ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Last name:

……

First name:

……

Section:

……

Cours de 3ème année,

Section d'Electricité

Systèmes Embarqués Microprogrammés

Date and place: *November 28th, 2014; ELG-124*

Duration: *1h45*

Grade:

……

Mid-term Exam

IMPORTANT NOTES:

- The skeleton project for the exam can be downloaded from the Moodle Site under the link ***"midterm_code"***. This project is similar to the ones provided during the practical sessions. In the source files there are placeholders to implement each exercise of the exam.
- The exercises must be implemented in the skeleton project and completed in the indicated order.
- Follow carefully all the instructions given in the current document and in the comments of the source code files of the skeleton project.
- A label "//…TO COMPLETE EXERCISE X" indicates where to write code for exercise X, X=1…5.
- The implemented exercises must work correctly in the NDS simulator to be considered as correctly done. However, the source code will be also evaluated after the exam.
- A compressed file of the project including the implemented code must be submitted by means of the different forms available on the Moodle Site after finishing each exercise.
- The presence in the exam counts as 1 point.

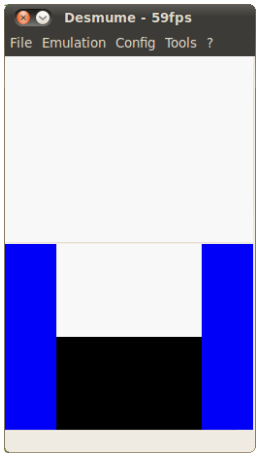
PROJECT DEFINITION:

The project consists in 5 exercises to implement a mini-game in which a random mosaic of white and black blocks will be shown and the player must choose the predominant color within 1.5 seconds by touching the corresponding region in the touchscreen. All the logic of the project is already implemented and only few placeholders need to be completed.

EXERCISE 1 (0.5 points)	Time:	Works: YES / NO	Teach. Sign.:

The bottom screen will be used to show the controls of the game in which the player will need to choose the predominant color. This will be done by means of an image (**bottom.png**) that you can download from the Moodle site. For this exercise the missing parts of the functions in the file *graphics_sub.c*, which are prototyped in the header *graphics_sub.h*, must be completed following the next steps:

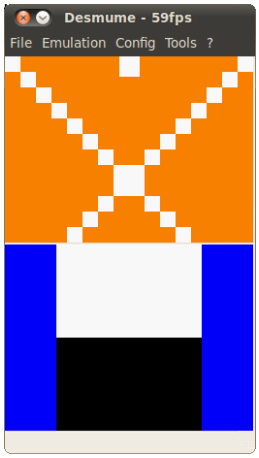
NOTE: the macros related to the SUB engine are followed by the suffix "_SUB" (i.e. BG_TILE_RAM_SUB, BG_PALETTE_SUB, BGCTRL_SUB, etc…).



- Uncomment the line related to exercise 1 in the **main()** function of the file *main.c*.
- Complete the function **configureGraphics_Sub()** in the file *graphics_sub.c* following the given comments to configure the SUB engine in mode 5 and activate the background BG2.
- Download the image **bottom.png** into the "data" folder of the project and create the configuration grit file in order to obtain the bitmap and the corresponding palette (therefore using pixels of 8bits length)
- Complete the function **configBG2_Sub()** in the file *graphics_sub.c* following the given comments to configure the background correctly and transfer the image information to the corresponding locations in memory.
- Compile the project and correct the possible errors.

EXERCISE 2 (1.5 points)	Time:	Works: YES / NO	Teach. Sign.:

The upper screen will be used to show a mosaic of white and black blocks. In order to do so, a function called **fillRectangle(…)** is called by the logic of the game to display the different blocks in the upper screen. In this exercise, the main engine must be configured to work in extended rotoscale mode using 16bit colors (i.e. emulating framebuffer mode) and the mentioned function must be implemented. Complete the following steps:



- Uncomment the line related to exercise 2 in the **main()** function of the file *main.c*.
- Complete the function **configureGraphics_Main()** in the file *graphics_main.c* following the given comments to configure the MAIN engine in mode 5 and activate the background BG2.
- Complete the function **configBG2_Main()** in the file *graphics_main.c* following the given comments to configure the background correctly.
- Implement the function **fillRectangle(…)** in the file graphics_main.c, which is prototyped and explained in detail in the file graphics_main.h.
- Compile the project and correct the possible errors. The upper screen of the simulator must display the blocks depicted on the image on the right. No extra squares should appear and the color must be as depicted.

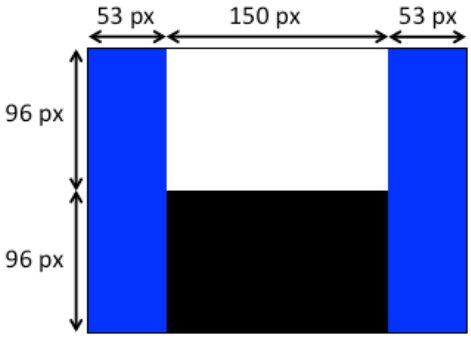
EXERCISE 3 (1 point)	Time:	Works: YES / NO	Teach. Sign.:

The mosaic on the upper screen must be visible for a short period of time (1.5 seconds). If the player does not choose a color within this time the game finishes and the player looses. In order to do so, a timer will be used. It is required to complete the functions in the source file *timer_management.c*, which are prototyped in the header file *timer_management.h*. Complete the following steps:

- Uncomment the line related to exercise 3 in the **main()** function of the file *main.c.*
- Complete the function **configureTimer()** in the file *timer_management.c* following the given comments in order to configure a timer to trigger an interrupt every 100 ms.
  NOTE: do not call **irqInit()** since the touchscreen will be used in the next exercise.
- Complete the Interrupt Service Routine of the timer **timerISR()** in the file *timer_management.c* so that after 1.5 seconds it disables the timer interrupt and ends the game by calling the function **playerLoses()**, which is already implemented within the project and prototyped in the header file *game.h*.
- Compile the project, correct the possible errors and check that after 1.5 seconds the mosaic of blocks disappears and the screen is filled with the red color (which is the effect of calling the function **playerLoses()**).

EXERCISE 4 (1 point)	Time:	Works: YES / NO	Teach. Sign.:

The player must choose a color by means of touching either the black area or the white one on the bottom screen. A touch outside those areas must not have any effect. Moreover, the game can be restarted at any time by pressing the START key. To do so, the function **exercise_4()** in the source file *main.c* must be completed. The size of the areas can be seen on the right. Follow the next steps:



- Uncomment the line related to exercise 4 in the **main()** function of the file *main.c.*
- Complete the function **exercise_4()** in the file *main.c* following carefully the comments given within the function.
- Compile the project, correct the possible errors and check that the game is working as expected. If the correct color is chosen, the upper screen will be filled in green. Otherwise, it will be fully filled in red as in the case of the timeout (exercise 3).

NOTE: The upper screen will show a random mosaic of black and white blocks.

EXERCISE 5 (1 point)	Time:	Works: YES / NO	Teach. Sign.:

The upper screen is divided into 16x12 blocks that compose the mosaic as shown in Fig.1. As the upper screen of the Nintendo DS is a matrix of 256x192 pixels, each of the blocks consists on a square of 16x16 pixels. In this exercise, a superposed background (BG0) in **tiled mode** will be used to display a partially transparent grid transforming the underlying squared blocks in "rounded" dots, as depicted in Fig. 2. Therefore, each block will be covered by 4 tiles, as shown in the diagram of Fig. 3.

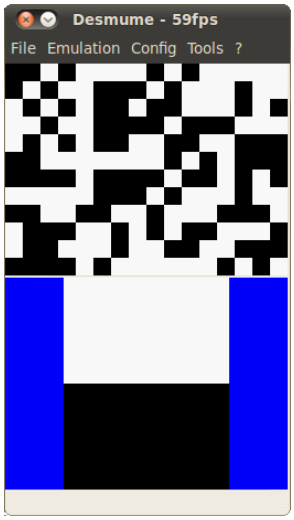


Fig 1: Only background 2 is active and the 16x12 mosaic is displayed on it

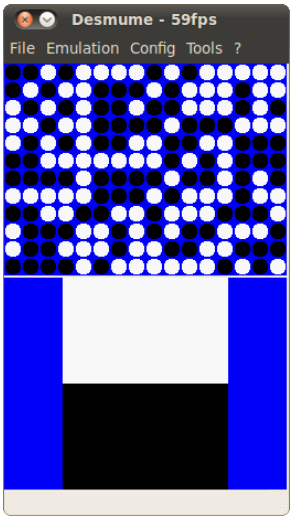


Fig 2: Background 0 is also active and superposed. It displays a partially transparent grid.

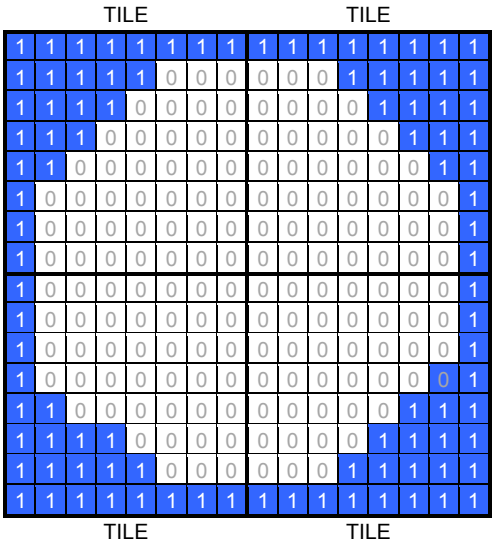


Fig 3: Each of the blocks in background 2 is covered by 4 tiles of background 0. The pixels of color 0 (white in this image) are transparent

In order to complete the exercise follow the next steps:

- Modify the function **configureGraphics_Main()** in the file *graphics_main.c*, such that background 0 is active (do not deactivate the background used in exercise 2).
- In order to leave space at the beginning of the VRAM bank for the tiles and the map of background 0, change the configuration of background 2 in the function **configBG2_Main()** implemented in exercise 2 so that the BMP_BASE number 1 is used instead of BMP_BASE 0. Modify the implementation of the function **fillRectangle(…)** accordingly. Check that everything is working as before starting this exercise (i.e. blocks are painted correctly as in Fig. 1)

HINT: Use the macro BG_BMP_RAM(1) to access the buffer where pixels are stored when painting the rectangles in the function fillRectangle(…)

- Declare and fill the necessary tile(s) according to the diagram depicted above in the file *graphics_main.c*.
- Complete the function **configBG0_Main()** following the given comments in the source files to configure the background BG0 in tiled mode using a 32x32 tile grid and 256 colors. Transfer the tile(s) defined in the previous step, assign the corresponding colors to the used component(s) of the palette and create the map to generate the semitransparent grid.

HINT: In order to not use overlapping memory regions, use the TILE_BASE 0 and one MAP_BASE between the 1 and the 7

- Compile the project, correct the possible errors and check that the grid is correctly depicted transforming the underlying blocks into rounded dots.

PLEASE MAKE SURE YOU HAVE SIGNED AT THE END OF THE EXAM AND ALL THE TEACHER SIGNATURES IN ALL THE EXERCISES SLOTS ARE COMPLETED BEFORE YOU LEAVE THE EXAM, OTHERWISE THE EXERCISES MAY NOT BE COUNTED FOR THE FINAL MARK	
FINAL TIME:	Student Sign.: