Technická dokumentace

MicroZed APO – Julia set

Program představuje implementaci zobrazení Juliovy množiny na přípravku MicroZed APO. Juliova množina představuje fraktál, složitý geometrický objekt tvořený všemi body komplexní roviny, pro kterou posloupnost nediverguje.  
Samotný fraktál je poté zobrazen na LCD displeji (480×320 bodů) v maximu z podporované barevné škály (RGB565). Poloha zobrazení počátku souřadného systému a parametry výpočtu jsou voleny otočnými voliči.

***Struktura programu:***

Program se skládá z hlavního zdrojového souboru – apo\_semestral.c, ve které se nachází hlavní programová logika.

Dále obsahuje několik hlavičkových souborů pro lepší přehlednost a možnou budoucí rozšiřitelnost programu.

***Pomocná rozhraní:***

**computation.h**

= rozhraní umožňující hlavní výpočetní operace Julia setu  
- definuje nový datový typ *computation*, který obsahuje proměnné reprezentující výpočetní parametry Julia setu  
- definuje výčtový typ *param*, pro pohodlnější adresování parametrů  
- definuje výchozí hodnoty parametrů

int compute (double cR, double cX, double pR, double pI, int n);

= funkce, která vrací počet iterací, kterých se dosáhlo při zjišťování konvergence komplexní posloupnosti bodu (pR, pI)

display\* computeImage(display\* img, computation parameters);

= funkce, která vrací spočítá barvy jednotlivých pixelů pro celý obrázek

computation initParameters();

= funkce, která inicializuje strukturu výpočetních parametrů, kde použije výchozí hodnoty parametrů

**knobs.h**

= rozhraní umožňující práci s tlačítky  
- definuje nový datový typ *knobs\_t*, který obsahuje staré a nové hodnoty tlačítek, inkrementy a namapovanou adresu  
- definuje výčtový typ rgb, pro pohodlnější adresaci jednotlivých tlačítek

bool getPUSH(uint32\_t knobs, rgb col);

= funkce, která testuje, zda-li určité tlačítko bylo stisknuto

knobs\_t\* getIncrement (knobs\_t\* k, uint32\_t knobs);

= funkce, která vypočte inkrementy na jednotlivých tlačítkách z nově vzatých hodnot

knobs\_t\* setInitialvalues(knobs\_t\* k, uint32\_t knobs);

= funkce, která inicializuje strukturu knobs\_t, do které uloží počáteční hodnoty a vrací ukazatel na ni

**screen.h**

= rozhraní umožňující hlavní operace související s obsluhou displeje   
- definuje nový datový typ *display* který obsahuje výšku a šířku displeje, namapovanou adresu a bitmapu daného obrázku  
- definuje nový typ *pixel*, který představuje jednotlivý pixel bitmapy

uint16\_t rgbTo16(uint8\_t r, uint8\_t g, uint8\_t b);

= funkce, která převede pixel do 16 bitové reprezentace určené pro MZApo

void repaintScreen(display\* display);

= funkce, která vykreslí bitmapu (display type) na screen MZApo

void showScreenPPM (display\* display);

= funkce, která uloží aktuální bitmapu jako ppm soubor

display\* initdisplay();

= funkce, která inicializuje strukturu typu display do které uloží počáteční výšku a šířku displeje, zabere paměť pro příslušnou bitmapu a nastaví všechny pixely na černou barvu

display\* setBlackScreen(display\* dis);

= funkce, která nastaví bitmapu (display type) na černou barvu

display\* writeText(display\* dis, int x, int y, char\* text);

= funkce, která nastaví celou bitmapu na černou barvu

display\* setPixel(display\* dis, int x, int y, uint8\_t r, uint8\_t g, uint8\_t b);

= funkce, která nastaví daný pixel na určitou barvu

display\* writeLetter(display\* dis, int x, int y, unsigned char c);

= funkce, která vypíše daný textový charakter na pozici x, y na displeji

display\* showStatus (display\* dis, int n, double cR, double cI, double aR, double aI, double bR, double bI);

= funkce, která vytvoří obrázek, který zobrazuje status výpočtu

display\* showMenu (display\* dis, menuicon icon);

= funkce, která vytvoří obrázek, který zobrazuje menu s aktuálně vybranou ikonou

**netcom.h**

= rozhraní umožňující komunikaci přes UDP protokol

int initCommunication();

= funkce, která zahájí síťovou komunikaci

int receiveBytes(int SOCKET, char\* address, char\* buffer);

= funkce, která přijme data ze síťového spojení

***Hlavní program***

**1. Main thread**

Na začátku hlavního vlákna se namapují adresy pro knoflíky a obrazovku, vytvoří datové struktury určené pro práci s nimi (knobs\_t, display). Dále se vytvoří synchronizační primitiva pro ochranu dat během programu a konečně ostatní vlákna.

**2. UDP thread**

Vlákno, které nejdříve vytvoří síťové spojení (socket) pomocí metod deklarovaných v **netcom.h** a dále čeká na přijetí parametrů, které chce uživatel změnit (v textové podobě).

**3. Knobschecker**

Vlákno, které pravidelně sleduje stav tlačítek a získá inkrementy pomocí funkcí z **knobs.h**. Při jejich změně provádí rozhodovací logiku, podle módu, ve kterém se zrovna nachází (COMPUTATION x MENU).

V MENU módu dochází k překreslení aktuálně používané položky menu, kdy se mění parametr *menuicon*. V COMPUTATION módu jsou dále měněny samotné parametry výpočtu.

Při každé změně tlačítek informuje vlákno knobschecker pomocí conditional variable a změny boolean proměnné *repaint* vlákno printer o překreslení displeje. Pokud se navíc nachází program ve fázi COMPUTATION je změněna také proměnná *compute* a dochází k přepočítání fraktálu.

**4. Printer**

Vlákno, které překresluje displej pomocí funkcí definovaných v **screen.h**. Překreslení displeje je závislé na aktuálním módu, ve kterém se program nachází.

**5. Counter**

Vlákno, které přepočítává fraktál pomocí funkcí definovaných v **computation.h**. Přepočítání displeje probíhá vždy pouze v módu COMPUTATION.