APO Documentation

Structure of the code

The program code is organized into 4 source files.
The main.c uses:

- functions from leds_interaction.c to operate the LED #1, LED #2 and LED line on MZ_APO;
- functions from font_functions.c to display menu messages and game information on MZ_APO LCD;
- functions from snake.c to initialize and control game.

Files	Description
leds_interaction.c	Groupes all fuctions necessary for managing interaction with LEDS on MZ_APO board.
font_functions.c	Consists of fuctions which take care of printing to the MZ_APO LCD display: be it pixels or whole words (using wTahoma_88 font). Functions printing texts to the display were grouped to form functions printing whole menus.
snake.c	Consists of fuctions necessary for initializing game board with snakes and borders, controlling movements of snakes, performing moves, generating food pieces, processing eating, and keeping the state of the game.
main.c	Runs the game.

Figure #1: List of source files of the program and description of their purpose.

Header file name	***leds_interaction.h***
functions	void lightGreenLED(unsigned char * mem_base, int ledNumber);
	void lightBlueLED(unsigned char * mem_base, int ledNumber);
	void lightRedLED(unsigned char * mem_base, int ledNumber);
	void lightDownLED(unsigned char * mem_base, int ledNum);

Figure #2: List of function declarations in one of the header files: leds_interaction.h

Header file name	***font_functions.h***
functions	int getCharWidth(font_descriptor_t* fdes, int ch);
	void drawChar(font_descriptor_t* fdes, uint16_t * board, int charWidth, int ch, int posX, int posY);
	void drawCharLarger(font_descriptor_t* fdes, uint16_t * board, int charWidth, int ch, int posX, int posY);
	void printText(char * str, int length, int posX, int posY, uint16_t * board);
	void printMenuMode(uint16_t * board);
	void printMenuAppleCount(uint16_t * board);
	int getKeyboardMenuInput();
	void cleanBoardArr(uint16_t * board);

	void printBoard(uint16_t * content, unsigned char * parlcd_mem_base);
	void printSnakeLengths(uint16_t * board, int len1, int len2, double time, unsigned char *parlcd_mem_base);
	void runMenu(int * mode, int * applesCount, uint16_t * board, unsigned char *parlcd_mem_base);

Figure #3: List of function declarations in one of the header files: font_functions.h.

Header file name	***snake.h***
functions	void printBoardToLcd(uint16_t * content, unsigned char * parlcd_mem_base);
	void blackLcd(unsigned char * parlcd_mem_base);
	bool isInRange(int currentRow, int currentCol, int pointX, int pointY, int range);
	void initializeSnakeAndDirection(int initX, int initY, int snakeLength, uint16_t * snakeArr, Cell ** directionArr);
	void initializeBorders(uint16_t * snakeArr);
	void redrawSnakeCell(uint16_t * snakeArr, int posX, int posY, uint16_t color);
	int addApple(uint16_t * snakeArr);
	void distributeApples(uint16_t * snakeArr, int * applesArr, int applesCount);
	void shiftDirCell(Cell * cell, unsigned char prevCellDir);
	bool isWithinLCD(int posX, int posY);
	bool isCellOccupied(uint16_t * snakeArr, Cell * head);
	bool isApple(uint16_t * snakeArr, Cell * head);
	bool snakeMakeMove(uint16_t * snakeArr, Cell ** directionArr, int * length, unsigned char * mem_base, bool * isEaten);
	void updateDirection(Cell * cell, unsigned char newDirection);
	unsigned char getRandomDirection(unsigned char currentDir);
	unsigned char mapKeyToDirection(unsigned char lastDirection, char pressedKey);
	unsigned char getKeyboardInput(unsigned char lastDirection);
	int isDirPossible(char lastDir, char desiredDir);
	char generateComputerMoveDir(uint16_t * board, int * chosenAppleIndex, int * applesArr, Cell * head, int applesCount);
	bool isAnyAppleLeft(int * applesArr, int len);
	void playRandomVsSSH(unsigned char *mem_base, unsigned char *parlcd_mem_base, int * snakeLengthS1, int * snakeLengthS2, uint16_t * board, Cell ** directionArrS1, Cell ** directionArrS2, int applesCount);
	void playRandomVsRandom(unsigned char *mem_base, unsigned char *parlcd_mem_base, int * snakeLengthS1, int * snakeLengthS2, uint16_t * board, Cell ** directionArrS1, Cell ** directionArrS2, int applesCount);

Figure #4: List of function declarations in one of the header files: snake.h.

Usage of Peripherals

LED #1 and LED #2

- display the state of snake #1 and snake #2 as follows:
    - blue color
        - if the snake eats a piece of food

        - red color
            - if the snake dies
        - green color
            - if the snake is alive and has not just eaten

LED line

- lights up if any snake eats a piece of food (0xFFFFFFFF)
- dark otherwise (0x0)

Manual

Game can be played in 2 modes:

Mode 1: Computer vs Computer / AI vs AI

- user just pasively watches 2 snakes move using computer-generated moves
- each snake randomly chooses one of the food pieces on the board, and moves towards it, whenever a food piece is eaten, another target is chosen

Mode 2: Computer vs Person

- user uses '*a*' and '*d*' keys to control movements of the snake that initially appears at the bottom left corner of the MZ_APO display

Rules

- user scores:
    - when its snake eats a food piece, which increases the snake's length by one cell
- snake dies:
    - if it touches borders (white lines at the edges of the display)
    - if it touches a body part of another snake
- game ends:
    - when all food pieces have been eaten
    - when both snakes had died
      (whichever happens first)

Gameplay

The game consists of 3 phases:

Phase #1 - Menu

Firstly, the user is asked to input the number of the desired Game Mode, that is,

- Mode 1: Computer vs Computer / AI vs AI
    - type '1' press Enter on keyboard
    - mode description: find above
- Mode 2: Computer vs Person
    - type '2' press Enter on keyboard
    - mode description: find above

Secondly, the user is asked to input number of food pieces that will be featured in the game

    - user can choose a number in the interval <5, 25>
    - type the chosen number and press Enter on keyboard

Phase #2 – Game

Mode 1

- user does nothing

Mode 2

- user can control the snake movements:
    - **press ‘a’**
        - to turn the snake head to left
    - **press ‘d’**
        - to turn the snake head to the right
    - **press ‘e’**
        - to leave the game
    - **press any other key**
        - to not wait for the end of the waiting for a key press, meaning that the snake will immediately move in the direction it is facing

Phase #3 – Game Summary

Length of each snake is displayed, along with the duration of the game in seconds.

Compilation & Running the Project

As the Makefile is included in the uploaded zip, the project can be run by “make”.

To use “make run”:

- set CTU account to yours in:
    - SSH_OPTIONS=-o 'ProxyJump=ctu_account@postel.felk.cvut.cz'
- the settings in Makefile suppose that the user has mzapo-root-key present in the ssh agent
- TARGET_IP is set to: 192.168.202.211, can be changed if there is a need

Git

Remote is present in my official CTU account on GitLab in the APO repository (in folder project).