Šola

Risanje fraktalov v Pythonu

Maturitetna seminarska naloga iz Informatike

Avtor: Ime in Priimek

Mentor: Ime in Priimek

Kraj in datum: Kraj, 17. april 2018

1 Kazalo

1.1 Kazalo vsebine

1.2 Kazalo slik

2 Povzetek

V nalogi sem predstavil kaj so fraktali na splošno, kaj je Mandelbrotova množica, kaj je Juliajeva množica in kaj je trikotnik Sierpinskega. Opisal sem tudi, kako sem uporabil moč *Python*-a in *OpenGL*-a za učinkovito risanje teh fraktalov. V *Python*-u sem izdelal tudi grafični uporabniški vmesnik (*GUI*), ki omogoča gledanje narisanih fraktalov in spreminjanje nekaterih izmed fraktalovih lastnosti (npr: stopnja trikotnika Sierpinskovega in število iteracij pri Juliajevi in Mandelbrotovi množici).

In this paper I present what fractals are in general, what the Mandelbrot set is, what the Julia set is and what Sierpinski's triangle is. I also present how I use the power of Python and OpenGL for efficient rendering of these fractals. I also present the graphical user interface (GUI), which enables us to view the rendered fractals and allows us to change specific properties and attributes of the rendered image and fractal (e.g. the level of Sierpinski's triangle and the number of iterations of the Julia and Mandelbrot set).

3 Ključne besede

Python, Mandelbrotova množica, Juliajeva množica, Sierpinskov trikotnik, OpenGL.

4 Uvod

V seminarski nalogi bom predstavil risanje fraktalov v programskem jeziku *Python*. Fraktali, ki jih bom narisal so Mandelbrotova množica, Juliajeva množica in trikotnik Sierpinskega. Risanje fraktalov v živo je za *CPU* zelo zahtevno, zato bom uporabil moč paralelizacije grafičnih kartic (*GPU*) z uporabo grafičnega vmesnika *OpenGL*. Za delovanje napisanega programa je torej najprej potrebno naložiti knjižnice za *Python*: *Pyglet (v 1.3.0)* in *ModernGL (v 5.0.7)*. Ti dve knjižnici omogočata uporabo *OpenGL*-a znotraj *Pythona* in pa risanje grafičnega uporabniškega vmesnika (*GUI*), s katerim lahko uporabnik spreminja različne parametre fraktalov.

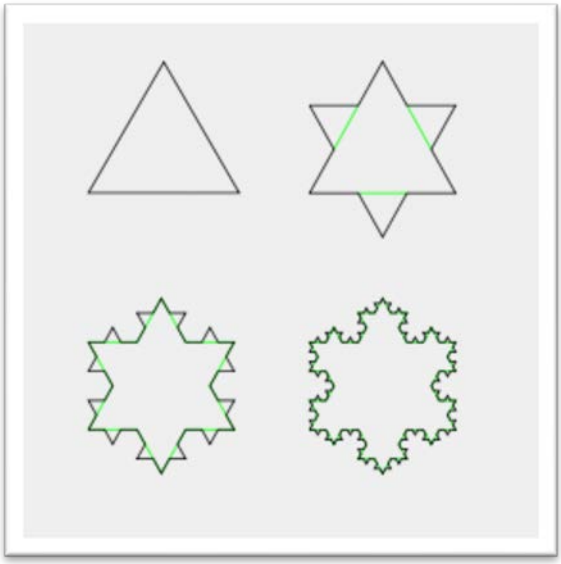
5 Risanje fraktalov

5.1 Kaj so fraktali?

»Fraktal je v matematiki objekt, ki ima vsaj eno od naslednjih lastnosti: vsebuje podrobnosti pri poljubni veliki ali majhni povečavi, je preveč nepravilne oblike za opis z običajnimi geometrijskimi prijemi, je natančno ali statistično samopodoben, njegova razsežnost je večja od njegove topološke razsežnosti ali pa je določen rekurzivno.« (Fraktal, 2015).

Fraktali so neskončni vzorci. So kompleksni vzorci, ki so si podobni na poljubni povečavi. Ustvarjeni oziroma določeni so s ponavljanjem istega procesa, ki se nanaša na isto funkcijo. Proces imenujemo *rekurzija* (beseda izhaja iz latinščine in pomeni nanašajočega na samega sebe). Fraktalni vzorci se pogosto pojavljajo v naravi in jih lahko najdemo v drevesih, rekah, obalah ... Bolj abstraktni fraktali – abstraktni v smislu, da ne obstajajo v naravi, so ustvarjeni z računalnikom, ki rekurzivno ponavlja isto funkcijo. Pomembna razlika med obema je, da naravnih fraktalov ne moremo povečevati v neskončnost, medtem ko matematične lahko.

Matematika fraktalov se je prvič začela razvijati v 17. stoletju, ko je matematik in filozof Gottfried Leibniz razmišljal o rekurzivnosti in samopodobnosti. Naredil je napako in mislil, da je samo premica fraktal, ampak je bil prvi bolj znani znanstvenik, ki je razmišljal v tej smeri. Zaradi odpora do novih nepoznanih konceptov so jih pogosto imenovali matematične pošasti.

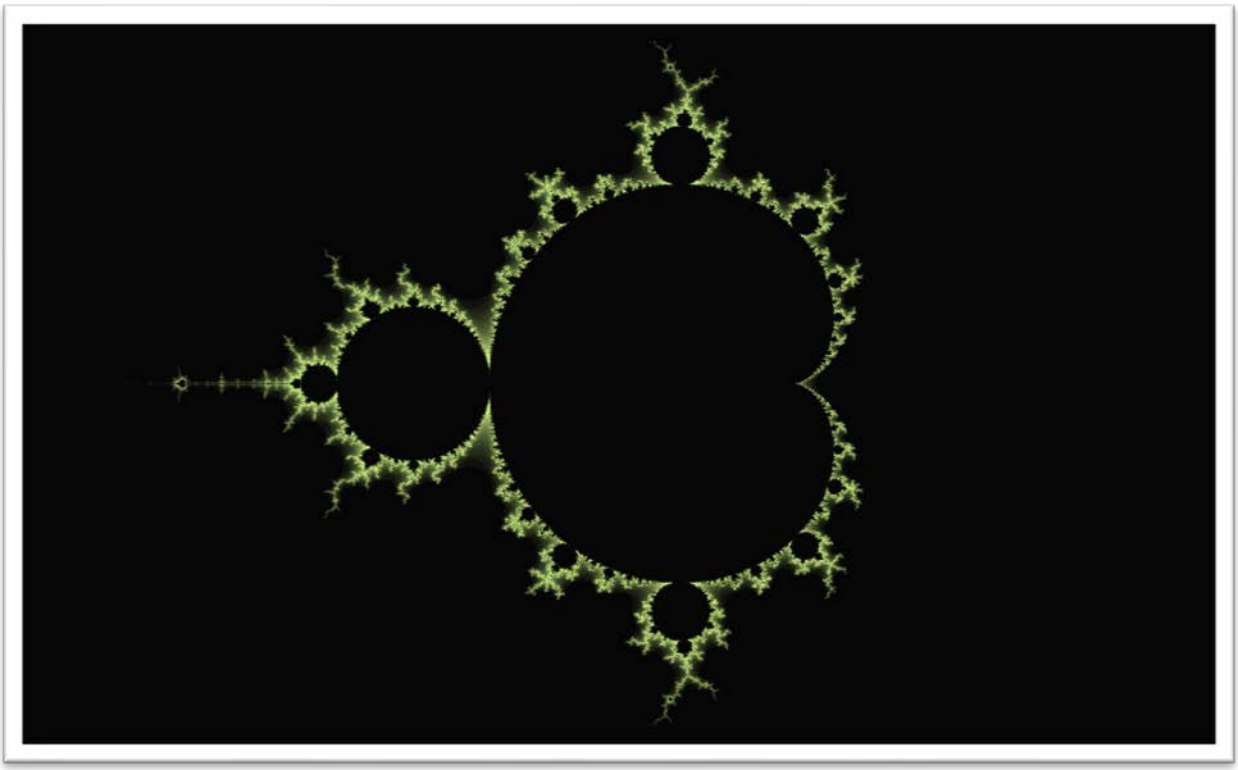


Slika 1 - Kochova snežinka

5.2 Mandelbrotova množica

5.2.1 Kaj je Mandelbrotova množica?

Mandelbrotova množica je v matematiki množica točk v kompleksni ravnini, katere meja tvori fraktal in njena matematična definicija je: »množica kompleksnih vrednosti *c*, za katere orbita vrednosti 0 pod iteracijo kompleksnega kvadratnega polinoma $z_{n+1} = z_n^2 + c$ ostaja omejena. Oziroma, kompleksno število c leži v Mandelbrotovi množici, če začnemo z $z_0 = 0$ in ponavljamo iteracijo, absolutna vrednost z_n nikoli ni večja od določenega števila (odvisnega od *c*) ne glede na to kako velik je *n*.« (Mandelbrotova množica, 2014). Množica je ime dobila po francosko-ameriškem matematiku Benoîtu B. Mandelbrotu. Znana je po svoji estetični privlačnosti ter zapleteni zgradbi, ki temelji na preprosti definiciji.



Slika 2 - Mandelbrotova množica

5.2.2 Risanje Mandelbrotove množice

Risanje Mandelbrotove množice je računsko zelo zahteven postopek. Naiven pristop bi bil vsako kompleksno število spremeniti v piksel in ga ustrezno pobarvati vsakega posebej. Vendar ta način bi bil prepočasen za risanje »v živo« z uporabo *CPU*-ja. Zato sem se odločil uporabiti sposobnost grafične kartice (*GPU*), ki omogoča visoko paralelizirano izvajanje programčkom, ki jim pravimo »*shader*«-ji. *Shader*-ji so napisani

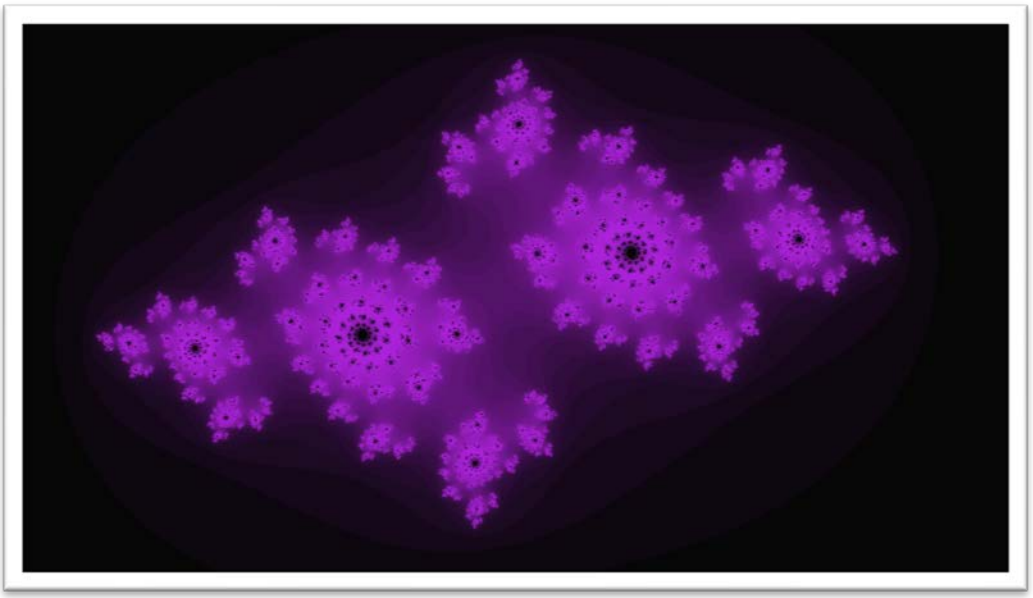
v programskem jeziku podobnem *C*-ju. Na ta način vsak *shader* dobi kot vhodni podatek koordinate enega piksla, ki ga potem pretvori v kompleksno število in po definiciji Mandelbrotove množice izvede iteracijo ter tako ugotovi, kako pobarvati ta piksel. Piksle sem se odločil pobarvati glede na število iteracij, ki jih je bilo potrebnih, da je program ugotovil ali je kompleksno število del množice.

```
float aspect_ratio = w_width / float(w_height);
float set_width = 2.5f * aspect_ratio;

// pretvorba koordinat pikslov v kompleksna števila
vec2 c = vec2((gl_FragCoord.x / w_width - 0.5f) * set_width * zoom,
                (gl_FragCoord.y / w_height - 0.5f) * 2.5f * zoom)
                - center;
vec2 z = vec2(0.0f, 0.0f);

int i;
for (i = 0; i < iterations; i++) {
     z = vec2(z.x * z.x - z.y * z.y, 2 * z.x * z.y) + c;
     if (length(z) >= 2) break;
}

// barvanje
if (i == iterations)
     frag_color = vec4(0, 0, 0, 1);
else
     frag_color = vec4(rgb, 1) * cos(90 * (1 - i /
float(iterations)));
```

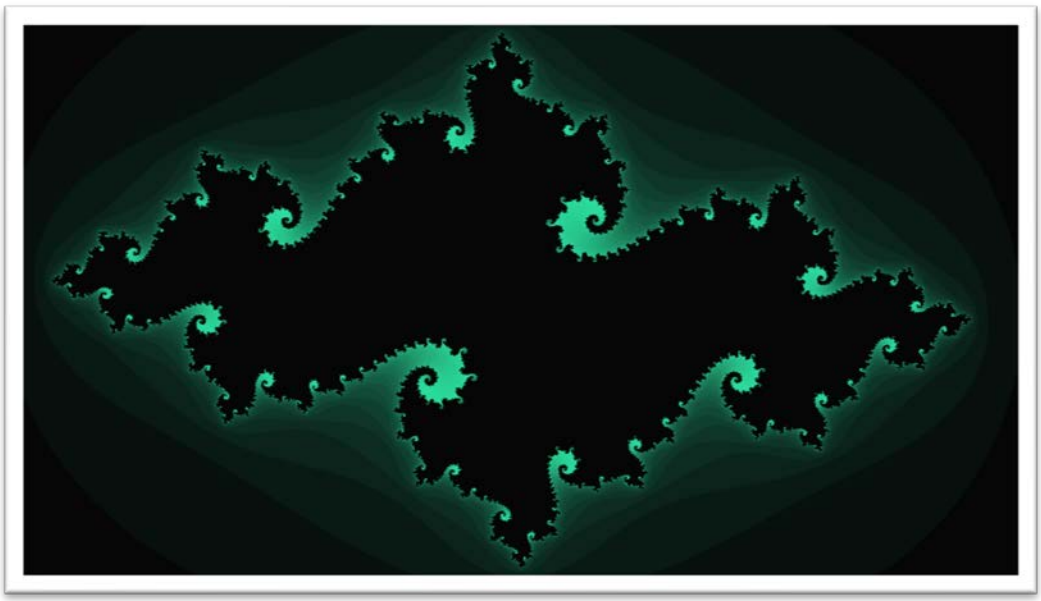
5.3 Juliajeva množica

5.3.1 Kaj je Juliajeva množica?

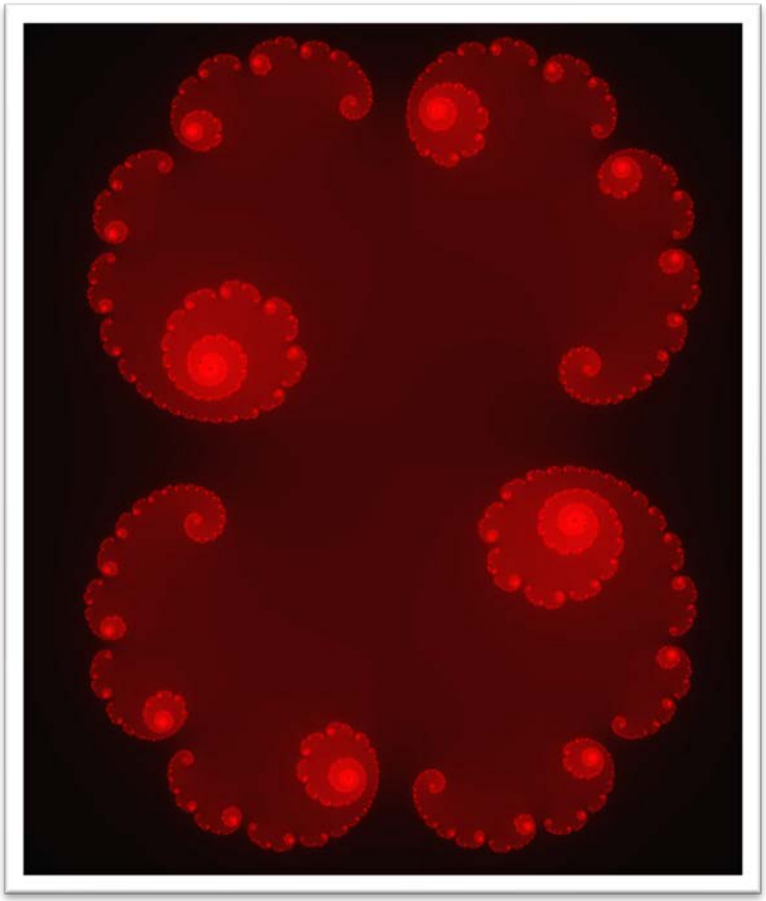
Juliajeve množice so množice kompleksnih števil definirane z iteracijo kompleksnih funkcij ($f_c(z) = z^2 + c$). Definiramo jo kot množico vseh kompleksnih števil, katerih orbite pri iteraciji ostanejo omejene in ne pobegnejo v neskončnost. Rob te množice pa je fraktal, ki ga imenujemo Juliajeva množica. S spreminjanjem konstante *c*, dobimo različne vzorce.



Slika 3 - Juliajeva množica (c = -0,735 - 0,3071i)



Slika 4 - Juliajeva množica (c = -0,735 + 0,1179i)



Slika 5 - Juliajeva množica (c = 0,29 - 0,0071i)

5.3.2 Risanje Juliajeve množice

Risanje Juliajeve množice je zelo podobno risanju Mandelbrotove množice. Razlika je le v začetnih vrednostih kompleksnih števil *z* in *c*. Program omogoča, da število *c* spreminjamo z uporabo tipkovnice, število *z* pa je le koordinata trenutnega piksla v *shader*-ju. Piksle program tudi pobarva podobno kot pri Mandelbrotovi množici, vendar pri tem uporabi funkcijo *sin x* namesto *cos x*.

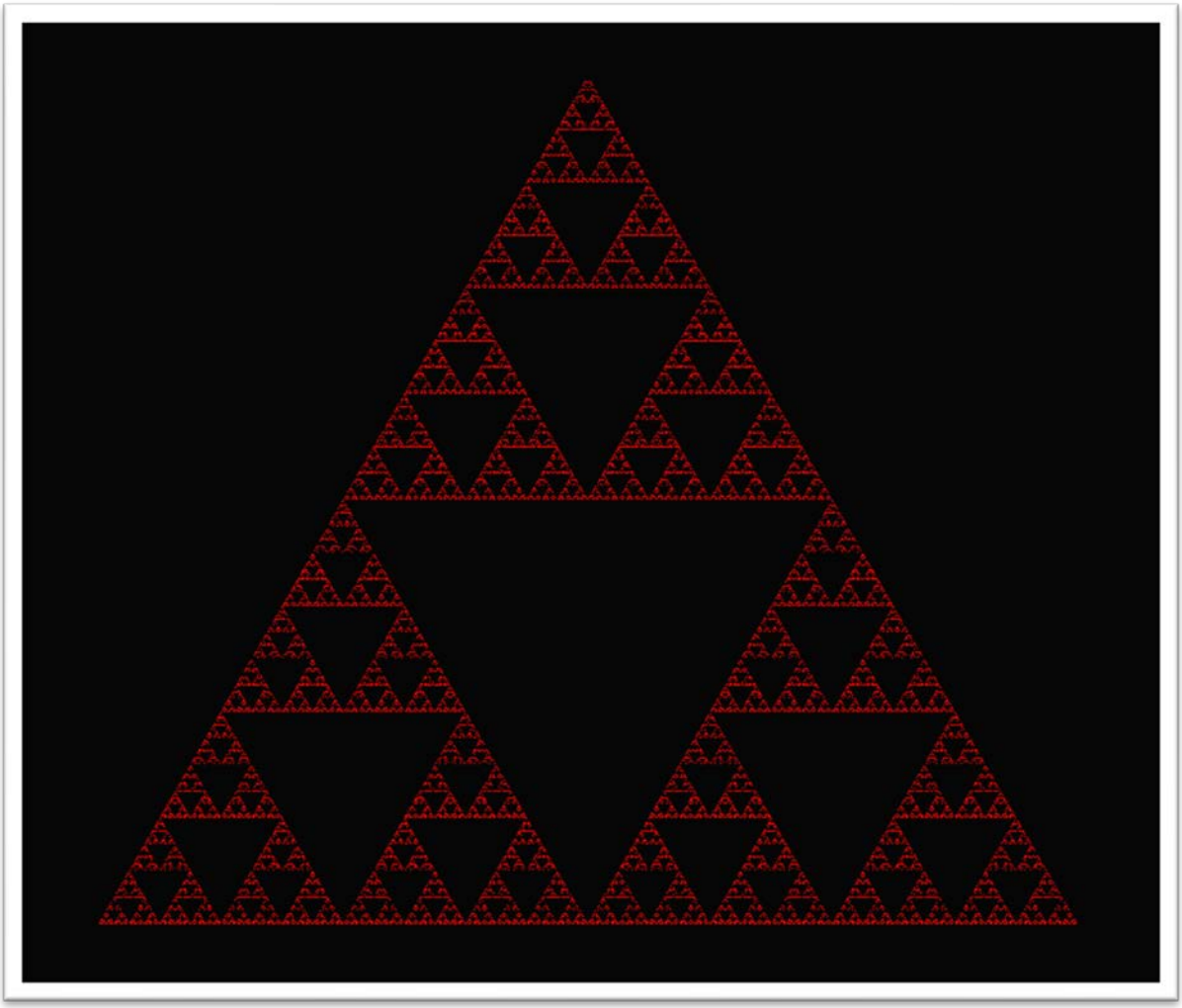
```
float aspect_ratio = w_width / float(w_height);
float set_width = 2.5f * aspect_ratio;

// pretvorba koordinat pikslov v kompleksna števila
vec2 z = vec2((gl_FragCoord.x / w_width - 0.5f) * set_width * zoom,
              (gl_FragCoord.y / w_height - 0.5f) * 2.5f * zoom)
              - center;

int i;
for (i = 0; i < iterations; i++) {
     z = vec2(z.x * z.x - z.y * z.y, 2 * z.x * z.y) + c;
     if (length(z) >= 2) break;
```

```
}

// barvanje
if (i == iterations)
     frag_color = vec4(0, 0, 0, 1);
else
     frag_color = vec4(rgb, 1) * sin(180 * (i / float(iterations)));
```

5.4 Trikotnik Sierpinskega

5.4.1 Kaj je trikotnik Sierpinskega?

Trikotnik Sierpinskega je fraktal z osnovno obliko enakostraničnega trikotnika, ki je rekurzivno razdeljen na manjše enakostranične trikotnike. Matematično je to preprost primer oblike, ki se jo da ponoviti na poljubni povečavi ali pomanjšavi. Imenovana je po Poljskem matematiku Wacławu Sierpińskemu, vendar je bila oblika uporabljena kot okras že stoletja prej.



Slika 6 - Sierpinskov trikotnik (stopnja 9)

5.4.2 Risanje Sierpinskovega trikotnika

Za risanje trikotnika Sierpinskega sem se odločil za drugačen pristop, kot za risanje ostalih fraktalov. Pri risanju npr. Mandelbrotove množice je bilo namreč potrebno za vsak piksel posebej izračunati, kako ga pobarvati. Pri risanju Sierpinskovega trikotnika pa gre za iterativno oziroma rekurzivno risanje trikotnikov, za katere je potrebno

shraniti tri koordinate točk oglišč trikotnikov. Glede na podano stopnjo fraktala, program izračuna in shrani koordinate oglišč vseh trikotnikov vključenih v Sierpinskov trikotnik in nato vse te trikotnike izriše in pobarva. Pri tem uporabljam *GPU* za risanje trikotnikov in *CPU* za izračun oglišč trikotnikov.

Algoritem za izračun vseh točk Sierpinskovega trikotnika izgleda tako:

1. Začnemo z enakostraničnim trikotnikom.
2. Trikotnik razdelimo na štiri enako veliko enakostranične trikotnike in odstranimo sredinski trikotnik.
3. Drugi korak ponavljamo v neskončnost (ali do neke vrednosti).

Slika 7 - Grafični prikaz algoritma za risanje Sierpinskovega trikotnika

```
def _create_sierpinski_array(self):
     vert_array = []

      self.add_sierpinski_vertices(vert_array, 0, Vec2(-0.5, -
0.433013), Vec2(0.5, -0.433013), Vec2(0.0, 0.433013))

      # seznam je potrebno preoblikovati v obliko primerno za OpenGL
     return struct.pack('{}f'.format(len(vert_array)), *vert_array)

def add_sierpinski_vertices(self, array, level, v1, v2, v3):
        """ Funkcija, ki rekurzivno dodaja tocke sierpinskovega
trikotnika v seznam 'array'. Tocke so v obliki Vec2. """

     if level >= self.sierpinski_levels:
          array.extend((v1.x, v1.y, v2.x, v2.y, v3.x, v3.y))
          return

     self.add_sierpinski_vertices(array, level + 1, v1, Vec2((v1.x +
v2.x) / 2, v2.y), Vec2((v1.x + v3.x) / 2, (v1.y + v3.y) / 2))
     self.add_sierpinski_vertices(array, level + 1, Vec2((v1.x +
v2.x) / 2, v1.y), v2, Vec2((v2.x + v3.x) / 2, (v1.y + v3.y) / 2))
     self.add_sierpinski_vertices(array, level + 1, Vec2((v1.x +
v3.x) / 2, (v1.y + v3.y) / 2), Vec2((v2.x + v3.x) / 2, (v1.y + v3.y)
/ 2), v3)
```

6 Grafični uporabniški vmesnik (*GUI*)

6.1 Opis

Za izdelavo grafičnega uporabniškega vmesnika sem uporabil odprtokodno knjižnico za *Python: pyglet v1.3.0*. Knjižnica mi omogoča risanje z *OpenGL*-om, uporabo tipkovnice in miške, osveženje slike, spreminjanje velikosti prikazanega okna in druge standardne operacije grafičnih uporabniških vmesnikov.

6.2 Uporaba

Za uporabo programa se večinoma uporablja tipkovnica in miška. Miška omogoča samo spreminjanje povečave z uporabo kolesca. Tipkovnica omogoča spreminjanje lastnosti trenutno prikazanega fraktala, zato nekatere tipke delujejo samo pri določenem fraktalu. Tipke skupne vsem fraktalom so:

- *Puščica levo*: premik kamere v levo,
- *Puščica desno*: premik kamere v desno,
- *Puščica gor*: premik kamere gor,
- *Puščica dol*: premik kamere dol,
- *Q*: pri Mandelbrotovi in Juliajevi množici zmanjša število iteracij, pri Sierpinskovem trikotniku pa zmanjša stopnjo rekurzije,
- *E*: pri Mandelbrotovi in Juliajevi množici poveča število iteracij, pri Sierpinskovem trikotniku pa zviša stopnjo rekurzije,
- *N*: naslednji fraktal,
- *P*: izpis lastnosti trenutnega fraktala na standardni izhod,
- *R*: naključna sprememba barve fraktala,
- *1*: spreminjanje rdeče komponente fraktala,
- *2*: spreminjanje zelene komponente fraktala,
- *3*: spreminjanje modre komponente fraktala.

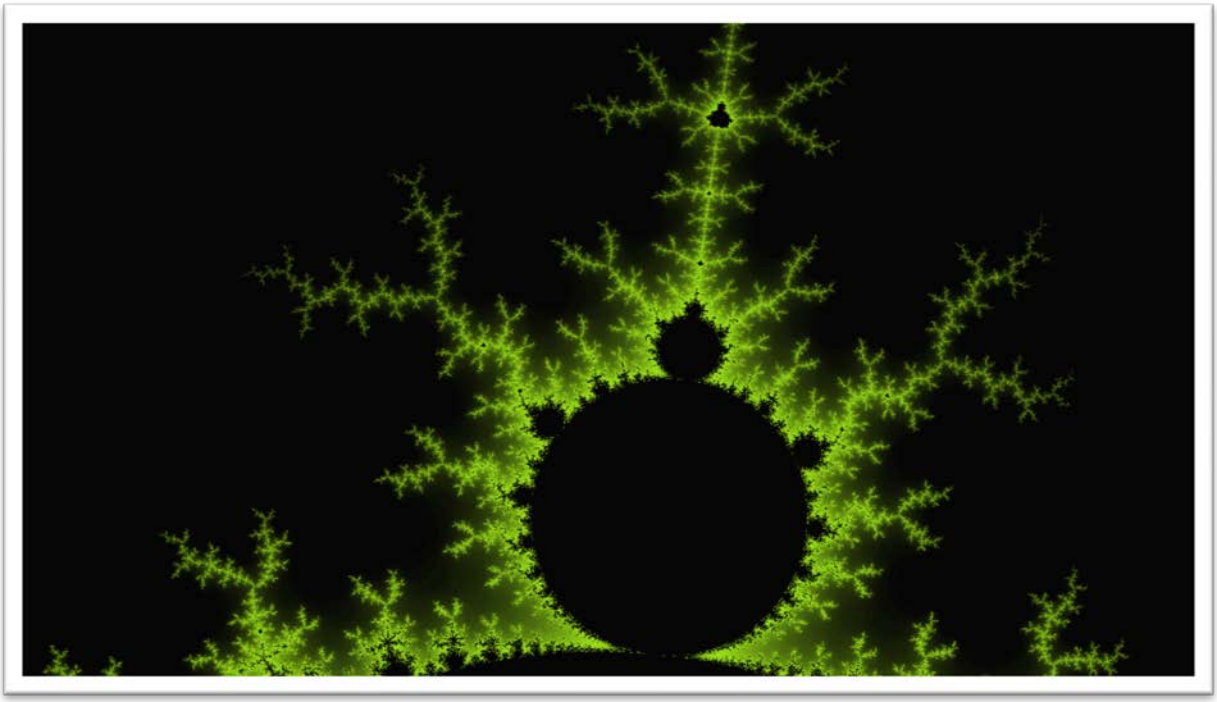
Tipke, ki delujejo samo pri Juliajevem fraktalu za spreminjanje c parametra so:

- *W*: višа imaginarno komponento *c* parametra,
- *S*: manjša imaginarno komponento *c* parametra,
- *A*: manjša realno komponento *c* parametra,
- *D*: višа realno komponento *c* parametra.

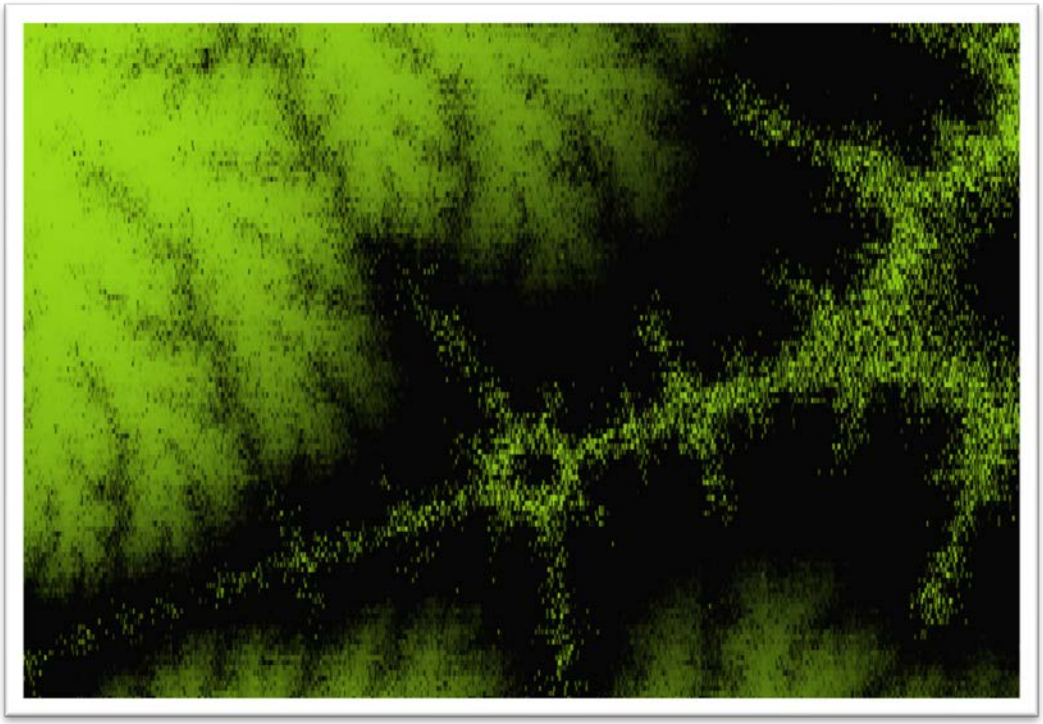
7 Zaključek

Fraktali so estetsko zelo privlačni liki, ki temeljijo na preprostih matematičnih definicijah. Njihova algoritmična implementacija je zato pogosto tudi preprosta, vendar pa je za njihovo učinkovito risanje potrebno veliko računalniške moči. Kljub temu, se jih danes da učinkovito risati na skoraj vsakem računalniku.

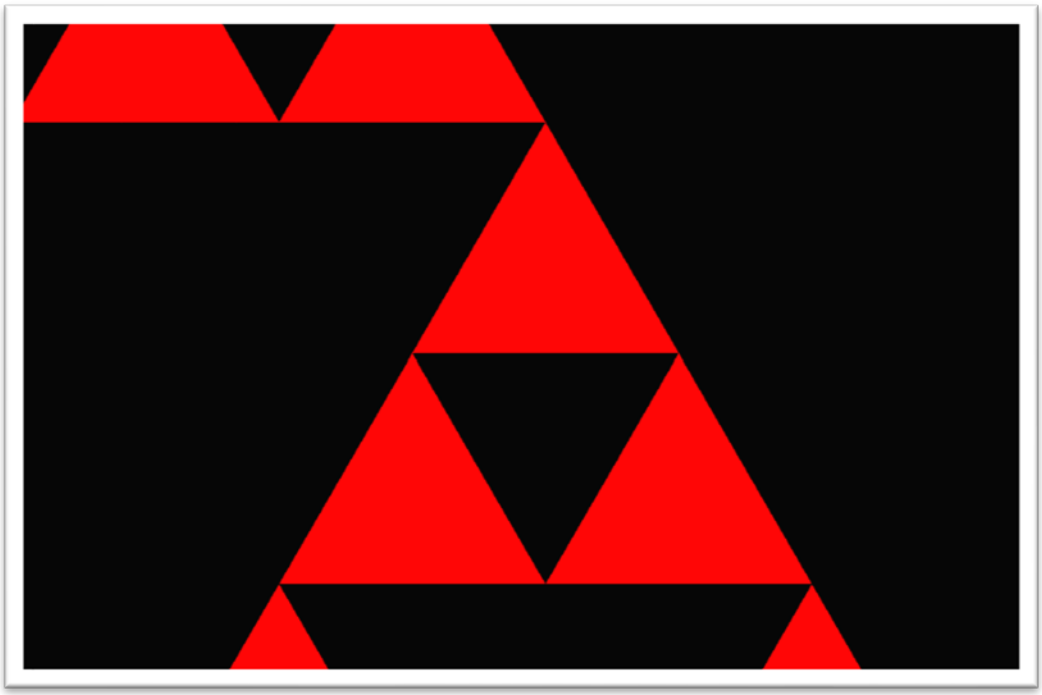
A vendar moja rešitev ni popolna. Zaradi omejenosti natančnosti podatkovnih tipov (*float*) v *shader*-jih, se grafični prikaz deformira ob preveliki povečavi. Tega problema praktično ni pri risanju Sierpinskovega trikotnika, saj tam ne uporabljam *shader*-je, temveč koordinate oglišč trikotnikov, ki jih pobarva grafična kartica. Vendar je pa zato veliko večja poraba pomnilnika in risanje se ne ponavlja v neskončnost, temveč je omejeno s podano stopnjo rekurzije. Hkrati se nisem ukvarjal s pretirano optimizacijo algoritmov, saj to ni bil cilj naloge.



Slika 8 - Mandelbrotova množica pri majhni povečavi



Slika 9 - Deformiranost pri visoki povečavi (Mandelbrotova množica)



Slika 10 - Problemov zaradi povečave pri Sierpinskovem trikotniku ni

8 Literatura

3.6.5 Documentation. 2018. [internet]. [citirano 5. 4. 2018]. Dostopno na naslovu: https://docs.python.org/3/.

Fractal Foundation. 2013. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: http://fractalfoundation.org/.

Fractal. 2015. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: https://en.wikipedia.org/wiki/Fractal.

Fraktal. 2015. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: https://sl.wikipedia.org/wiki/Fraktal.

Julia set. 2018. [internet]. [citirano 5. 4. 2018]. Dostopno na naslovu: https://en.wikipedia.org/wiki/Julia_set.

Koch Flake. 2017. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: https://commons.wikimedia.org/wiki/File:KochFlake.svg.

Mandelbrotova množica. 2014. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: https://sl.wikipedia.org/wiki/Mandelbrotova_mno%C5%BEica.

ModernGL – ModernGL 5.0.6 documentation. 2018. [internet]. [citirano 5. 4. 2018]. Dostopno na naslovu: https://moderngl.readthedocs.io/en/stable/.

Pyglet Documentation – pyglet v1.3.2. 2017. [internet]. [citirano 5. 4. 2018]. https://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/.

Sierpinski triangle – Wikipedia. 2018. [internet]. [citirano 5. 4. 2018]. Dostopno na naslovu: https://en.wikipedia.org/wiki/Sierpinski_triangle.

Wechtersbach, R. 2005. *Informatika: učbenik za srednje izobraževanje.* Ljubljana: Saji.

What are Fractals? 2013. [internet]. [citirano 9. 12. 2017]. Dostopno na naslovu: http://fractalfoundation.org/resources/what-are-fractals/.

9 Priloga

V prilogi je datoteka *fraktali_pyglet.py*. Za zagon potrebujete *Python 3* in knjižnici *Pyglet 1.3.0* in *ModernGL 5.0.7*.

Namesto zagona *python* datoteke lahko na operacijskem sistemu *Windows* zaženete priložen program *Fraktali.exe*, ki sem ga zgradil z uporabo *PyInstaller*-ja.