ŠOLSKI CENTER VELENJE
Elektro in računalniška šola
Trg mladosti 3, 3320 Velenje
Mladi raziskovalci za razvoj Šaleške doline

Raziskovalna naloga

Ustvarjanje in 3D – vizualizacija fraktalov z Blenderjem

Tematsko področje: TEHNIŠKE VEDE

Avtor: David Mikek, 4. letnik

Mentor: Nedeljko Grabant, dipl. inž.

Velenje, 2017

Raziskovalna naloga je bila opravljena na ŠC Velenje, Elektro in računalniška šola, 2017.

Mentor: Nedeljko Grabant, dipl. inž.

Datum predstavitve: marec 2017

By: David Mikek, Nedeljko Grabant

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2016/2017

KG Blender | Python | Python skripte | parametrično modeliranje | generiranje fraktalov | fraktalna grafika | fraktalno programiranje

AV MIKEK, David

SA GRABANT, Nedeljko

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2017

LI 2017

IN USTVARJANJE IN 3D – VIZUALIZACIJA FRAKTALOV Z BLENDERJEM

TD Raziskovalna naloga

OP *IX, 41 str., 1* tab., *12 graf., 63 slik., 1 pril., 24 vir*

IJ SL

JI sl

AI

V raziskovalni nalogi so na kratko predstavljeni programsko 3D-okolje Blender, programski jezik Python in geometrijske oblike, imenovane fraktali. Fraktali so samopodobni geometrijski objekti. To pomeni, da se vzorec ponavlja pri poljubno veliki ali majhni povečavi; povedano drugače, objekt je sestavljen iz (približno ali popolnoma enakih) kopij samega sebe z različnimi ali enakimi parametri. Konceptualno lahko definiramo fraktal kot samo sebi podobno strukturo.
Uporaba fraktalne geometrije zajema danes področje fizike, ekonomije, biologije, medicine, računalništva (kompresijo slik, računalniško grafiko, posebne učinke v filmih, generiranje glasbe, klasificiranje vzorcev...) in druga področja.
V tej raziskovalni nalogi je predstavljeno pisanje skript v Pythonu in te se upodabljajo kot 3D-objekti v Blenderju. V raziskovalni nalogi smo preverili zastavljene hipoteze o primernosti uporabe Blenderja in Pythona za ustvarjanje in vizualizacijo fraktalov ter spoznavanje njihovih oblik. Potrjevanje hipotez je temeljilo na anketi in praktičnem ustvarjanju skript v Pythonu in uporabi le-teh za upodabljanje fraktalov kot 3D-objektov v Blenderju.

KEY WORDS DOCUMENTATION

ND ŠC Velenje, 2016/2017

CX Blender | Python | Python scripts | parametric modeling | fractal generating | fractal graphics | fractal programming

AU MIKEK, David

AA GRABANT, Nedeljko

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2017

PY 2017

TI CREATING AND 3D VISUALIZATION OF FRACTALS IN BLENDER

DT RESEARCH WORK

NO IX, 41 p., 1. tab., 63 img., *12* graf., 1 ann., 24 ref.

LA SL

AL sl/en

In this research paper, the open-source 3D environment Blender, the programming language Python and geometrical shapes, called the »fractals«, are presented. Fractals are self-repeating geometrical objects, which means that their pattern can be seen at any magnification; in other words, an object is made from (roughly or completely identical) copies of itself with different or same parameters. Conceptually, we can define fractals as a structure similar to itself.

Fractal geometry is currently used in physics, economy, biology, medicine, computing (compressing the pictures, computer graphics, special effects, generating sounds) and in some other areas.

In this research paper, writing scripts in Python and their results - 3D objects in Blender- are shown. We also tested hypotheses about suitability of Blender and Python for creating and visualizing fractals and general knowledge of fractals among my peers. Confirming the hypotheses was based on the survey, as well as on the creation of scripts in Python and 3D objects made with those scripts in Blender.

KAZALO KRATIC

% – odstotek
€ – evro
3D – tridimenzionalno
angl. – prevod iz angleškega jezika
API – Application Programming Interface (vmesnik za programiranje aplikacij)
ASCII – American Standard Code for Information Interchange
BY – priznanje avtorstva
CC – angl. Creative Commons, kreativna skupnost
cca – približno, okoli
dipl. – diplomirani
ERŠ – Elektro in računalniška šola
g. – vnspod
ga. – gospa
GUI – Graphical User Interface (uporabniški grafični vmesnik)
HTML – Hyper Text Markup Language (jezik za označevanje nadbesedila)
http – hiper text transfer protocol (nadbesedilni prenosni protokol)
HTTP – Hyper Text Transfer Protocol (nadbesedilni prenosni protokol)
inž. – inženir
IRQ – Interupt Request (zahteva za prekinitev)
ISO – Interational Standard organisation
LGPL – Lesser General Public License
NaN – Not A Number (podjetje Nismo številka)
npr. – na primer
NURBS – Non Uniform Rational Basis Spline (vektorska krivulja)
oz. – oziroma
POV-ray – Persistence of Vision Raytracer (program za upodabljanje)
RAD – Rapid Application Development (hiter razvoj aplikacij)
sl. – slovensko
spl. – splet
ŠC Velenje – Šolski center Velenje
t. i. – tako imenovani
URL – (angl. Uniform Resource Locator) enolični krajevnik vira, je naslov spletne strani v svetovnem spletu
wiki – Wikipedia
www – world wide web - svetovni splet
XML – Extensible Markup Language (XML) – razširljiv označevalni jezik

KAZALO VSEBINE

Kazalo slik

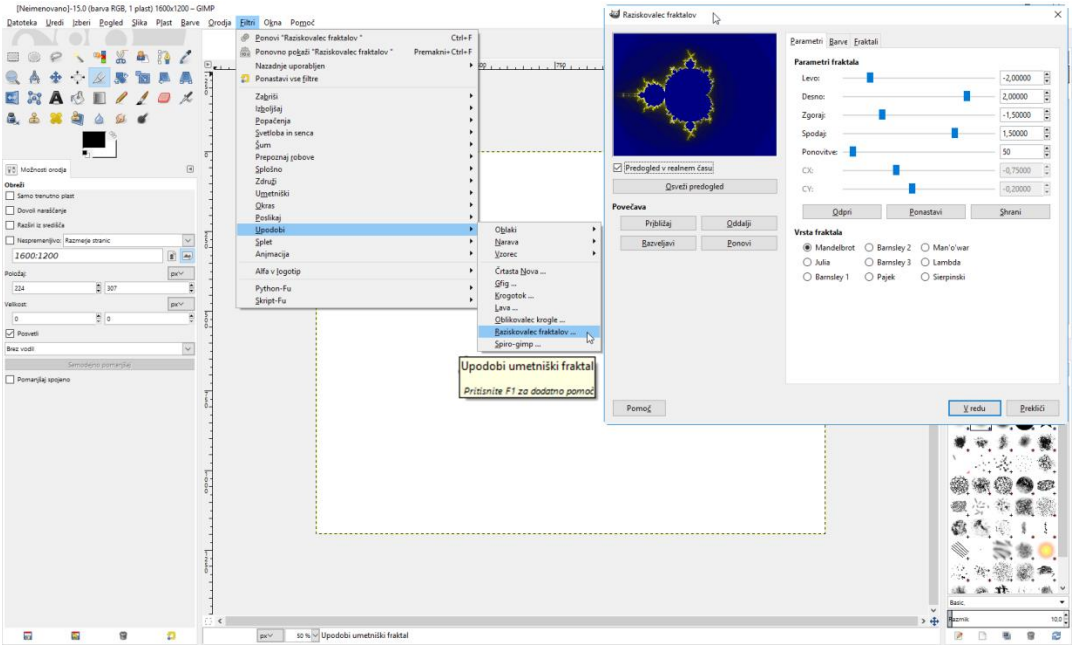
Kazalo grafov

Kazalo tabel

Kazalo kod

1 UVOD

3D-računalniška grafika ali 3D-modeliranje (krajše modeliranje) je proces razvoja matematične predstavitve katerekoli tridimenzionalne površja objekta (nežive ali žive) s pomočjo posebne programske opreme. Izdelek modeliranja se imenuje 3D-model.
Ročno 3D-oblikovanje je lahko zanimivo, a zahtevno, zato pri srednješolcih večinoma ni posebej priljubljeno. Kombinacija med programiranjem in modeliranjem pa je še bolj zahtevna, čeprav postane tudi zanimiva, ko so osnove enkrat osvojene in stvar približno deluje.
Fraktali so zanimive oblike in vzorci, pri katerih matematika sreča grafiko (slika 1).



Slika 1: Raziskovalec fraktalov je programski filter, ki mogoča ustvarjanje 2D-fraktlov v programu GIMP, vir: lastna slika

V Blenderju je možno modelirati (ustvarjati 3D-predmete ali modele), animirati, upodabljati (ustvariti filme ali slike), izvesti kompozicijo in filmsko montažo vključno z zvokom in 3D-interaktivno vsebino oz. igro (angleško je na kratko to model – shade – animate – render – composite – interactive – 3D). Python je programski jezik, katerega Blender uporablja. Tako lahko oblikujemo s skripto in ne ročno, kot je to običajni postopek modeliranja.
Če pa se vprašamo, ali bi vse te tri stvari lahko združili in kakšen bi bil rezultat, dobimo idejo za raziskovalno nalogo.
Tako smo se v tej raziskovalni nalogi posvečali vprašanju, kako lahko matematično formulo preslikamo v vizualno predstavitev na zaslonu.

1.1 NAMEN RAZISKAVE

Raziskovati smo začeli zaradi želje po znanju, in ker smo se želeli naučiti več o fraktalih ter kako jih programsko zapisati ter upodobiti. Želeli smo napisati čim več skript, s katerimi lahko v Blenderju vizualiziramo znane fraktale, in ustvariti nekaj lastnih.

1.2 HIPOTEZE

Izbrali smo si pet hipotez:

1. Blender je primeren program za ustvarjanje fraktalov.
2. Blender je primeren program za vizualizacijo fraktalov.
3. Python je primeren jezik za pisanje skript, ki ustvarjajo fraktale.
4. S pisanjem skript smo pri 3D-oblikovanju lahko bolj natančni in prihranimo s časom.
5. Za fraktale je slišala vsaj tretjina anketirancev, a manj kod desetina ve, kaj fraktali so.

2 PREGLED STANJA TEHNIKE

V nadaljevanju sta opisana Python in Blender in njuna zgodovina [21] ter kaj so fraktali. Opisane so osnove programiranja v Pythonu za ustvarjanje fraktalov v Blenderju.

2.1 Kaj je Python?

Python je enostaven programski jezik, primeren tako za učenje kot tudi za pisanje zahtevnejših programov (slika 2), s katerimi rešimo kompleksne probleme v podjetjih. Zaradi svoje enostavnosti je primeren predvsem za začetnike programiranja, saj se osnov hitro naučimo. Njegova priljubljenost pa ni le v tem, da je lahek za učenje, ampak tudi v tem, da je zelo primeren za hiter razvoj oz. RAD (angl. Rapid Application Development) kot tudi za pisanje skript ali »lepljenja« že obstoječih komponent programa. Python podpira različne module in pakete, kar spodbuja programsko modularizacijo in ponovno uporabo že napisane kode. Njegov prevajalnik in obsežne standardne knjižice so dostopne v izvorni ali binarni obliki brez stroškov na vseh večjih platformah oz. operacijskih sistemih in se jih lahko prosto razširja.

Slika 2: Logotip Pythona, vir: [1]

Pri programerjih je priljubljen predvsem zaradi povečane produktivnosti, ki jo prinese s seboj. Ker v njem ni koraka prevajanja, je krog (uredi – testiraj – odpravi napake) izjemno hiter. Razhroščevanje oz. odpravljanje napak v Pythonu je izjemno lahko: napaka ali napačen vnos ne bo nikoli povzročil napake t. i. segmentacije. Namesto tega prevajalnik takrat, ko odkrije napako, postavi izjemo IRQ. Ko program te izjeme ne ulovi, prevajalnik izpiše sled sklada. Prevajalnik je napisan povsem v Pythonu, kar priča o moči ter poglobljenosti tega programskega jezika.

2.2 Kaj je Blender?

Blender [4] je odprtokodno programsko orodje za grafično 3D-modeliranje, animiranje, komponiranje s post produkcijo, 3D-manipulacijo v realnem času … V Blenderju (slika 3) je vgrajen programski jezik Python kot API, s katerim lahko uporabnik avtomatizira in dodatno razširi možnost edinstvenega programa. Blender uporablja OpenGl za izris uporabniškega vmesnika in upodabljanje.

Slika 3: Logotip programa Blender, vir: [2]

2.2.1 Krajša zgodovina Blenderja

Blender [4] je ustvaril nizozemski animacijski studio. Glavni razvijalec programa je bil Ton Roosendaal. Yellova pesem z albuma Baby je Dance tako navdušila, da so dali programu ime Blender. Ko je Neo Geo kupilo drugo podjetje, sta Ton Roosendaal in Frank van Beek junija leta 1998 ustvarila Not a Number Technologies (NaN), da bi Blender še naprej razvijala. Leta 2002 je podjetje NaN bankrotiralo, zato je Roosendaal začel kampanjo Osvobodimo Blender. Septembra 2002 so uporabniki programa zbrali dovolj denarja, da je Blender postal odprtokodni program.

2.3 Kaj so fraktali?

Fraktali so samopodobni geometrijski objekti. To pomeni, da se vzorec ponavlja pri poljubno veliki ali majhni povečavi; povedano drugače, objekt je sestavljen (iz približno ali popolnoma enakih) kopij samega sebe. Fraktale torej lahko poljubno mnogokrat povečamo, podrobnosti pa se ohranjajo. Poleg tega za fraktale veljajo še druge lastnosti:

- so preveč nepravilne oblike za opis z običajnimi geometrijskimi prijemi, čeprav so pogosto zelo simetrični;
- njihova fraktalna dimenzija je večja od topološke razsežnosti;
- so določeni rekurzivno.

Prvič je izraz fraktal uporabil Benoit Mandelbrot in izhaja iz latinske besede fractus, ki pomeni nepravilen oz. razbit.

2.3.1 Fraktali v vsakdanjem življenju

Fraktali nas obkrožajo, a jih po navadi sploh ne opazimo ali pa ne vemo, da so to fraktali. Opazimo jih lahko:

1. v gozdu (slika 4),

Slika 4: Gozd, vir[3]

2. med nevihto v oblakih in strelah (slika 5),

Slika 5: Nevihta, vir[4]

3. med sneženjem v snežinkah in pri strukturah nalaganja snega (slika 6);

Slika 6: Sneženje, vir [5]

4. pa tudi v video igricah (slika 7).

Slika 7: Igra Sid Meirer's Civilization VI, vir: lasten

A včasih si moramo stvari le pogledati bližje. Poglejmo, ali lahko fraktal najdemo v gozdu (slika 8 in slika 9).

Slika 8: Gozd od blizu, vir [3]

Lep primer fraktala, ki ga najdemo v gozdu, je list praproti (slika 9).

Slika 9: Praprot, vir [6]

Pri njem lepo vidimo, kako se vzorec lista počasi zmanjšuje (slika 10).

Slika 10: Računalniška ustvarjena praprot, vir [7]

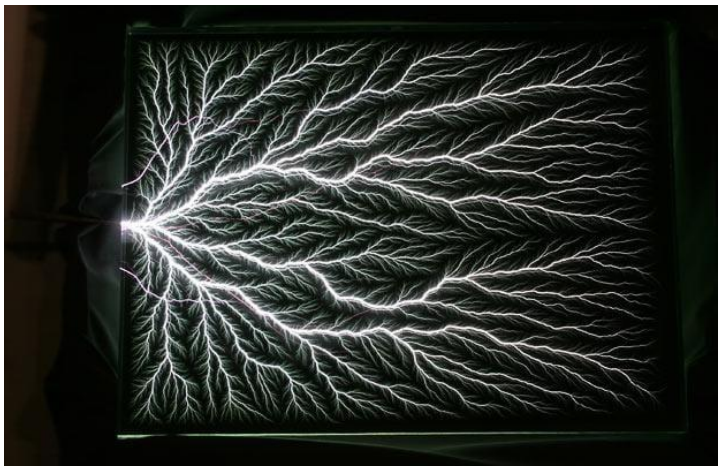
Podrobnejši prikaz nevihte (slika 11)

Slika 11: Nevitha od blizu, vir [4]

Strela je lahko lep primer fraktala (slika 12).

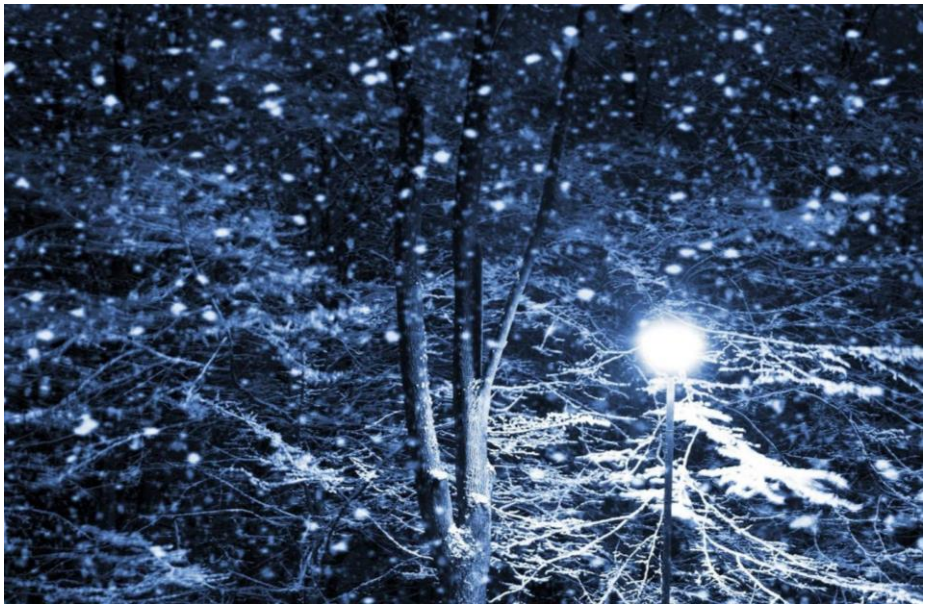
Slika 12: Strele, vir [4]

Benoit Mandelbrot začel svojo knjigo The Fractal Geometry of Nature (Fraktalna geometrija narave) z besedami: *"Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line."* (Oblaki niso sfere, gore niso stožci, obale niso krožne in lubje ni gladko, prav tako pa strela ne potuje v ravni črti (slika 13).



Slika 13: Umetna razelektritev, vir[8]

Sneženje (slika 14)



Slika 14: Sneženje, bližje, vir [5]

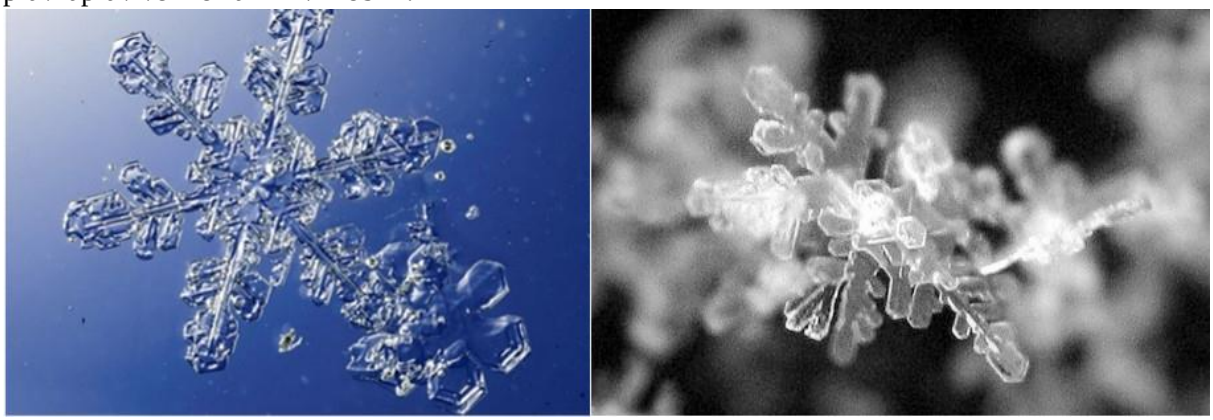
Sneženje še malo bližje (slika 15)

Slika 15: Snežinka od daleč, vir [5]

Če postopek nadaljujemo, pridemo do zanimive oblike snežinke (slika 17);

Slika 16: Snežinka od blizu, vir [9]

pravzaprav veliko zanimivih oblik.



Slika 17: Snežinki, vir [10]

Pravijo, da je vsake snežinka posebna in unikatna. Zagotovo pa je posebna Kochova snežinka. Kochova snežinka, ki ji lahko rečemo tudi Kochova obala, krivulja ali otok, je eden najbolj znanih fraktalov.

Slika 18: Kristalčki, vir [11]

V naravi lahko najdemo fraktale še pri školjkah, brokoliju, rasti kristalov, pa tudi formacija galaksij poteka v fraktalih.

Slika 19: Brokoli, vir [12]

2.4 Znanstveniki, ki so raziskovali fraktale

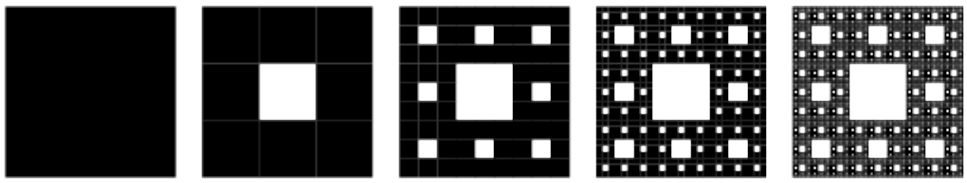
Me najbolj znanimi so Waclaw Sierpisnki, Karl Megner, Helge von Koch in Benoit Mandelbrot.

2.4.1 Waclaw Sierpisnki

Poljski matematik Waclaw Frančišek Sierpinski (1882–1969) [22] je odkril fraktal, ki mu rečemo preproga Sierpinskega.

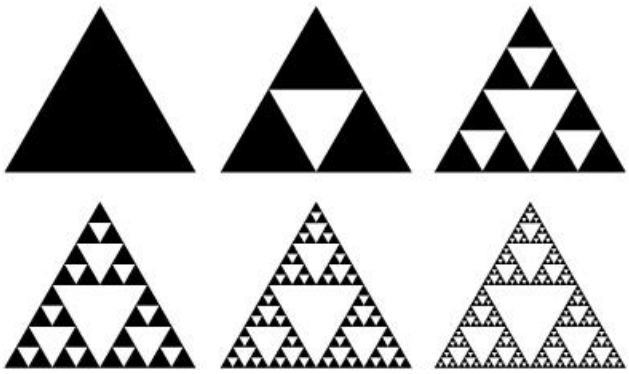
Slika 20: Sierpinski, vir [13]

Izgleda nekako tako: sredi kvadratne podloge imamo kvadrat. Okoli njega pa je devet manjših kvadratov – eden desno, eden levo, eden zgoraj, eden spodaj, eden spodaj levo, eden spodaj desno, eden zgoraj levo, eden zgoraj desno. Okoli vsakega tega kvadrata je znova devet manjših kvadratov. In okoli vsakega tega spet. In okoli vsakega tega spet. In to se ponavlja naprej in naprej v neskončnost.



Slika 21: Nastajanje preproge, vir [14]

Znan je še tudi po svoji krivulji in trikotniku. Po njem se imenujejo števila Sierpinskega in problem Sierpinskega. Raziskoval je teorije množic, števil, funkcij in topologij. Napisal je več kot petdeset knjig.



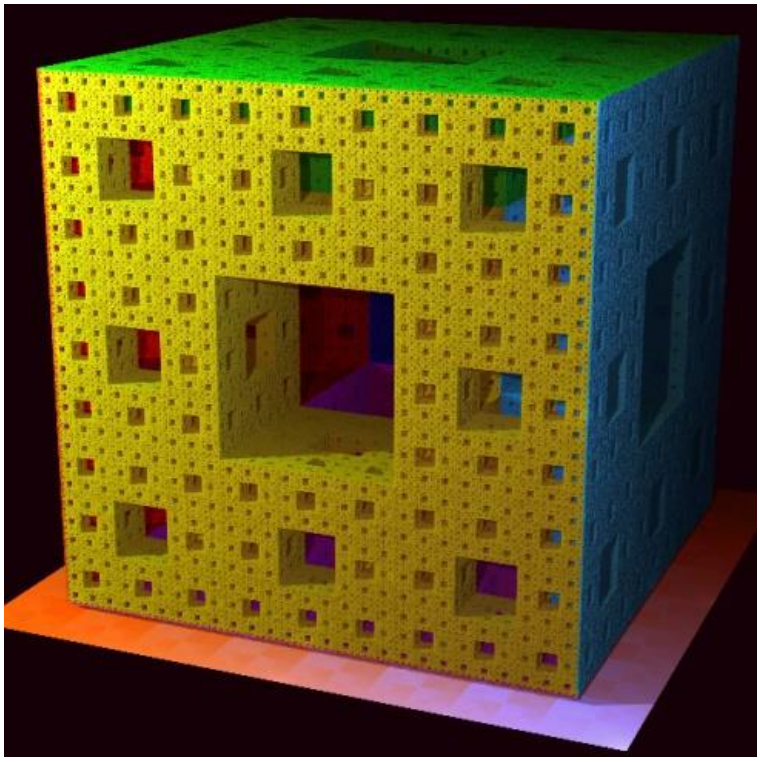
Slika 22: Nastajanje trikotnika, vir [15]

2.4.2 Karl Megner

Karl Menger [22] (1902–1985) je bil ugleden matematik dvajsetega stoletja (slika 23), ki je veliko prispeval k mnogim področjem matematike, vključno s teorijo razsežnosti, logike, teorijo rešetke (angl. Lattice theory) diferencialne geometrije in teorijo grafov.

Slika 23: Karl Megner, vir [16]

Megner je ustvaril telo, ki ima za stranice preproge Sierpinskega. Telesu pravimo Megnerjeva spužva (slika 24). To telo ima izjemno veliko površino, hkrati pa ima prostornino izredno zelo majhno (ta skoraj limitira proti ničli).



Slika 24: Megnerjeva spužva, vir [17]

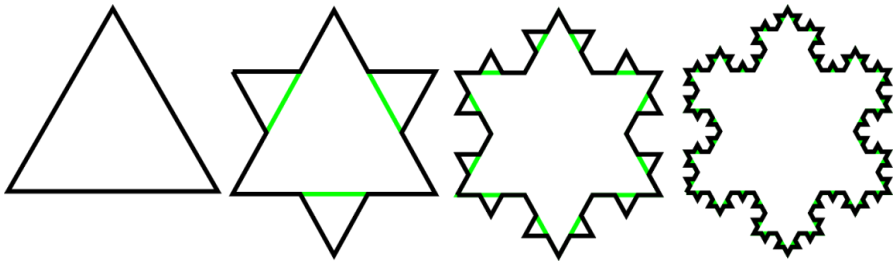
2.4.3 Niels Fabian Helge von Koch

Niels Fabian Helge von Koch (1870–1924) [22] se je rodil v švedski plemiški družini (slika 25). Šolal se je na Univerzi v Uppsali, kasneje pa je učil matematiko v Stockholmu. Posvečal se je teoriji števil.

Slika 25: Niels Fabian Helge von Koch, vir [18]

Kochova snežínka ali Kochova zvézda (slika 26) je eden prvih odkritih fraktalnih likov. Leta 1904 jo je opisal Niels Fabian Helge von Koch v članku O zvezni krivulji brez tangente, dobljeni z elementarno geometrijsko konstrukcijo (Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire).

Najbolj znan je po svoji krivulji (slika 27), ki ji lahko rečemo tudi obala, otok, največkrat pa snežinka.

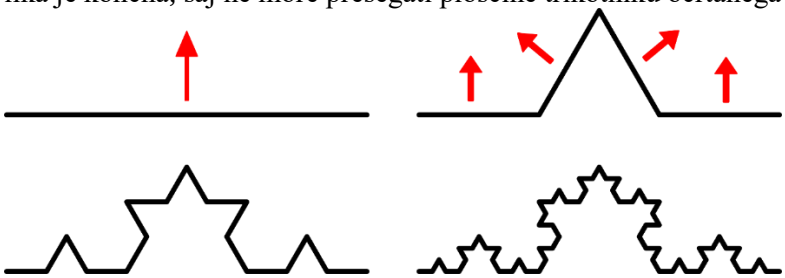


Slika 26: Prve štiri iteracije pri konstrukciji Kochove snežínke

Kochovo snežinko naredimo takole:

- narišemo enakostranični trikotnik s stranico a;
- vsako stranico razdelimo na tri enake dele in nad srednjim narišemo enakostranični trikotnik z dolžino stranice a/3. Tako dobimo šestkrako zvezdo;
- postopek lahko ponavljamo na vsaki stranici neskončnokrat.

Lik, ki smo ga tako dobili, ima neskončen obseg, saj ta meri (4/3) n, kjer je n število korakov. Ploščina lika je končna, saj ne more presegati ploščine trikotniku očrtanega kroga.



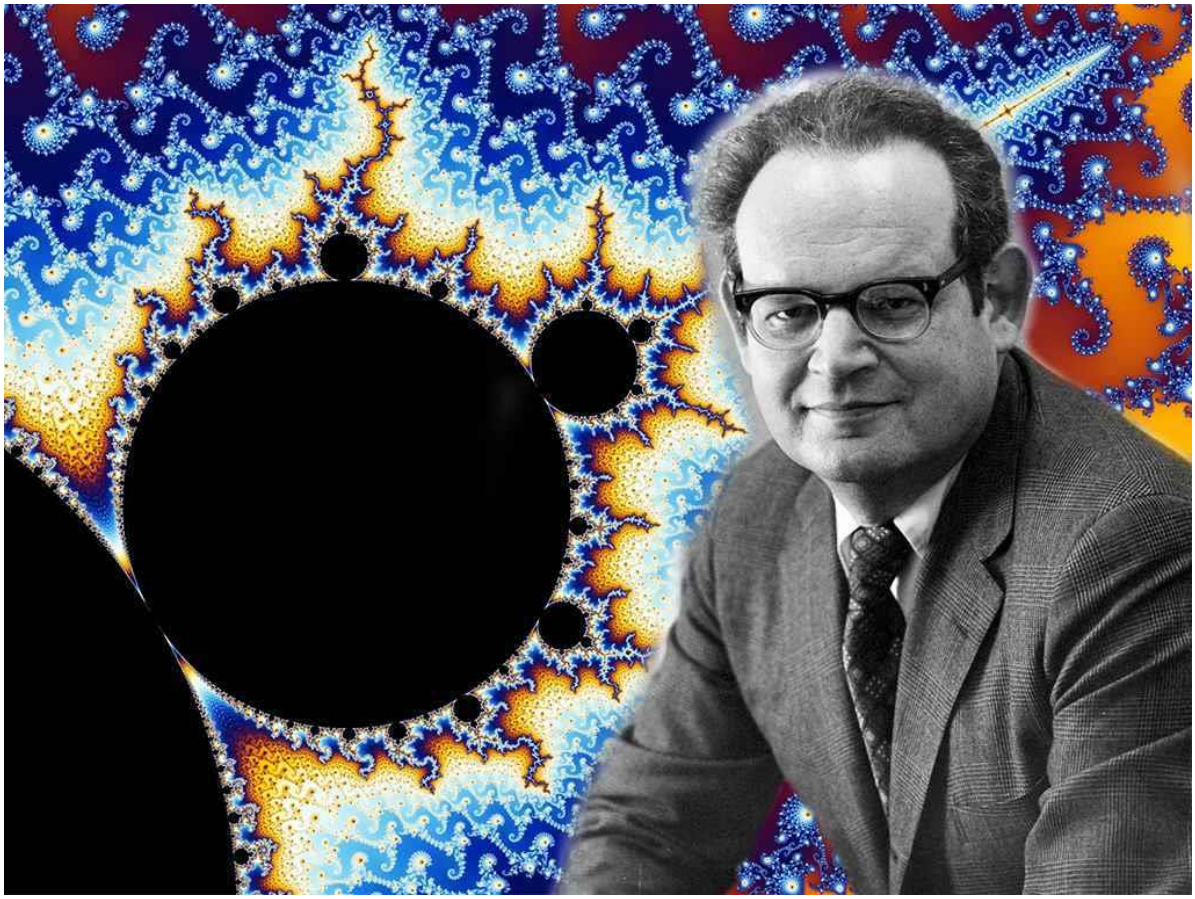
Slika 27: Ustvarjanje Kochove krivulje, lastna risba

Krivulji natančne dolžine ne moremo določiti. Manj znana je Kochova krivulja, ki je enaka Kochovi snežinki, s tem da se namesto z enakostraničnim trikotnikom začne z daljico. Kochova krivulja je poseben primer de Rhamove krivulje. Temu se je posvetil tudi Mandelbrot.

2.4.4 Benoit Mandelbrot

Benoit Mandelbrot (1924–2010) [22] se je rodil na Poljskem. Študiral je v Parizu in Združenih državah Amerike. Znamenito Mandelbrotovo množico je odkril leta 1979. Z njo je pokazal, kako lahko preprosta matematična pravila naredijo zelo kompleksno vizualno podobo (slika 28). Sam sebi je pravil Fraktalist.

Madelbrotova mnóžica [24] je v matematiki množica točk v kompleksni ravnini, katere meja tvori fraktal. Imenuje se po francosko-ameriškem matematiku Benoîtu B. Mandelbrotu. Matematično lahko Madelbrotovo množico definiramo kot množico 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.
Če postavimo na primer c = 1, dobimo zaporedje 0, 1, 2, 5, 26 …, ki narašča v neskončnost. Ker je neomejeno, 1 ni element Mandelbrotove množice.
Na drugi strani, če je c = i (kjer je i imaginarna enota, določena kot $i^2 = -1$), dobimo zaporedje 0, i, (− 1 + i), − i, (− 1 + i), − i …, ki je omejeno, zato i pripada Mandelbrotovi množici.
Če Mandelbrotovo množico izračunamo in jo izrišemo na kompleksni ravnini, ima zamotano mejo, ki se ob poljubni povečavi ne poenostavi, zaradi tega je njena meja fraktal (slika 28).
Mandelbrotova množica je zunaj matematike postala priljubljena tako zaradi svoje estetske privlačnosti kot tudi zaradi svoje zapletene zgradbe, ki izhaja iz preproste definicije. Velja za enega najbolj znanih zgledov matematičnega predočenja.



Slika 28: Mandelbrot v svoji množici, vir [20]

2.4.5 Kako dolga je Britanska obala?

Mandelbrot je vprašal: »Kako dolga je obala Britanije?« Vprašanje se zdi enostavno in pričakovali bi tudi enostaven odgovor. A ga ne dobimo. Obala Britanije bi prav lahko bila neskončna, odvisno od tega, kako pogledamo. Če pogledamo z raketoplana, vidimo le osnovno obliko obale. Če pogledamo iz letala, vidimo več. Če gledamo z balona, opazimo že kar nekaj zalivčkov, ki jih iz letala nismo mogli. Če merimo na tleh, vsak zalivček posebej z metrskim ravnilom, bi dobili manjši rezultat, kot pa če bi ga merili na centimetrski. Tako bi lahko nadaljevali do atomske ravni in naprej in odgovor bi bil vsakič večji. Njegova krivulja nastaja tako (slika 27): sredi stranice naredimo trikotno izboklino in nenadoma se število stranic poveča za štiri. Zdaj sredi vsake manjše stranice naredimo izboklino in število stranic se znova poveča za štiri.
Število stranic za vsako začetno stranico je enako štiri na n-to potenco (4^n), kjer je n število ponovitev.

3 METODOLOGIJA

Kodo smo pisali v programskem jeziku Python zato, ker je Blender API in GUI pisan v tem jeziku; sicer bi lahko pisali tudi v C-ju in to vključili kot knjižnico oz. razširitev.
Za pisanje kode smo imeli na izbiro številne urejevalnike oz. programe..

3.1 Python in Notepad ++

Kodo bi lahko pisali neposredno v Blenderju. Pri pisanju ne bi bilo zastojev, shranili pa bi lahko hkrati kodo in 3D-objekt (predmet ali telo), ki ga koda naredi. Vendar je pisanje kode neposredno v Blenderju lahko časih problematično. Če bi se v kodi zmotili in ustvaril neskončno zanko, bi se Blender nehal odzivati, mi pa bi bili prisiljeni program zapreti in s tem izgubiti celo zapisano skripto. Temu bi se lahko izogibali z rednim shranjevanjem, a nismo hoteli tvegati.
Kode smo večinoma pisali v programu Notepad ++, ki smo ga poznali že nekaj časa (slika 29). Pri pisanjih so bili majhni zastoji: Python je za razliko od drugih programskih jezikov, s katerimi smo delali, zelo občutljiv na zamike, saj ne pozna zavitih oklepajev in se namesto tega zanaša na pravilne zamike. Notepad++ je zaznal, ko smo začel s pisanjem zanke, in je v vsaki naslednji vrstici naredil zamik. Problem je bil to, da je delal zamik s tabulatorjem, uporaba tabulatorja v skripti pa je v Blenderju pomenila napako. Tabulatorje smo zaradi tega morali brisati in jih nadomestiti s presledki. Je pa Notepad ++ imel eno veliko prednost. V njem smo lahko imeli odprtih več zavihkov. Da bi si ogledali, kako sem napisal del kode pri drugem fraktalu, nam ni bilo treba celega projekta shraniti, zapreti, odpreti novega projekta, pogledati, zapreti in spet odpreti, ampak smo lahko le kliknili na zavihek, pogledali in kliknili nazaj na prvotni zavihek. S tem je bilo olajšano tudi kopiranje del že napisane kode in sprememba le-te v novem delu kode ali zavihka (slika 29).

serpinskiv1.py | koch1.py | spuzva.py | serpinskiv2.py | stozeckrogv2.py | stozeckrogv3.py

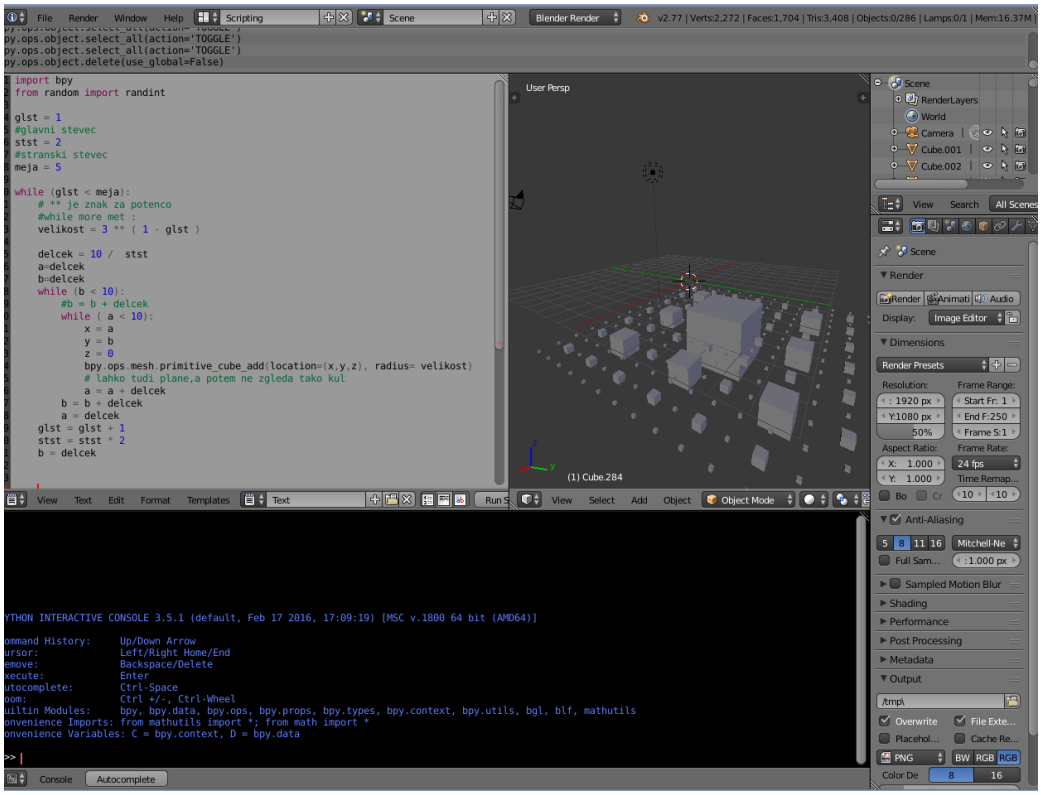
```
1  import bpy
2  from random import randint
3  bpy.ops.mesh.primitive_cube_add()
4
5  glst = 1
6  stst = 2
7  meja = 3
8
9  while (glst < meja):
10     # ** je znak za potenco
11     #while more met :
12     velikost = 3 ** (1 - glst )
13
14     delcek = 10 % stst
15     a=delcek
16     b=delcek
17     for b in range(0, 10):
18         for a in range(0, 10):
19             x = a
20             y = b
21             z = 0
22             bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
23             a = a + delcek
24         b = b + delcek
25     glst = glst + 1
```

Slika 29: Notepad++, lasten vir

Podobno kot pri zavihkih na spletnih brskalnikih.
V večini kod nastopajo kocke, ki so se nam zdele primeren osnovni gradnik za 3D-fraktale.

3.2 Pisanje kode v Blenderju

Glavna prednost pisanje kode v Blenderju je bila to, da smo lahko takoj videli rezultat kode (slika 30).



Slika 30: Pisanje in upodabljanje kode v Blenderju, vir: lasten

3.3 Anketa

Ankete smo se poslužili zato, da bi ugotovili, koliko dijaki in drugi anketiranci vedo, kaj so fraktali. Vprašanja v anketi so bila postavljena zvito. Anketiranec ni mogel razbrati odgovora na določeno vprašanje s tem, da je šel brat naslednja vprašanja.
Najprej smo v anketi povprašali po starosti in spolu (slika 31). Nato so sledila vprašanja, povezana s fraktali.

Ste že slišali za besedo "Fraktal"?

- O Da
- O Ne
- O Morda

Kaj od spodaj naštetega je najustreznejša definicija fraktala?

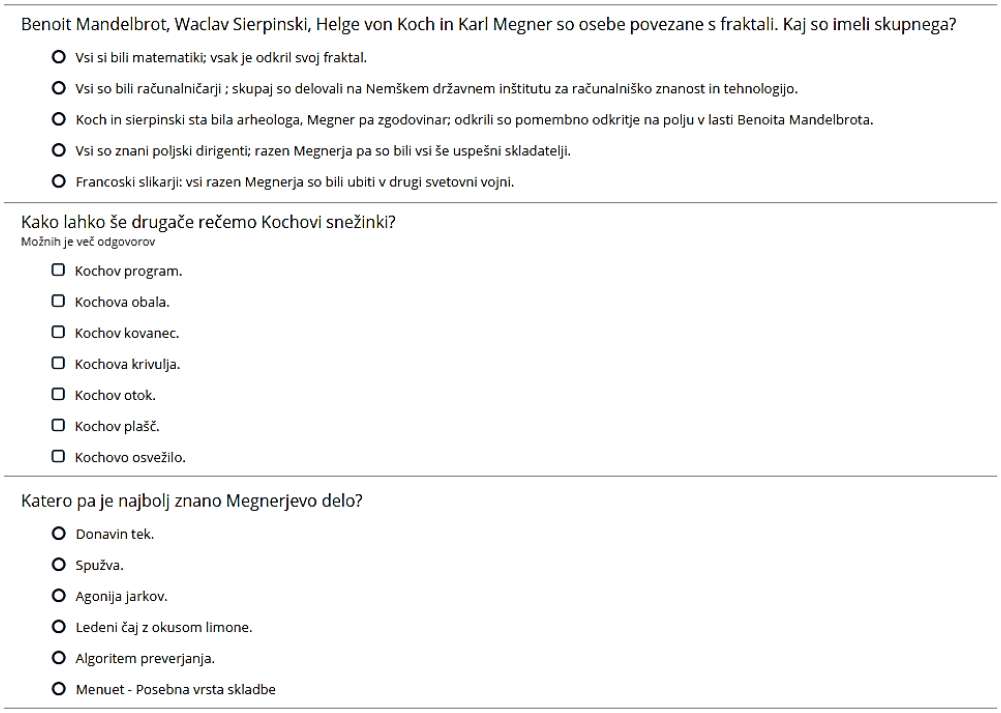
- O Svečano moško oblačilo, ki ga ponavadi nosijo dirigenti.
- O Program za defragmentacijo diska.
- O Denar, ki so ga uporabljali Franki.
- O Matematični objekt s ponavljajočim se vzorcem.
- O Znamka brezalkoholnih pijač.
- O Tehnika slikanja, ki se je razširila med vojnama v Franciji.
- O Programski jezik, namenjem designerjem in 3D oblikovalcem.

Od spodaj naštetih prostorov, kje bi najverjetneje našli fraktal?

- O V muzeju zgodovine evropskih narodov.
- O V hladilniku.
- O Na polju kjer gojijo brokoli Ramanesco.
- O V omari, na obešalniku.
- O Na C: disku.
- O V muzeju moderne umetnosti.

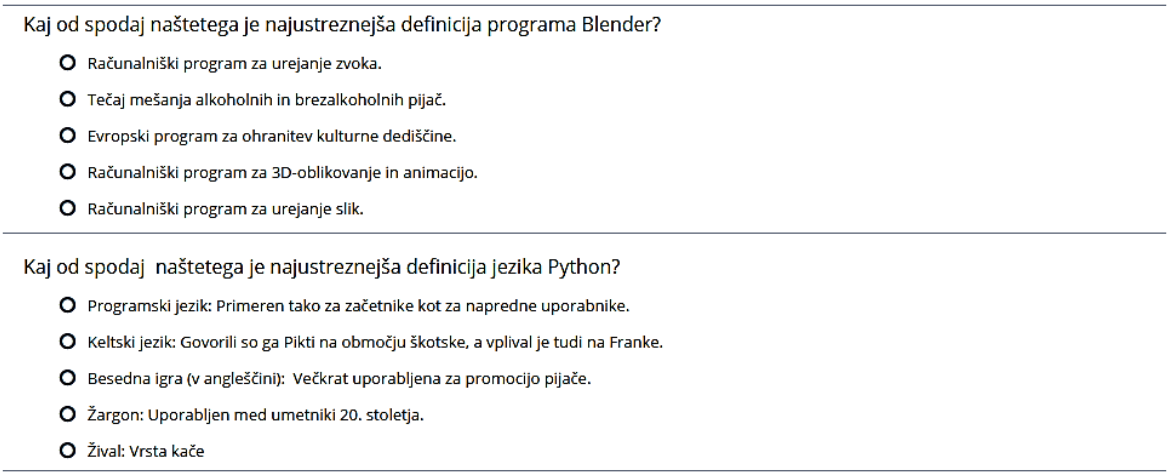
Slika 31: Anketa 1. del

Na *naslednji* sliki (slika 32) ne opazimo dveh vprašanj, ki anketiranca sprašujeta po tem, če pozna kakšen program za ustvarjanje 2D-fraktalov in kakšnega za ustvarjanje 3D-fraktalov.



Slika 32: Anketa 2. del

Vprašanji se pojavita le, če anketiranec izbere pravilen odgovor – matematični objekt. Zadnji dve vprašanji pa se nanašata na Blender in Python (slika 33).



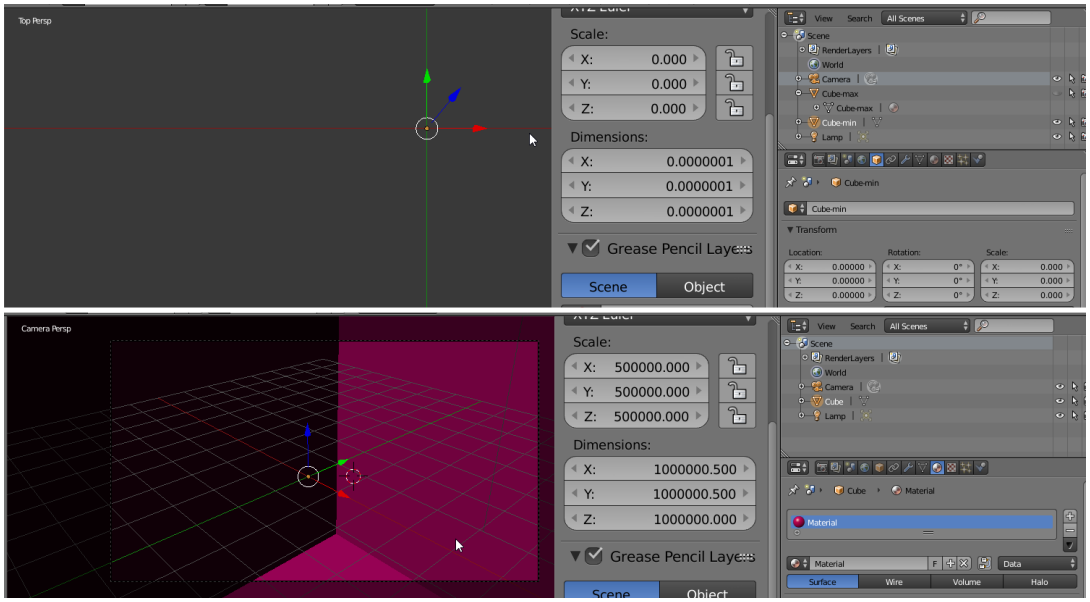
Slika 33: Anketa 3. del

3.4 Potek dela

Najprej smo si izbrali 3D-telo oz. model, ki smo ga želeli narediti. Nato smo na papirju načrtovali, kako se bomo lotili skripte. Ko se nam je zdelo, da je ideja za zanko ustrezna, smo na računalniku v Notepadu ++ napisali skripto. Potem smo preverili, ali je v skripti kaj napak, ki bi onemogočile delovanje programa. Skripto smo nato kopirali v Blender in jo zagnali. Enkrat smo spregledal napako – v programu smo imeli neskončno zanko (ponavljanje v neskončnost, brez meje, ko se ponavljanje zanke ne ustavi). Blender se je nehal odzivati, zato smo ga morali prisilno zapreti. Fraktali, kot jih poznamo, se po navadi z matematičnega stališča ponavljajo v neskončnost, v računalništvu pa imajo končne meje. Blender teoretično lahko stvar ponovi stokrat ali več milijonkrat (za to potem potrebuje dolgo časa), neskončnokrat pa resda ne more. Smo si pa o Blenderju postavili še eno vprašanje.

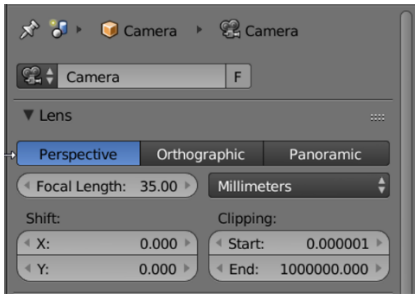
3.4.1 Kje so meje pri modeliranju v Blenderju?

Najmanjša možna dimenzija kocke v Blenderju je X=0. 0000001, Y= 0. 0000001, Z=0. 0000001 Blender enot (zgornji del – slika 34), ki je praktično ne vidimo, ker je oranža pika na sliki. Največja možna dimenzija kocke v Blenderju je X=1 000 000, Y= 1 000 000, Z = 1 000 000 Blender enot (spodnji del – slika 34), ki povzroči, da smo v objektu in kamera mora biti pravilno nastavljena. To območje velikosti objekta je torej 10^{12}, kar je za prakso upodabljanja veliko preveč.



Slika 34: Najmanjša in največja velikost objekta v Blenderju (zgornji in spodnji del slike)

Pri eni izmed skript se kocke, ki jih dodajamo telesu, v vsaki ponovitvi zmanjšajo na tretjino velikosti. Začetna kocka je dimenzije 1 x 1 x 1 Blender enot.
Po praktičnem izračunu smo prišli do tega, da lahko naredimo 7 ponovitev (če ne štejemo dodajanja začetne kocke).
To zadostuje za vizualizacijo, saj so kocke že od četrte ponovitve težko vidljive pri navadnem pogledu brez povečave. Druga hipoteza s tem še ni ovržena.
Za samo upodabljanje slike je prav tako pomembna nastavitev območja vidnosti (angl. Clipping) pri kameri v Blenderju in ta znaša od 0. 0000001 do 1 000 000 Blender enot (slika 35). To je območje v razponu 10^{13}, kar je z našim očesom in v praksi nemogoče sočasno videti (sočasen pogled od mikro v makro svet).



Slika 35: Območje vidnosti (angl. Clipping) pri kameri v Blenderju

Po drugi strani pa smo potrdil tretjo hipotezo – **Python je primeren jezik za pisanje skript, ki ustvarjajo fraktale.** Python je primeren jezik za ustvarjanje skript za fraktale, saj z njim nismo imeli problemov, ko smo delal fraktale, pa tudi neskončno zanko smo lahko napisali v njem. Je pa Python edini programski jezik, pri katerem lahko napišemo zanko »while (true or false):«, pa nam ne bo javil napake.

4 REZULTATI

Najprej sledi krajša razlaga lastno napisanih Python kod za ustvarjanje: 3D-oblike vrtavke, 3D-Koch iz kock, mreža kock, Koch kocke in Sierpinski kocke ter podpoglavje z naslovom Fraktali in barve. Temu sledi bolj podroben opis kodiranja in upodabljanja Miklavinskega stopnišča.

4.1 Skripte

V tej raziskovalni nalogi smo veliko delali s potencami, tako da smo eksponentno enačbo imeli v vsaki izmed naših skript.
Pokazali in razložili bomo 3 glavne skripte, iz katerih so nastajali rezultati naše raziskave.

4.1.1 Koda za vrtavko

Ta skripta bo dodajala Mesh krožnice po z-osi navzgor. Krožnice se bodo eksponento večale, dokler ne dobijo največjega polmera – 3. Od takrat naprej se začnejo eksponentno manjšati. Manjšajo se, dokler ne pridejo do nič. Število ponovitev je odvisno od vrednosti spremenljivke »meja« (koda 1).

```
1  import bpy
2  from random import randint
3
4  meja = 1
5
6  x = 0
7  y = 0
8  z = 0.0
9
10 # obratni z
11 nz = 0.0
12 # obmocni z
13 az = 0.0
14
15 velikost = 2 ** ( z) -1
16
17 while (meja > 0):
18     meja = meja - 1
19     while (az < 2):
20         bpy.ops.mesh.primitive_circle_add(location=(x,y,z), radius= velikost)
21         z = z + 0.01
22         az = az + 0.01
23         velikost = 2 ** ( az ) -1
24     nz = az
25     while (2 < az < 4):
26         bpy.ops.mesh.primitive_circle_add(location=(x,y,z), radius= velikost)
27         z = z + 0.01
28         az = az + 0.01
29         nz = nz - 0.01
30         velikost = 2 ** (  nz ) -1
31     az = 0
```

Koda 1: Koda za vrtavko

4.1.2 Koch kocke

Kodo smo poimenovali Koch kocke (koda 2), saj ima podoben koncept kot Kochova snežinka. Kocki smo na vsako od stranic dali manjšo kocko, vsaki od teh kock pa na ploskev, ki gleda proč od velike kocke, še manjšo kocko in še tem kockam dodajali manjše (koda 2).

```
import bpy
from random import randint
bpy.ops.mesh.primitive_cube_add(location=(0,0,0))

meja = 5
glst = 0
skupnavelikost = 0
z = 0

while (glst < meja):
    glst = glst + 1
    velikost = 2 ** ( - glst )
    skupnavelikost = 3 * velikost + skupnavelikost
    y = 0
    x = skupnavelikost
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    x = - skupnavelikost
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    y = skupnavelikost
    x = 0
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    y = - skupnavelikost
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    y = 0
    z = skupnavelikost
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    z = - skupnavelikost
    bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
    z = 0
```

Koda 2: Koch kocke

Razlika med tem in snežinko je v tem, da snežinka deluje kot 1/3 delitev na vsaki novi stranici, v zgornji se število kock povečuje eksponentno, pri nas pa se število kock povečuje linearno. Dvanajsta vrstica spremenljivka velikost določa velikost kocke, ki pri iteraciji potem eksponentno pada. Položaj kocke določa spremenljivka »skupna_velikost«.

4.1.3 Mreža kock Sierpinski

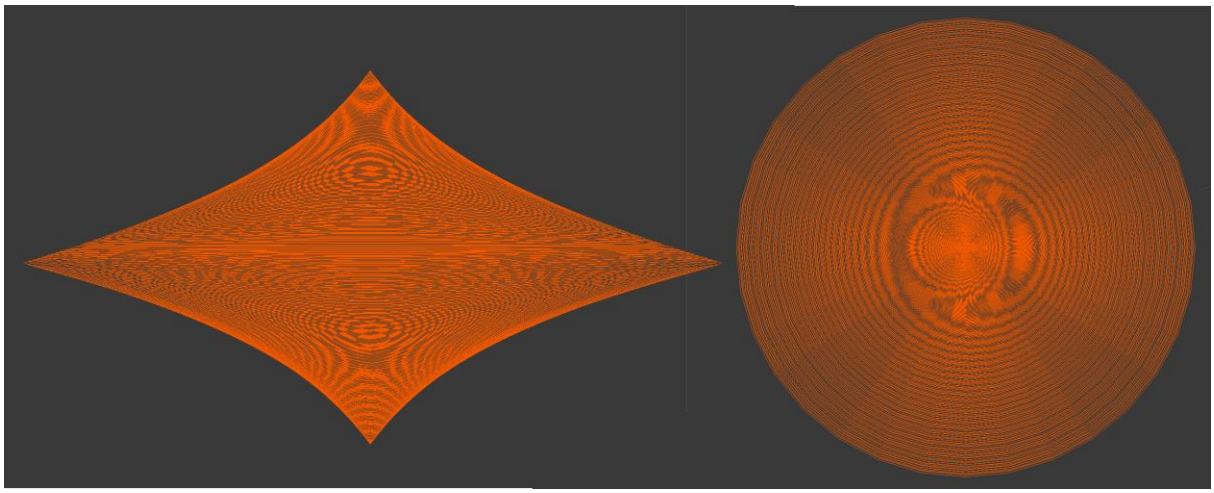
Pri tej kodi (koda 3) smo se zgledovali po preprogi Sierpinskega in ustvarili smo zelo podoben fraktal, ki smo ga poimenovali mreža kock Sierpinski. Velikost kock se eksponentno zmanjšuje. Število kock narašča eksponentno.
Iz te kode smo med raziskanjem prišli do zelo zanimivega fraktala, ki je predstavljen kasneje.

```
import bpy
from random import randint
glst = 1
#glavni stevec
stst = 2
#stranski stevec
meja = 5

while (glst < meja):
    # ** je znak za potenco
    #while more met :
    velikost = 3 ** ( 1 - glst )

    delcek = 10 /  stst
    a=delcek
    b=delcek
    while (b < 10):
        #b = b + delcek
        while ( a < 10):
            x = a
            y = b
            z = 0
            bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
            # lahko tudi plane,a potem ne zgleda tako kul
            a = a + delcek
        b = b + delcek
        a = delcek
    glst = glst + 1
    stst = stst * 2
    b = delcek
```

Koda 3: Sierpinski kocke

4.2 Oblika vrtavke

S to kodo smo naredili obliko (koda 1), ki spominja na vrtavko.
Kot je vidno na spodnji sliki, je ta objekt v celoti narejen iz krožnic (slika 36).

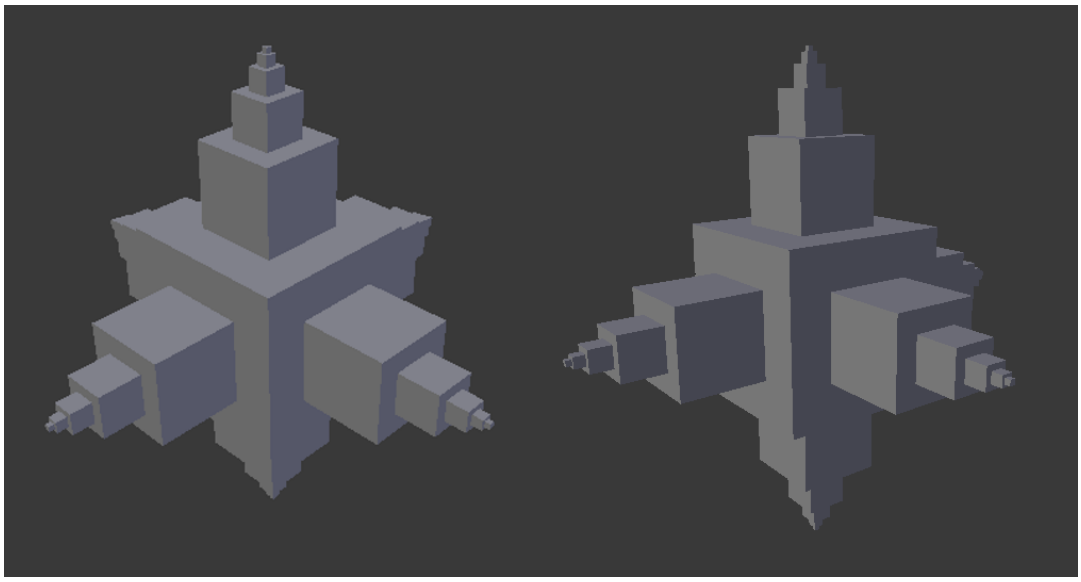


Slika 36: Pogled na vrtavko s strani in od zgoraj (z leve na desno), vir: lasten

Vrtavka nam grafično prikazuje, kako naraščajo potence števila dva (vse do števila štiri), po obliki pa spominja na vrtavko, po čemer smo je tudi poimenovali.
Kodo lahko z lahkoto spremenimo, da pokaže potence drugih števil, a najlepšo vrtavko dobimo s številom dva.

4.2.1 3D-Koch iz kock

Naslednja slika prikazuje izgleda 3D-telesa, ki ga ustvari skripta Koch (koda 2 in slika 35).

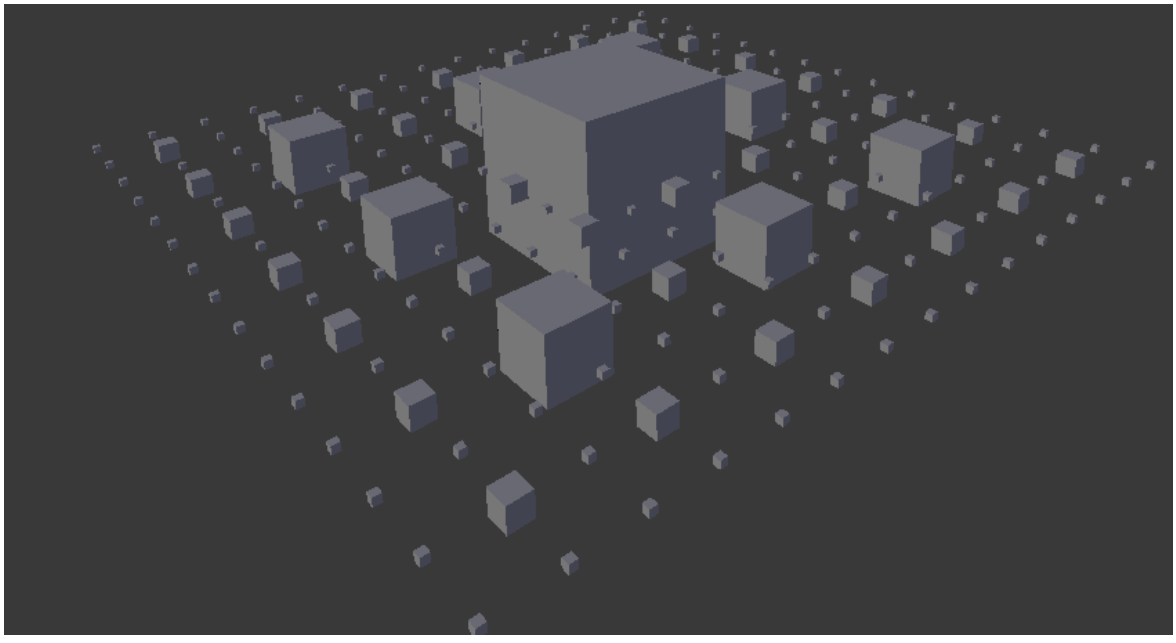


Slika 37: Telesi 3D-Koch iz kock, vir: lastni

Pri tej skripti lahko Blender izvede 8 ponovitev, saj je vsaka kocka v naslednji ponovitvi manjša za eno polovico in ne eno tretjino, tako kot pri Sierpinskem.

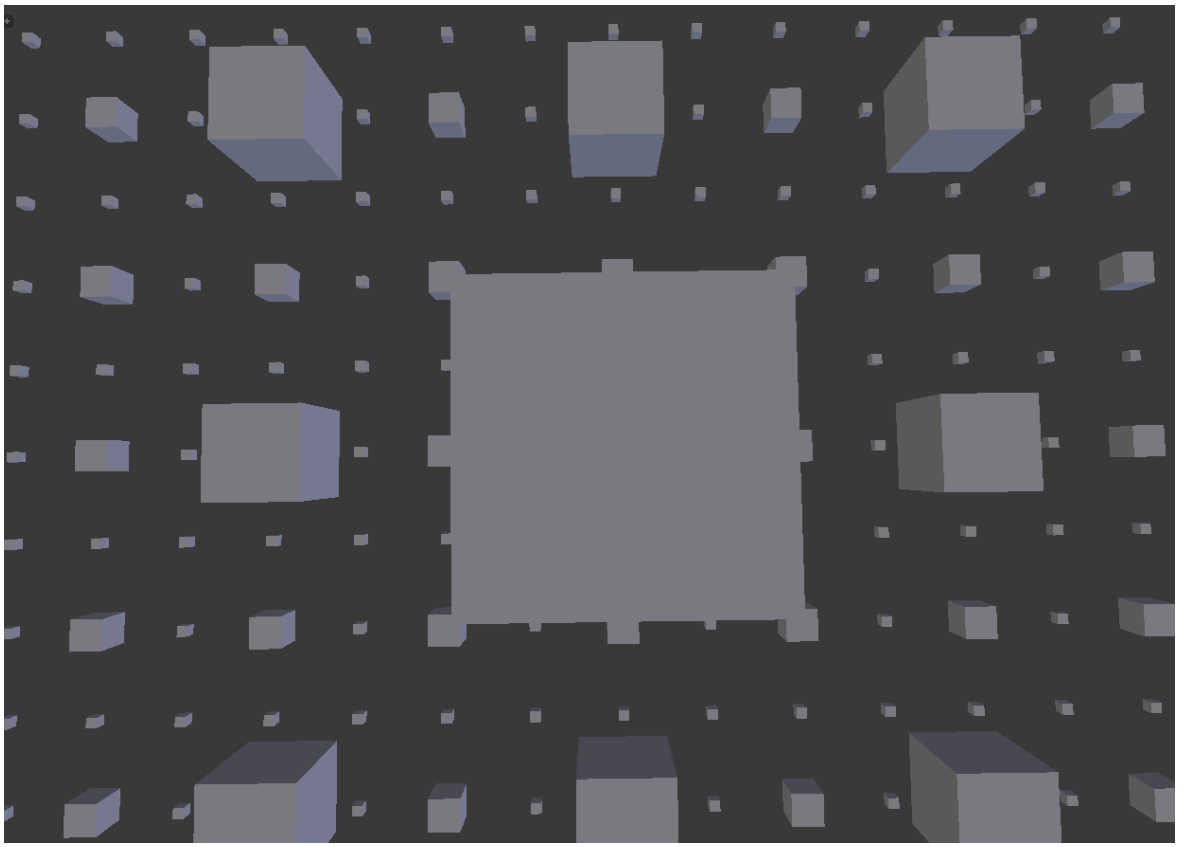
4.2.2 Mreža kock Sierpinski

To telo mreža kock Sierpinski (slika 38) smo delali po zgledu preproge Sierpinskega.



Slika 38: Mreža kock Sierpinski, vir: lasten

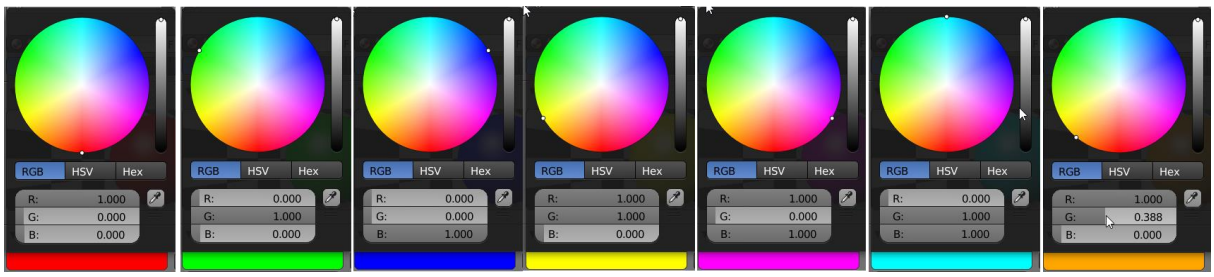
Tako zgleda naša verzija preproge Sierpinskega (slika 38 in slika 39). Je zelo podobna originalu, ni pa ji identična (poleg tega, da je narejena v 3D-prostoru).



Slika 39: Mreža kock Sierpinski, od zgoraj, vir: lastni

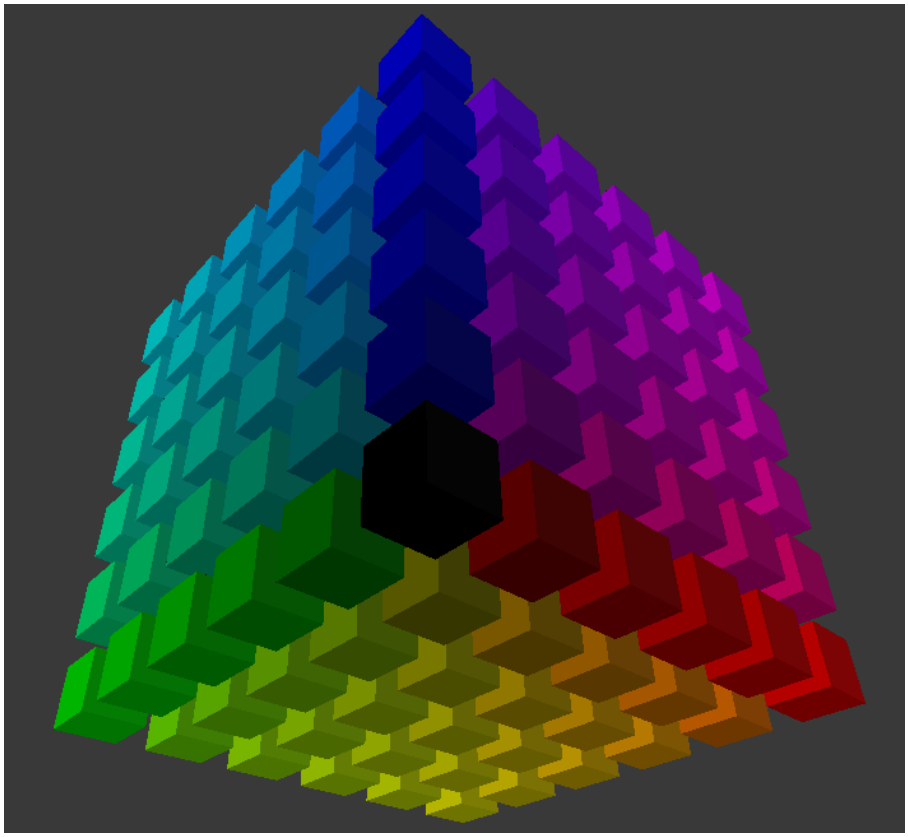
4.3 Fraktali in barve

Pri Blenderju so barve določene pod lastnosti materiala objekta, kot triplet s tremi vrednostmi RGB-barvnega modela (slika 40). To je trikanalni barvni model, vsak kanal pripada eni barvi – rdeči, zeleni ali modri. Vrednost se mora gibati med 0 in 1 (na ali več decimalna mesta – npr. cian). Če je ena barvni kanal enak ena in drugi dve nič, dobimo nasičeno barvo (prva šest okna na sliki 38). Če so vse vrednosti enake ena, dobimo belo barvo. Če so vse vrednosti enake nič, dobimo črno.



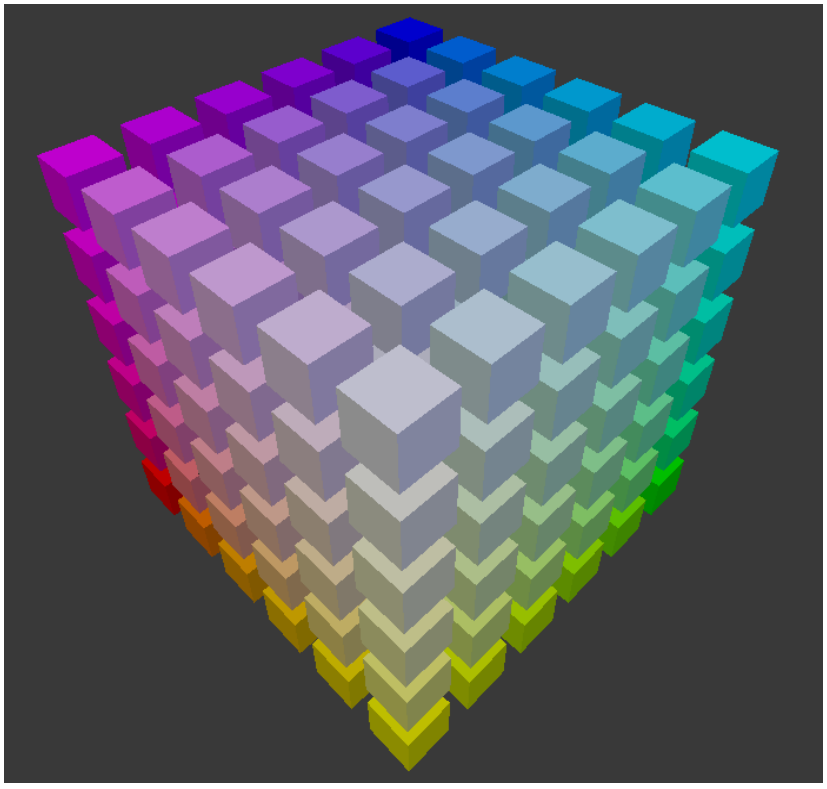
Slika 40: Izbira barv v RGB-modelu programa Bender, lastna slika

Barve lahko tudi poljubno vmes izbiramo (slika 41).



Slika 41: 3D-barvna paleta 216 barvnih odtenkov spodnji del, vir: lasten

To telo sicer ni fraktal, lahko pa z njim pokažemo, kako delujejo barve. V kocki na vrhu diagonalno na nevidnem delu zgornje slike so vrednosti vseh barv enake 1 (slika 42), na nasprotnem koncu (sredina zgornjega dela slike) pa so vse vrednosti barv enake 0 (slika 41).



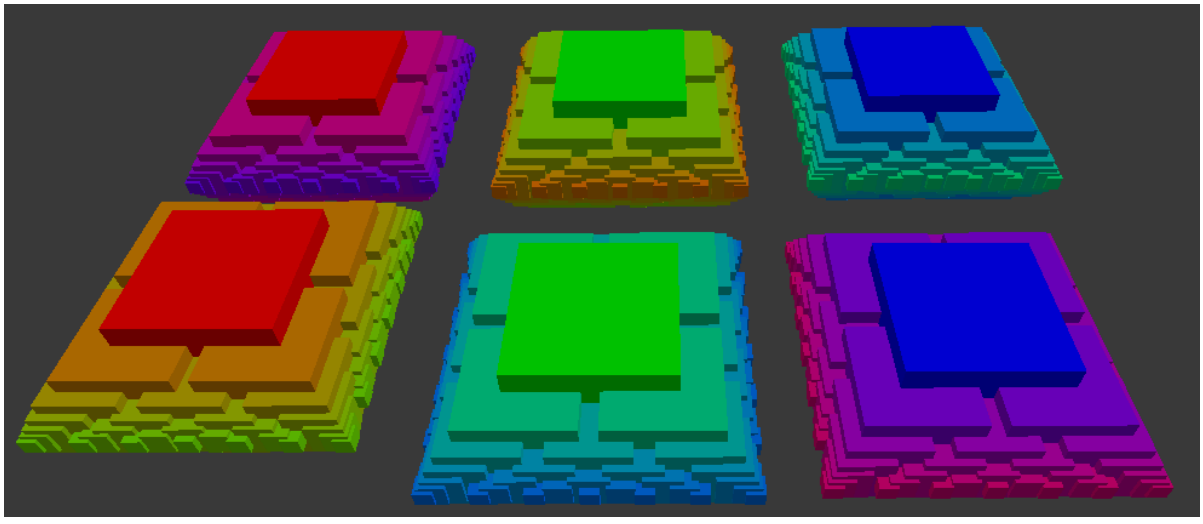
Slika 42: 3D-barvna paleta na spletu varnih 216 barvnih odtenkov — zgornji del, vir: lasten

Bolj kot je posamezna kocka oddaljena od te bele kocke, več barve od posameznega kanala ima (koda 4).

```
import bpy
from random import randint

glst = 1

meja = 2

while (glst < meja):
    delcek = 3
    a = 0
    b = 0
    c = 0
    while (b < 16):
        while ( a < 16):
            while (c < 16 ):
                x = a
                y = b
                z = c
                red = x / 15
                gre = y / 15
                blu = z / 15
                bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= 1 )
                activeObject = bpy.context.active_object
                mat = bpy.data.materials.new(name="MaterialName")
                activeObject.data.materials.append(mat)
                bpy.context.object.active_material.diffuse_color = (red, gre, blu)
                c = c + delcek
            a = a + delcek
            c = 0
        b = b + delcek
        a = 0
    glst = glst + 1
    b = 0
```

Koda 4: Koda za ustvarjanje barvne palete216 barvnih odtenkov, lastna koda

Tako izgleda koda za zgornje telo (slika 40). Ta skripta nam ni vzela dosti časa, saj smo takrat, ko smo to kodo pisali, že vedeli kar nekaj stvari o Pythonu. Bili smo pa gotovi veliko hitreje, kot če bi vsako kocko ročno dodali in vsaki šli dodajat svoj material. S tem pa smo potrdili četrto hipotezo – s pisanjem skript smo pri 3D-oblikovanju lahko bolj natančni in prihranimo s časom.
A tu se z barvami nismo ustavili. Hoteli smo, da bi naše oblike dobivale barve, ki bi se spreminjale. In to nam je tudi uspelo (spodnja slika) – primarne barve (v Blenderju) postopoma prehajajo v sekundarne.
Telesa na naslednji sliki so fraktali, ki smo jih odkrili (slika 43).

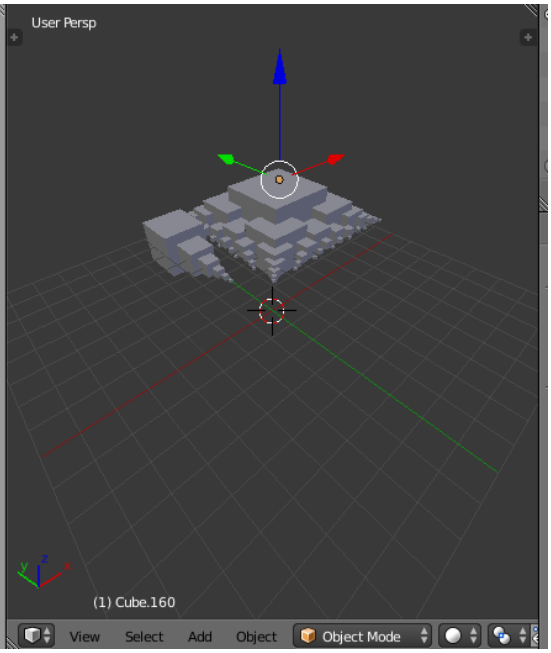


Slika 43: Prehodi barvnih odtenkov, vir: lasten

4.4 Miklavisnko stopnišče

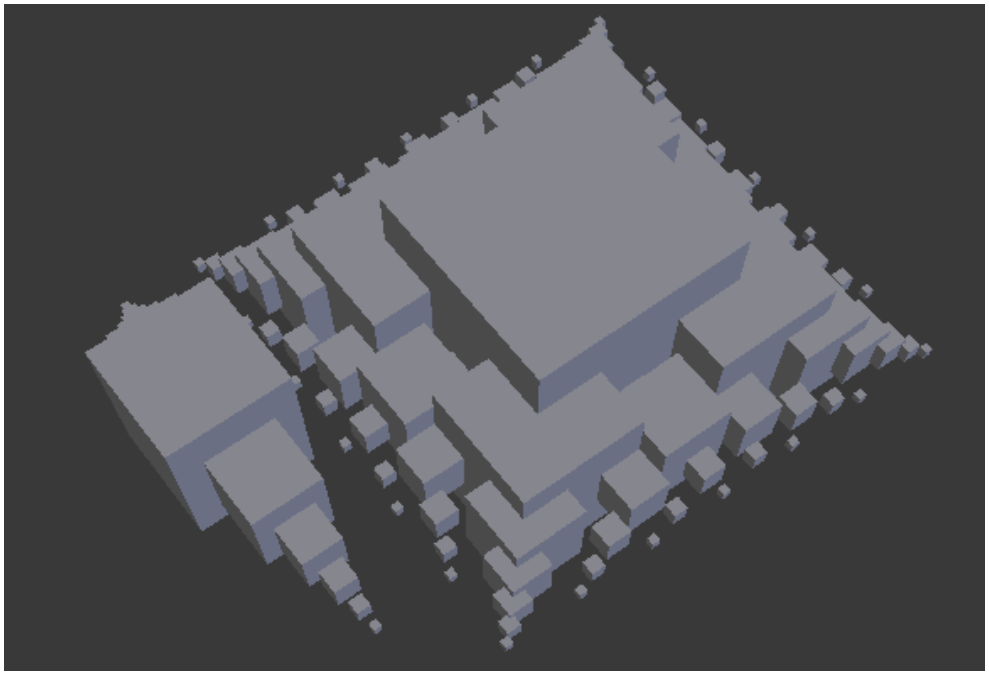
Ko smo spreminjali vrednost eksponentne enačbe (potenca) pri kodi Sierpinski, smo dobili presenetljiv rezultat (slika 44 in koda 5).

```
import bpy
from random import randint

glst = 1
#glavni stevec
stst = 2
#stranski stevec
meja = 8

red = 1.0
gre = 0.0
blu = 0.0

while (glst < meja):
    velikost = 3 ** ( 1 - glst /2)
    delcek = 10 / stst
    a=delcek
    b=delcek
    while (b < 10):
        #b = b + delcek
        while ( a < 10):
            x = a
            y = b
            z = 0
            bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
            a = a + delcek
        b = b + delcek
        a = 0
        # lahko pa damo na delcek, a tako izgubimo drugo telo
    glst = glst + 1
    stst = stst + 1
    b = 0
    red = ((red * 3 ) / 4 )
    blu = 1 - red
```



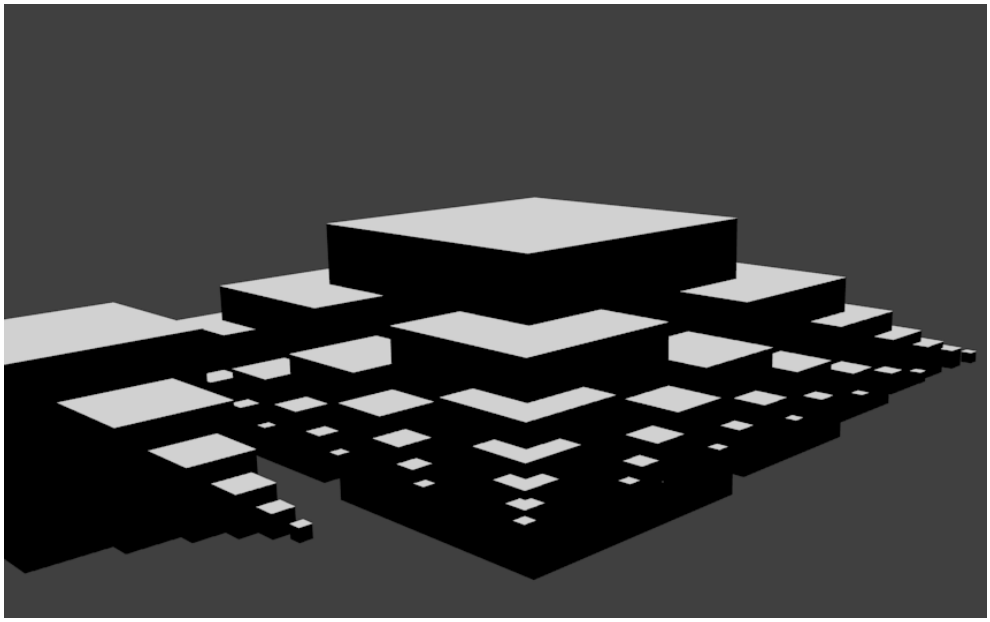
Slika 44: Koda Miklavinsko stopnišča (1 — x/2), vir: lasten

Gmota kock na levi, ki je ločena od glavnega telesa (slika 45), je rezultat napake v kodi, a ker se nam zdi, da ne izgleda slabo, smo se odločili, da je ne bomo popravili (smo pa jo pri nadaljnjih verzijah).



Slika 45: Miklavinsko stopnišče (1 — x/2), od zgoraj, vir: lasten

Na naslednji sliki se vidi to telo, če je osvetljeno le od zgoraj (slika 44).



Slika 46: Miklavinsko stopnišče(1 — x/2), črnobelo, vir: lasten

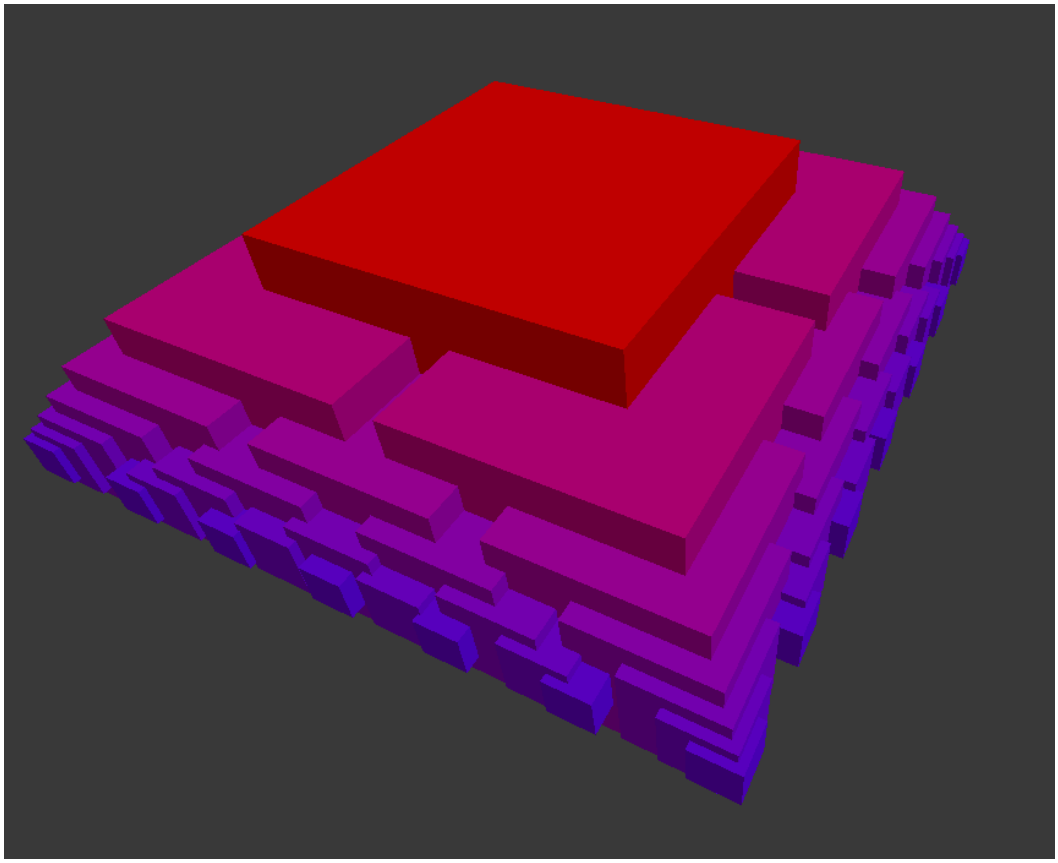
Nepričakovanega odkritja smo bili veseli, a se nismo ustavil le pri tem. Zamenjali smo cilj – namesto, da smo poskušali kopirati stvari, ki so že bile ustvarjene, smo raje raziskovali naprej. Kmalu smo odkrili še več zanimivih oblik.

V kodi (koda 4) smo spremenili potečno enačbo za velikost in dodali kodo za barve (koda 5).

```
import bpy
from random import randint

glst = 1
#glavni stevec
stst = 2
#stranski stevec
meja = 8

red = 1.0
gre = 0.0
blu = 0.0

while (glst < meja):
    velikost = 3 ** ( 1 - glst /3 )
    delcek = 10 / stst
    a=delcek
    b=delcek
    while (b < 10):
        #b = b + delcek
        while ( a < 10):
            x = a
            y = b
            z = 0
            bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
            activeObject = bpy.context.active_object
            mat = bpy.data.materials.new(name="MaterialName")
            activeObject.data.materials.append(mat)
            bpy.context.object.active_material.diffuse_color = (red, gre, blu)
            a = a + delcek
        b = b + delcek
        a = delcek
        # lahko pa damo na zacetek, a tako izgubimo drugo telo
    glst = glst + 1
    stst = stst + 1
    b = delcek
    red = ((red * 3 ) / 4 )
    blu = 1 - red
```

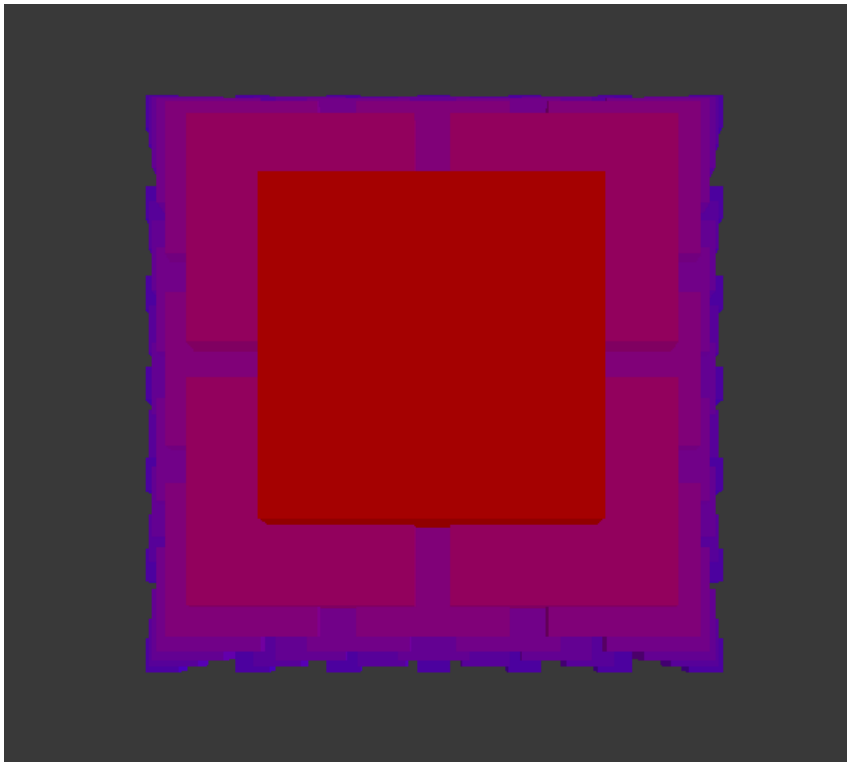
Koda 5: Miklavinsko stopnišče (1 — x/3)

Za prehod barv smo si večinoma izbirali rdečo, ki prehaja v vijolično. Za lahkoto bi spremenili kodo in dobili katero drugo kombinacijo prehajanja barvnih odtenkov (slika 47).



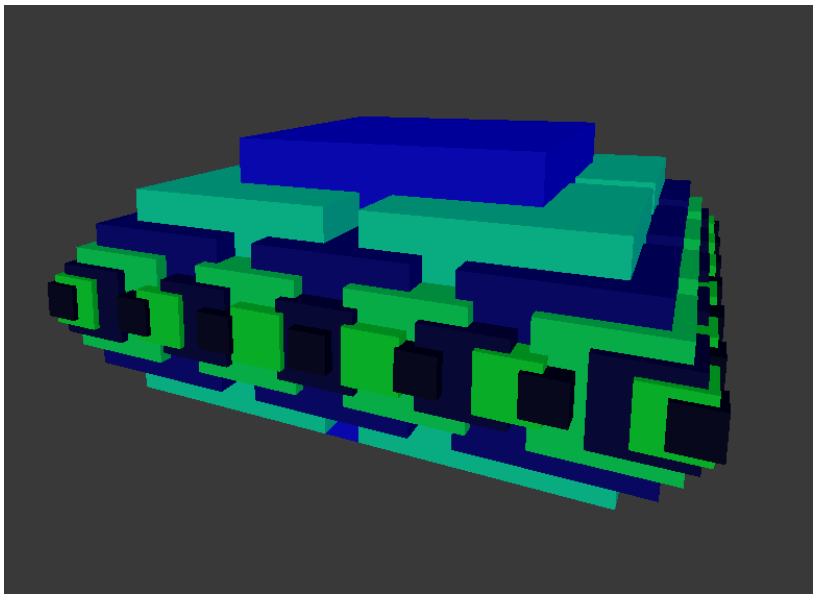
Slika 47: Miklavinsko stopnišče (1 — x/3), vir: lasten

S pogledom od zgoraj (naslednja slika) lahko vidimo, kam se postavljajo kocke med ponovitvami (slika 48).



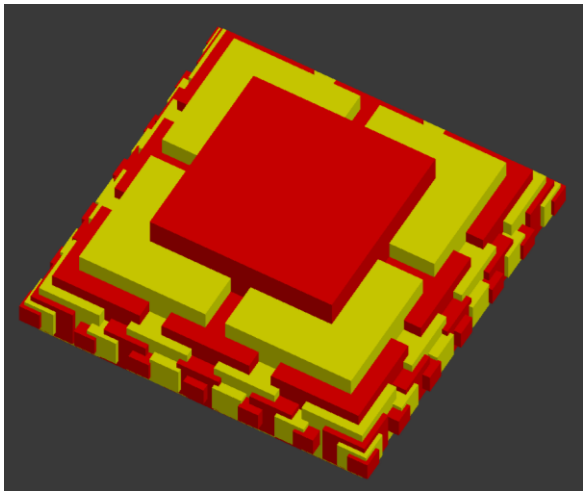
Slika 48: Miklavinsko stopnišče (1 — x/3), od zgoraj, vir: lasten

Eksperimentirali smo tudi z barvami. Na zgornji sliki (slika 48) smo povečevali vrednost zelene, modro pa dajali izmenično (na začetku, v drugi ponovitvi in v preostalih sodih ponovitvah).



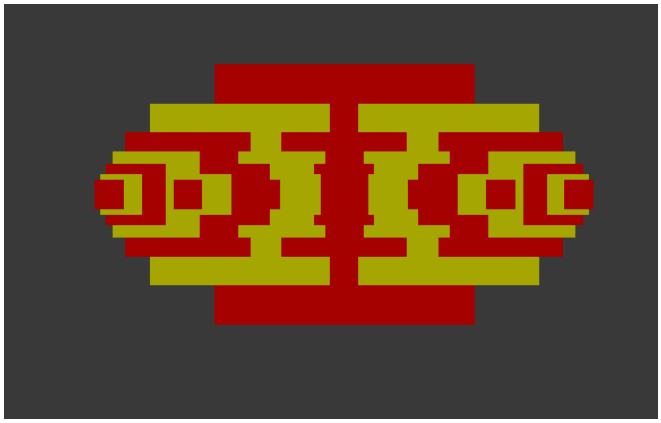
Slika 49: Miklavinsko stopnišče (1 — x/3), modrozeleno, vir: lasten

Še posebej všeč nam je bila rdeče-rumena kombinacija (slika 50).



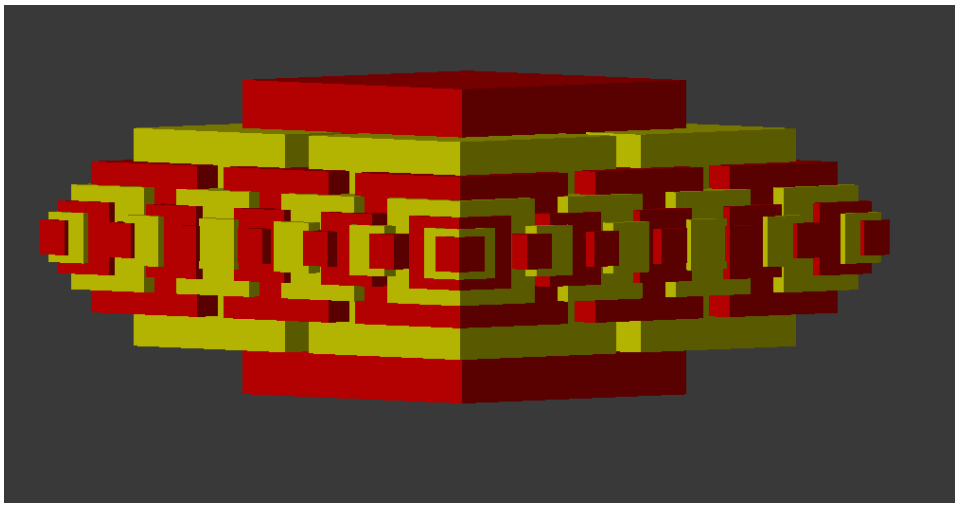
Slika 50: Miklavinsko stopnišče (1 — x/3), rdeče-rumeno, vir: lasten

Zanimiv pa je tudi pogled od strani (slika 51).



Slika 51: Miklavinsko stopnišče (1 — x/3), rdeče-rumeno, od strani, vir: lasten

Ta kombinacija je v kodi izgledala takole: Rdeča barva je vedno na maksimumu, zelena pa izmenično na maksimumu in minimumu (slika 52).



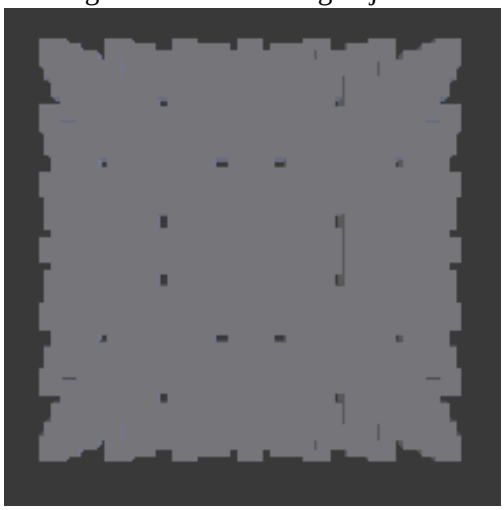
*Slika 52: **Miklavinsko stopnišče** (1 — x/3), rdeče-rumeno, rob, vir: lasten*

V tej kodi smo znova spremenili potenčno enačbo - spremenljivko »velikost« v 17. vrstici v kodi (koda 5).

```
1  import bpy
2  from random import randint
3
4  glst = 1
5  #glavni stevec
6  stst = 2
7  #stranski stevec
8  meja = 8
9
10 red = 1.0
11 gre = 0.0
12 blu = 0.0
13
14 while (glst < meja):
15     # ** je znak za potenco
16     #while more met :
17     velikost = 3 ** ( 0.5 - glst /4 )
18
19     delcek = 10 / stst
20     a=delcek
21     b=delcek
22     while (b < 10):
23         #b = b + delcek
24         while ( a < 10):
25             x = a
26             y = b
27             z = 0
28             bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
29             activeObject = bpy.context.active_object
30             mat = bpy.data.materials.new(name="MaterialName")
31             activeObject.data.materials.append(mat)
32             bpy.context.object.active_material.diffuse_color = (red, gre, blu)
33             a = a + delcek
34         b = b + delcek
35         a = delcek
36         # lahko pa damo na zacetek, a tako izgubimo drugo telo
37     glst = glst + 1
38     stst = stst + 1
39     b = delcek
40     red = ((red * 3 ) / 4 )
41     blu = 1 - red
```

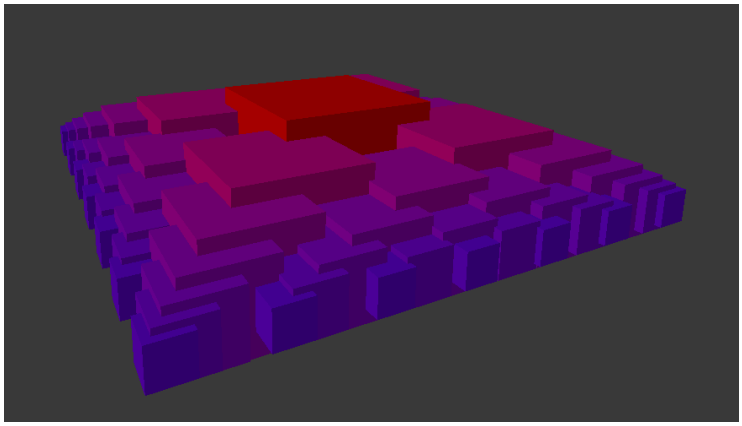
Koda 6: Miklavinsko stopnišče (0.5 — x/4)

Tako zgleda ta fraktal od zgoraj. Zanimivo je videti, kakšen vzorec delajo praznine (slika 53).



Slika 53: Nepobarvano Miklavinsko stopnišče (0.5 — x/4), vir: lasten

Tako pa zgleda od strani (slika 54).



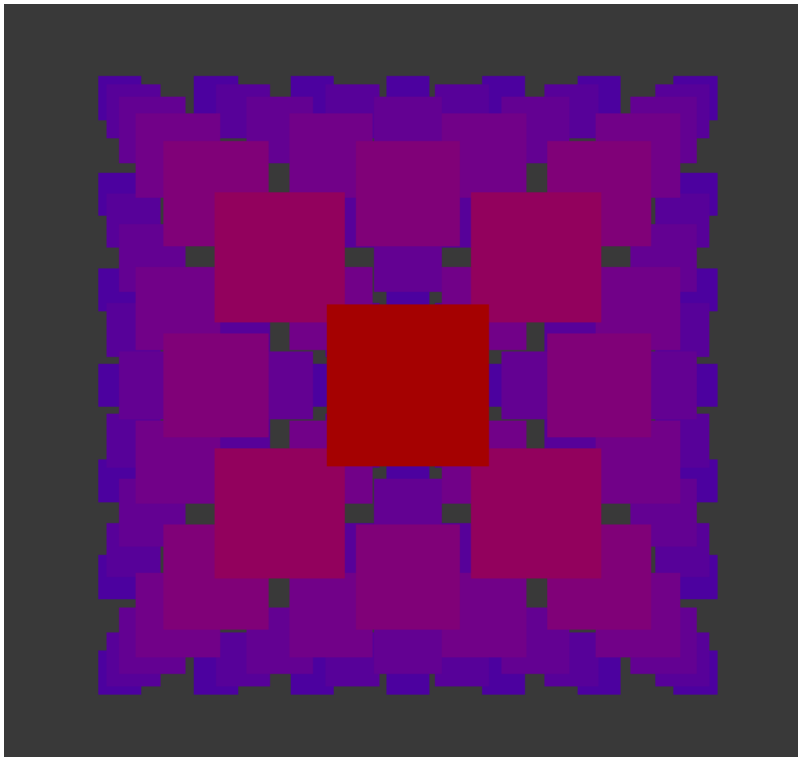
Slika 54: Miklavinsko stopnišče (0. 5 — x/4), barvno, vir: lasten

Še en primer kode in njenega rezultata (koda 7) je upodobljen na naslednjih slikah (slika 55 in slika 56).

```
import bpy
from random import randint

glst = 1
#glavni stevec
stst = 2
#stranski stevec
meja = 8

red = 1.0
gre = 0.0
blu = 0.0

while (glst < meja):
    # ** je znak za potenco
    #while more met :
    velikost = 3 ** ( 0.25 - glst /5 )

    delcek = 10 / stst
    a=delcek
    b=delcek
    while (b < 10):
        #b = b + delcek
        while ( a < 10):
            x = a
            y = b
            z = 0
            bpy.ops.mesh.primitive_cube_add(location=(x,y,z), radius= velikost)
            activeObject = bpy.context.active_object
            mat = bpy.data.materials.new(name="MaterialName")
            activeObject.data.materials.append(mat)
            bpy.context.object.active_material.diffuse_color = (red, gre, blu)
            a = a + delcek
        b = b + delcek
        a = delcek
        # lahko pa damo na zacetek, a tako izgubimo drugo telo
    glst = glst + 1
    stst = stst + 1
    b = delcek
    red = ((red * 3 ) / 4 )
    blu = 1 - red
```

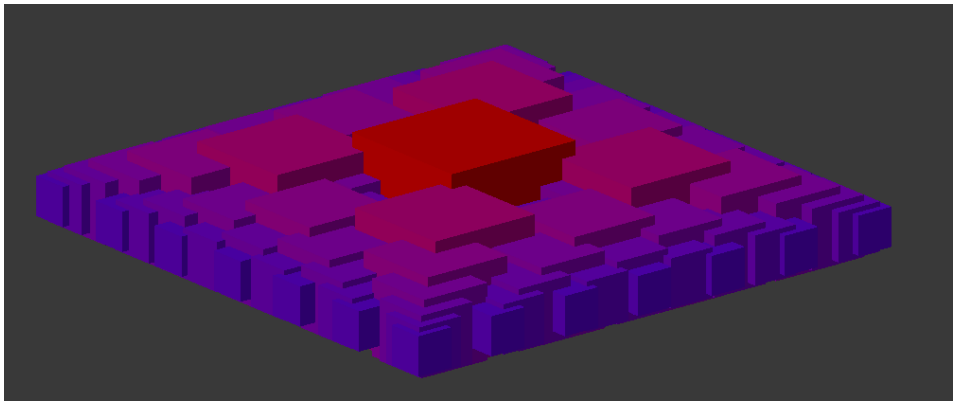
Koda 7: Miklavinsko stopnišče (0.25 — x/5)

In tudi naslednji fraktal je zanimiva oblika (slika 55).



Slika 55: Miklavinsko stopnišče (0. 25 — x/5), od zgoraj, vir: lasten

Pogled od strani na naslednji sliki (slika 56).



Slika 56: Miklavinsko stopnišče (0. 25 — x/5), od strani, vir: lasten

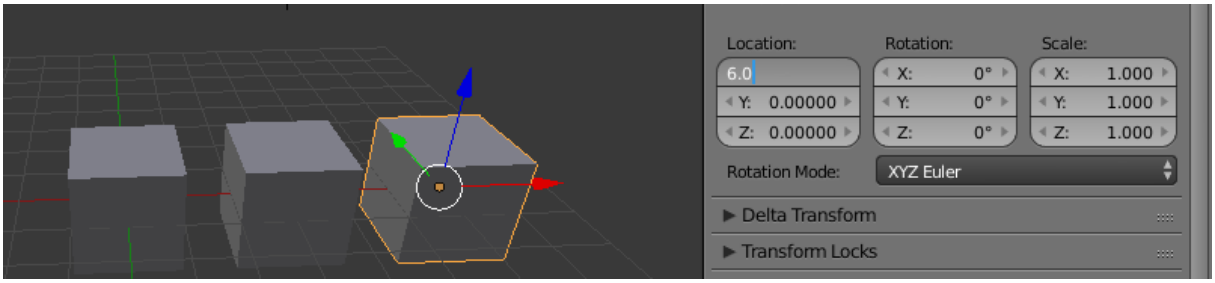
Glede na prejšnje slike lahko brez pomisleka potrdimo drugo in tretjo hipotezo: Blender je primeren program za vizualizacijo fraktalov. Python je primeren jezik za pisanje skript, ki ustvarjajo fraktale. Mladi raziskovalec pa je naš fraktal tudi poimenoval.Naš objekt se imenuje **Miklavinsko stopnišče.** Za ločevanje med posameznimi različicami objektov pa je najbolje, da uporabimo zapis s potenčno enačbo, ki določa velikost kock.

5 RAZPRAVA

Pri razpravi se bomo najprej ukvarjali s četrto hipotezo – s pisanjem skript smo pri 3D-oblikovanju lahko bolj natančni in prihranimo s časom.
Čeprav smo jo prej že potrdili, pa ji moramo vseeno nameniti nekaj pozornosti, saj nismo preverili vseh območij veljavnosti (definicijsko območje). Preverili smo le dodajanje končnega števila objektov in določanja atributov (v našem primeru barve) glede na položaj ter dodajanja novih objektov po določeni matematični formuli, zato smo se odločili še za en preizkus.
Izbrali smo si preproste naloge:

1. postaviti tri kocke v vrsto – ena od druge naj bodo oddaljene natančno eno enoto (slika 57);
2. postaviti stolp iz desetih kock, pri katerih je vsaka kocka manjša za eno četrtino (slika 59);
3. izdelati preprost model snežaka (tri krogle po vertikalni liniji in le stožec za nos – slika 61).

Nato smo šli vsako nalogo izvest enkrat z ročnim modeliranjem (slika 57, slika 59, slika 61) in enkrat s kodo (slika 58, slika 60, slika 62) ter ob tem merili čas (tabela 1).
Za prvo nalogo (kocke v vrsti) smo porabili malo več kot minuto (slika 57 in tabela 1).

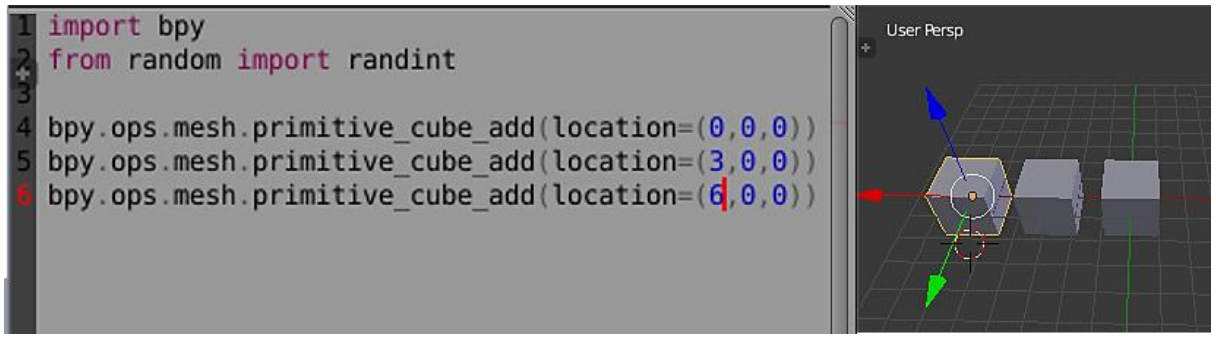


Slika 57: Tri kocke v vrsto – 1. naloga – ročno, vir: lasten

Tabela 1: Časi modeliranja

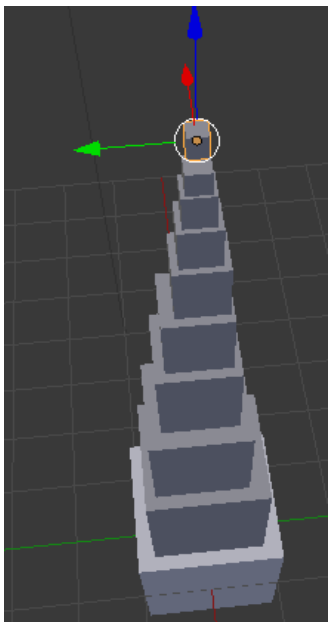
Ročno modeliranje (min.)	Modeliranje s kodo (min.)
1. 07	0. 57
5. 22	4. 09
1. 22	2. 23

Za prvo nalogo (kocke v vrsti) s kodo pa smo porabili malo manj kot minuto (slika 58 in tabela 1).



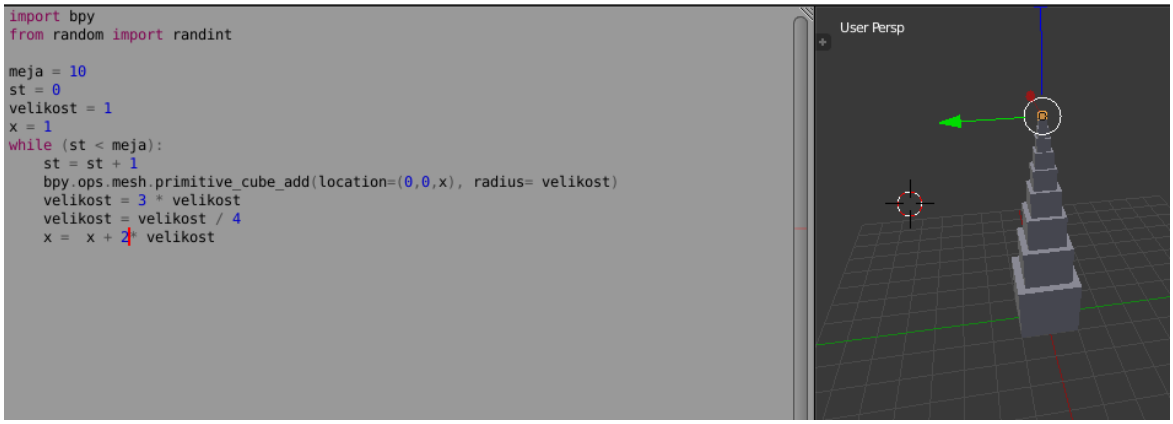
Slika 58: Tri kocke v vrsto – 1. naloga – s kodo, vir: lasten

Za drugo nalogo (stolp) smo ročno porabili več kot pet minut, rezultat pa bi lahko bil bolj natančen (slika 59 in tabela 1).



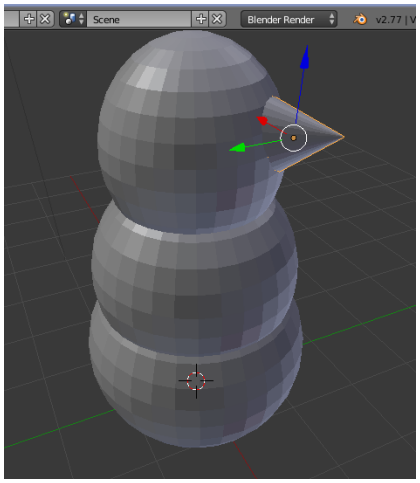
Slika 59: Stolp iz kock – 2. naloga – ročno modeliranje, vir: lasten

Za drugo nalogo (stolp) smo za kodo porabili malo več kot štiri minute, rezultat pa je bil natančen (slika 60 in tabela 1).



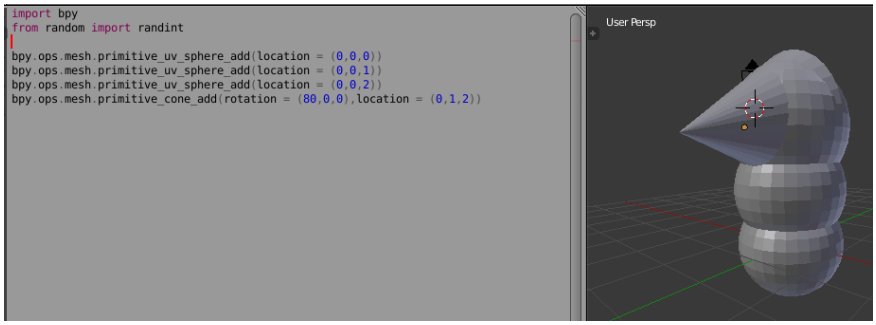
Slika 60: Stolp iz kock – 2. naloga – s kodo, vir: lasten

Za tretjo nalogo (snežaka) smo ročno porabili več kot minuto, rezultat pa je bil zadovoljiv (slika 61 in tabela 1).



Slika 61: Snežak – 3. naloga – ročno, vir: lasten

Za tretjo nalogo (snežak) smo za kodo porabili več kot dve minuti, rezultat pa je bil manj zadovoljiv, saj je stožec za nos prevelik (slika 62 in tabela 1).



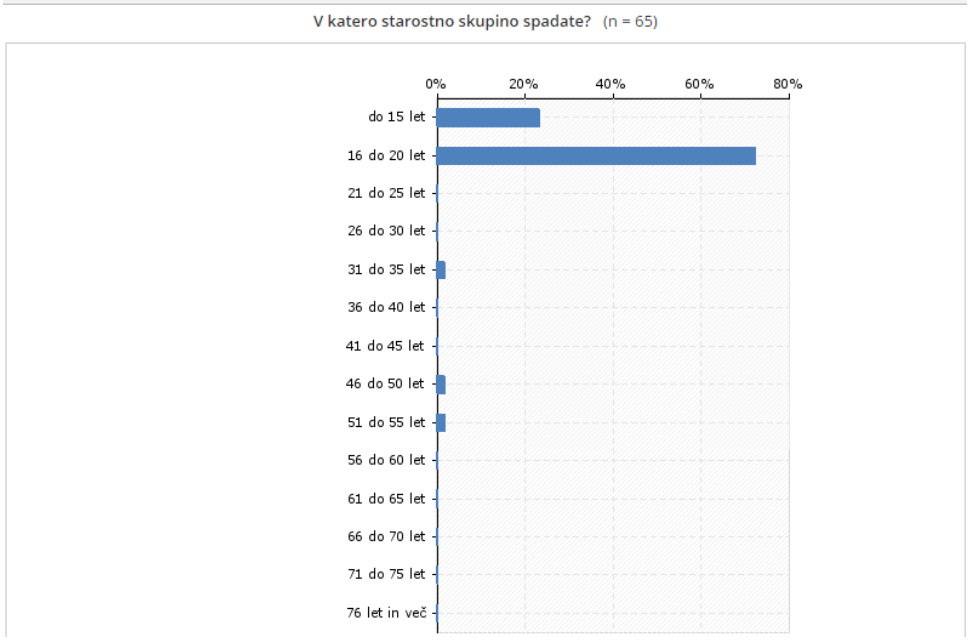
Slika 62: Snežak – 3. naloga – s kodo, vir: lasten

Pri prvi nalogi smo dobili enaka rezultata, le da smo s kodo bili hitrejši.
Pri drugi nalogi smo bili s kodo ne le hitrejši, ampak tudi natančnejši.
Pri tretji nalogi smo ročno dobili lepši rezultat in bili tudi hitrejši.

Vidimo lahko, da rezultata prvih dveh nalog potrjujeta hipotezo, rezultat tretje naloge pa postavljeni hipotezi nadsprotuje, zaradi tega to hipotezo lahko le pogojno potrjujemo.

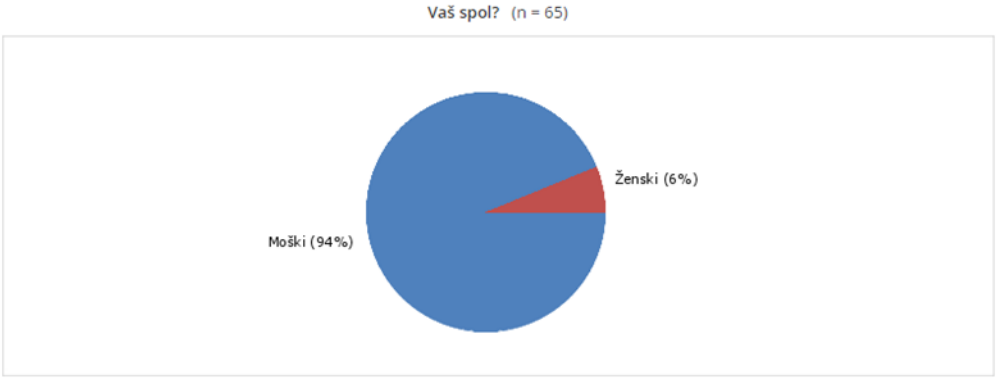
5.1 Rezultati ankete

Na anketo, ki smo jo izvajali od 15. 1. 2017 do 9. 2. 2017, so se odzvali sto štirje (104) anketiranci. Veljavnih anket je bilo petinšestdeset (65). Anketiranci so bili večinoma moškega spola, stari od 15 do 20 let (graf 1).



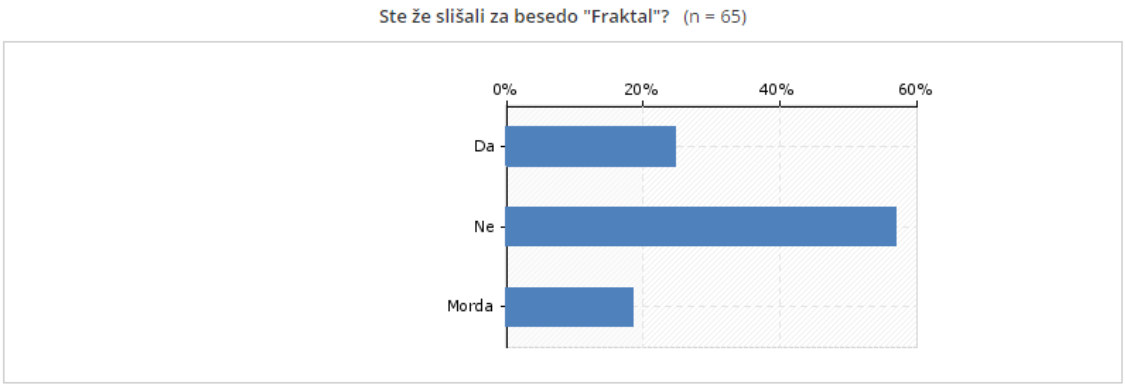
Graf 1: Starost anketirancev

Iz zgornjega in spodnjega grafa lahko vidimo, da je bila večina anketirancev po starosti in spolu enaka mlademu raziskovalcu (graf 2).



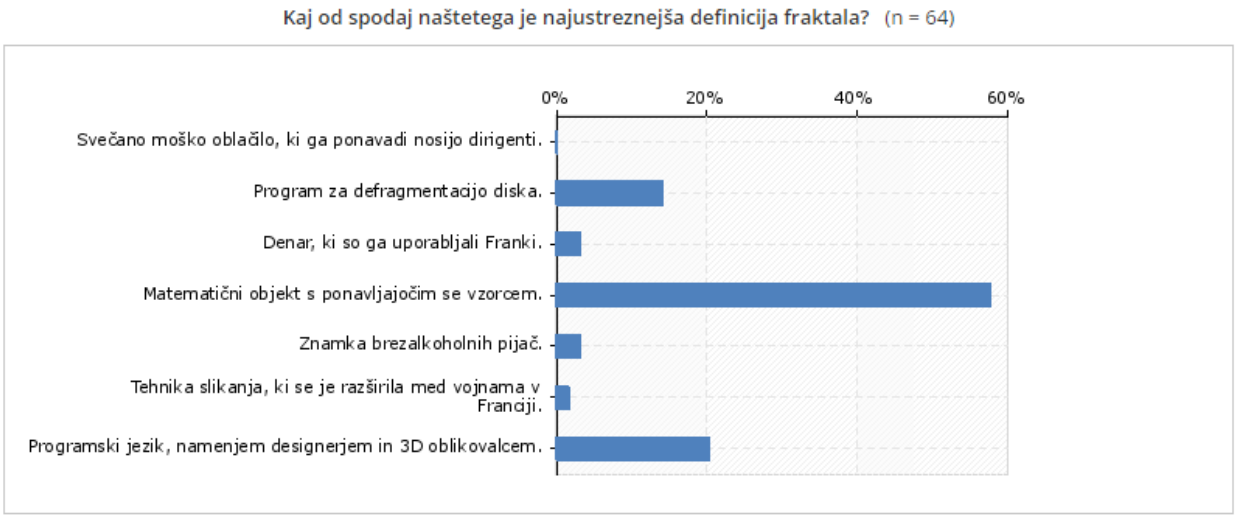
Graf 2: Spol anketirancev

Na 3. vprašanje – Ste že slišali za besedo fraktal? – je pritrdilno odgovorilo le 25 % anketirancev, kar je ena četrtina. To ovrže našo peto hipotezo – za fraktale je slišala vsaj tretjina anketirancev, a manj kot desetina ve, kaj fraktali so – saj smo predvidevali, da bo pritrdilnih odgovorov vsaj ena tretjina – 33 % (graf 3).



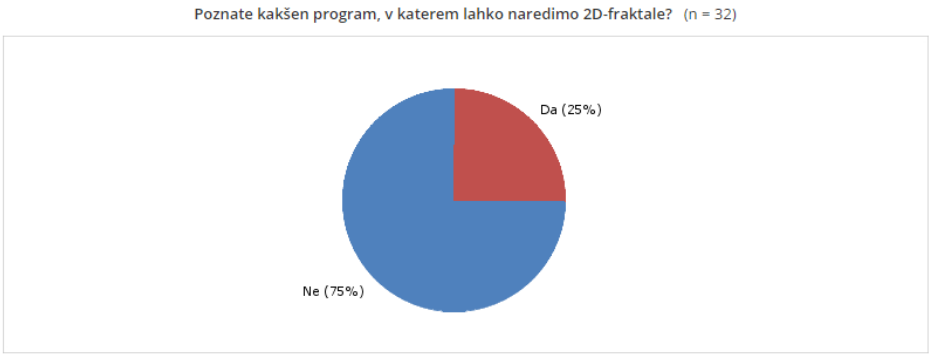
Graf 3: Ste že slišali za fraktale?

Na naše presenečenje je bila večina odgovorov pri zgornjem vprašanju (graf 4) pravilnih. Presenečeni smo tudi, da ni nihče izbral prvega. Pričakovali smo, da bo vsaj nekaj anketirancev zamenjalo »frak« in »fraktal«.



Graf 4: Kaj je fraktal?

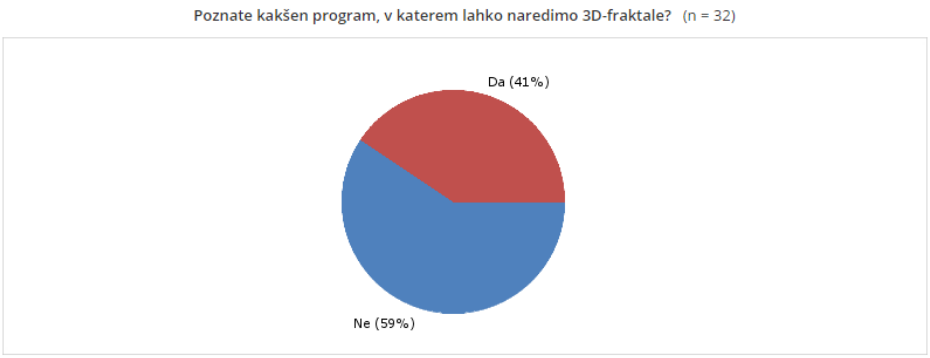
Iz dveh postavljenih podvprašanj smo izvedeli, da večina anketirancev ne pozna programov (graf 5 in graf 6), v katerih lahko ustvarjamo fraktale.



Graf 5: Poznate program za 2D-fraktale?

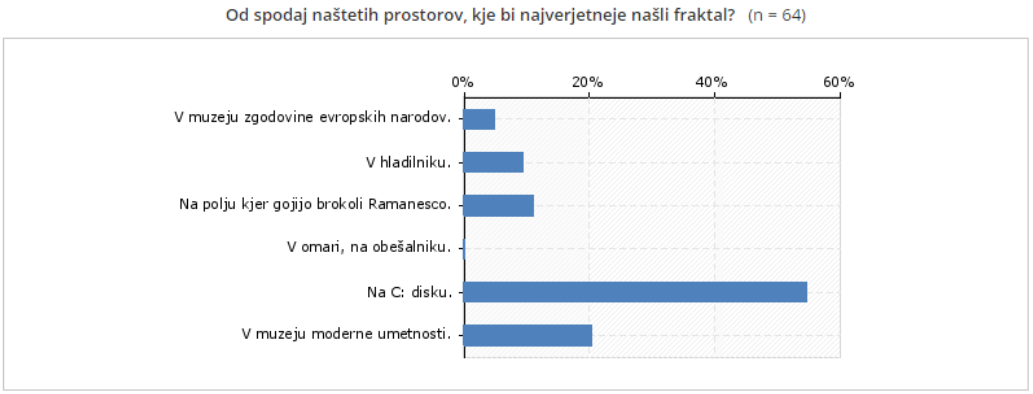
Podvprašanji sta se pokazali le, če je anketiranec pravilno odgovoril na vprašanje 4. Sklepamo, da je bil to tudi razlog, da je pri vprašanju 4 bilo toliko pravilnih odgovorov. Temu se ne bi mogli izogniti, če bi bili podvprašanji vidni v vsakem primeru, saj bi lahko anketiranci iz njiju lahko sklepali, kaj je pravilen odgovor pri vprašanju 4. Lahko pa bi se temu izognili, če bi se podvprašanji pojavili za vsak

odgovor, a na to idejo smo prišli prepozno. Tudi ta podvprašanja bi lahko ponujala dva odgovora, primeri vprašanj pa bi lahko bili: Ali je fraktal rjave ali črne barve?, Ali je fraktal odprtokoden program?, Ali je bil en fraktal vreden več ali manj od ene bizantinske lire?, Ali je fraktal boljši, če je serviran topel ali hladen?, Ali je bilo s fraktalno tehniko lažje slikati abstraktne slike? in Ali je jezik fraktal bolj primeren za ustvarjanje rastrskih ali vektorskih slik?



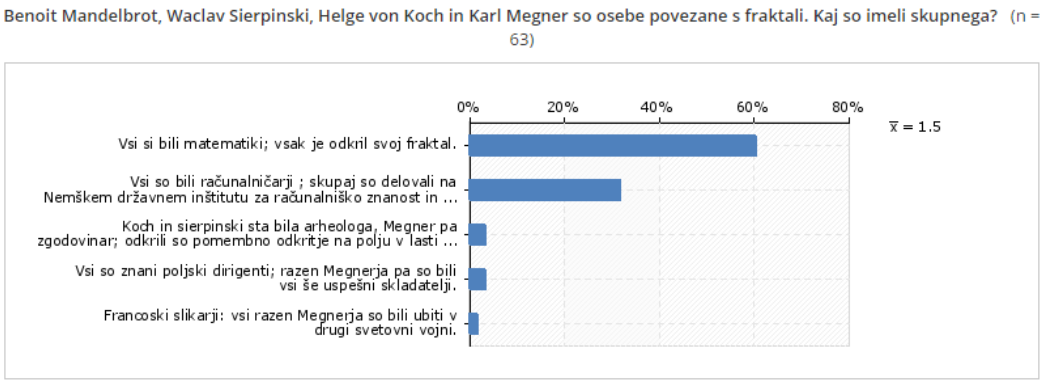
Graf 6: Poznate program za 3D-fraktale?

Vprašanje 5 – Kje bi lahko najverjetneje našli fraktal? – je bilo premeteno. Pravzaprav bi lahko našli fraktal v vsakem izmed naštetih prostorov, vendar bi ga samo na enem zagotovo – na polju brokolija sorte Ramanesco (slika 20). Lahko pa iz spodnjega grafa vidimo, da je večina anketirancev tu odgovorila narobe (graf 7).



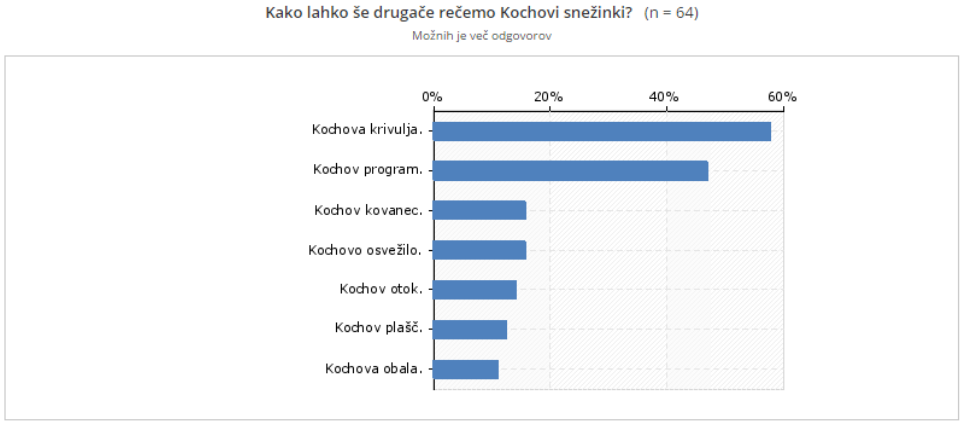
Graf 7: Kje bi našli fraktal?

Pri sledečem vprašanju pa je večina odgovorila pravilno, Fraktalisti so res bili matematiki, ki so odkrivali fraktale (graf 8). Iz množice odgovorov pa izstopa tudi, da so bili fraktalisti računalničarji – kar ne bi bilo tako narobe, če ne bi zraven pisalo, da so bili na »Nemškem državnem inštitutu za računalniško znanost in tehnologijo« – to pa ne bi bilo mogoče, saj smo si naziv za ta inštitut izmislili mi.



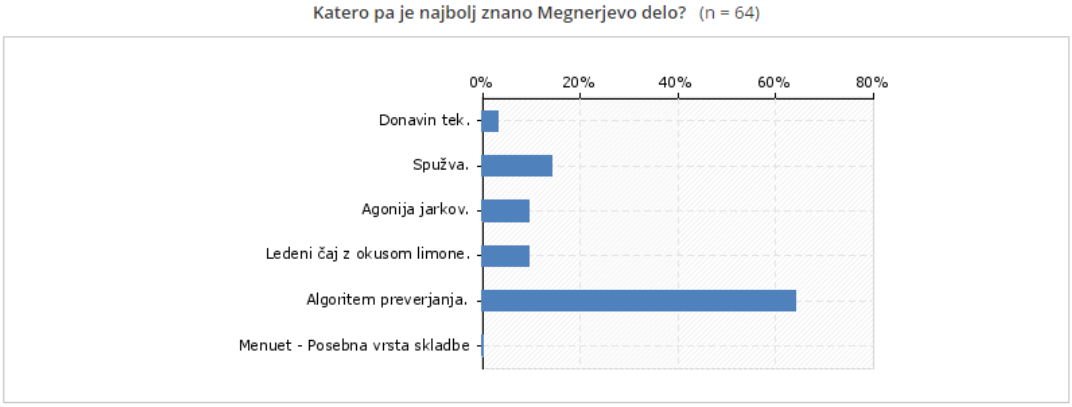
Graf 8: Kaj so fraktalisti?

Pri naslednjem vprašanju so bili pravilni trije odgovori (Kochova krivulja, otok in obala). Zanimivo je, da je bil eden izmed pravilnih odgovorov največkrat izbran, eden pa najmanjkrat (graf 9).



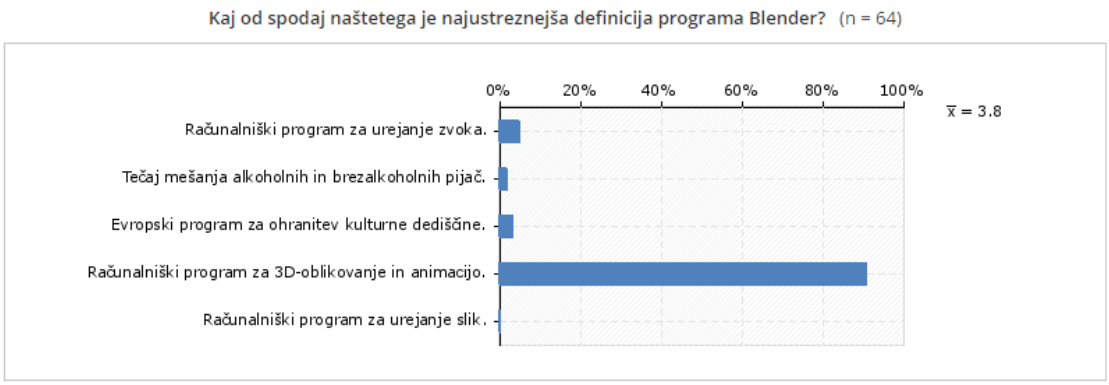
Graf 9: Kako lahko še rečemo Kochovi snežinki?

Tudi naslednje vprašanje je bilo zvito. Le kdo bi pomislil, da najbolj znano delo računalničarja in matematika ni noben algoritem, ampak spužva (graf 10).

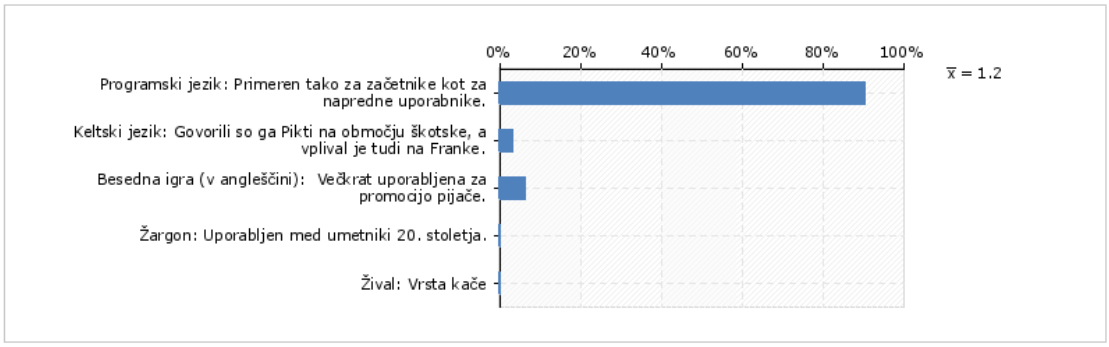


Graf 10: Megnerjevo najbolj znano delo?

Za konec pa smo vprašali še, kaj je Blender in kaj Python. Večina odgovorov je bila pravilna, kar pomeni, da sta Blender in Python med raziskovalčevimi vrstniki in drugimi dijaki šole dokaj poznana (graf 11 in graf 12).



Graf 11: Kaj je Blender?



Graf 12: Kaj je Python?

6 ZAKLJUČEK

V raziskovalni nalogi smo nekaj zastavljenih hipotez potrdili, nekaj pa ovrgli.
Prvo hipotezo **– Blender je primeren program za ustvarjanje fraktalov –** smo potrdili. Fraktali imajo vzorce, ki se ponavljajo v neskončnost in jih praksi računamo do končanega števila, saj želimo, da računalniški program konča z izračunom in izriše upodobljeno sliko fraktala.
Drugo hipotezo – **Blender je primeren program za vizualizacijo fraktalov –** smo potrdili. Resda ne more v celoti izdelati fraktala v neskončno število iteracij, lahko pa vseeno izdeluje vzorce, dokler so dovolj veliki, da jih vidimo. Kot primer smo izdelali fraktal preprogo Sierpinskega in nekaj lastnih fraktalov.
Tretjo hipotezo – **Python je primeren jezik za pisanje skript, ki ustvarjajo fraktale –** smo tudi potrdili. Python je programski jezik, ki pa s ponavljanjem česa v končnih zankah nima problema. Problem prepusti računalniškim programom, ki jih poganja. Z njim smo napisali skripte, ki so v Blenderju upodabljale napisane fraktale.
Četrto hipotezo – **s pisanjem skript smo pri 3D-oblikovanju lahko bolj natančni in prihranimo s časom** – smo v razpravi pogojno potrdili. S kodo smo lahko hitrejši v primeru, da moramo slediti matematični formuli. Če pa modeliramo naključen objekt, pa bomo z ročnim modeliranjem ne le prihranili s časom, temveč bomo imeli tudi večji nadzor nad končnim rezultatom.
Peto hipotezo **– za fraktale je slišala vsaj tretjina anketirancev, a manj kod desetina ve, kaj fraktali so –** smo ovrgli, saj je po anketi za fraktale slišala le četrtina anketirancev,

Mladi raziskovalec je vesel, da mu je uspelo dokazati, da lahko nekatere znane fraktale vizualiziramo v Blenderju s programom Python. Še posebej je navdušen, da mu je uspelo odkriti nov fraktal. In s čim mu je to uspelo? Z eksperimentiranjem, s spontanostjo in z naključjem.

O Pythonu in Blenderju smo se naučili veliko več, kot smo vedeli na začetku (kar je bilo skoraj nič), prišli smo tudi do nekaj pomanjkljivosti in težav. Te smo uspešno rešili z lastno iznajdljivostjo in viri na medmrežju in v knjigah. Smo pa ugotovili, da v povprečju ni veliko oseb, ki bi se s tem ukvarjale.

Ugotovili smo, da je za uspeh potrebno imeti pravo idejo, čas, vztrajnost in voljo, pa tudi zagnanost; da ideja ne ostane samo ideja ter postane v našem primeru program in upodobljena slika ali model fraktala.

7 ZAHVALA

Raziskovalna naloga ne bi bila v takšni obliki, če nam pri njenem nastajanju ne bi pomagalo veliko ljudi. Zahvala je torej namenjena naslednjim:

- mentorju Nedeljku Grabantu, dipl. inž., pomoč, voljo, vztrajnost, prosti čas in spodbudo pri nastajanju;
- staršem
- Sonji Lubej, prof. za lektoriranje;
- Jolandi Melanšek, prof. za lektoriranje angleškega povzetka;
- učiteljem ERŠ-a in ravnatelju Simonu Konečniku, univ. dipl. inž., za vso podporo in razumevanje;
- recenzentu raziskovalne naloge;
- komisiji Mladih raziskovalcev in koordinatorici gibanja Mladi raziskovalci Karmen Hudournik;
- dijakom, učiteljem in ostalim, ki so izpolnjevali anketo,
- vsem neomenjenim, ki so kakorkoli pomagali pri izdelavi naloge.

8 VIRI

[1] https://www. python. org/static/community_logos/python-logo-master-v3-TM.png, 7. 1. 2017
[2] http://blenderartists. org/forum/attachment. php?attachmentid=319036&d=1404598838, 23. 12. 2016
[3] http://7-themes. com/6792222-free-forest-moss-wallpaper.html, 14. 12. 2016
[4] http://www. pixelstalk. net/wp-content/uploads/2016/08/Cool-Lightning-Storm-Background.jpg, 14. 2. 2016
[5] http://wallpapercave. com/wp/0IQv6Ho. jpg, 14. 12. 2016
[6] http://www. hd-fractals. com/images/fern-leaf-web.jpg, 14. 12. 2016
[7] http://www. hd-fractals. com/images/fern. gif, 3. 1. 2017
[8] http://i. telegraph. co. uk/multimedia/archive/01535/white-lightning_1535353i. jpg, 14. 12. 2016
[9] https://s-media-cache-ak0. pinimg. com/564x/64/ad/ec/64adecdfe363357f374dc66c2111003d. jpg, 3. 1. 2017
[10] https://s-media-cache-ak0. pinimg. com/236x/f1/c6/a7/f1c6a70c0a8a1dd75fc8be7b8d4890ae.jpg, 3. 1. 2017
[11] https://i. kinja-img. com/gawker-media/image/upload/s--HBhBI6PB--/1464298697802922311.jpg, 3. 1. 2017
[12] https://www. fourmilab. ch/images/Romanesco/images/Lcr1.jpg, 3. 1. 2017
[13] https://upload. wikimedia. org/wikipedia/commons/5/5f/Wac%C5%82aw_Sierpi%C5%84ski.jpg, 3. 1. 2017
[14] http://mathworld. wolfram. com/images/eps-gif/SierpinskiCarpet_730. gif, 3. 1. 2017
[15] http://www. cplusplus. com/articles/LyTbqMoL/serpinskiTriangles.jpg, 3. 1. 2017
[16] https://upload. wikimedia. org/wikipedia/commons/9/95/Karl_Menger_1970_Shimer_College_Wiki.jpg, 3. 1. 2017
[17] https://upload. wikimedia. org/wikipedia/commons/9/9b/Menger_sponge_%28IFS%29. jpg, 3. 1. 2017
[18] http://www. shodor. org/media/N/D/Q/yNTUxOWI4MWI5OWU5Y2Y0MDU2NTg2ZjVlZDZhMjU.gif, 3. 1. 2017
[19] https://upload. wikimedia. org/wikipedia/commons/0/0e/Helge_von_Koch. jpg, 3. 1. 2017
[20] https://panoramamoments. files. wordpress. com/2015/01/benoit-mandelbrot_fractal-dimension_01.jpg?w=1024, 3. 1. 2017
[21] Raziskovalna naloga: 20160411_v2_3d-krivulje_blender_python, 3. 1. 2017
[22] https://en. wikipedia. org/wiki, 3. 1. 2017
[23] http://mathematica. stackexchange. com/questions/102697/revolution-of-koch-snowflake, 3. 1. 2017
[24] https://sl. wikipedia. org/wiki/Mandelbrotova_mno%C5%BEica, 4. 1. 2017

9 AVTOR RAZISKOVALNE NALOGE

David Mikek je dijak 4. letnika Elektro in računalniške šole (ERŠ) v Velenju. Za raziskovalno nalogo se je odločil, ker ga zanima programiranje in različni načini uporabe Blenderja s pomočjo Pythona. Zanimajo ga tudi drugi programski jeziki. V prihodnosti se želi ukvarjati s programiranjem, z umetno inteligenco in z zagotavljanjem varnosti v računalniških sistemih.

Slika 63: David Mikek, lastna slika

10 PRILOGA 1. BLENDER DATOTEKE IN PRIMERI 3D-FRAKTALOV