UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Antonio Martí Campoy
amarti@disca.upv.es

Game Library architecture
Embedded C

ESIGELEC

Introduction.

This document called “Game library” presents full detail for the use of constants and functions available in the software library *game.lib*. You can include this library in your project and use all the constants and functions defined in it. To use this library, copy the file game.lib in the folder of your project and don’t forget to include game.h.

The game is structured in two loops:
- The menu loop. Allows the user to choose the colour of the car and start de game.
- The game loop. Runs the game (draw the road and move forward the car). It is a time-triggered structure, that is, actions are executed periodically.

The library includes the following constants:

```
#define CLOCK 24000000
```
Frequency in Hz of the DCO, used for MCLK, HSMCLS and SMCLK.

```
#define NUM_OPTIONS 3
```
Number of options available in the menu.

The library includes the following functions:

```
uint8_t check_SysTick_flag(void);
```

The SysTick timer is configured to raise an interrupt every 40ms. The ISR for SysTick sets with value 1 a variable, private for the library called SysTick_flag. This function returns the value of Systick_flag and clear it to 0.
Calling this function inside game loop allows to execute some tasks every 40ms.

Parameters: none.
Return value: the value of SysTick_flag.

```
void Init_Game(void);
```

Initialise the TFT screen and the software structures to create the game. This function requires that global interrupts are enabled.
Parameters: none.
Return value: none.

```
void Create_Menu(void);
```

Draw in the screen the game menu. Call just once before entering in the loop for menu management.

Parameters: none.
Return value: none.

```
uint8_t Show_Menu(uint8_t option);
```

Highlight the chosen colour indicated by the parameter *option*. If option isn't in the proper range, does nothing.

Parameters: the number of the option. In the range [0, NUM_OPTIONS -1]
Return value: 0 if option is within the range, otherwise it returns 1.

```
void Set_Car_Color (uint8_t color);
```

Store the car colour chosen by the user, so it can be used when the game start. It is a number that correspond with the number of option in the menu. If the parameter is not in the range the behaviour is undefined.

Parameters: the number of the option. In the range [0, NUM_OPTIONS -1].
Return value: none.

```
void Draw_Car(void);
```

Draws the car in the screen. It must be called every 40ms.

Parameters: none.
Return value: none.

```
void Draw_Road(void);
```

Draw the road in the screen. It must be called every 40ms.

Parameters: none.
Return value: none.

```
void Steering_Wheel (int8_t offset);
```

Change the stored position of the car in the units indicated in the parameter *`offset`*. The position is moved to the left if the value of *`offset`* is negative. The position is moved to the right if the value of *`offset`* is positive.
This function doesn't update the position of the car in the screen.

Parameters: units the car is moved to the left of to the right.
Return value: none

```c
uint8_t change_speed(uint8_t units);
```

Change the stored speed of the car in the units indicated in the parameter *`units`*. The speed lowers if the value of *`units`* is negative. The speed is raised the value of *`units`* is positive. The function takes care keeping the speed within the range of minimum and maximum speed allowed.
This function doesn't update the screen.

Parameters: units the speed is lowered or raised.
Return value: always 0.

```c
void show_error(int8_t *l1, int8_t *l2);
```

Show the string l1 and l2 in the screen.
Example of use: `show_error("colour", "ERROR");`

Parameters: null terminated strings.
Return value: none

```c
uint8_t foo(void);
```

Does nothing.

Parameters: none.
Return value: always 0.