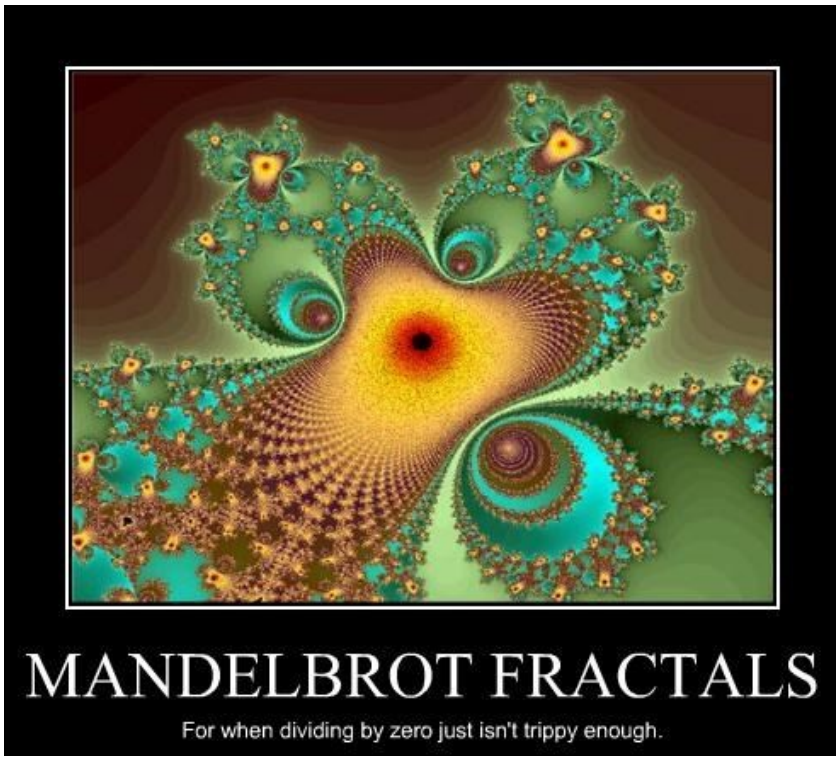
JEGYZŐKÖNYV

Nagyítható és navigálható Mandelbrot fraktál

Korszerű számítástechnikai módszerek a fizikában I.



- Készítette: Juharos Eszter
- Neptun-azonosító: F5UBDG

- Jegyzőkönyv leadásának időpontja: 2021. május 7.

1. Bevezetés

Nagy projektnek a Mandelbrot fraktál nagyítható és navigálható verziójának elkészítését választottam. A fraktálok ábrázolása egy kifejezetten érdekes és gyümölcsöző projekt kellő programozási háttértudással. Mindezek felett az elkészült ábra elénk tár egy igen érdekes jelenséget, az önhasonlóságot. A nagyítható ábrával ezt könnyű belátni.

Jelen jegyzőkönyvben részletezem a Mandelbrot fraktál elméleti hátterét, ami a kód elkészítésének alapjául szolgált, majd néhány nagyobb fejezetre felbontva a kód különböző szakaszait tárgyalom.

2. Elméleti háttér

A Mandelbrot- halmaz azon komplex számokból áll, melyekre az alábbi x_n rekurzív sorozat:

$$x_1 := 0$$

$$x_{n+1} := (x_n)^2 + c$$

nem tart végtelenbe, azaz abszolút értékben (hosszára nézve) korlátos.

A Mandelbrot halmaz a komplex számsíkon ábrázolva létrehozza az azonos nevű fraktált.

Essen néhány szó a fraktálokról is. A fraktálok végtelenül komplex geometriai alakzatok, amikre jellemző, hogy felületük illetve határaik végtelenül gyűröttek, érdesek vagy szakadásosak. Ezen kívül fontos tulajdonságuk a korábban már említett önhasonlóság. Ezeken felül remekül alkalmasak a "végtelen" mint fogalom megértésének segítésére és nem utolsó sorban esztétikai értékkel is bőven bírnak.

2.1. Az algoritmus

A kódom alapja az "escape time algorithm", aminek elve, hogy az összes pont x és y komponensére megtörténik a kiszámolás, és az érték alapján egy bizonyos színt hozzárendel a program a pixelértékhez.

Ez az x y koordinátájú pixel adja meg az iteráció kezdetét, majd a következő pixel számításához ezt a kezdőértéket veszi alapul. Minden lépésnél ellenőrzi a program, hogy a pixel nem lépte-e túl az "escape condition"-t, ha igen, akkor a pixel elkészül és a program továbblép a következőre.

A pixel színe ezek után az alapján kerül kiválasztásra, hogy mennyi iteráció alatt éri el az "escape condition"-t.

3. Program felépítése

A programot igyekeztem a tárgyon tanult elvek szerint strukturálni, így részét képezi egy header és egy main függvény, a kimenete pedig egy ppm formátumú képfájl.

3.1. Header

A headerbe csoportosítottam az érdemi számításokat végző függvényeket. Az összes függvényt templatekkel definiáltam a rugalmasabb használat érdekében.

A MaptoReal és a MaptoImaginary függvények, ahogy azt a nevük is sugallja, azzal az értékkel térnek vissza, amely a pixel tényleges koordináta értéke lesz x-re és y-ra, ami a komplex szám képzetes illetve valós része definíció szerint.

A findMandelbrot függvényben lényegében az algoritmus pszeudokódját valósítom meg. Fő része a while ciklus, aminek feltételei a fent már tárgyalt "escape condition"-ök. A while ciklus hasában a számítás a következő módon zajlik:

$$x = Re(z^2 + c) = x^2 - y^2 + x_0$$

$$y = Im(z^2 + c) = 2xy + y_0$$

Az itt említett függvények segítségével a main jelentősen lerövidül és egyszerűsödik, most már csak meg kell hívni.

3.2. Main

A Mandelbrot.cpp lényegében csak a main függvényt tartalmazza.

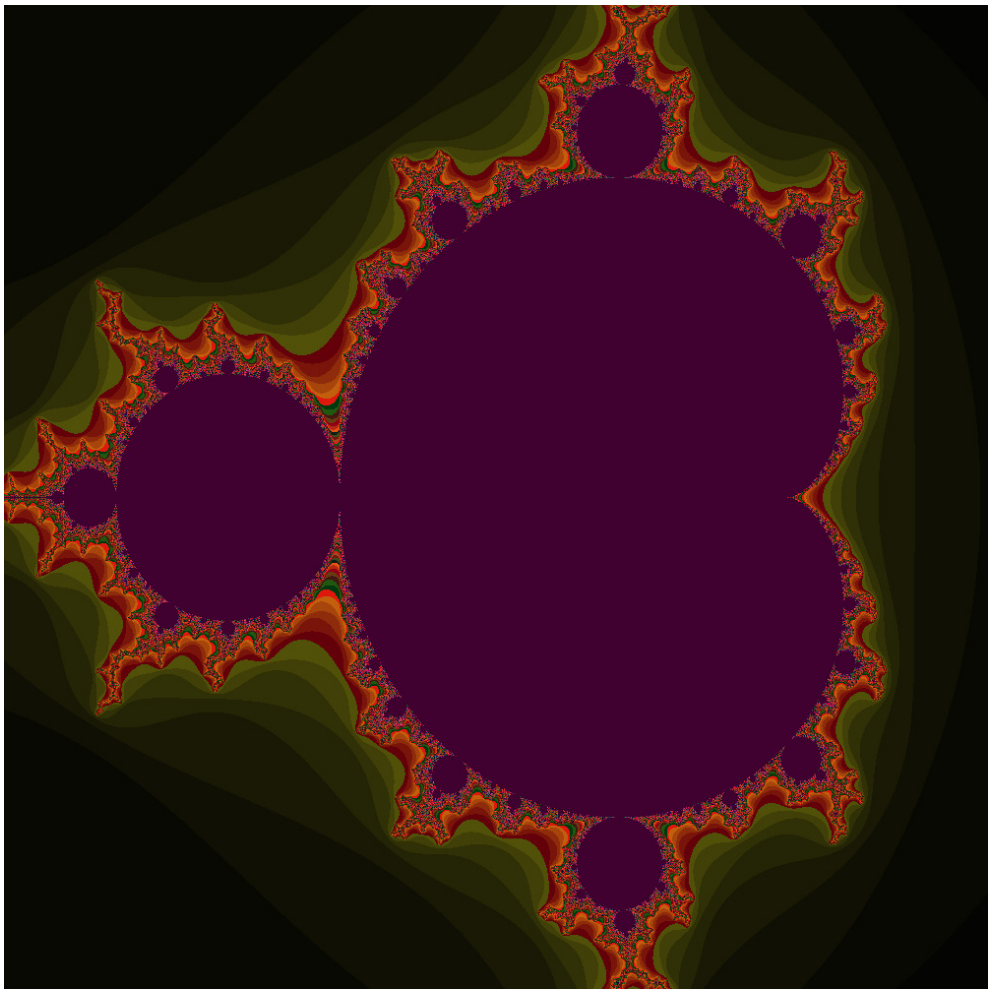
Ennek elején deklarálom a kép szélességét és magasságát, ezt ugyanis annyira nem fontos elállítani a továbbiakban, valamint deklarálom a maxN maximum iterációt. Utóbbi azért felel, hogy a Mandelbrot ábra mennyire legyen részletes. Kb 1000 nagyságrenű értéknél lesz a leg esztétikusabb az elkészült ábra. Ezzel az értékkel is lehet játszani, növelve vagy csökkentve az ábra komplexitását, de úgy gondoltam default értknek ez megfelel.

A kiadott feladatban benne volt az a kitétel, hogy az ábra nagyítható és navigálható legyen, ennek megvalósítása nagy fejtörést okozott, mivel az interaktív ábra készítése komolyabb módszerek alkalmazását igényelné, így végül azt a megoldást választottam, amit a korábbi házi feladatoknál sikeresen tudtam alkalmazni. Indításkor a program bekér 4 értéket, amik a minimum valós, maximum valós, minimum képzetes és maximum képzetes. Ezen paraméterek megválasztásával lehet a zoomolást és a navigációt kivitelezni, ugyanis a headerben definiált függvények a számítást ezek alapján eghatározott rangen végzik el.

Ha a sztenderd Mandebrot ábrát szeretnénk visszakapni, akkor a következő paramétereket ajánlom:

minReal	-1.5
maxReal	0.7
minImaginary	-1.0
maxImaginary	1.0

Természetesen ezek szabadon változtathatóak attól függően, hogy mennyire akarunk bele nagyítani illetve hogy az ábra melyik részére vagyunk kíváncsiak. A sztenderd paraméterekkel elkészített ábra:



A main függvény utolsó nagy állomása a színezés, amit a fent említett elv alapján RGB értékekkel hajtottam végre. Ennek előny, hogy teljesen szabadon

választható, hogy milyen színskálán fog mozogni a Mandelbrot, ugyanis az n értékének különböző manipulációit használom fel a keveréshez.

A cél egy pink-orange-lilac paletta lett volna, de nagyon megkedveltem ezt a furcsa színskálát, amit véletlenül, az RGB számolási értékek állítgatásával hoztam létre, így ezt hagytam a véglegesben is.

4. Diszkusszió

Összességében nagyon élveztem ezt a projektet és meg vagyok elégedve az eredménnyel. Fejlesztési lehetőségnek látom még a konzol applikáció elkészítését, amit esztétikus ábrák iránti érdeklődésemből fakadóan a jövőben valószínűleg meg is fogok csinálni.

Kifejezetten megkedveltem a fraktálok témakörét és könnyedén ment a C++ kód elkészítése, amit szintén nagy eredményenek könyvelek el.

Felhasznált irodalom

- Mandelbrot-halmaz- Wikipédia
- Plotting algorithms for the Mandelbrot set- Wikipédia
- Aesthetic Function Graphposting repository- Masterdesky
- Mandelbrot set in C++ from scratch- Medium
- Különböző fórum posztok