Dokumentácia k projektu do predmetu Soft computing: Vizualizácia *Mandelbrotovej množiny*

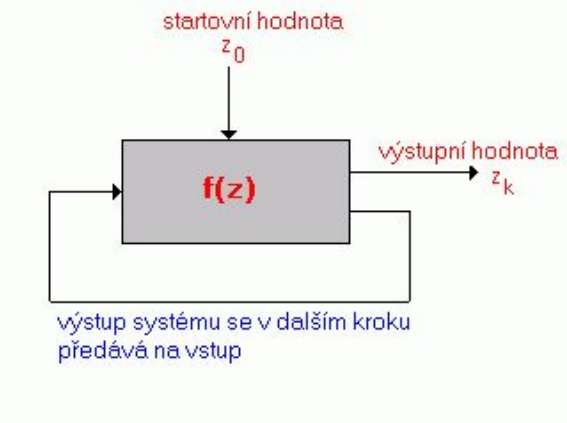
Autor: Filip Gulán (*xgulan00*)

1 Úvod

Úlohou tohoto projektu bolo vytvoriť pôvodný funkčný program, ktorý mal demonštrovať činnosť niektorého zo soft-computingových prístupov. V rámci projektu bolo vybrané zadanie číslo 123 Vizualizácia *Mandelbrotovej* množiny. Samotná realizácia prebehla pomocou jazyka *Java* a grafickej knižnice *JavaFX*. Naimplementovaná aplikácia slúži ako prehliadač *Mandelbrotovej* množiny, v ktorom je možné sa po množine pohybovať, približovať ju, odďaľovať ju, nastavovať rôzne farebné parametre zobrazenej množiny, nastavovať presnosť výpočtu a nakoniec uložiť zaujímavý grafický výsledok ako obrázok vo formáte *png*.

2 Popis riešeného problému

Mandelbrotova množina je jeden z najznámejších fraktálov. Je vytvorená pomocou jednoduchého dynamického systému, ktorý je založený na iterácii funkcie komplexnej paraboly $z_{n+1} = z_n^2 + c$ kde z a c ležia v komplexnej rovine. Tento systém je možné vidieť na obrázku 1.



Obrázok 1: Iteračný výpočet bodov ležiacich v *Mandelbrotovej* množine. Zdroj [1]

V priebehu výpočtu sa hodnota z mení, zatiaľ čo hodnota c zostáva rovnaká a mení sa iba pri presune výpočtu na nový bod. Iteračný proces začína hodnotou z_0 a systém postupne generuje sekvenciu hodnôt $z_1, z_2, z_3, \ldots z_k$, ktoré sa tiež nazývajú *orbit*. Pri práci s týmto systémom nás zaujíma, či pre danú štartovaciu hodnotu z_0 a konštantu c postupnosť konverguje, alebo diverguje. *Mandelbrot* sa obmedzil iba na prípad, kedy počiatočná hodnota z_0 je nulová a pre každý počítaný bod sa mení iba konštanta c. Iteračným výpočtom potom vzniknú iba *orbity*

nuly, ktoré je možné rozdeliť podľa ich vlastností: pre *c* je buď *orbit* konečný (všetky hodnoty z_k sú konečné, patria sem aj hodnoty, ktoré oscilujú), alebo je *orbit* nekonečný (po určitej dobe rastú hodnoty z_k nad všetky medze). *Mandelbrotova* množina je teda definovaná ako $M = \{c \mid P_c^n(0) \neq \infty \; \forall n \to \infty;\; c, z \in C\}$, kde $P_c^n(0)$ znamená hodnotu z_n pre danú konštantu *c* a $z_0=0$ [1].

3 Popis implementácie

Program je implementovaný v jazyku *Java* a pre vizualizáciu a užívateľské rozhranie bola použitá grafická knižnica *JavaFX*. Celá aplikácia je rozdelená do balíčkov a tried. Hlavná trieda *Mandelbrot*, ktorá je spustená ako prvá sa nachádza v koreni balíčku *sfc*. Táto trieda zabezpečí správne inicializovanie grafického rozhrania a ďalšie základné nastavenia GUI. V balíčku *sfc.controllers* sa nachádza trieda *MainController*, ktorá slúži ako kontrolér pre layout *mainLayout* a nachádzajú sa v ňom všetky operácie s GUI, ako napríklad odchytávanie udalostí (kliky na tlačítká, posúvanie posuvníkov...). V balíčku *sfc.models* sa nachádzajú triedy pre prácu s dátami. Konkrétne ide o triedy *CanvasModel* a *MandelbrotModel*. *CanvasModel* slúži na prácu a vykresľovanie *Mandelbrotovej* množiny do obrázka, presnejšie do *ImageView* prvku. Pri implementácii sa ukázalo, že vykresľovanie do dátového typu *WritableImage* a následné jeho zobrazenie v *ImageView* je omnoho rýchlejšie ako kreslenie priamo na *Canvas*. Posledná trieda *MandelbrotModel* slúži na testovanie bodu, či bod patrí do *Mandelbrotovej* množiny, alebo naopak nepatrí.

Po spustení aplikácie triedou *Mandelbrot* je riadenie predané triede *MainController*. Tá udržuje stav aplikácie, vytvára instanciu triedy *CanvasModel*, nastaví všetky nastavenia na východzie hodnoty a následne spustí metódu *CanvasModel*-u menom *rendercanvas()*. Táto metóda riadi vykresľovanie Mandelbrotovej množiny v separátnom vlákne, aby nebolo blokované GUI. V prvom rade sa zdistretizuje aktuálny "výsek" komplexnej roviny, aby ju bolo možné vykresliť po pixeloch. Potom sa vypočíta prírastok, ktorý sa bude pridávať k imaginárnej alebo reálnej zložke komplexného čísla. Tento prírastok sa počíta v metóde *getStepValue()*. Metóda *getStepValue()* funguje tak, že sa získa vzdialenosť medzi 2 bodmi na imaginárnej alebo reálnej ose a následne sa toto číslo vydelí hodnotou šírky alebo dĺžky vykresľovacej plochy. Samotné vykresľovanie prebieha v 2 vnorených *for* cykloch, ktoré prechádzajú body po pixeloch a zároveň prechádzajú body komplexnej roviny posúvaním sa o vypočítaný prírastok, ktorý vrátila spomínaná funkcia *getStepValue()*. V každej iterácií vnoreného *for*-u sa zisťuje, či daný bod komplexnej roviny je bodom, ktorý patrí do *Mandelbrotovej* množiny alebo nie. Tento test bodu nastáva v rovnomennej metóde *test()* triedy *MandelbrotModel*, ktorá ako parametre berie imaginárnu a reálnu zložku bodu v komplexnej rovine. Pre vyhodnocovanie testu na nekonečnú *orbitu* a zostavenie algoritmu bol použitý dôkaz, že pre $|c|<=|z|$ a $|z|>2$ postupnosť diverguje [1]. V prípade, že funkcia zistí, že sa nejedná o bod, ktorý leží v *Mandelbrotovej* množine, tak vráti počet iterácií, ktoré boli potrebné na to, aby bola táto skutočnosť zistená (v opačnom prípade vracia 0). Práve táto hodnota počtu iterácií je použitá pri ofarbovaní. Metóda na testovanie náležitosti bodu do

Mandelbrotovej množiny je priložená aj v tejto dokumentácii, nakoľko sa jedná o významný úsek kódu.

```
/**
* Test if point belongs to Mandelbrot or not
* @param cReal real part of complex number
* @param cIm imaginary part of complex number
* @return number of iterations
*/
int test(double cReal, double cIm) {
   double zReal = 0.0;
   double zIm = 0.0;
   double zRealTemp;
   double zImTemp;
   int i = 0;

   do {
       zImTemp = zIm * zIm;
       zRealTemp = zReal * zReal;
       zIm = 2.0 * zReal * zIm + cIm;
       zReal = zRealTemp - zImTemp + cReal;
       i++;
       if (zRealTemp + zImTemp >= 4.0) { //it is not mandelbrot point
           return i;
       }
   } while (i <= this.maxIterations);
   return 0; //it is mandelbrot point
}
```

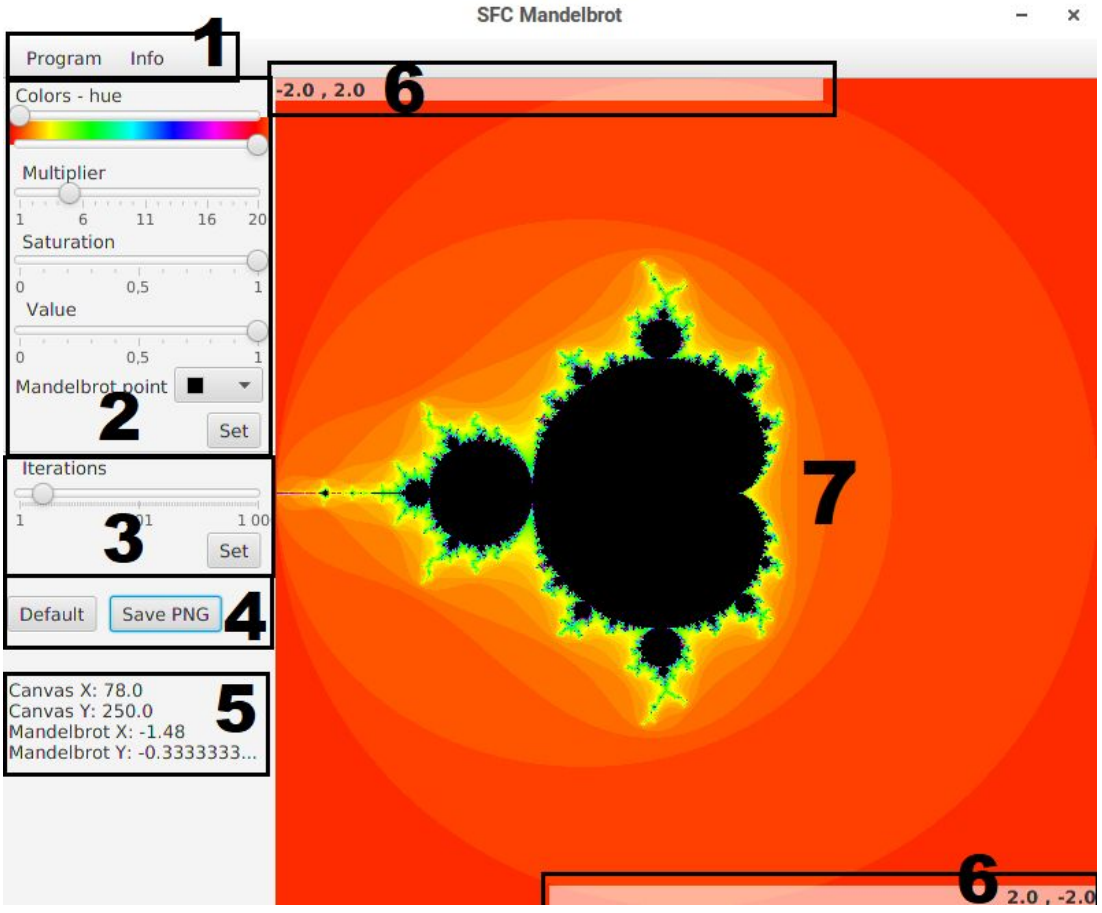
Pre vykresľovanie a ofarbovanie výsledku bol použitý farebný model *HSV* (*hue, saturation, value*). Hodnota, ktorá je vrátená funkciou *test()*, je použitá pre hodnotu farebného tónu (*hue*), pritom môže byť ešte obmedzovaná, alebo násobená, čo záleží od toho, čo daný užívateľ nastaví v GUI. Sýtosť farby (*saturation*) a hodnota jasu (*value*) sú taktiež brané podľa nastavení, ktoré zvolil daný užívateľ.

V prípade, že sa užívateľ naviguje v množine, buďto približovaním/odďaľovaním alebo posunom, tak sa vždy volá metóda *setupFrame()* triedy *CanvasModel*, ktorá nastaví nové súradnice zobrazovanej plochy a následne sa volá už hore spomínaná metóda *renderCanvas()* a proces vykresľovania sa opakuje.

4 Manuál k programu

Program je pred prvým spustením nutné preložiť priloženým ant build súborom. Tento preklad je možné spustiť zadaním príkazu *ant* v koreni adresára projektu. Po preklade vznikne zložka *build*, v ktorej sa nachádza zložka *classes* a *dist*. V zložke *classes* sa nachádzajú *Java* triedy a v zložke *dist* už výsledný program v *jar* formáte. Program je následne možné spustiť príkazom *java -jar Mandelbrot.jar* v zložke *dist* (v prípade *java -jar build/dist/Mandelbrot.jar* z koreňa adresára projektu).

Manipulácia s *Mandelbrotovou* množinou sa uskutočňuje prostredníctvom grafického užívateľského rozhrania. Celé grafické rozhranie je rozdelené na niekoľko blokov, tieto bloky je možné vidieť na obrázku 2. Blok 1 obsahuje *MenuBar*, v ktorom je možné zavrieť program alebo zobraziť modálne okno s informáciami o autorovi a projekte. Blok 2 obsahuje nastavenia, ktoré ovplyvňujú, ako sa budú aplikovať farby na získaný výsledok. *Colors - hue* obsahuje farebné spektrum a je tu možné nastaviť rozpätie, z ktorého sa budú aplikovať farby. *Multiplier* slúži na väčšiu variabilitu farieb. Čím je *multilier* väčší, tým je výsledný obraz farebnejší, poprípade farebný prechod je výraznejší. Pomocou *saturation* sa nastavuje sýtosť farby a pomocou *value* zase hodnota jasu. Položka *Mandelbrot point* slúži na výber farby, ktorou budú ofarbené body, ktoré patria do *Mandelbrotovej* množiny. Pre aplikovanie zmien je nutné kliknúť na tlačidlo *Set*. Blok 3 obsahuje nastavenia iterácií. Čím viac iterácií je nastavených, tým je *Mandelbrotova* množina detailnejšia (hlavne pri väčšom priblížení), avšak na úkor času potrebného na vykreslenie. Blok 4 obsahuje tlačidlá *Default* a *Save PNG*. Tlačidlo *Default* spôsobí, že všetky nastavenia, ako aj vykreslený obrázok, budú vrátené do východzieho stavu. Tlačítko *Save PNG* zase umožňuje uložiť zobrazenú grafiku do súboru. Blok 5 obsahuje informácie o polohe myši, ako v rámci vyrkeslovacieho plátna, tak aj v rámci komplexnej roviny. Bloky 6 obsahujú informácie o zobrazenom detaile, konkrétne súradnice rohov v rámci komplexnej roviny. Posledný blok 7 obsahuje vygenerovanú grafiku. V bloku 7 je možné meniť stred zobrazenia klikom myši (v podstate je to posúvanie sa), alebo je možné komplexnú rovinu približovať a odďaľovať koliečkom myši.



Obrázok 2: Grafické rozhranie aplikácie.

5 Záver

Aplikácia implementovaná v rámci tohoto projektu dokáže vyzualizovať *Mandelbrotovu* množinu. Užívateľ je pomocou *GUI* schopný nastavovať rôzne farebné nastavenia, presnosť výpočtu, a je mu umožnené sa dynamicky pohybovať, približovať, poprípade oddaľovať komplexnú rovinu.

Program bol riadne otestovaný na operačnom systéme *Zorin OS 12* a na operačnom systéme *CentOS*. Behom testovania sa nevyskytli žiadne chyby a všetky testy dopadli úspešne.

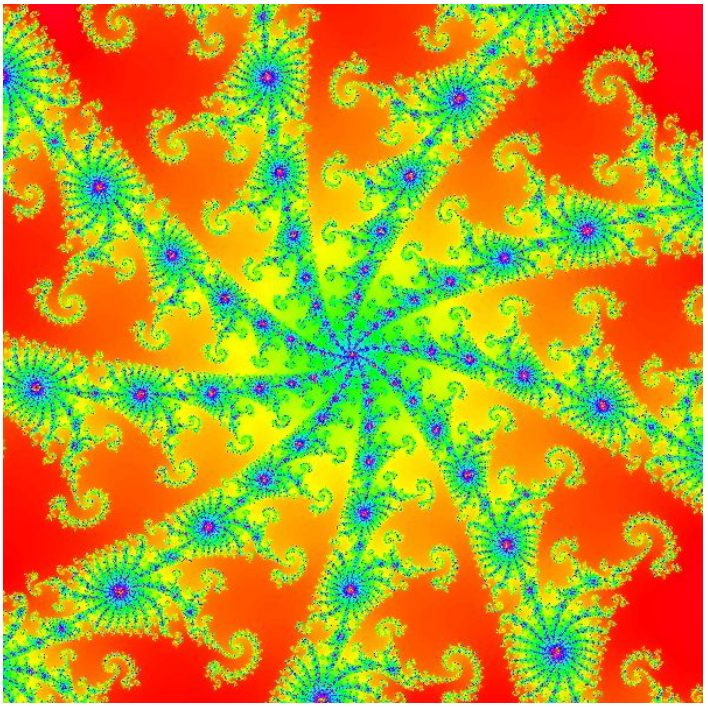
Referencie

[1] TIŠNOVSKÝ, Pavel. *Fraktály v počítačové grafice XII* [online]. 2006 [cit. 2017-10-15]. Dostupné z: https://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-xii/

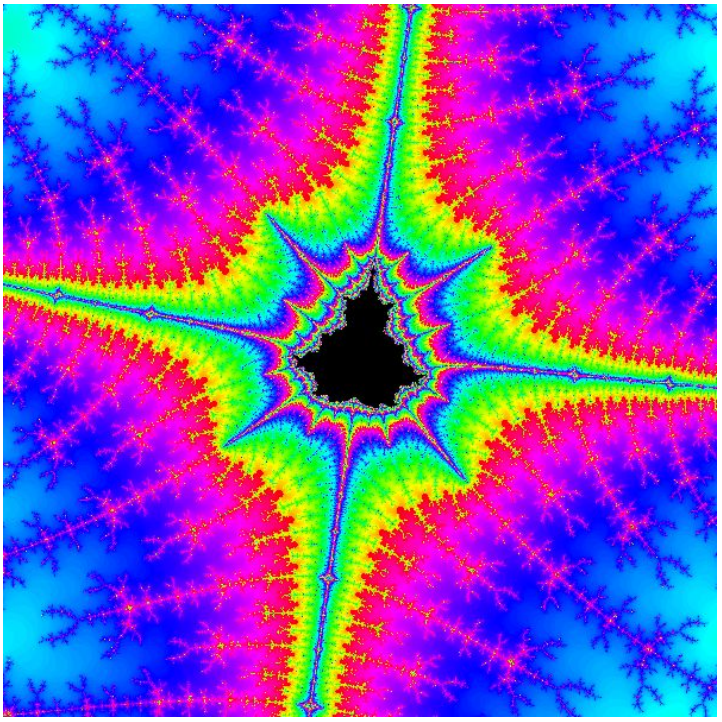
[2] *12. Chaos: SFC slidy k prednáškam* [online]. s. 39-40 [cit. 2017-10-15]. Dostupné z: https://www.fit.vutbr.cz/study/courses/SFC/private/16sfc_12.pdf

Príloha

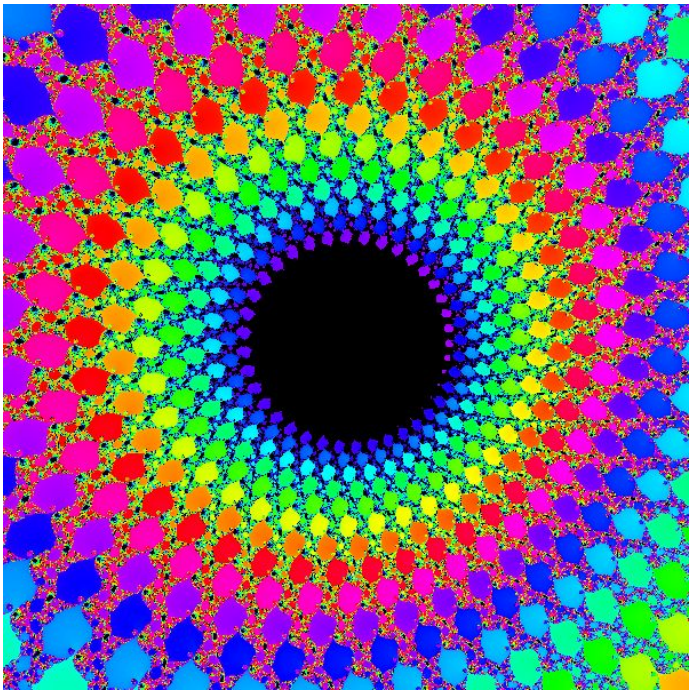
V tejto prílohe sú vybrané detaily *Mandelbrotovej* množiny, ktoré prišli subjektívne zaujímavé autorovi tejto práce. Pod obrázkom, ktorý je detailom množiny, sú uvedené súradnice ľavého horného rohu a pravého dolného rohu zobrazeného detailu.



-0.7985227782258434, -0.1659731400565913
-0.798517284133938, -0.16597863414849673



-0.106513068148097, -0.916581264394972
-0.10651252552173596, -0.916581807021333



0.25688912229464383. -0.0008951069961060542
0.2568914089176631, -0.0008973936191253698