Fakultet elektrotehnike i računarstva

Računalna grafika, 3. lab. vježba:

WebGL istraživač Mandelbrotovog skupa korištenjem sjenčara

Petar Mihalj, 0036497900

Siječanj 2021.

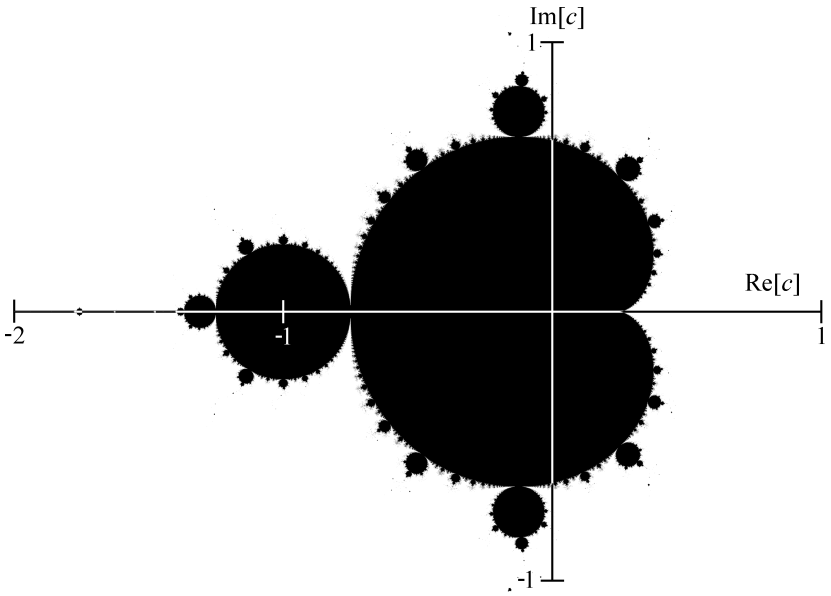
1. Matematička pozadina

Mandelbrotov skup definiramo kao skup kompleksnih brojeva, za koje sljedeća iteracijska formula ne divergira:

$$z_i = z_{i-1}^2 + c$$
$$z_0 = 0$$

Konstanta c predstavlja broj kojem testiramo pripadnost u Mandelbrotovom skupu.

Budući da ova definicija nije pogodna za praktično testiranje pripadnosti, koristimo činjenicu da svi kompleksni brojevi kojima je apsolutna vrijednost veća od 2 nužno divergiraju. Naravno, ako u bilo kojem koraku iteracije z postane apsolutne vrijednosti veće od 2, iteracijska formula će divergirati.



Slika 1: Mandelbrotov skup
(https://math.stackexchange.com/a/1732973)

Na Slici 1 vidljiv je prikaz Mandelbrotovog skupa u kompleksnoj ravnini. Crni dijelovi predstavljaju one točke koje nakon nekog fiksnog broja iteracija nisu divergirale u osnovnoj formuli.

Alternativni prikaz M. skupa temelji se na bojenju i onih točaka koje su divergirale, ali tako da boja ovisi o broj iteracija koje su točki trebale da izađe iz kruga radijusa 2, centriranog u nuli.

Jasno je da za svako izračunavanje izgleda M. skupa treba iterirati istu formulu na mnogo točaka; ovo je idealan problem za korištenje sjenčara, minimalnih programa koji se izvršavaju na grafičkoj kartici.

2. Funkcionalnosti sustava

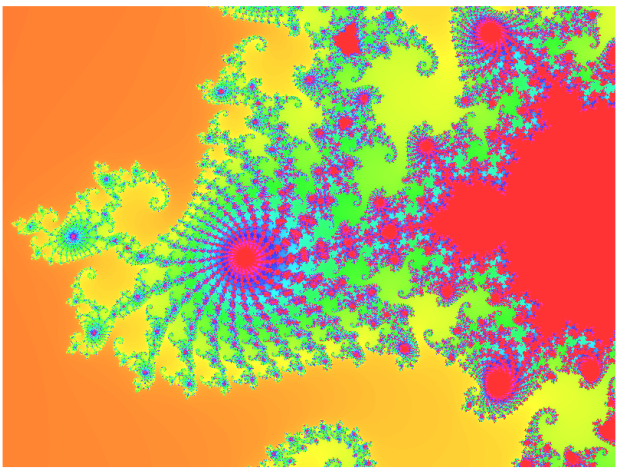
WebGL istraživač M. skupa ostvaren je korištenjem WebGL-a, grafičke knjižnice namijenjene korištenju u web preglednicima.

Korisničko sučelje nudi intuitivne komande za zumiranje skupa.

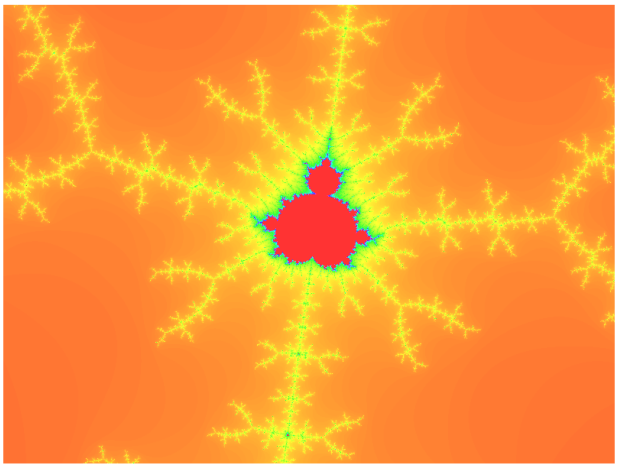
Sustav je responzivan čak i s visokim rezolucijama, vrijeme za crtanje jedne scene neprimjetno je korisniku, čak i kad je pozadinski procesor iz obitelji Intel i7, a ne dedicirana grafička kartica.

Slika 2: Korisničko sučelje

U nastavku su dani neki zanimljivi dijelovi skupa:



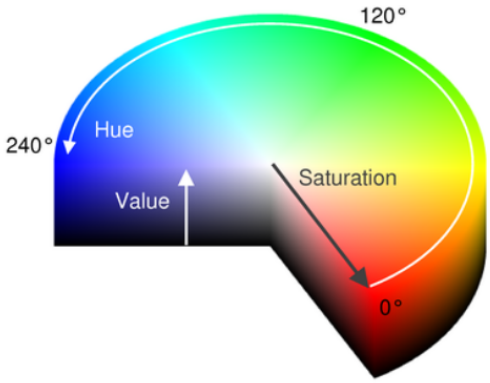
Slika 3



Slika 4

Na slici 4 uočavamo zanimljiv fenomen; Mandelbrotov skup u sebi sadrži dijelove koji sliče njemu samom! Ovo je naravno razlog zašto ga svrstavamo u fraktale.

Preslikavanje iz broja iteracija u boju temelji na prelasku u HSV reprezentaciju boja:



Slika 5: HSV (https://dsp.stackexchange.com/a/30263)

Broj iteracija potrebnih za divergenciju točke skaliramo na interval [0,1]. Ako točka ne divergira, broj iteracija definiramo maksimalnim, i takva točka dobiva vrijednost 1.

Taj interval koristimo za uzorkovanje boje s vrijednosti *hue*. Dakle, točka koja treba 0 iteracija bit će crvena, a ona koja treba otprilike pola od maksimalnog broja, bit će svijetlo plave boje (suprotno od crvene). Ostale dvije komponente HSV reprezentacije fiksirane su na vrijednosti koje daju dobar efekt (saturation = 0.8, value = 1.0).

3. Analiza sjenčara

```
1 attribute vec2 coords;
2 attribute vec2 c;
3 varying vec2 fragC;
4 void main(void) {
5     fragC = c;
6     gl_Position = vec4(coords, 0.0, 1.0);
7 }
```

Slika 6: sjenčar vrhova

Sjenčar vrhova ima minimalnu funkcionalnost. Njemu je zadaća primiti koordinate u prostoru ekrana ([-1,1] x [-1,1] koordinatni sustav) ovdje nazvane coords, i njih postaviti za izračunatu poziciju točke.

Naime, sustav funkcionira tako da prikaže dva trokuta koja čine kvadrat koji obuhvaća koordinatni sustav ekrana. Ta dva trokuta osim svojih pozicija imaju i pozicije u kompleksnoj ravnini, c, koje se sjenčaru fragmenata šalju u obliku varying varijable. Svaki pixel trokuta u sjenčaru vrhova dobiva interpolirane vrijednosti vrhova trokuta - tako realiziramo “prebacivanje” svih točaka kompleksne ravnine na GPU.

Sjenčar fragmenata treba, za svaki kompleksni broj fragC izračunati boju. Provodimo osnovnu rekurzivnu formulu dok ne dođemo do zadanog maksimalnog broja iteracija. Boja se iz broja iteracija izračunava na već spomenuti način. Sjenčar fragmenata realiziran je u skladu s ograničenjima jezika GLSL; na primjer, ograničeni smo konstantnim izrazima u petlji.

```
precision highp float;
precision highp int;

varying vec2 fragC;
uniform int max_iters;
// https://github.com/hughsk/glsl-hsv2rgb
vec3 hsv2rgb(vec3 c) {
    vec4 K = vec4(1.0, 2.0 / 3.0, 1.0 / 3.0, 3.0);
    vec3 p = abs(fract(c.xxx + K.xyz) * 6.0 - K.www);
    return c.z * mix(K.xxx, clamp(p - K.xxx, 0.0, 1.0), c.y);
}

void main(void) {
    vec2 z = vec2(0.0,0.0);
    vec2 z_new = vec2(0.0,0.0);
    float count = 0.0;
    for(int i=0;i<1000000;++i){
        if (i == max_iters){
            count = float(max_iters);
            break;
        }
        z_new[0] = z[0] * z[0] - z[1] * z[1] + fragC[0];
        z_new[1] = z[0] * z[1] * 2.0 + fragC[1];
        z = z_new;
        float abs_val = z_new[0]*z_new[0]+z_new[1]*z_new[1];
        if (abs_val >= 4.0){
            count = float(i);
            break;
        }
    }

    float max_iters_fl = float(max_iters);
    float hue = count / max_iters_fl;
    gl_FragColor = vec4(hsv2rgb(vec3(hue, 0.8, 1.0)), 1.0);
}
```

Slika 7: sjenčar fragmenata

4. Upute za pokretanje:

Kod cijelog projekta, uključujući ovu dokumentaciju, dostupan je na:

https://github.com/PetarMihalj/MandelbrotExplorer

Nakon preuzimanje programskog koda, pokrenite ga otvaranjem main.html datoteke u svom web pregledniku. Naravno, preglednik mora imati podršku za WebGL.

Zahvala:

Hvala kolegi Filipu Husnjaku (https://github.com/FilipHusnjak) na ispravku programa da bi bio kompatibilan s operacijskim sustavom Windows (promjena dizajna sjenčara).