EPFL

Last name:

……

First name:

……

Section:

……

Cours de 3ème année,

Section d'Electricité

Systèmes Embarqués Microprogrammés

Date and place: *November 29th, 2024; MED 2 2524*

Duration: *1h45minutes (from 14h15 to 16h00)*

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 exam exercise.
- 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 the comments of the source code.
- 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 also be evaluated after the exam and must be uploaded in Moodle.
- After finishing each exercise, a compressed file of the project, including the implemented code, must be submitted using the different forms available on the Moodle Site.
- The presence in the exam counts as 1 point.

PROJECT DEFINITION:

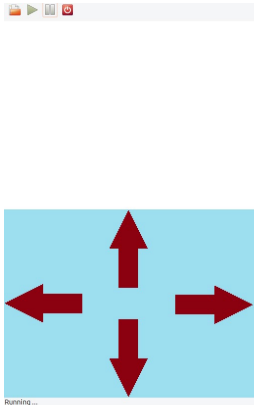
The project consists of 5 exercises to implement a mini-game in which we move a character on the main screen using the touchpad arrows. Moreover, the player must press *START* or *A* within 3 seconds after launching the game to continue playing. Finally, we will implement a portal to teleport the character. The project's logic has already been implemented, and only a few placeholders (indicated in the provided skeleton) must be completed.

EXERCISE 1 (0.75 points)	Time:	Works: YES / NO	Teach. Sign.:

The bottom screen will show the game's controls, namely the directional arrows for the character movement. This will be done using an image (arrows.png) already provided under the folder *data*. For this exercise, the missing parts of the functions in the file *graphics.c*, which are prototyped in the header *graphics.h*, must be completed following the next steps:

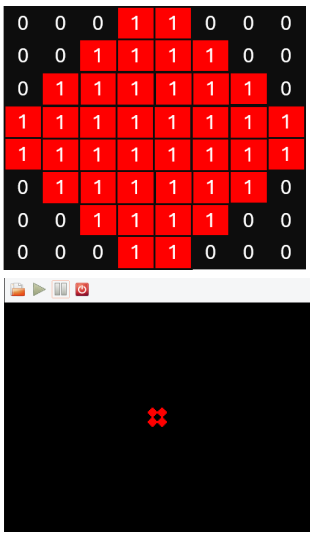
- Complete the function **configureGraphics_Sub()** in the file *graphics.c* following the comments to configure the SUB engine in mode 5 and activate the background BG2.
- Create the configuration grit file inside the *data* folder to obtain the bitmap and the corresponding palette (therefore using pixels of 8bits length)
- Complete the function **configBG2_Sub()** in the file *graphics.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.

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…).



EXERCISE 2 (1 point)	Time:	Works: YES / NO (PC: Y / N)	Teach. Sign.:

The upper screen will be used to show the main character. To do so, the main engine must be configured to work in tiled mode, the tiles should be manually designed and mapped, and the character should be placed in the middle of the screen. Complete the following steps:

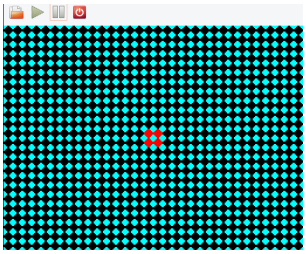


- Complete the function **configureGraphics_Main()** in the file *graphics.c* following the comments to configure the MAIN engine in mode 5 and activate the background BG0.
- Complete the function **configBG0_Main()** in the file *graphics.c* following the given comments to configure the background correctly.
- Create the main character's tile as a red diamond complemented by transparent pixels (top image). Draw the main character using four identical tiles placed in the center of the main screen while filling the rest with empty tiles (as provided).
- Compile the project and correct the possible errors. The upper screen of the simulator must only display the main character in the center (bottom image). The rest of the screen should appear black.

EXERCISE 3 (1.25 points)	Time:	Works: YES / NO (BD: Y / N)	Teach. Sign.:

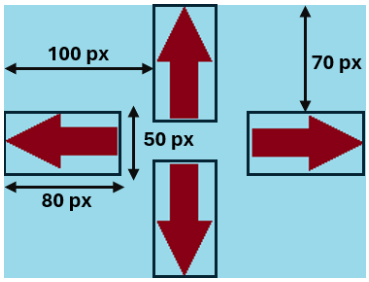
The black background on the upper screen must be visible for a short period of time (3 seconds). If the player does not press *START* or *A* within this time, the game finishes and the player loses. A timer will be used to do so. Completing the functions in the source file *timer.c* is required. Perform the following steps:

- Complete the function **initTimer()** in the file *timer.c* following the given comments to configure a timer to trigger an interrupt every 200 ms. For maximum points, choose the divider that offers the best resolution for the indicated time interval. Then, associate the timer with **timerISR()** and enable the interrupts.
  NOTE: do not call **irqInit()** since the touchscreen will be used later.
- Complete the Interrupt Service Routine of the timer **timerISR()** in the file *timer.c* so that after 3 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*.
- Complete the code in the **main()** function inside the while loop to call the function Initgame() in case the *START* or *A* key were pressed.
  NOTE: In the simulator, X is bound to *A* and ENTER to *START by default.*
- Compile the project and correct any possible errors. Check that after 3 seconds without pressing the START or A key, the upper screen becomes red, as shown in the top image (effect of calling playerLoses()). Check if the START or A key is pressed within 3 seconds, and the upper screen must transform, as shown in the bottom image below (effect of calling **gameInit()**).



EXERCISE 4 (1.25 points)	Time:	Works: YES / NO (SB: Y / N)	Teach. Sign.:

The player can move the character around by touching the arrows on the bottom screen. To do so, you must complete the code for scanning the touch position in the source file *main.c* and the logic inside the functions **movePlayer<direction>()** in *game.c*. Follow the next steps:



- Complete the code inside the while loop in the file *main.c* following carefully the comments given within the function. The size of the touch areas of the arrows can be seen on the right. Assume the boxes are 80x50px and touch the screen edges.
  NOTE: We do not require a pixel-perfect touch precision.
- Complete the code inside the functions **movePlayerUp(), movePlayerDown(), movePlayerLeft(), movePlayerRight().** Follow carefully the comments to move the character correctly (one tile displacement at each arrow press).
  HINT: Use the given variables playerX1, playerX2, playerY1, and playerY2 where (X1, Y1) and (X2, Y2) are the coordinates of the top-left and bottom-right player's tiles, respectively.
- For maximum points, ensure that the character remains within the screen borders.
- Compile the project and correct the possible errors. Ensure that the character is smoothly moving around.

EXERCISE 5 (0.75 points)	Time:	P1: Y / N, P2: Y / N, TP: Y / N	Teach. Sign.:

In this exercise, a superposed background (BG2) in **extended rotoscale mode** will display a portal on the right edge of the upper screen. The portal transports the character to the left edge of the screen when the character attempts to pass through it. Follow the next steps:

- Modify the function **configureGraphics_Main()** in the file *graphics.c* to activate background 2 (**do not deactivate background 0**).
- Complete the function **configBG2_Main()** in the file *graphics.c* following the instructive comments. **Emulate the framebuffer mode** to draw a magenta column (portal) with a width equal to 1 tile on the right edge of the screen, as shown in **Figure 1**. The magenta is defined in *colors.h.*
  HINT: Use the macro BG_BMP_BASE() wisely to configure BGCTRL_2 and avoid overlapping the map of this background with the tiles and tile map of background 0 (set in exercise 2).
- Uncomment the definition of *PORTAL_ACTIVE* in *game.h*.
- Modify the priority of BG 2 to appear on top of BG 0, as in **Figure 2**.
  HINT: Use the macros BG_PRIORITY(0) and BG_PRIORITY(1) to configure the registers BGCTRL_2 and BGCTRL_0, respectively.
- Complete the code inside the function **movePlayerRight()** in the file *game.c* to **teleport the full character to the left edge of the screen when the right-most tiles of the character are about to enter the portal. The character should not overlap with the portal or be split at any moment.** The y position of the character must be the same after the teleportation move. The teleportation result is shown in **Figure 3** below.
- Compile the project and correct any possible errors. Check that the magenta portal appears on the right edge of the screen and that the character teleports correctly when passing through it.

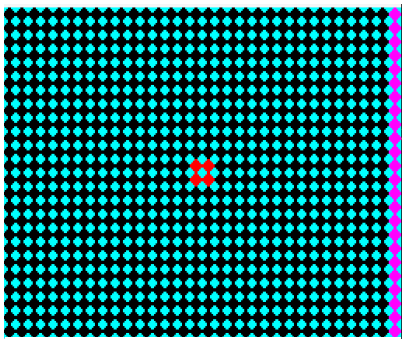


Figure 1 – Initial portal

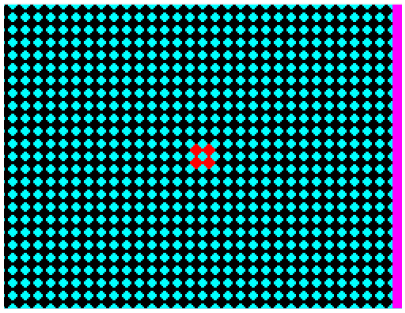


Figure 2 – Updated portal

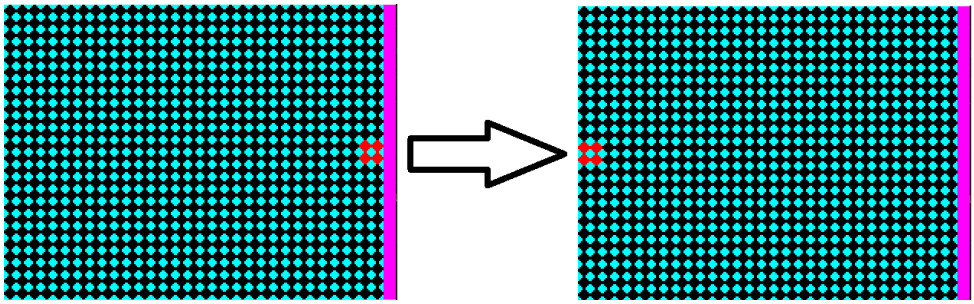


Figure 3 – Teleportation example

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 SOME OF THE EXERCISES MAY NOT BE COUNTED FOR THE FINAL MARK	
FINAL TIME:	Student Sign.: