Počítačové zobrazování fraktálních množin

Jiří Bednář, SPŠSaD Děčín, programagor@gmail.com
Veronika Chloubová, EOA Děčín, NikaEsterez@gmail.com
Jana Sotáková, Gymnázium Brno, Sotakova.Jana@seznam.cz

Abstrakt:

Cílem našeho miniprojektu bylo bližší seznámení s fraktálními množinami, napsání programu, jehož výstupem by měl být fraktál Buddhabrot, který bude popisován v práci.

1. Úvod

Fraktál je geometrický útvar lišící se od ostatních tím, že nemá jednoduchý tvar, vypadá nahodile a chaoticky. Název fraktál v překladu znamená rozbitý či rozlámaný, což odpovídá jeho podobě. My se zaměříme na Juliovu a Mandelbrotovu množinu a pokusíme se napsat program generující Buddhabrota.
Pro fraktály lze obecně říct následující pravidla

- soběpodobnost – menší části fraktálu připomínají fraktál jako celek
- složitá struktura jevící se jako chaotická, avšak definovatelná jednoduchou matematickou funkcí
- Hausdorffova dimenze je větší, než dimenze topologická

2. Teoretická část

Fraktální množiny jsou součástí fraktální geometrie. Za zakladatele se považuje Benoit B. Mandelbrot, který poprvé matematicky definoval fraktál. Ty ale byly známy ještě před ním, například v podobě přírodních útvarů.

2.1. Využití fraktálů v praxi

- počítačová grafika – počítačové hry (krajiny, stromy, říční systémy)
- předpovídání chování složitých a chaotických systémů
- úmění

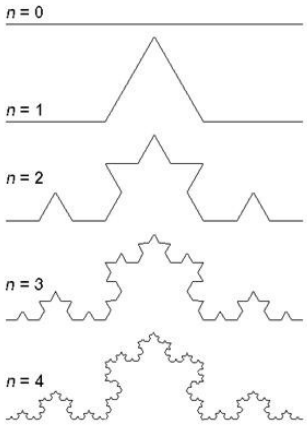
2.2. Jednoduché fraktální útvary

Cantorova množina

Vytvoříme jí nekonečným opakováním daného předpisu. Předpokládejme, že máme úsečku dané délky. Rozdělíme jí na tři stejné díly a prostřední vyjmeme. Výsledkem jsou dvě stejně velké úsečky, které opět rozdělíme na tři stejně velké díly …

Kochova křivka

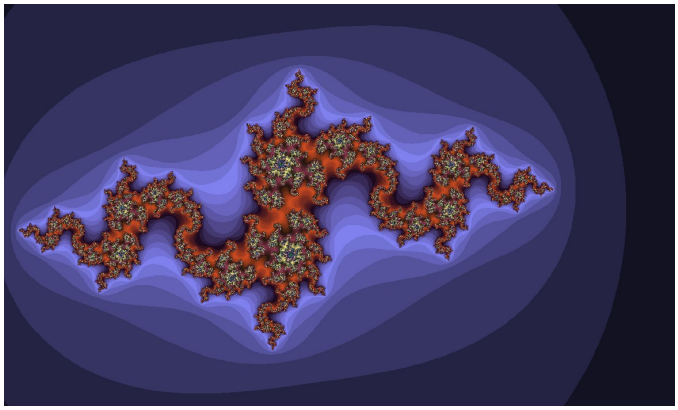
Přímku dané délky rozdělíme na tři části a prostřední část vyměníme za dvě úsečky poloviční délky svírající 60°. Proces se opakuje na všech úsečkách v útvaru.



2.3. Komplexní fraktální útvary

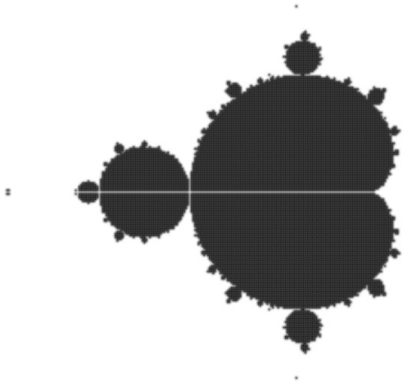
Juliova množina

Vyberme libovolný bod na gaussově rovině a označme jej *c*. Následně vybírejme všechny body gaussovy roviny a provádějme na nich tuto funkci: $Z_{n+1} = Z_n^2 + c$ Pokud tento bod po nekonečnu iterací této funkce dojde k nekonečnu, bod do množiny nepatří. V opačném případě tam patří. Konstanta *c* je pro celou množinu stejná a může být libovolná. Proto je Juliových množin nekonečně mnoho.



Mandelbrotova množina

Vytváří se podle iterace komplexn množiny $Z_{n+1} = Z_n^2 + c$, přičemž konstanta *c* je pro každý počáteční bod rovna právě tomuto bodu.



2.4. Způsoby zobrazování těchto množin

2.4.1. Černobílé fraktály

Spočívá v pouhém rozlišení, zda vybraný bod do fraktálu patří (zobrazen černě) či nikoliv (zobrazen bíle)

2.4.2. Obarvování vnějšku fraktálu

Lze určit barvu daného bodu podle počtu iterací kterých bylo potřeba k určení toho, že do množiny nepatří.

2.4.3. Trasování cest bodů

Způsob zobrazení kdy je na gaussovu rovinu zanesen každý bod Z_n. Tímto znikne mapa míst, kde se body Z_n vyskytují nejvíce, než dojdou k nekonečnu.

3. Praktická část

V rámci praktické části jsme se rozhodli napsat vlastní algoritmus na zobrazení tzv. Buddhabrotu, což je způsob zobrazení Mandelbrotovy množiny.

Náš algoritmus funguje tak, že nejprve vytvoříme dostatečně náhodná komplexní čísla omezená ImMIN< imaginární část <ImMAX, ReMIN< reálná část<ReMAX.
Nejprve sestrojíme reálné a imaginární složky tak, že vybereme náhodné číslo z intervalu 0 - 100 000, které podělíme 100 000, čímž získáváme číslo od 0 do 1. To pak vynásobíme délkou intervalu, ze kterého chceme vybírat komplexní(imaginární) složku čísla. To číslo ještě posuneme o začátek intervalu a tím jsme sestrojili hodnoty, které jsou z našeho pohledu vhodná a dostatečně náhodná.
Tímto algoritmem sestrojíme N čísel, jejichž trasy budeme zakreslovat.

```
int n;
double x,y;
srand(time(NULL));
for (n=0; n<N; n++)
{
  x = ((double)(rand()%100000)/100000.0)*(ReMAX-ReMIN)+ReMIN;
  y = ((double)(rand()%100000)/100000.0)*(ImMAX-ImMIN)+ImMIN;
}
```

Pro každé námi vytvořené číslo zjistíme, zda leží v M. množině nebo ne, vytvoříme proto funkci check, která nám zjistí, zda bod do M. množiny patří nebo ne. Parametry funkce check: reálná a imaginární část daného komplexního čísla, počet iterací, maximální absolutní hodnota, pro kterou budou stále prováděny výpočty, konstanta c, která je v tuto chvíli nulová.
Funkce vrací 1 pokud toto číslo do množiny M. patří a 0, pokud do množiny nepatří a pokud . Výsledek si uložíme do proměnné value.

```
int check(SDL_Surface* screen, double r, double i, int iter, int
bail, Uint32 c)
{
     double zr,zi,zrp; int a;
     zr=r;
     zi=i;
     a=0;
```

Cyklus kontroluje zda bod stále patří do M. množiny dokud není dosáhnuto maximální absolutní hodnoty - pak program vrací 0 a my víme, že toto číslo do M. množiny nepatří, anebo dokud není dosaženo maximálního počtu iterací, program vrací 1 a bod do M. množiny patří

```
	do {
		zrp=zr*zr-zi*zi+r;
		zi=2.0*zr*zi+i;
		zr=zrp;
		a++;
```

Pro zjednodušení je funkce complex funkce, kterou na určeném místě (screen) vykreslí komplexní číslo s reálnou složkou zr a imaginární složkou zi na nějaké souřadnice podle toho čísla, ale pokud již bylo vykresleno, zvýší se odstín barvy o c

```
		if (c>0) {complex(screen, zr, zi, c);}
	}
	while ( (a<iter) && ((zi*zi)+(zr*zr) < (bail*bail)) );
	if ( a>=iter) {return 1;}
		else {return 0;};
}
```

Nyní se podíváme, jakou hodnotu nám pro to naše číslo tato funkce vrací. Přidáme podmínku:

```
	if (value) {check(screen, x, y, iter, bail, c);}
	else {check(screen, x, y, iter, bail, c);}
```

Z nichž vybereme, jestli chceme zobrazovat body, které v množině leží nebo neleží, a naše c nyní bude nenulové a bude označovat barvu, kterou máme bod vykreslit. Voláme tedy funkci check znova, která nyní ale po každé iteraci zakresluje stopu našeho čísla.

Poděkování:

Rádi bychom poděkovali našemu supervizorovi Petru Paušovi, který nás důkladně seznámil s problematikou fraktální geometrie provedl nás všemi úskalími a pomohl nám navrhnout program, jehož výsledkem byl Buddhabrot. Dále bychom rádi poděkovali Linusu Tovarsovi za poskytnutí Kernelu, Richardu Matthew Stallmanovi a jeho vývojovému týmu za posytnutí utilit GNU a v neposlední řadě také všem vývojářům gcc, bez kterýchžto by byl náš svět ochuzen o tolik krás, jako je například obrázek do hloubky prokresleného Buddhabrota.

Reference:

[1] P. Pauš, http://kmlinux.fjfi.cvut.cz/~pauspetr/html/skola/fraktaly/reserse.htm [cit 15. 6. 2010]
[2] Peitgen H.-O., Jurgens H., Saupe D.: "Chaos and Fractals: New Frontiers of Science", Springer-Verlag, New York, 1992