CAs – Extra Points 2 – Tech Notes

The behaviour of the implemented game

The application implements the game "Blind Labyrinth" and makes use of the LCD display on board the LandTiger. The project includes some pre-created libraries for managing the LCD and Touch Panel.

The logic and the code behind it

In particular, the LCD library exposes trough the GLCD.h header file some useful functions, and I extended it by adding another source file (GLCD_POWER.c) that adds some functions. In particular, we have:

- LCD_DrawRectangle: this is capable of **drawing rectangles given a starting point, a length and a width**. In order to make the drawing faster (the use of the default exposed functions makes the drawing too slow), I had to expose from GLCD.c some functions (initially declared as static so no globally available) like LCD_SetPoint, LCD_WriteIndex and LCD_WriteData. **This let me write directly inside the GRAM without using external functions** (the GRAM linearize the display as a C-like matrix is linearized in RAM).
  In particular, using LCD_SetPoint is possible to set the starting "y" and "x" of a line of the rectangle to draw. With LCD_WriteIndex I begin a communication towards the GRAM (address 0x0022), finally with LCD_WriteData I can send all the new pixels of my rectangle in a row.
- LCD_DrawSpaceship: this is capable of **drawing the spaceship from Space Invaders in the four directions** (N, S, W, E) given a single matrix of pixels and rotating it by use of strategic indexes. This **represents the robot and is possible to draw it with two different background** (in order to differentiate the EXPLORE and MOVE modes).
- LCD_DrawSpacemonster: like the spaceship, **draws the monster from Space Invaders representing an obstacle** of the labyrinth. The function is capable of **drawing both the red version and green version** based on an argument.
  This function is used with this **argument generated "pseudo-randomly" taking the LSB of the Timer0 counter** when needed. In memory, **only the red green monster is stored**. The red one is possible thanks to a live replace of each colour with its "twin". **Because of the symmetry of the image, only half of it is saved in memory in order to save space** and during its drawing the remaining half is drawn reading the saved image in reverse.

The display shows the labyrinth map. **Each cell has a resolution of 15x15 pixel**. Some macros are available (inside the file lab_map.h) useful to map a position of the labyrinth to the relative position on the LCD, and this is useful especially with functions like LCD_DrawSpaceship.

Both the RIT and Timer0 are used. **The RIT is used with a period of 20 ms and it pools the joystick and the Touch Screen**. When a direction is selected in MOVE, it enables Timer0 that moves the robot along its direction with a period of 200 ms.

For the Clear operation, it loops through all the labyrinth (not through the display) and for each obstacle found (except for the one in front of the robot) it draws a void rectangle at its position on the LCD (even if there is no obstacle shown). **This makes the clear operation faster, thanks also to the optimization done in LCD_DrawRectangle**.

All the "labyrinth behaviours" like obstacle identification or robot moving, are handled by very specific function library under "blind_labyrinth.h". Timers make use of these functions in order to handle and display everything correctly.