which has been set is MODE_5_2D (in some previously discussed examples the mod was MODE_0_2D). The difference is that in MODE_0_2D all available backgrounds work in Text mode. In MODE_5_2D, backgrounds 2 and 3 work in extended rotation mode.

```
unsigned int angle = 0;
int scrollX = 0;
int scrollY = 0;
int scaleX = intToFixed(1,8);
int scaleY = intToFixed(1,8);

while(1) {
    scanKeys();
    u32 keys = keysHeld();

    if ( keys & KEY_START ) break;

    if ( keys & KEY_L ) angle+=64;
    if ( keys & KEY_R ) angle-=64;

    if ( keys & KEY_LEFT ) scrollX++;
    if ( keys & KEY_RIGHT ) scrollX--;
    if ( keys & KEY_UP ) scrollY++;
    if ( keys & KEY_DOWN ) scrollY--;

    if ( keys & KEY_A ) scaleX++;
    if ( keys & KEY_B ) scaleX--;

    if( keys & KEY_X ) scaleY++;
    if( keys & KEY_Y ) scaleY--;

    swiWaitForVBlank();


    bgSetRotateScale(bg3, angle, scaleX, scaleY);
    bgSetScroll(bg3, scrollX, scrollY);
    bgUpdate();
}
```

To modify the text - rotate, translate, scale, this example use simple solution. There are some dedicated variables to keep the rotate angle, position and scale factor for each dimension. Then inside the main loop, based on pressed key, this values are modified and whole background is modified using new values.

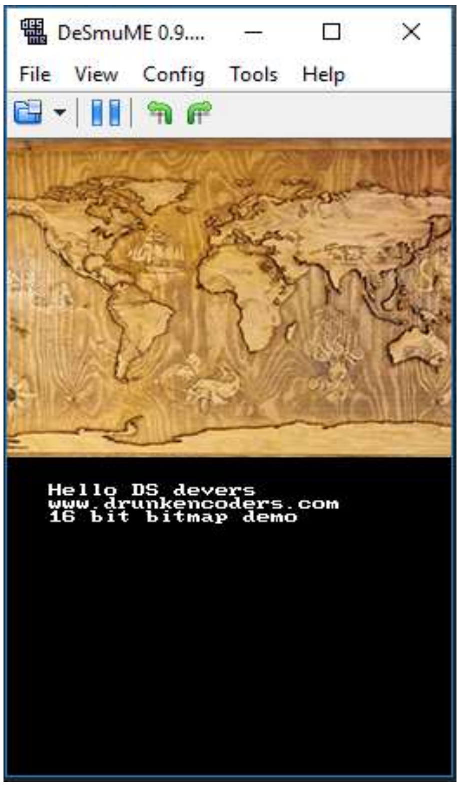
## 2D Graphic

### Exercise 5:

```
if( keys & KEY_L ) angle+=20;
if( keys & KEY_R ) angle-=20;
if( keys & KEY_LEFT ) scrollX++;
if( keys & KEY_RIGHT ) scrollX--;
if( keys & KEY_UP ) scrollY++;
if( keys & KEY_DOWN ) scrollY--;
if( keys & KEY_A ) scaleX++;
if( keys & KEY_B ) scaleX--;
if( keys & KEY_START ) rcX ++;
if( keys & KEY_SELECT ) rcY++;
if( keys & KEY_X ) scaleY++;
if( keys & KEY_Y ) scaleY--;
```

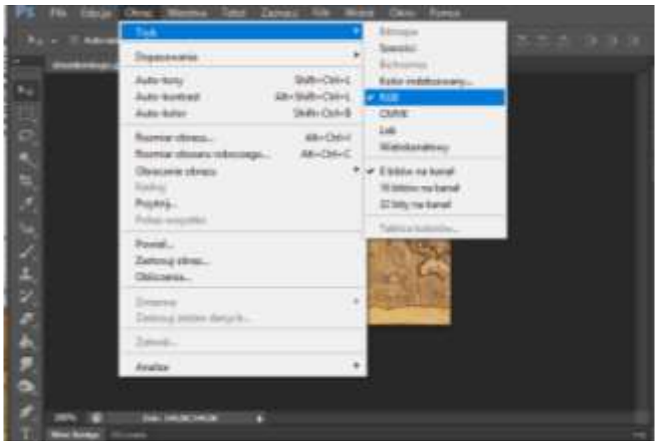
This screenshot represents all modifications of background which can be performed by pressing keys. Based on that we can rotate (keys: R, L), translate (keys: left, right, up, down), scale (keys: A, B, X, Y ) and move the root center point (keys: start, select).

### Exercise 6:



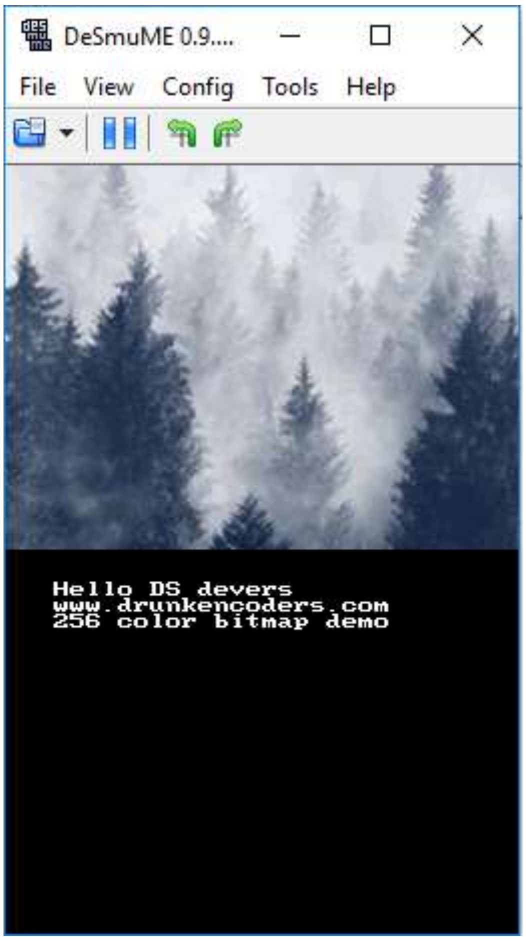
```
Hello DS devers
www.drunkencoders.com
16 bit bitmap demo
```

This exercise was about replacing background with the new one created in graphic editor.



## Exercise 7:

In this exercise I had to try replace bg image but for 256-bit color bitmap.



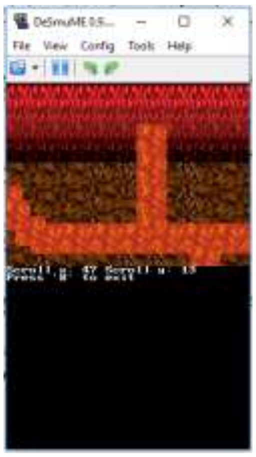
```
Hello DS devers
www.drunkencoders.com
256 color bitmap demo
```

The difference which I noticed in that exercise is that it was impossible to finish make until I removed all other files from the data directory (previously I renamed original image background to name backup.png and in this exercise I did the same, but the make script wanted special rules for the file backup.png).

The another difference which I noticed is in the code in the way, how the background is initialize.

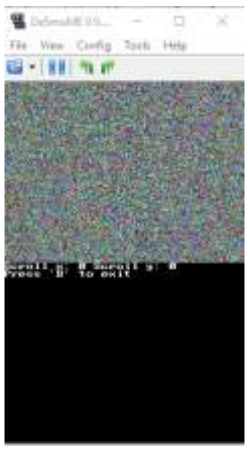
Previously, we used method called decompress, int that exercise we used method dmaCopy which copy directly from source to destination point the content. And in that case we copied the reference to bitmap, and the color palette for that bitmap.

## Exercise 8:



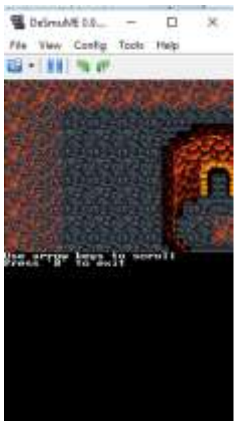
```
Scroll x: 47 Scroll y: 13
Press 'B' to exit
```

Basic->Extended Rotation 256x256



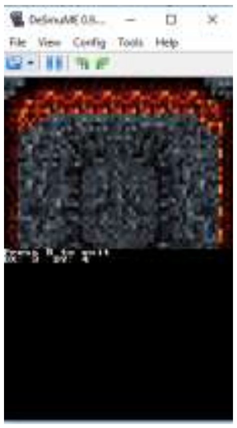
```
Scroll x: 0 Scroll y: 0
Press 'B' to exit
```

Bitmap -> Bitmap 256 color 512x512



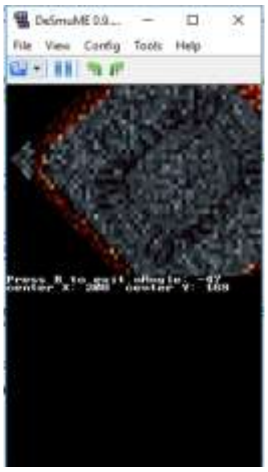
```
Use arrow keys to scroll
Press 'B' to exit
```

Scrolling -> Horizontal scrolling < ExRot >



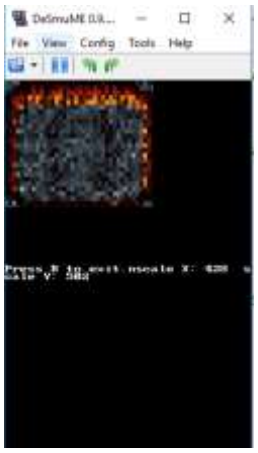
```
Press B to exit
DX: 5 DY: 4
```

Advanced -> Mozaic



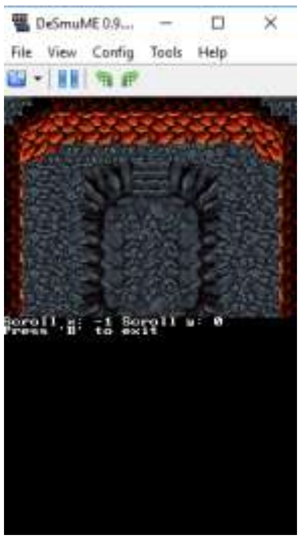
```
Press B to exit nAngle: -47
center X: 208 center Y: 168
```

Advanced -> Rotation



```
Press B to exit nscale X: 628 s
cale Y: 384
```

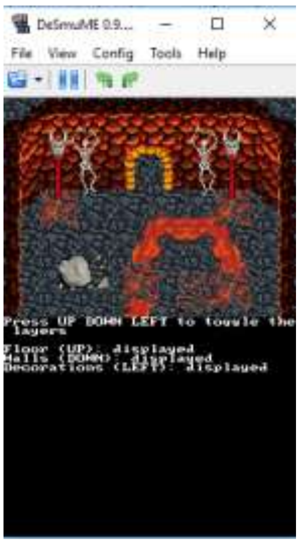
Advanced-> Scaling



```
Scroll x: -1 Scroll y: 0
Press 'B' to exit
```

Advanced -> Extended Pallete

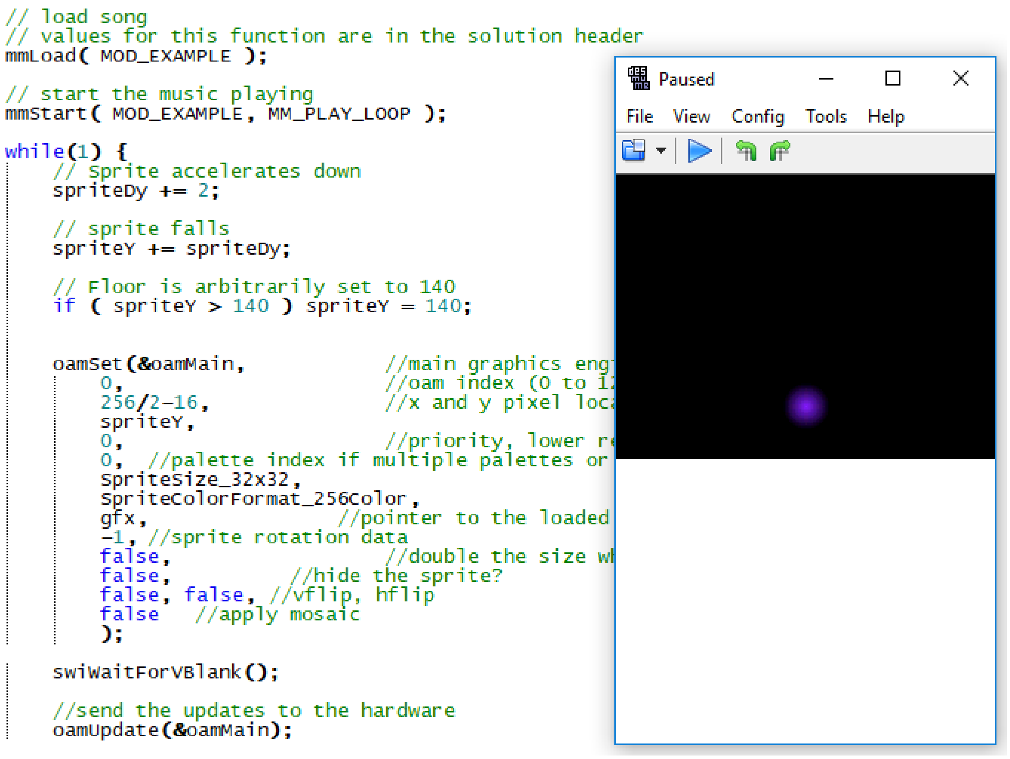
Advanced -> Multiple Text Layers



```
Press UP DOWN LEFT to toggle the
 layers

Floor (UP): displayed
Walls (DOWN): displayed
Decorations (LEFT): displayed
```

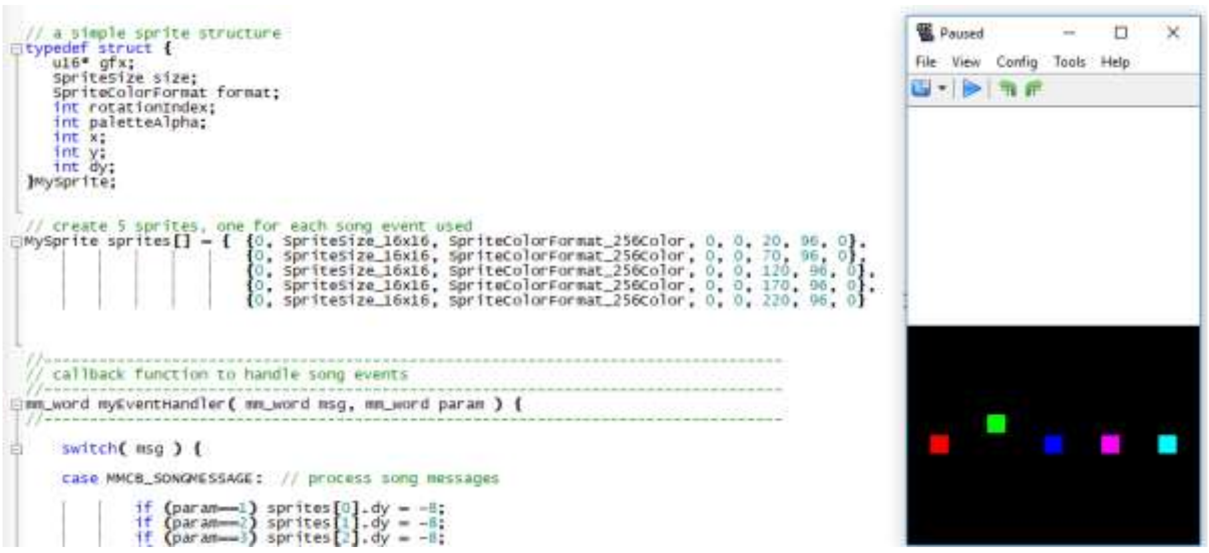
## Exercise 9:

In the example song_event_example the graphic is loaded from memory, where was inserted at the beginning. Then, during the main program loop, there are modified only position attributes and the graphic is displayed again but in new location.



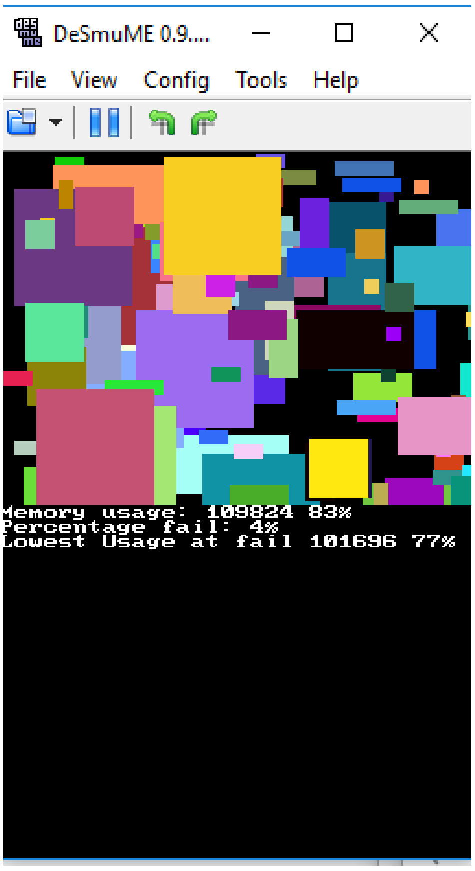
```
// load song
// values for this function are in the solution header
mmLoad( MOD_EXAMPLE );

// start the music playing
mmStart( MOD_EXAMPLE, MM_PLAY_LOOP );

while(1) {
    // Sprite accelerates down
    spriteDy += 2;

    // sprite falls
    spriteY += spriteDy;

    // Floor is arbitrarily set to 140
    if ( spriteY > 140 ) spriteY = 140;


    oamSet(&oamMain,             //main graphics eng
        0,                       //oam index (0 to 12
        256/2-16,                //x and y pixel loca
        spriteY,
        0,                       //priority, lower re
        0,  //palette index if multiple palettes or
        SpriteSize_32x32,
        SpriteColorFormat_256Color,
        gfx,                   //pointer to the loaded
        -1, //sprite rotation data
        false,                   //double the size wh
        false,              //hide the sprite?
        false, false, //vflip, hflip
        false   //apply mosaic
        );

    swiWaitForVBlank();

    //send the updates to the hardware
    oamUpdate(&oamMain);
```

In the example song_event_example2 all graphics are created in the code (defining the array "sprites") and during the main program loop, they are displayed in different position, depends on the music.



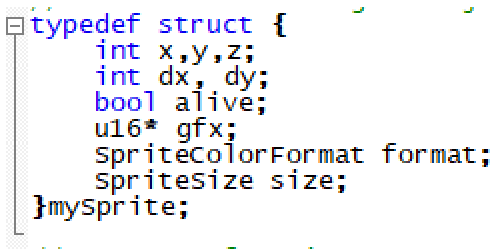
```
// a simple sprite structure
typedef struct {
    u16* gfx;
    SpriteSize size;
    SpriteColorFormat format;
    int rotationIndex;
    int paletteAlpha;
    int x;
    int y;
    int dy;
}MySprite;


// create 5 sprites, one for each song event used
MySprite sprites[] = {  {0, SpriteSize_16x16, SpriteColorFormat_256Color, 0, 0, 20, 96, 0},
                        {0, SpriteSize_16x16, SpriteColorFormat_256Color, 0, 0, 70, 96, 0},
                        {0, SpriteSize_16x16, SpriteColorFormat_256Color, 0, 0, 120, 96, 0},
                        {0, SpriteSize_16x16, SpriteColorFormat_256Color, 0, 0, 170, 96, 0},
                        {0, SpriteSize_16x16, SpriteColorFormat_256Color, 0, 0, 220, 96, 0}


//---------------------------------------------------------------------------------
// callback function to handle song events
//---------------------------------------------------------------------------------
mm_word myEventHandler( mm_word msg, mm_word param ) {
//---------------------------------------------------------------------------------

    switch( msg ) {

    case MMCB_SONGMESSAGE:  // process song messages

            if (param==1) sprites[0].dy = -8;
            if (param==2) sprites[1].dy = -8;
            if (param==3) sprites[2].dy = -8;
```

## Exercise 10:



```
Memory usage: 109824 83%
Percentage fail: 4%
Lowest Usage at fail 101696 77%
```

In that example was created structure to keep information about sprites.



```
typedef struct {
    int x,y,z;
    int dx, dy;
    bool alive;
    u16* gfx;
    SpriteColorFormat format;
    SpriteSize size;
}mySprite;
```

When each sprite is created there is also allocated space in graphic memory for it by calling method oamAllocateGfx and analogously when they are killed, the memory is released by calling method oamFreeGfx.

During main loop each sprite object is updated, in fact, the memory is updated based on new values from each sprite object. This is done by calling method oamSet (at this moment the memory is only prepared for updating)

Finally at the end of oop the memory is updated by calling method oamUpdate.
So, there were some methods with oam prefix. Let's see their specifications.

| | | | |
|---|---|---|---|
| **u16* oamAllocateGfx** | **(** | **OamState *** | ***oam,*** |
| | | **SpriteSize** | ***size,*** |
| | | **SpriteColorFormat** | ***colorFormat )*** |

| | |
|---|---|
| *oam* | must be: &oamMain or &oamSub |

| | |
|---|---|
| *size* | the size of the sprite to allocate |
| *colorFormat* | the color format of the sprite |

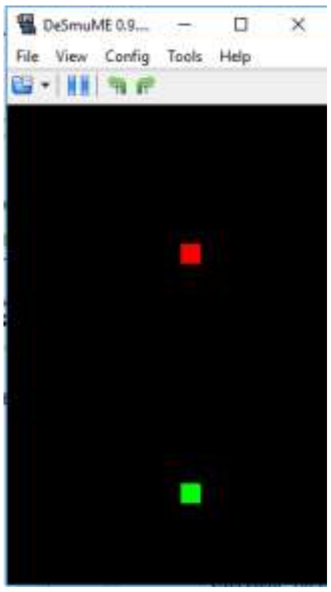
```
void oamFreeGfx   (  OamState *     oam,
                     const void *   gfxOffset  )
```

| | |
|---|---|
| *oam* | must be: &oamMain or &oamSub |
| *gfxOffset* | a vram offset obtained from oamAllocateGfx |

```
void oamSet    (  OamState *           oam,
                  int                  id,
                  int                  x,
                  int                  y,
                  int                  priority,
                  int                  palette_alpha,
                  SpriteSize           size,
                  SpriteColorFormat    format,
                  const void *         gfxOffset,
                  int                  affineIndex,
                  bool                 sizeDouble,
                  bool                 hide,
                  bool                 hflip,
                  bool        vflip,
                  bool        mosaic )
```

| | |
|---|---|
| *oam* | must be: &oamMain or &oamSub |
| *id* | the oam number to be set [0 - 127] |
| *x* | the x location of the sprite in pixels |
| *y* | the y location of the sprite in pixels |
| *priority* | The sprite priority (0 to 3) |
| *palette_alpha* | the palette number for 4bpp and 8bpp (extended palette mode), or the alpha value for bitmap sprites (bitmap sprites must specify a value > 0 to display) [0-15] |
| *size* | the size of the sprite |
| *format* | the color format of the sprite |
| *gfxOffset* | the video memory address of the sprite graphics (not an offset) |
| *affineIndex* | affine index to use (if < 0 or > 31 the sprite will be unrotated) |
| *sizeDouble* | if affineIndex >= 0 this will be used to double the sprite size for rotation |
| *hide* | if non zero (true) the sprite will be hidden |
| *vflip* | flip the sprite vertically |
| *hflip* | flip the sprite horizontally |
| *mosaic* | if true mosaic will be applied to the sprite |

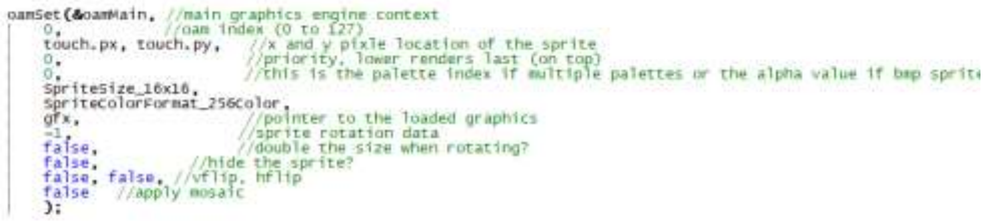
## Exercise 11:



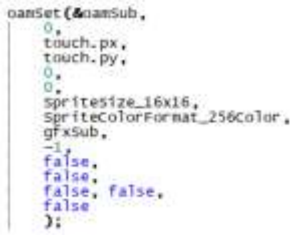
In this example we are using only two different graphics gfx. Both of them are stored in the graphic memory (at the beggining of the program the space was allocated for them by calling oamAllocateGfx).

Then during the main program loop, for each sprite the memory update is prepared by calling method oamSet with new location and later, the memory is updated by caling oamUpdated.

Each sprite is using different graphic engine context (oamMain, oamSub) so they are independent.

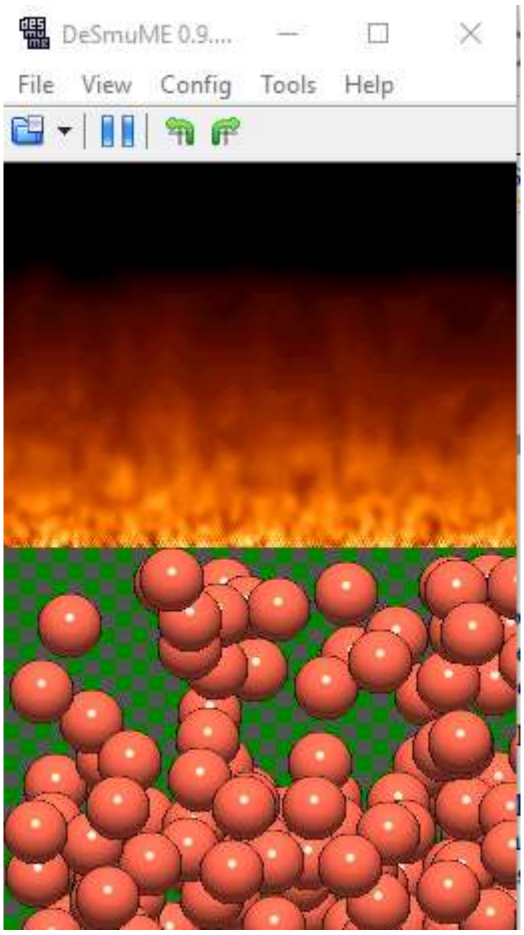


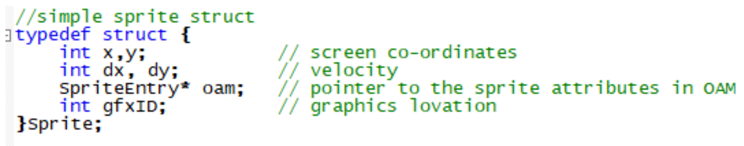
```
oamSet(&oamMain, //main graphics engine context
    0,              //oam index (0 to 127)
    touch.px, touch.py,   //x and y pixle location of the sprite
    0,                    //priority, lower renders last (on top)
    0,                    //this is the palette index if multiple palettes or the alpha value if bmp sprit
    SpriteSize_16x16,
    SpriteColorFormat_256Color,
    gfx,                  //pointer to the loaded graphics
    -1,                  //sprite rotation data
    false,               //double the size when rotating?
    false,          //hide the sprite?
    false, false, //vflip, hflip
    false   //apply mosaic
    );
```



```
oamSet(&oamSub,
    0,
    touch.px,
    touch.py,
    0,
    0,
    SpriteSize_16x16,
    SpriteColorFormat_256Color,
    gfxSub,
    -1,
    false,
    false,
    false, false,
    false
    );
```

**Exercise 12:**





```
//simple sprite struct
typedef struct {
    int x,y;            // screen co-ordinates
    int dx, dy;         // velocity
    SpriteEntry* oam;   // pointer to the sprite attributes in OAM
    int gfxID;          // graphics lovation
}Sprite;
```

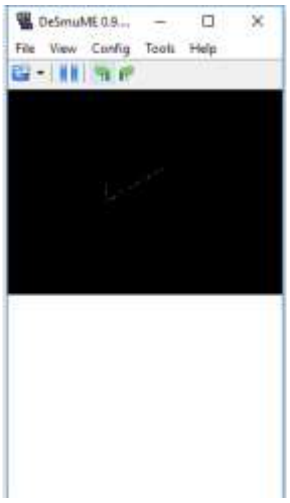
In that example we have two different ways to generate the view. In first case, the lower screen is based on sprites objects based on structures presented above. Each sprite load the graphic from the file, and they are stored in the graphic memory. During the main loop the properties of each sprites are modified and the memory is updated.

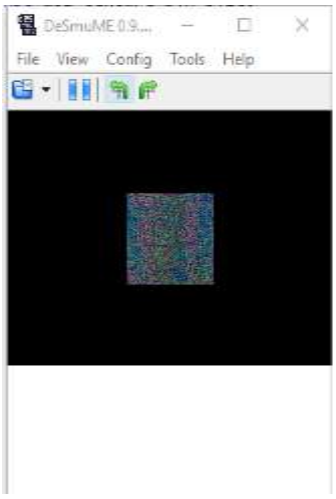
This is very similar to previous examples.

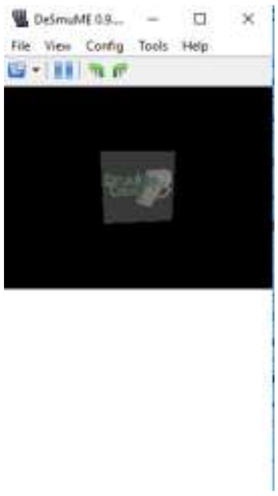
Something new happened with the second view. The fire effect is the graphic completely generated runtime. The idea is simple - first we generate the pixels at the bottom of the screen, based on the previously defined fire color palette, and then from bottom to top, each row has calculated color. Whole graphic is represented by array modified by code.

## 3D Graphic

### Exercise 13:





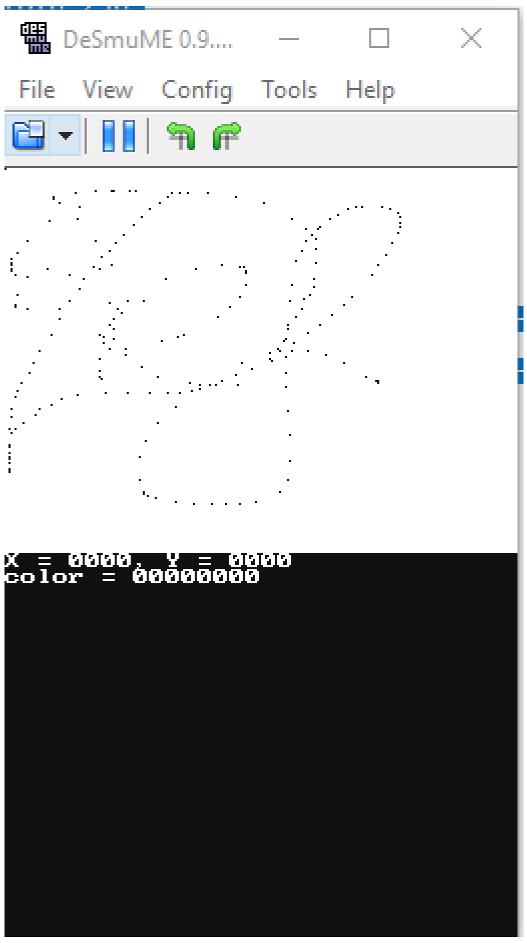


In this exercise I explored the usability of 3D graphics. In both cases I changed the source file (texture.bin, drunkenlogo.pcx) but probably I used wrong converter for each file so the graphics didn’t display correctly – in the first case I only can see some pixels on the border, In second case whole are is set by “random” pixels.

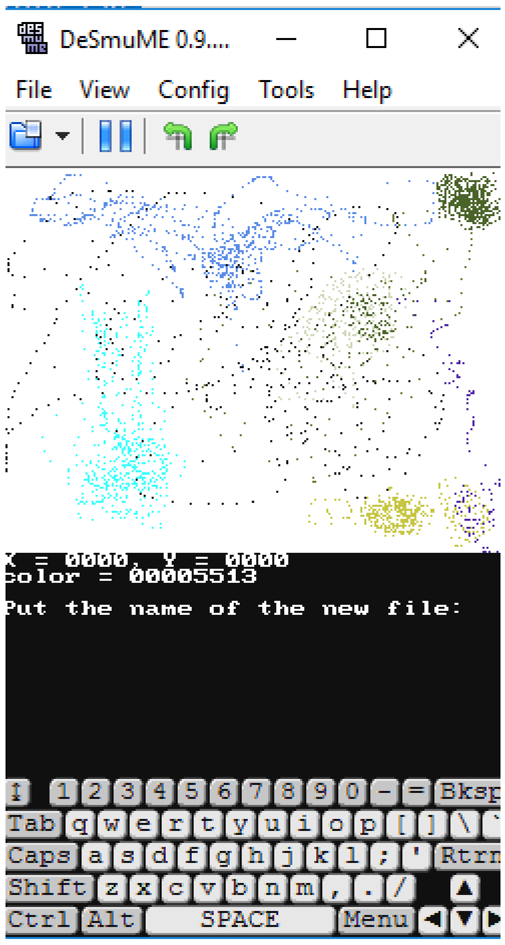
## Exercise: graphic

*( portfolio/projects/lab3 )*

```
#include <nds.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define WIDTH 256
#define HEIGHT 256

//function prototypes
void clearCanvas();
u16 generateColor();
void saveImage();
void initBackground();
void drawPixel(int,int,u16);
void savePPMImage(char*);

//----------------------------------------------------------------
------
int main(void) {
//----------------------------------------------------------------
------
      int keys;
      int x,y;
      u16 color;
      touchPosition touch;

       initBackground();
      consoleDemoInit();
      clearCanvas();
      color = RGB15(0,0,0);

      //main loop
      while(1) {
            scanKeys();
            touchRead(&touch);
            keys = keysDown();
            x = touch.px;
            y = touch.py;
            drawPixel(x,y,color);
            iprintf("X = %04i, Y = %04i\n", x, y);
            iprintf("color = %08i", color);
            if(keys & KEY_A) color = generateColor();
            if(keys & KEY_B) clearCanvas();
            if(keys & KEY_SELECT) saveImage();
            if(keys & KEY_START) break;
            swiWaitForVBlank();
            consoleClear();
      }
      return 0;
}

void OnKeyPressed(int key) {
   if(key > 0)
      iprintf("%c", key);
}

void saveImage() {
      //get the name
      char fileName[256];


      iprintf("\n\nPut the name of the new file:\n");
      Keyboard *kbd = keyboardDemoInit();
      kbd->OnKeyPressed = OnKeyPressed;
      scanf("%s", fileName);

      iprintf("\nNew file name%s", fileName);
      //save in the root as ppm file
      savePPMImage(fileName);



}
//-----------------------------------------------------------------
------
void clearCanvas() {
      int i;
      for(i = 0; i < WIDTH*HEIGHT; i++)
            VRAM_A[i] = RGB15(31,31,31);
}

void drawPixel(int x,int  y,u16 color) {
      int position = y*HEIGHT+x;
      VRAM_A[position] = color;
}

u16 generateColor() {
      int r = (abs(rand()) % 32);
      int g = (abs(rand()) % 32);
      int b = (abs(rand()) % 32);
      return RGB15(r,g,b);
}

void initBackground() {
      videoSetMode(MODE_FB0);
      vramSetBankA(VRAM_A_LCD);
}

void savePPMImage(char* fileName) {
      FILE *fp;
      fp = fopen("file.ppm", "w+");
      //PPM header
      fprintf(fp, "P6\n %d %d 255\n", WIDTH, HEIGHT);
      //PPM content
      for (int i = 0; i < WIDTH * HEIGHT; i++) {
            fprintf(fp, "%c", VRAM_A[i]);
      }
      fclose(fp);
}
```



```
X = 0000, Y = 0000
color = 00000000
```



```
X = 0000, Y = 0000
color = 00005513
Put the name of the new file:
```

## Synthesis by tone generation

### Exercise 1:

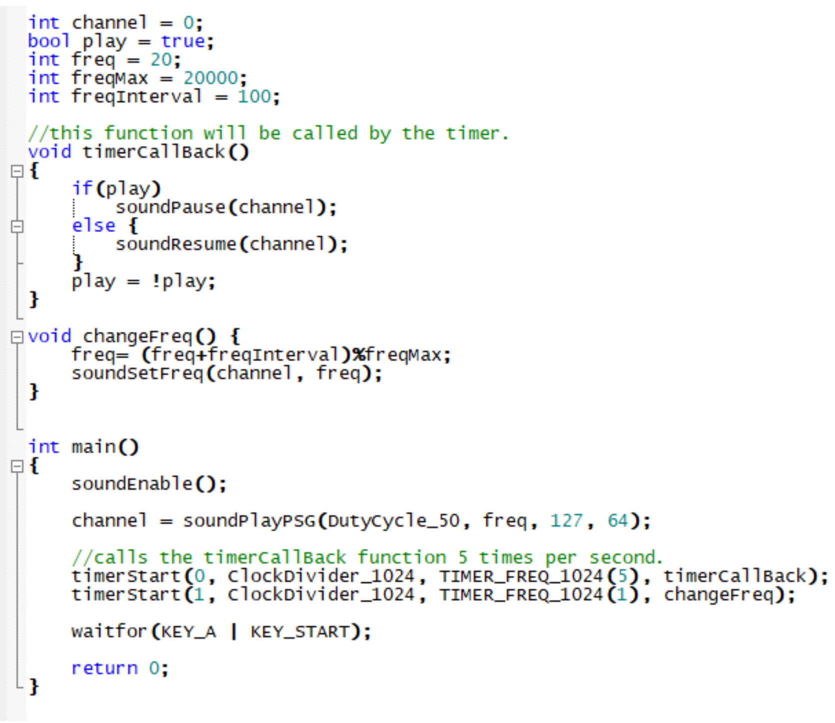
```
int soundPlayPSG   (  DutyCycle   cycle,

                      u16         freq,


                      u8          volume,


                      u8          pan )
```

| | |
|---|---|
| *cycle* | The DutyCycle of the sound wave |
| *freq* | The frequency in Hz of the sample |
| *volume* | The channel volume. 0 to 127 (min to max) |
| *pan* | The channel pan 0 to 127 (left to right with 64 being centered) |

## Exercise 2:

In that exercise I had to modified a code of given example to being able to changing frequency in range from 20Hz to 20kHz.

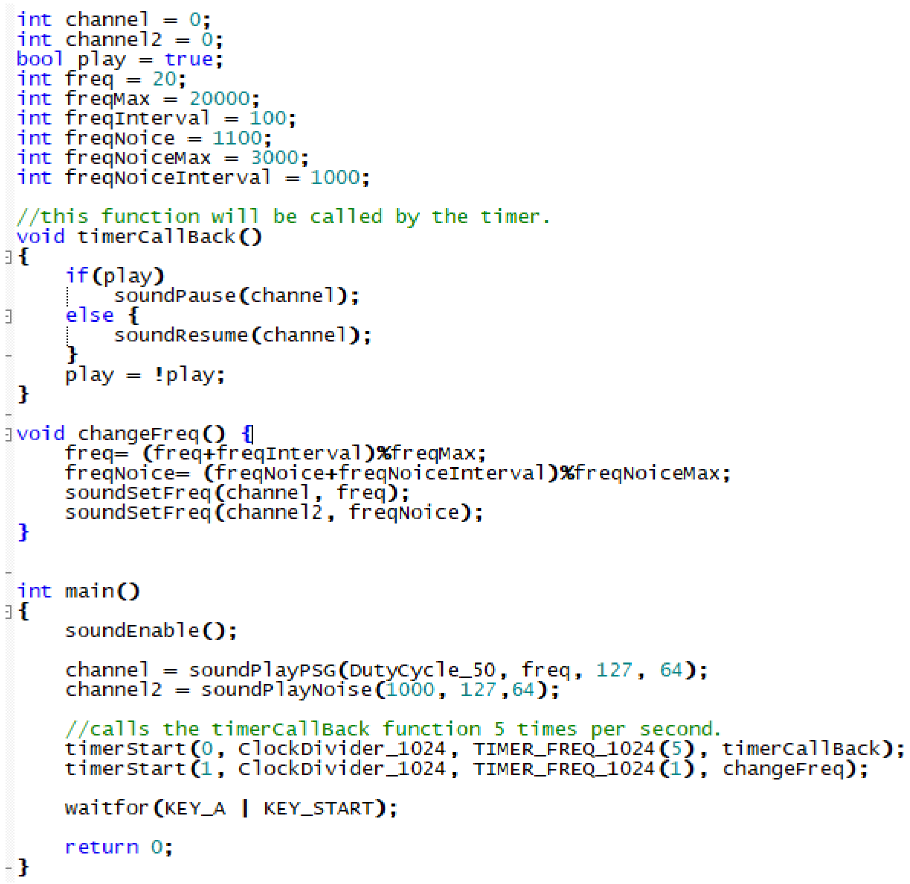
I did it by adding second timer which once per second increment frequency by set Interval and modifies the frequency by calling method soundSetFreq() as a id passing channel.



```
int channel = 0;
bool play = true;
int freq = 20;
int freqMax = 20000;
int freqInterval = 100;

//this function will be called by the timer.
void timerCallBack()
{
    if(play)
        soundPause(channel);
    else {
        soundResume(channel);
    }
    play = !play;
}

void changeFreq() {
    freq= (freq+freqInterval)%freqMax;
    soundSetFreq(channel, freq);
}

int main()
{
    soundEnable();

    channel = soundPlayPSG(DutyCycle_50, freq, 127, 64);

    //calls the timerCallBack function 5 times per second.
    timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5), timerCallBack);
    timerStart(1, ClockDivider_1024, TIMER_FREQ_1024(1), changeFreq);

    waitfor(KEY_A | KEY_START);

    return 0;
}
```

## Exercise 3:

In that exercise I used the same technique as before (frequency manipulation) to hear the differences.
Paramethers in method soundPlayNoise are almost the same as in soundPlayPSG - we can define the frequency, volume in range from 0 to 127 and the pan from 0 to 127 and the 64 value is in the center.



```
int channel = 0;
int channel2 = 0;
bool play = true;
int freq = 20;
int freqMax = 20000;
int freqInterval = 100;
int freqNoice = 1100;
int freqNoiceMax = 3000;
int freqNoiceInterval = 1000;

//this function will be called by the timer.
void timerCallBack()
{
    if(play)
        soundPause(channel);
    else {
        soundResume(channel);
    }
    play = !play;
}

void changeFreq() {
    freq= (freq+freqInterval)%freqMax;
    freqNoice= (freqNoice+freqNoiceInterval)%freqNoiceMax;
    soundSetFreq(channel, freq);
    soundSetFreq(channel2, freqNoice);
}

int main()
{
    soundEnable();

    channel = soundPlayPSG(DutyCycle_50, freq, 127, 64);
    channel2 = soundPlayNoise(1000, 127,64);

    //calls the timerCallBack function 5 times per second.
    timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5), timerCallBack);
    timerStart(1, ClockDivider_1024, TIMER_FREQ_1024(1), changeFreq);

    waitfor(KEY_A | KEY_START);

    return 0;
}
```

## Microphone:

### Exercise 4:

In that exercise the main method response for recording sound is

```
int soundMicRecord    (   void *          buffer,
                          u32             bufferLength,
                          MicFormat       format,
                          int             freq,
                          MicCallback     callback
                      )
```

| | |
|---|---|
| *buffer* | A pointer to the start of the double buffer |
| *bufferLength* | The length of the buffer in bytes (both halfs of the double buffer) |
| *format* | Microphone can record in 8 or 12 bit format. 12 bit is shifted up to 16 bit pcm |
| *freq* | The sample frequency |
| *callback* | This will be called every time the buffer is full or half full |

For playing the recorded sounds was used method:

```
int soundPlaySample   (   const void *    data,
                          SoundFormat     format,
                          u32             dataSize,
                          u16             freq,
                          u8              volume,
                          u8            pan,
                          bool          loop,
                          u16           loopPoint
                      )
```

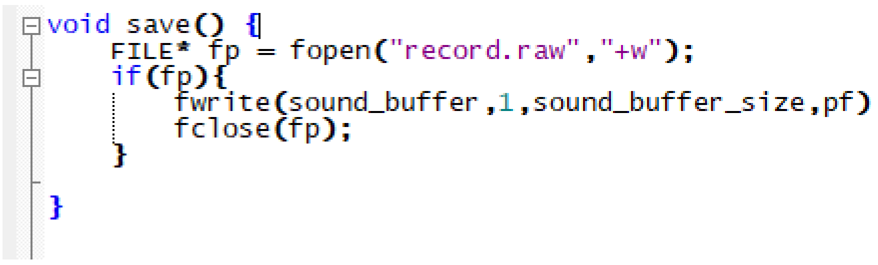
| | |
|---|---|
| *data* | A pointer to the sound data |
| *format* | The format of the data (only 16-bit and 8-bit pcm and ADPCM formats are supported by this function) |
| *dataSize* | The size in bytes of the sound data |
| *freq* | The frequency in Hz of the sample |
| *volume* | The channel volume. 0 to 127 (min to max) |
| *pan* | The channel pan 0 to 127 (left to right with 64 being centered) |
| *loop* | If true, the sample will loop playing once then repeating starting at the offset stored in loopPoint |
| *loopPoint* | The offset for the sample loop to restart when repeating |

The variable sound_buffer_size is initialized in followed way:

sound_buffer_size = 8 * k * s
where k is the factor which allows us to use 8bit, 16bit, 32bit audio
s is the factor which describe how long audio in miliseconds we want to have

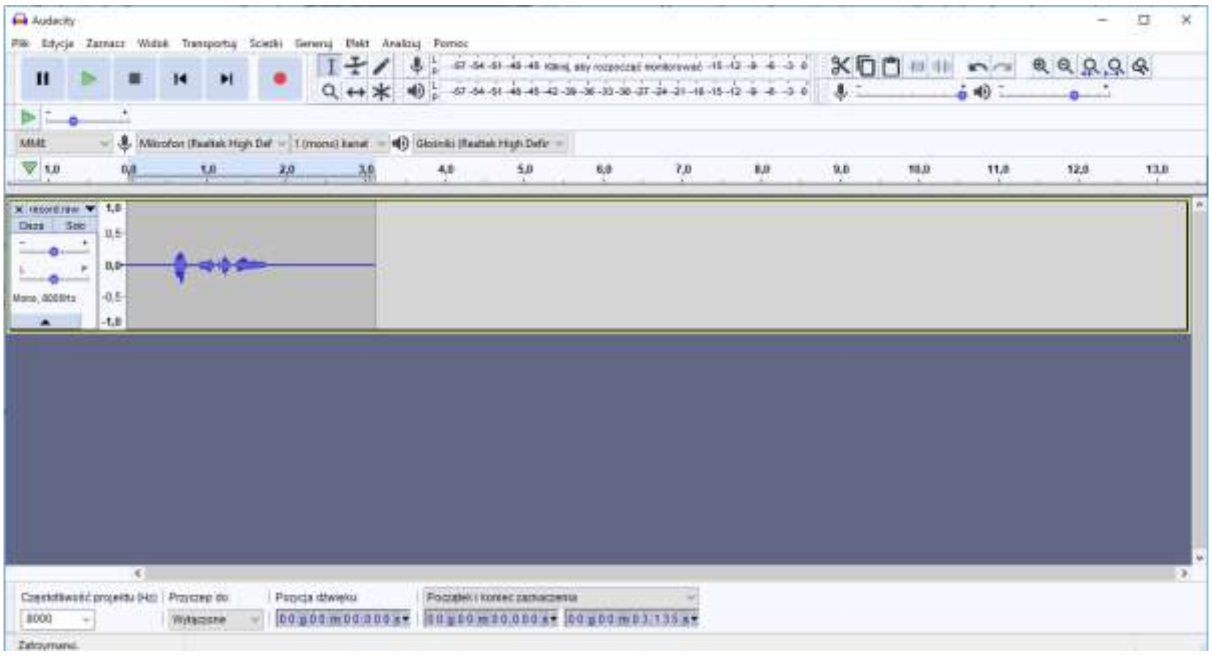
## Exercise 5:

To save recorder sound, I created method save().



```
void save() {
    FILE* fp = fopen("record.raw","+w");
    if(fp){
        fwrite(sound_buffer,1,sound_buffer_size,pf)
        fclose(fp);
    }

}
```

## Exercise 6:


Audacity
Plik Edycja Zaznacz Widok Transportuj Ścieżki Generuj Efekt Analizuj Pomoc
MME
Mikrofon (Realtek High Def
1 (mono) kanał
Głośniki (Realtek High Defir
record.raw
Cisza Solo
Mono, 8000Hz
Częstotliwość projektu (Hz)
8000
Przyciąganie do
Wyłączone
Pozycja dźwięku
Początek i koniec zaznaczenia
Zatrzymano.

## Maxmod

### Exercise 7:

First to add functionality of the maxmod library first it has to be included to the project.
#include <maxmod9.h>

To start working with that library the maxmod has to by initialized by calling method
void mmInitDefault( char* soundbank );
Soundbank is fileName of the sound file which can be created with the Maxmod utility.

### Exercise 8:

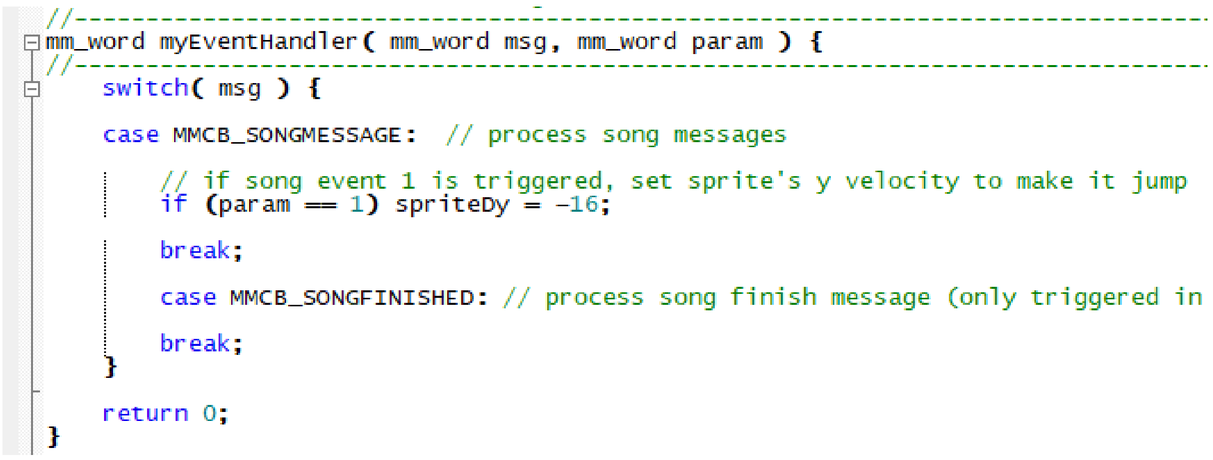
To prepare our Makefile to working with maxmode and allow building music directly from to music file we have to add following rule to the Makefile

```
#-----------------------------------------------------------------
------# rule to build soundbank from music files
#-----------------------------------------------------------------
------
soundbank.bin soundbank.h : $(AUDIOFILES)
#-----------------------------------------------------------------
------
     @mmutil $^ -d -osoundbank.bin -hsoundbank.h
```

## Event synchronization:

### Exercise 9:

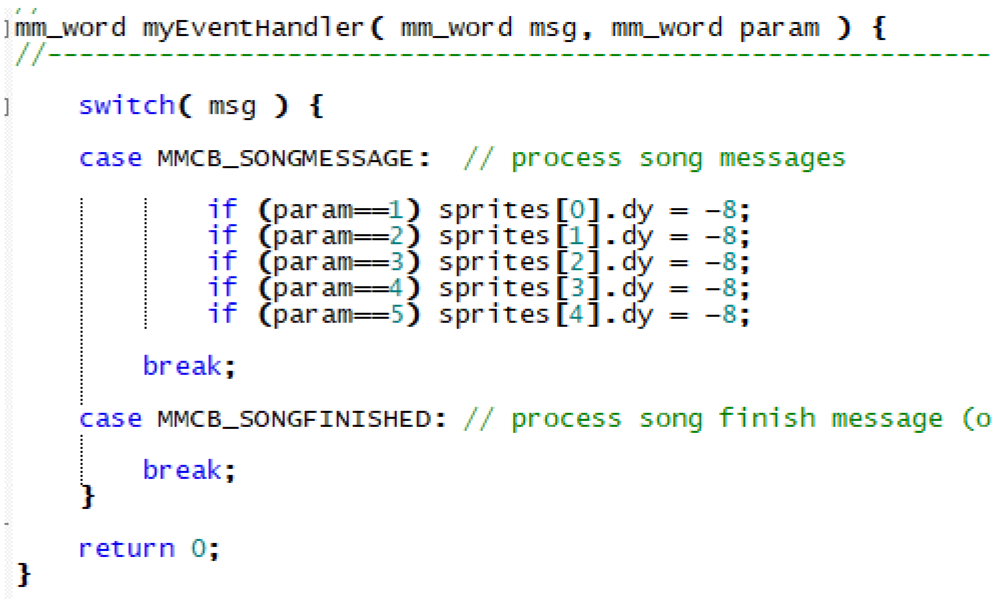
mmSetEventHandler( myEventHandler );



```
//---------------------------------------------------------------------------------
mm_word myEventHandler( mm_word msg, mm_word param ) {
//---------------------------------------------------------------------------------
    switch( msg ) {

    case MMCB_SONGMESSAGE:  // process song messages

        // if song event 1 is triggered, set sprite's y velocity to make it jump
        if (param == 1) spriteDy = -16;

        break;

        case MMCB_SONGFINISHED: // process song finish message (only triggered in

        break;
    }

    return 0;
}
```

This part of the example is response to manipulating the sprite localization. First as a EventHandler our method has been set, which depends on the message value (in that case MMCB_SONGMESSAGE ) manipulate the sprite location, In Maxmode there are only 2 types of message and both has been used in example above.

## Exercise 10:

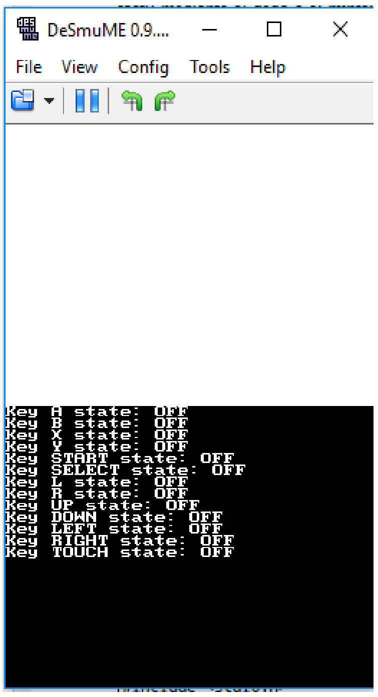


```
mm_word myEventHandler( mm_word msg, mm_word param ) {
//-----------------------------------------------------------

    switch( msg ) {

    case MMCB_SONGMESSAGE:  // process song messages

            if (param==1) sprites[0].dy = -8;
            if (param==2) sprites[1].dy = -8;
            if (param==3) sprites[2].dy = -8;
            if (param==4) sprites[3].dy = -8;
            if (param==5) sprites[4].dy = -8;

        break;

    case MMCB_SONGFINISHED: // process song finish message (o

        break;
    }

    return 0;
}
```

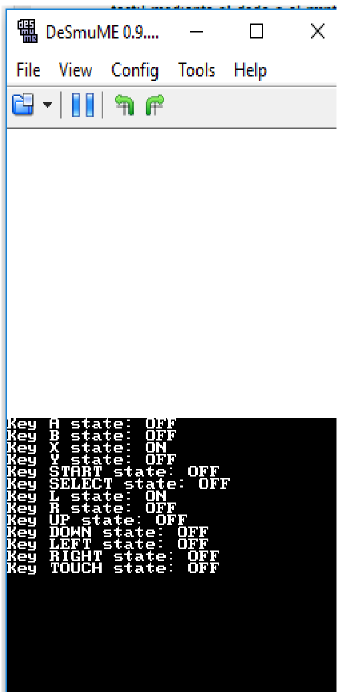
In that example to distinguish which sprite should be moved has been used attribute param delivered by callback function. In the Maxmode documentation there is no information about what exactly the param might be.

## Input and output of data using the text

### Exercise 1:



```
Key A state: OFF
Key B state: OFF
Key X state: OFF
Key Y state: OFF
Key START state: OFF
Key SELECT state: OFF
Key L state: OFF
Key R state: OFF
Key UP state: OFF
Key DOWN state: OFF
Key LEFT state: OFF
Key RIGHT state: OFF
Key TOUCH state: OFF
```



```
Key A state: OFF
Key B state: OFF
Key X state: ON
Key Y state: OFF
Key START state: OFF
Key SELECT state: OFF
Key L state: ON
Key R state: OFF
Key UP state: OFF
Key DOWN state: OFF
Key LEFT state: OFF
Key RIGHT state: OFF
Key TOUCH state: OFF
```

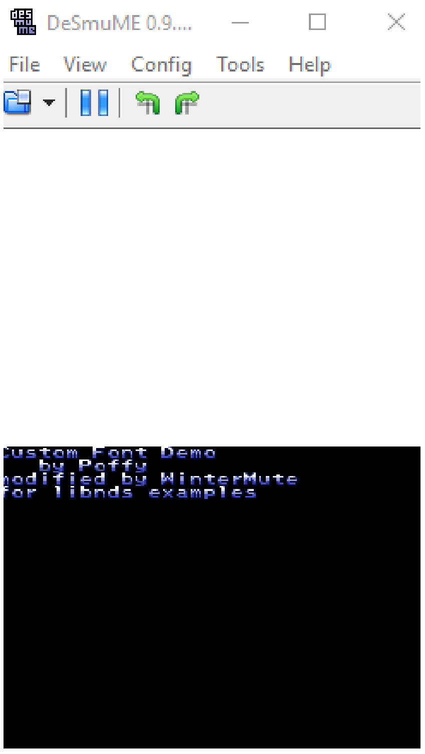
That exercise was really easy. I attached below only part of the code to present the idea how the program works, but whole function is so long and repeatedly.

```
int held;
char ON[] = "ON";
char OFF[] = "OFF";
char* keystate_a = OFF;
char* keystate_b = OFF;
char* keystate_x = OFF;
char* keystate_y = OFF;
char* keystate_start = OFF;
char* keystate_select = OFF;
char* keystate_stop = OFF;
char* keystate_l = OFF;
char* keystate_r = OFF;
char* keystate_top = OFF;
char* keystate_bottom = OFF;
char* keystate_left = OFF;
char* keystate_right = OFF;
```

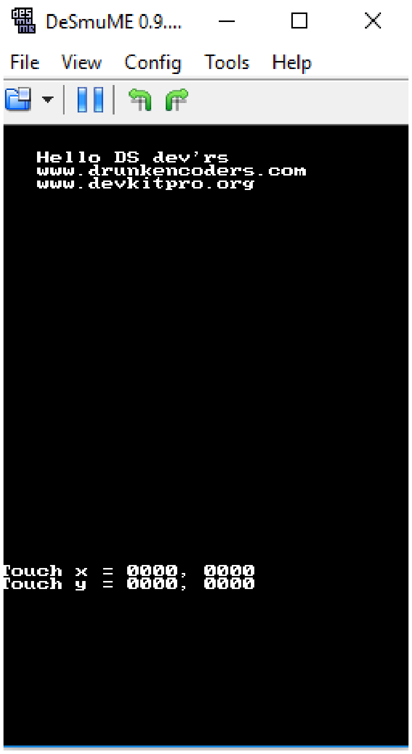
```
iprintf("Key A state: %s\n", keystate_a);
iprintf("Key B state: %s\n", keystate_b);
iprintf("Key X state: %s\n", keystate_x);
iprintf("Key Y state: %s\n", keystate_y);
iprintf("Key START state: %s\n", keystate_start);
iprintf("Key SELECT state: %s\n", keystate_select);
iprintf("Key L state: %s\n", keystate_l);
iprintf("Key R state: %s\n", keystate_r);
iprintf("Key UP state: %s\n", keystate_top);
```

```
consoleDemoInit();
while(1)
{
    scanKeys();
    held = keysHeld();
    if( held & KEY_A)
        keystate_a = ON;
    else
        keystate_a = OFF;

    if( held & KEY_B)
        keystate_b = ON;
    else
        keystate_b = OFF;

    if( held & KEY_X)
        keystate_x = ON;
    else
        keystate_x = OFF;

    if( held & KEY_Y)
        keystate_y = ON;
    else
        keystate_y = OFF;
```

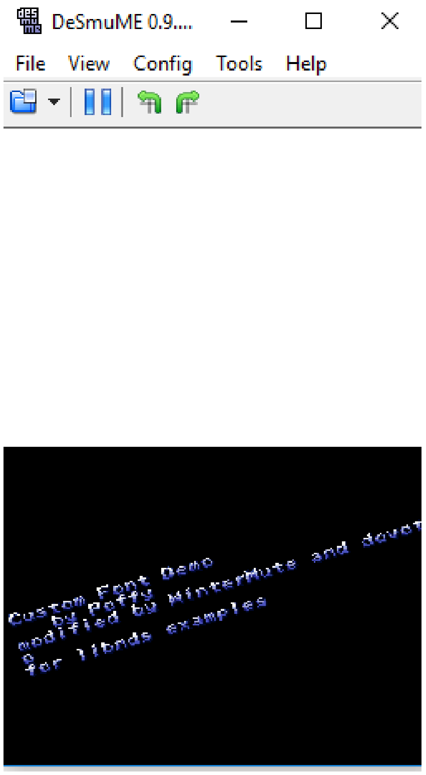
## Exercise 2:



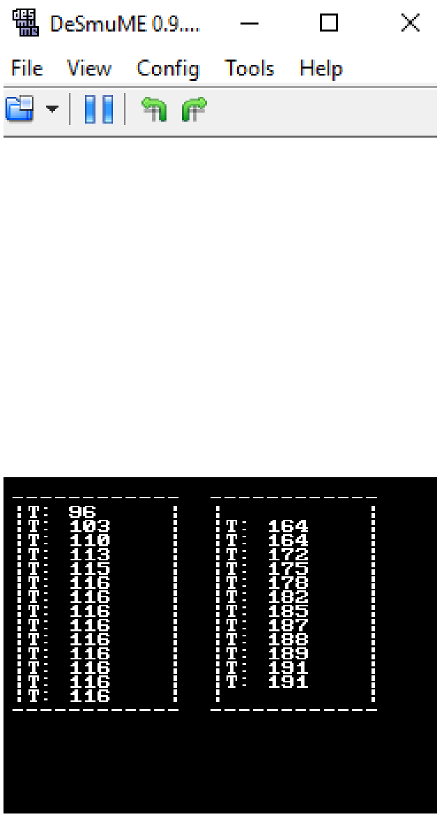
```
ustom Font Demo
  by Poffy
odified by WinterMute
or libnds examples
```



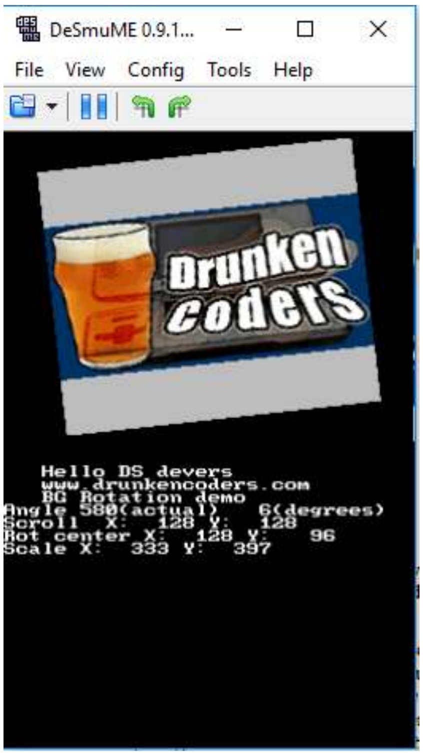
```
    Hello DS dev'rs
    www.drunkencoders.com
    www.devkitpro.org

ouch x = 0000, 0000
ouch y = 0000, 0000
```



```
Custom Font Demo
  by Poffy
modified by WinterMute and devot
for libnds examples
```



```
------------    ------------
|T:  96    |    |          |
|T:  103   |    |T:  164   |
|T:  110   |    |T:  164   |
|T:  113   |    |T:  172   |
|T:  115   |    |T:  175   |
|T:  116   |    |T:  178   |
|T:  116   |    |T:  182   |
|T:  116   |    |T:  185   |
|T:  116   |    |T:  187   |
|T:  116   |    |T:  188   |
|T:  116   |    |T:  189   |
|T:  116   |    |T:  191   |
|T:  116   |    |T:  191   |
|T:  116   |    |          |
------------    ------------
```



```
   Hello DS devers
   www.drunkencoders.com
   BG Rotation demo
Angle 580(actual)    6(degrees)
Scroll  X:   128 Y:  128
Rot center X:  128 Y:    96
Scale X:  333 Y:  397
```

All of that examples presents different functionalities, some of them I analyzed during previous exercises. We can learn from them how to:

- split the screen and use both parts simultaneously,
- use font defined in the graphic file
- Use console to display  simple information (mostly properties)
- How to transform the background using scale, translation, rotation
- How to read the information from touchpad inbuilded to the screen.