Dokumentace pro APONGO

B0B35APO - Architektura počítačů

Rozálie Bílková
bilkoroz@fel.cvut.cz

Jakub Adamec
adamej14@fel.cvut.cz

Testováno na desce č. 0926 0009 1144, která má jiný displej. V `perifs_handle.c` je proto na **25.** řádku příkaz `parlcd_hx8357_init(parlcd_base);` zakomentovaný pro korektní funkci.

Kompilace je doporučena pomocí přiloženého Makefile.

Spuštění

Aplikace se spouští běžným spuštěním programu z příkazové řádky, tedy `./ubongo`.

Menu

Po spuštění aplikace nejdřívě naběhne synchronizační funkce, která zajistí, že výstup z ovládacích knobů je připraven k používání. Během inicializace je na displeji vypsán text „`Loading! Please wait.`“.

Po úspěšné inicializaci naběhne hlavní menu programu. Zde jsou na výběr tlačítka **Play**, **Settings** a **Quit**.

Pohyb v menu

Pohyb je zajištěn pomocí otáčení zeleného knobu. Tato informace je také vypsána na displeji. Pravotočivý pohyb znamená pohyb směrem vpravo v menu, respektive vlevo. Kurzor je zajištěn fialovým obtáhnutím daného tlačítka s textem. Výběr a spuštění akce je provedeno zmáčknutím zeleného knobu.

Možnosti volby

Jestliže uživatel v nastavení nenastaví obtížnost anebo scaling písma a zmáčkne tlačítko **Play**, nastaví se přednastavené hodnoty, tedy obtížnost `Potato` a scaling `2`.

Po zmáčknutí tlačítka **Settings** se uživatel dostane do submenu, ve kterém je na výběr změna obtížnosti **Difficulty** nebo změna scalingu písma v celém programu **Scale**. Zároveň má možnost se vrátit do hlavního menu pomocí **Back** tlačítka.

V **Difficulty** jsou na výběr obtížnosti: `Potato` (defaultní), `Nightmare` a `Hell`. Po zvolení obtížnosti se uživatel vrátí do hlavního menu a program si uloží nově nastavenou obtížnost. Obtížnost mění velikost mapy, případně její samotnou náročnost.

V **Scale** podmenu má uživatel na výběr dvě velikosti scalingu, tedy `Scaling 1` a `Scaling 2` (defaultní).

Po zvolení tlačítka `Quit` v hlavním menu se program ukončí a displej se nastaví na černou obrazovku.

Praha, jaro 2023

Ovládání aplikace během hry

Na implementaci samotné herní mechaniky bohužel nezbyl čas.

Průběh hry byl zamýšlen: Náhodně se vybere jedna z 6 předpřipravených map s příslušnými herními dílky, které řeší danou mapu. Následně bude mít hráč umístěn kurzor na prvním z nich. Pomocí zeleného knobu se pohybuje kurzorem po herních dílcích. Stiskem zeleného knobu se potvrdí výběr dílku. Následně se kurzor spolu s dílkem umístí na hrací plochu.

Ovládání v této sekci se přepne na posun po **x** ose pomocí červeného knobu, posun po **y** ose pomocí zeleného knobu a **rotaci** pomocí modrého knobu. Zmáčknutí červeného tlačítka vrátí hráče s neumístěným dílkem zpět do stacku s dílky. Zmáčknutím zeleného knobu s dílkem na hrací ploše se provede place, při kterém se zkontroluje, jestli je tah validní. Pokud ano, zároveň se otestuje, jestli řeší danou mapu, čímž se rozsvítí jedna sekce na LED pásku. Zmáčknutí modrého knobu kdykoliv během hry vrátí hráče zpět do hlavního menu.

Vlastní soubory

Repozitář je dostupný https://github.com/knedl1k/MZAPO_SEM,
případně mirror na https://gitlab.fel.cvut.cz/adamej14/MZAPO_SEM.

ubongo.c

Hlavní soubor, obsahuje kromě hlavní funkce `main()` také inicializace periférií. Zároveň spouští funkci `menuReaction()`, která zajišťuje chod úvodního menu.

colors.h

Soubor, ve kterém je uložena definice `union rgb`.

perifs_handle.c

Soubor funkcí, které korektně inicializují a pracují s periferiemi.

- `union pixel fb[LCD_WIDTH][LCD_HEIGHT]`
  Frame buffer. Globální proměnná, která je sdílená i mezi dalšími soubory pomocí klíčového slova `extern`. Zapisují se do ní všechny změny, které se následně mají propsat na LCD.
- `void lcdReset(int color)`
  Funkce restartuje celý displej a nastaví na celý displej barvu ze vstupu.
- `void lcdRefresh(void)`
  Aktualizuje změny z framebufferu a propíše je na displej.
- `struct rotation_t updateKnobValues(void)`
  Pomocí struktury `rotation_t`, do které ukládá aktuální data o zpracování pohybu knobů, a pomocné struktury `knob_t`, vyhodnocuje, jestli se jednalo o skutečné otočení voliče, či jen o zákmit. Také zpracovává informace o zmáčknutí knobů.
- `void rgb1(union rgb color)` & `void rgb2(union rgb color)`
  Funkce, které mají jako vstupní argument barvu, kterou následně zobrazí na LED diodě.

drawing.c

Soubor zaměřen na funkce, u nichž je primární funkce kreslení tvarů, či výpomoc při barvení pixelů.

- `void colorPixel(union rgb color, int x, int y)`
  Na posici (x,y) obarví daný pixel vstupní hodnotou `color`.

Praha, jaro 2023

- `void drawRectangle(union rgb color, int x, int y, int width, int height)`
  Na vstupních (x,y) souřadnicích nakreslí obdelník o rozměrech (width, height) s barvou hrany `color`.

maps.c

Zdrojový soubor pro všechny důležité funkce, které vykreslují herní mapy a jejich řešící herní dílky.

- funkce `void drawBoard1(int edge)` až `void drawBoardn(int edge)`
  Nakreslí předem definové hrací plochy s černou hranou a červeným podkresem.
- `void manageStack(uint8_t board, uint8_t piece, _Bool in_stack, uint8_t cursor)`
  Spravuje render dílku na hratelném stacku pro danou mapu. V momentální implementaci pouze vykreslí hratelné dílky.

text_display.c

Zde jsou funkce, které zapisují příslušné stringy nebo chary daného fontu do frame bufferu.

- `void printChar(char c, int x, int y, union rgb color,unsigned char scale)`
  Na příslušnou (x,y) posici vypíše znak **c** barvy **color** se scalingem **scale**.
- `void printString(char *word, int x, int y,union rgb color, unsigned char scale)`
  Jako vstupní parametr má string **word**. Výpis na displej probíhá tak, že se postupně volá funkce `printChar` dokud se nevypíší všechny chary příslušného stringu.
- `void drawRectangleWithText(char *str, int x, int y, union rgb color,unsigned char scale, _Bool selected)`
  Vypíše string **str** na displej a dynamicky okolo něj vykreslí obdelníkový rámček barvy **color**. Vstupní parametr **selected** zajišťuje vykreslování případného kursoru. Pokud je **selected** `true`, pak okolo rámečku textu vykreslí jeden trochu větší fialové barvy, znázorňující kursor.

menu_handle.c

Funkce, které se starají o správné fungování a reakce menu na uživatelský vstup.

- `void menuReaction(void)`
  Funkce s hlavní smyčkou menu. Zpracovává uživatelský vstup ze zeleného knobu. Dle potřeby volá funkce na nastavení obtížnosti, scalingu, spouští a ukončuje program. Zároveň zařizuje korektní posouvání fialovým kursorem po tlačítkách.
- `static void render<>Menu(uint8_t selected)`
  <>={Main, Settings, Diff, Font}
  Soubor funkcí, které vyrenderují všechna tlačítka pro příslušné menu/submenu s nadpisem. Vstupní parametr **selected** zajišťuje render kursoru okolo tlačítka.
- možnost nastavit scaling fontu na 2 (defaultní) a 1.
- vzhledem k nezvládnuté implementaci samotné herní mechaniky je submenu s nastavením obtížnosti fakticky jen ukázkové, protože není co nastavovat.

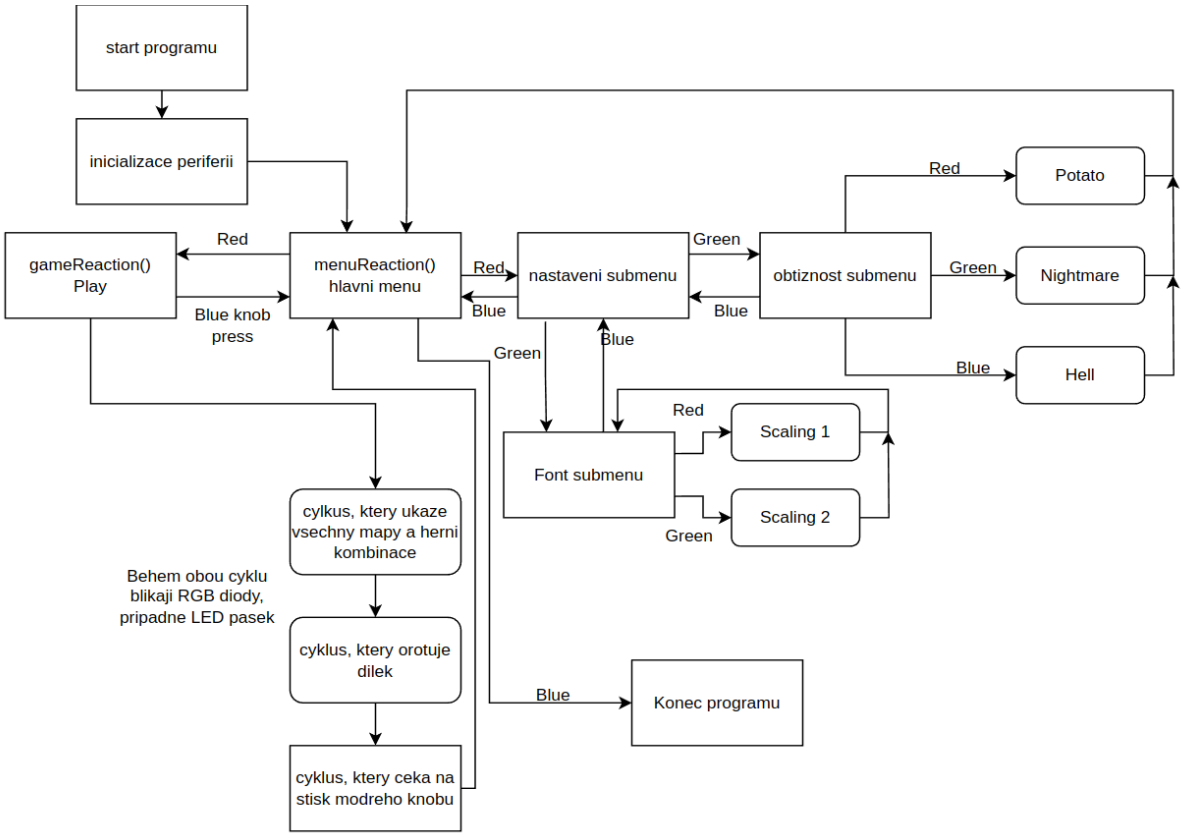
game_handle.c

- `void gameReaction(void)`
  V aktuální implementaci se zde nachází cyklus, který v určitém intervalu prolistuje všechny herní mapy a jejich dílkové kombinace. Zároveň se zde také nachází rotace dílku okolo své osy pro ukázku funkčnosti.

  Zamýšlená byla jako funkce, ve které je hlavní herní smyčka. Zpracovávala by vstup z knobů a řídila spouštění pomocných herních mechanik.

Praha, jaro 2023

UML diagram aktuálního stavu programu



Obrázek 1: UML diagram běhu programu a reakce jednotlivých funkcí.

Praha, jaro 2023