

TDK DOLGOZAT

Barcsák Csaba

Intézmény: Miskolci Egyetem

Kar: Gépészmérnöki és Informatikai Kar

Tanszék: Alkalmazott Matematikai Tanszék

Szerző: Barcsák Csaba

Témavezető: Dr. Agbeko Kwami Nuteke, egyetemi docens

Dolgozat címe:

Determinisztikus hiperkomplex fraktálok háromdimenziós megjelenítése sugárkövetés
alapú térfogatvizualizációval

Köszönetnyilvánítás

Köszönetemet fejezem ki témavezetőmnek, Dr. Agbeko Kwami Nutefe tanár úrnak, aki figyelemmel kísérte munkámat, sokat segített az optikai modellek megértésében, és hasznos tanácsokkal látott el. Köszönettel tartozom Dr. Juhász Imre tanár úrnak, aki dolgozatom elolvasása után építő javaslatokat tett.

A bemutatott kutató munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Tartalomjegyzék

1. Bevezetés	1
2. Kaotikus dinamikus rendszerek	2
2.1. Hausdorff-dimenzió	2
2.2. Kaotikus dinamikus rendszerek a síkon	3
2.2.1. Júlia-halmazok	3
2.2.2. Mandelbrot-halmaz	4
2.3. Kaotikus dinamikus rendszerek magasabb dimenziókban	4
3. Megjelenítés	5
3.1. Sugárkövetés alapú térfogatvizualizáció	6
3.2. A térfogatvizualizációs integrál	7
3.2.1. Fényelnyelés	7
3.2.2. Fénykibocsátás	7
3.2.3. Fénykibocsátás és fényelnyelés	8
3.2.4. Árnyékolás	8
3.3. Hiba analízis	9
3.3.1. Trapéz-formula	9
3.3.2. Simpson-formula	9
3.3.3. Megoldások összevetése	10
3.4. Megvilágítási modellek	12
3.5. Osztályozás	13
3.5.1. Átviteli függvények	13
3.6. Globális megvilágítási modellek	14
3.6.1. Ambient Occlusion	14
3.6.2. Lokális Ambient Occlusion	14
3.6.3. Dinamikus Ambient Occlusion	15
3.6.4. Az optimális mérték	16
4. Elkészült alkalmazások	17
4.1. Java alkalmazás	17
4.2. CUDA alkalmazás	18
5. Konklúzió	19
Függelék:	
A.	22

1. fejezet

Bevezetés

Benoit Mandelbrot 1975-ben publikálta első könyvét a fraktálokról. A fraktál szó is tőle származik, a latin fractus szóból képezte a tört dimenziók megjelölésére. Mandelbrot eredetileg olyan halmazokat definiált fraktáloknak, melyeknek Hausdorff dimenziója meghaladja a topológiai dimenziójukat, de később minden önhasznó és majdnem önhasznó halmazt is ide sorolt. A fraktálok új területet nyitottak a tudomány világában, mely felkeltette matematikusok, fizikusok, informatikusok és művészek érdeklődését egyaránt. A természetben sok fraktálokkal leírható alakzatot és jelenséget találunk, ilyenek például egy fa ágai, az óceán hullámmozgása, örvények egy turbulens folyadékban.

A fraktálokról viszonylag kevés ismeretanyaggal rendelkezik a matematika, mivel a terület eléggé újkeletű. Megismerésük egyik módja, hogy megjelenítjük őket. Dolgozatunkban a determinisztikus, hiperkomplex iterációval előállítható fraktálok megjelenítésére helyeztük a fő hangsúlyt. Ezen halmazok vizualizációja jóval összetettebb, mint a komplex számsíkon megadható fraktáloké, emiatt az algoritmusok hatékonysága sokkal nagyobb jelentőséggel bír. Az irodalomban használt megoldások közül a felületalapú megjelenítési eljárások váltak elterjedtté melyeknek hátránya az, hogy csak egy felületet képesek kiemelni a térfogati függvényből, ezáltal jóval kevesebb információt mutatnak meg, mint amennyi rendelkezésre áll, valamint az sem garantált, hogy a térfogati függvény rendelkezik jellegzetes felülettel. Munkánk során azt vizsgáltuk, hogy milyen eredmény érhető el a megjelenítés tekintetében, ha a kevésbé ismert térfogati megjelenítést alkalmazzuk a felületalapú megoldás helyett, eközben nagy figyelmet fordítottunk a térfogati megjelenítő algoritmusban felhasznált lokális megvilágítási modell megoldásának hiba analízisére.

A hiba analízis során bizonyítást nyert, hogy az általunk használt összetett Simpson-formula alkalmazása kedvezőbb a hiba szempontjából, mint az irodalomban használt megoldás és az algoritmus sebességére nincs negatív hatással. Globális megvilágítási modellek esetén a statikus Ambient Occlusion algoritmusban alkalmazásra került az optimális átlagok elmélete, dinamikus Ambient Occlusion esetén új megközelítés kerül bemutatásra a lokális hisztogramok előállításához, mely kedvezőbb az irodalomban használt megoldásnál, mivel levezetése az elméleti modellből történt.

Munkánk során elkészült egy Java alkalmazás mely bemutatja az elért eredményeket. Mivel az algoritmusok CPU-n történő számítása relatíve lassúnak bizonyult egy GPU-t használó CUDA alkalmazás is fejlesztésre került, valamint shader-ek a Voreen nyílt forráskódú vizualizációs engine-hez, és Matlab program, melynek segítségével háromdimenziós adathalmazt állíthatunk elő a Mandelbulb fraktálról.

2. fejezet

Kaotikus dinamikus rendszerek

Kaotikus dinamikus rendszerek alatt olyan determinisztikus rendszereket értünk, melyek látszólag véletlenszerűen működnek. A természetben sok ilyen jelenséggel találkozhatunk. Például a csapból kifolyó víz esetén a nehézségi erő miatt folyamatosan vékonyodó vízsugarat kellene kapnunk, de a turbulens jelenségek fellépése miatt a rendszer viselkedése véletlenszerűvé válik. Ebben a fejezetben a kaotikus dinamikus rendszerekkel kapcsolatos fogalmak kerülnek bevezetésre [1] alapján, valamint a magasabb dimenziós rendszerekről is olvashatunk.

2.1. Hausdorff-dimenzió

Az irodalom a fraktál fogalmat többféleképpen definiálja, nincs egységesen elfogadott álláspont. Az egyik gyakran használt definíció bevezetéséhez szükséges ismernünk a Hausdorff dimenzió fogalmát. A topológiai dimenzió definíciója a következő:

Definíció 2.1 *A pont 0 dimenziós. Egy alakzat n dimenziós ha két részre lehet vágni egy $n - 1$ dimenziós alakzattal, de kevesebb dimenzióssal nem.*

Ezen definíció alapján a dimenziószám csak természetes szám lehet. A dimenzió-fogalom általánosítható oly módon, hogy a dimenziószám nem csak természetes számot vehet fel értékül. Ezt az általánosított dimenzió fogalmat nevezzük Hausdorff-dimenziónak, mely az önhasonlóság fogalmán alapul.

Vegyünk egy egydimenziós szakaszt és az $r = 1/n$ hasonlósági transzformációval kicsinyítsük le. Az így kapott szakaszból $N = n$ darab összeragasztásával megkapjuk az eredeti szakaszt. Hasonló kísérletet elvégezhetünk egy kétdimenziós téglalappal is. Az $r = 1/n$ hasonlósági transzformációval kapott kis téglalapból $N = n^2$ szükséges az eredeti téglalap lefedéséhez. Egy téglatest $r = 1/n$ hasonlósági transzformációval kicsinyített változatából $N = n^3$ szükséges az eredeti téglatest felépítéséhez. Ebből az következik, hogy D -dimenzióban az r kicsinyítési arány és a kicsinyített változathoz az eredeti lefedéséhez szükséges darabszám között a

$$N = \frac{1}{r^D} \quad (2.1)$$

összefüggés állítható fel. Ezen összefüggés alapján definiálhatjuk egy pontthalmaz Hausdorff-dimenzióját:

Definíció 2.2 Egy ponthalmaz Hausdorff-dimenzióján a

$$D = \frac{\log N}{\log 1/r} \quad (2.2)$$

menyiséget értjük.

A fraktál fogalmat Mandelbrot a következőképp definiálta:

Definíció 2.3 A fraktálok olyan halmazok, melyeknek Hausdorff-dimenziója nagyobb mint a topológiai dimenziója.

A későbbiekben minden önhasnó és kvázi önhasnó halmazt is a fraktálok közé sorolt. Az önhasnótság értelmezhető térben és időben, de a fogalom még tágabb értelmezést nyer a "hasnó" szó különbözı definiálása esetén [2].

2.2. Kaotikus dinamikus rendszerek a síkon

Számítógép segítségével lehetıségünk van kaotikus dinamikus rendszerek szimulációjára illetve megjelenítésére. Síkon ábrázolható kaotikus dinamikus rendszerek képként jeleníthetünk meg. A sík egy pontját algebrailag megadhatjuk egy x, y valós számpárral vagy egy z komplex számmal. A rendszer változása során a sík pontjai pályákat járnak be melyeknek megjelenítésével képeket kaphatunk.

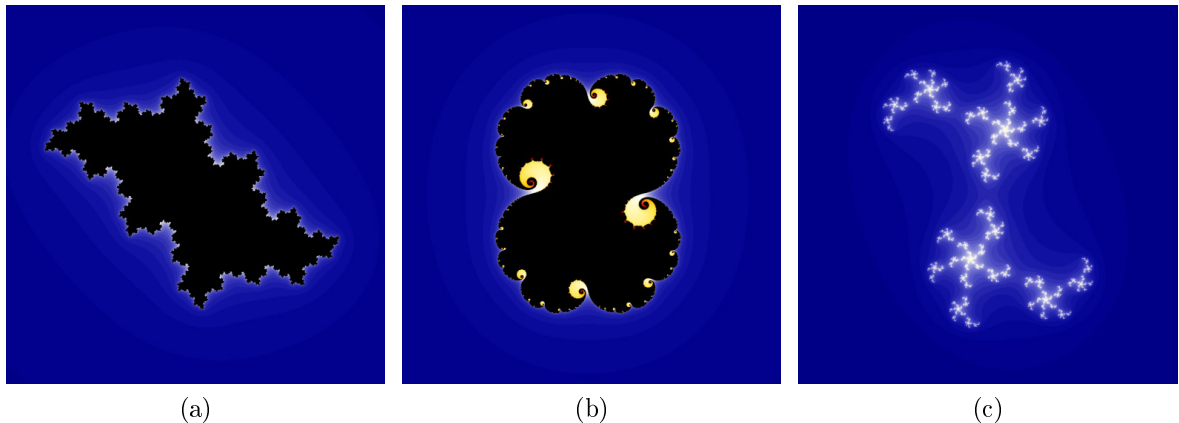
2.2.1. Júlia-halmazok

Definíció 2.4 A Júlia-halmazok azon pontok halmazai a komplex számsíkon melyekre a

$$z_{n+1} = z_n^2 + c \quad (2.3)$$

iteráció konvergens, ha n tart a végtelenbe, ahol c tetszıleges komplex szám.

Ezen halmazok megjelenítésére az irodalomban több algoritmust is találunk, ezek közül az egyik legnépszerűbb az ún. Escape Time algoritmus [3]. Az algoritmus a konvergenciát csak közelítıleg vizsgálja.



2.1. ábra: Komplex Júlia-halmazok különbözı c értékekkel. Az (a) ábrán $c = -0.258 + 0.6i$, a (b) ábrán $c = 0.258 + 0.002i$, a (c) ábrán $c = 0.47 + 0.4i$.

A fenti ábrákon jól látható, hogy különbözı c értékek esetén a halmazok nagymértékben eltérnek egymástól.

2.2.2. Mandelbrot-halmaz

A különböző c komplex számokkal előállított Júlia halmazokat vizsgálva megállapítható, hogy azok vagy összefüggő halmazok, vagy egymással nem érintkező pontok halmazai. Mandelbrot arra keresett választ, hogy c -nek milyen tulajdonságúnak kell lennie ahhoz, hogy a Júlia-halmaz összefüggő legyen. Az ilyen tulajdonságú komplex számok halmazát Mandelbrot-halmaznak nevezzük.

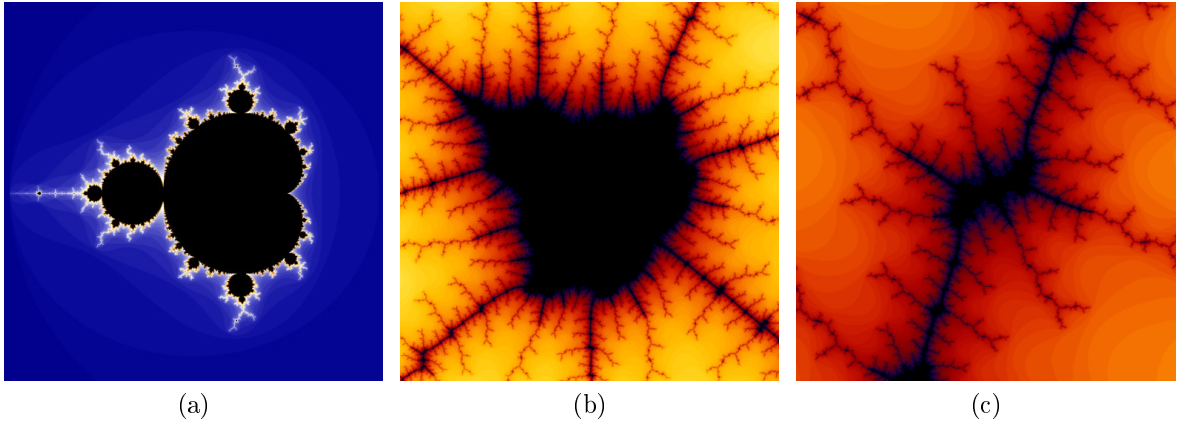
Definíció 2.5 *A Mandelbrot-halmaz azon pontok halmaza a komplex számsíkon melyekre a*

$$z_1 = c \quad (2.4)$$

$$z_{n+1} = z_n^2 + c \quad (2.5)$$

iteráció konvergens, ha n tart a végtelenbe.

A formula hasonlít a Júlia-halmazoknál használt leírásra, a különbség az, hogy a c komplex szám itt már nem tetszőleges, hanem függ a helykoordinátáktól. Az Escape-Time algoritmus jól használható a halmaz megjelenítésére.



2.2. ábra: Mandelbrot-halmazról készült képek. Az (a) ábrán 1.3 szoros, a (b) ábrán 7300-szoros, a (c) ábrán 100000-szeres nagyításban láthatók a halmaz egyes részei.

2.3. Kaotikus dinamikus rendszerek magasabb dimenziókban

Az előző alfejezetben bemutatott kaotikus dinamikus rendszerek a síknál magasabb dimenziókban is léteznek. Ilyen rendszereket úgy is létrehozhatunk, hogy az előbbiekben bemutatott halmazokat nem a komplex számok halmazán, hanem kvaterniókon vagy Cayley-féle számokon értelmezzük. Az Escape Time algoritmus segítségével kétdimenziós metszeteket számolhatunk a magasabb dimenziós rendszerekből, melyekből háromdimenziós adathalmazt állíthatunk elő. Ezen adathalmazok megjelenítésére többféle eljárás található az irodalomban, melyek közül a felület alapú módszerek váltak elterjedtté [4]. A magasabb dimenziós rendszerek megjelenítése jóval összetettebb mint a komplex számsíkon megadott társaiké, emiatt a vizualizációs algoritmusok hatékonysága sokkal nagyobb jelentőséggel bír.

3. fejezet

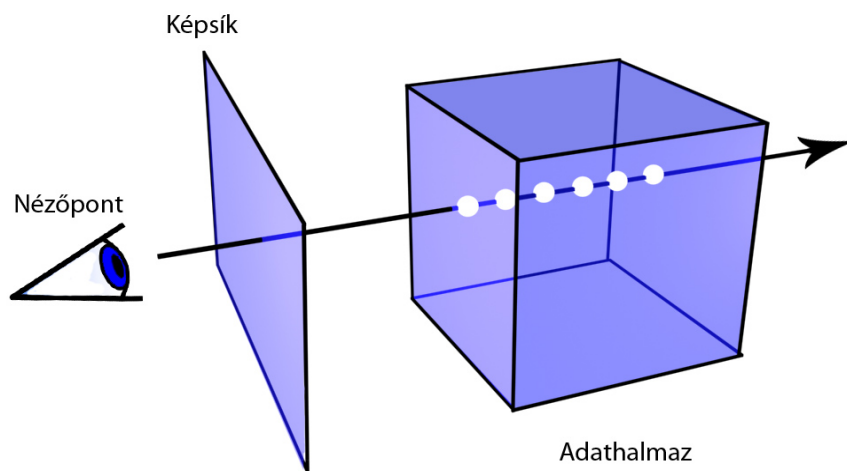
Megjelenítés

A megjelenítéshez felhasználható módszerek két nagyobb csoportra oszthatók. Léteznek felület-alapú és térfogat-alapú technikák. Felület-alapú megjelenítés esetén az adatokat felületekkel adjuk meg, melyek reprezentálhatók implicit függvényekkel, parametrikus görbékkel illetve poligonokkal. Térfogat-alapú módszereknél az adatokat térfogati elemek un. voxelek reprezentálják. A voxel szó az angol volume element kifejezésből származik, jelentése elemi térrész. A felület alapú megjelenítési módszerek csak felületeket modelleznek, nem veszik figyelembe a felület által közrezárt térrész anyagtulajdonságait, vagy homogénnek tekintik azokat ellentétben a térfogatot szemléltető megoldásokkal. A térfogati megjelenítési módszerek segítségével emiatt sokkal több információt mutathatunk meg, hátrányuk viszont az, hogy számításigényesek. A térfogati vizualizációs technikákat alkalmazzák CT és MRI vizualizációnál, geológiai és régészeti adathalmazok megjelenítésénél, kozmológiai és áramlástani szimulációkban és több más területen is. A magasabb dimenziós fraktálokról előállított adathalmazok térfogati függvények, ezekből lehetőségünk van felületeket elkülöníteni és megjeleníteni, de ez információvesztést okoz, mivel egy térfogati függvény áll a rendelkezésünkre. A térfogati vizualizációs módszerek képesek teljes egészében megjeleníteni térfogati függvényeket, emiatt esett a választás a rájuk. Ezen technikákat három nagyobb csoportra oszthatjuk. Léteznek a problémát a grafikai objektumok szemszögéből megközelítő un. object order, a kép szemszögéből megközelítő un. image order és hibrid térfogatvizualizációs megoldások.

Az object order technikák az adatokból előállított térfogati modellt rávetítik a képsíkra, egyik ilyen népszerű eljárás a volume splatting [5]. Az image order megoldások a képsík pixeleinek szemszögéből közelítik meg a vizualizációs problémát. Minden pixelen át sugarakat indítanak és vizsgálják, hogy a sugár milyen grafikai objektumokkal ütközik, és az így kinyert adatok segítségével állítják elő a végleges képet. Az image order eljárásokat szokás sugárkövetés alapú eljárásoknak is nevezni. A hibrid technikák az image order és object order eljárások keverékei. Mindhárom csoportnak megvannak a maga előnyei és hátrányai. Az object order technikák segítségével gyorsabban ki tudjuk számolni a végleges képet, viszont a kapott kép minősége alulmarad a sugárkövetés alapú eljárásokkal szemben. A sugárkövetés alapú eljárások számításigényesebbek, viszont fotorealistikus képet adnak, és jól paraméterezhetők (pixelenkénti sugarak száma, sugárankénti minták száma, stb.), a hibrid eljárások pedig az előbb említett két módszer jó tulajdonságait próbálják ötvözni [6]. A következő alfejezetekben a sugárkövetés alapú eljárások kerülnek részletes bemutatásra és elemzésre.

3.1. Sugárkövetés alapú térfogatvizualizáció

A sugárkövetés alapú térfogat vizualizáció az elmúlt néhány évben vált elterjedtté a CT és MRI képsorozatok megjelenítésénél a GPU-k teljesítményének növekedése következtében. Az eljárás jól párhuzamosítható GPU-n, nagy számításigénye miatt CPU-n történő implementáció esetén lassú és nem alkalmazható hatékonyan valós idejű vizualizációra. Az előnye az, hogy nagyon jól paraméterezhető és fotorealistikus képet ad. A módszer lényege, hogy a nézőpontból a képsík minden pixelén át sugarakat indítunk, és ezen sugarak mentén diszkrét pontokban mintákat veszünk a térfogati modellből, és a vett minták segítségével becsülhetjük az adott pixel színét. Ezt szemlélteti a következő ábra.



3.1. ábra: A sugárkövetés alapgondolata

Az előző fejezetben említésre került, hogy az Escape Time algoritmus segítségével kétdimenziós metszeteket tudunk előállítani magasabb dimenziós fraktálokról és ezen metszetek segítségével háromdimenziós adathalmaz hozható létre. Ezen adathalmazt úgy hozhatjuk létre, hogy a metszeteket adott tengely mentén sorba rendezzük, és betöltjük egy háromdimenziós adatstruktúrába. Így egy háromdimenziós hálót kapunk melynek rácspontjaiban találhatók az intenzitás értékek. Ez az adatstruktúra skalármezőként is értelmezhető. A kétdimenziós képeket érdemes szürkeárnyaltos képként menteni, mivel a vizualizációnál a képsorozat tárolása kevesebb memóriát igényel mint ha RGB formátumban mentettük volna el. Térfogatvizualizációnál a memóriával történő gazdálkodás kulcskérdés, mivel az adathalmazok rendszerint nagy méretűek.

A sugárkövetés alapú térfogatvizualizáció menete részleteiben a következő: Mintavételi pontok kiválasztása a sugár mentén, mintavételezés interpolációval. (Az interpolációra amiatt van szükség mivel a mintavételi pontok sok esetben nem a rácspontokban találhatók.) Ezután ha szükséges akkor gradienst becsülünk. A skalármező adott pontokban vett gradienseire egyes árnyalási modelleknél lehet szükség, erről a későbbiekben még lesz szó. Utána következik az osztályozás (classification). Az osztályozás során a vett mintákhoz szín és átlátszatlanság értéket társítunk. Ezután következhet az árnyalás, majd legvégül az összeállítás (compositing). Az összeállítás során számítjuk ki az előállítandó kép pixeleinek színét (a pixelek színét a mintákhoz rendelt szín és átlátszatlanság értékekből állítjuk össze).

3.2. A térfogatvizualizációs integrál

Ahhoz, hogy térfogatvizualizáció segítségével képeket készítsünk, szükségszerű matematikai modelleket definiálnunk melyek az adathalmazban található intenzitás értékekhez optikai tulajdonságokat rendelnek. A modellt annak függvényében kell megalkotnunk, hogy mennyire szeretnénk a valóságot pontosan modellezni, mennyire fotorealisztikus képet szeretnénk kapni végeredményként. Mérlegelnünk kell azt is, hogy egy összetettebb, valósághoz közelebb álló modell alkalmazása számítás és memória-igényes lehet. Ebben az alfejezetben az egyik legelterjedtebb modell kerül bemutatásra melynek alapötletét Blinn [8] dolgozta ki.

A térfogatvizualizációs modellek legtöbb esetben a geometriai optikai modellek, ahol a fény egyenes vonalak mentén terjed, közben kölcsönhatásba lép a grafikai objektumokkal. Általában a következő kölcsönhatásokat veszik számításba: fénykibocsátás, fényelnyelés, fénytörés. Ezen kölcsönhatások befolyásolják a fény energiáját. A fénykibocsátás növeli, az elnyelés csökkenti, a fénytörés pedig növeli és csökkenti is a fény energiáját az egyenes mentén. A fény energiája a radianciával jellemezhető, ami az A egységnyi felületre egységnyi idő alatt jutó Q kisugárzott energia az Ω térszög függvényében.

$$I = \frac{dQ}{dA_{\perp} d\Omega dt} \quad (3.1)$$

A $\perp$ jelölés azt jelzi, hogy a felületet mint vetületet kell mérnünk az egyenes mentén $A_{\perp} = A \cos \theta$, ha θ a szintfelület normálvektora és a fényforrás irányvektora által közbezárt szög. A radianciát intenzitás néven is említi az irodalom. Blinn modellje az előbb említett kölcsönhatások közül csak a fénykibocsátást és fényelnyelést veszi figyelembe. A fénykibocsátást a voxel színe, míg az elnyelést a voxel átlátszósága határozza meg.

3.2.1. Fényelnyelés

Abban az esetben, ha a voxelek csak fényelnyelő tulajdonsággal rendelkeznek, fénykibocsátással nem, a rendszert a következő differenciálegyenlet írja le:

$$\frac{dI}{ds} = -\tau(s)I(s) \quad (3.2)$$

Ahol s egy előre definiált ponttól vett távolság a sugár (félegyenes) mentén, $I(s)$ a fény intenzitása az adott pontban, $\tau(s)$ az adott pontban vett abszorpció együttható, $s \in [0, \varphi]$, $\varphi \geq 0$ és $I(0) = I_0$. Az egyenlet megoldása:

$$I(s) = I_0 e^{-\int_0^s \tau(\bar{s}) d\bar{s}} \quad (3.3)$$

Ahol I_0 a fény intenzitás értéke a grafikai objektumba történő belépéskor.

3.2.2. Fénykibocsátás

Csak a fénykibocsátást figyelembe véve a következő differenciálegyenlethez jutunk:

$$\frac{dI}{ds} = q(s) \quad (3.4)$$

Ahol $q(s)$ a kibocsátott fény mértéke, $s \in [0, \varphi]$, $\varphi \geq 0$ és $I(0) = I_0$. Az egyenlet megoldása a következő:

$$I(s) = I_0 + \int_0^s q(\tilde{h}) d\tilde{h} \quad (3.5)$$

3.2.3. Fénykibocsátás és fényelnyelés

A valóságot jobban közelítő modellhez jutunk, ha a fénykibocsátást és elnyelést egyidejűleg vesszük figyelembe, a fénytörést elhanyagoljuk. A voxelek elnyelik a rajtuk keresztülhatoló fény egy részét, eközben fényt bocsátanak ki.

$$\frac{dI}{ds} = q(s)\tau(s) - I(s)\tau(s) \quad (3.6)$$

A megoldás a következő [9][7], ha $s \in [0, \varphi]$, $\varphi \geq 0$ és $I(0) = I_0$:

$$I(s) = I_0 e^{-\int_0^s \tau(\tilde{s}) d\tilde{s}} + \int_0^s q(\tilde{h}) \tau(\tilde{h}) e^{-\int_h^s \tau(\tilde{s}) d\tilde{s}} d\tilde{h} \quad (3.7)$$

Ezt a formulát nevezzük térfogatvizualizációs integrálnak. Ez az egyik legelterjedtebb modell. A formula a következőképp írható fel diszkrét alakban [7]:

$$I = \sum_{i=0}^n q_i \prod_{j=i+1}^n (1 - \alpha_j) \quad (3.8)$$

Ahol $q_0 = I_0$ és α_j a mintákhoz tartozó átlátszatlanság. A gyakorlatban ezt a formulát alkalmazzák a legtöbb esetben összeállításra (compositing), mivel az irodalomban elterjedté vált.

3.2.4. Árnyékolás

Az előbbieken bemutatott modellt kiegészíthetjük árnyékolással. Ez sokkal számításigényesebbé teszi az algoritmust, de az előnye, hogy realiztikusabbá teszi a képet és sokkal több vizuális információt ad. Árnyékolást úgy számolhatunk ki, hogy minden mintavételi pontból a fényforrás irányába sugarakat indítunk, és becsüljük, hogy a fényforrásból érkező fényből mennyi nyelődik el amíg a mintavételi ponthoz ér.

$$\Psi(s) = e^{-\int_s^L \tau(\tilde{s}) d\tilde{s}} \quad (3.9)$$

Ahol L a fényforrás pozíciója. Ezzel a formulával kiegészíthetjük a térfogatvizualizációs integrált. Az árnyékolás szebbé tehető, ha az adott mintavételi pontból a fényforrás felé több sugarat is indítunk (ügyelnünk kell arra, hogy a sugarak ne legyenek párhuzamosak egymással), és ezen sugarak esetén kapott fényelnyelési értékeket átlagoljuk. Ekkor az árnyékok lágyabbá válnak, ez a módszer jobban közelíti a valóságot, de a számításigénye magasabb mint abban az esetben, amikor csak egy sugart indítunk a fényforrás felé mintavételi pontonként. Árnyékok megvalósítására többféle technika is létezik, ezek közül néhány megoldásról a 3.6. fejezetben olvasható részletes ismertetés. Árnyékolással készült képek a függelékben találhatók.

3.3. Hiba analízis

A térfogatvizualizációs integrál kiszámítására az irodalomban a 3.8-as formula vált elterjedtté. Ezen formulánál lehetőségünk van kisebb hibával rendelkező becslést is adni az elméleti modell kiszámítására. A trapéz-formula és a Simpson-formula a Newton-Cotes formulák közé tartoznak melyek hibájának kiszámítására létezik zárt formula ellenben az irodalomban használt megoldással [10].

3.3.1. Trapéz-formula

Legyen $x_1 = a$ és $x_2 = b$! A két pontra támaszkodó elsőfokú Lagrange-interpolációs polinom:

$$p(x) = f(x_1) \frac{x - x_2}{x_1 - x_2} + f(x_2) \frac{x - x_1}{x_2 - x_1} \quad (3.10)$$

melynek határozott integrálja:

$$\begin{aligned} \int_{x_1}^{x_2} p(x) dx &= \left[f(x_1) \frac{(x - x_2)^2}{2(x_1 - x_2)} + f(x_2) \frac{(x - x_1)^2}{2(x_2 - x_1)} \right]_{x_1}^{x_2} \\ &= \frac{x_2 - x_1}{2} [f(x_1) + f(x_2)] \end{aligned} \quad (3.11)$$

Ha $f(x_1)$ és $f(x_2)$ előjele azonos, akkor az eredmény az $(x_1, 0)$, $(x_2, 0)$, $(x_2, f(x_2))$, $(x_1, f(x_1))$ pontok által határolt trapéz területe. Ha az $[a, b]$ intervallumot felosztjuk az

$$a = x_1 < x_2 < \dots < x_{n+1} = b \quad (3.12)$$

ekvidisztáns pontokkal n részintervallumra, akkor megkapjuk az összetett trapézformulát, mely a következőképp írható fel:

$$\int_a^b f(x) dx \approx T_n(f) = \frac{h}{2} \left[f(x_1) + 2 \sum_{i=2}^n f(x_i) + f(x_{n+1}) \right] \quad (3.13)$$

Ahol $h = \frac{b-a}{n}$. Az összetett trapézformula hibájára fennáll, hogy

$$\left| \int_a^b f(x) dx - T_n(f) \right| \leq \frac{M_2(b-a)h^2}{12} = \frac{M_2(b-a)^3}{12n^2} \quad (3.14)$$

ahol M_2 az $f(x)$ függvény második deriváltjának maximum értéke az $[a, b]$ intervallumon, feltéve, hogy $f(x)$ legalább kétszer folytonosan differenciálható.

3.3.2. Simpson-formula

Legyen $x_1 = a$, $x_2 = \frac{a+b}{2}$ és $x_3 = b$! Tekintsük a három pontra támaszkodó másodfokú Lagrange interpolációs polinomot:

$$\begin{aligned} p(x) &= f(x_1) \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_2 - x_3)} + f(x_2) \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + \\ &+ f(x_3) \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \end{aligned} \quad (3.15)$$

Ennek az $[a, b]$ intervallumon vett integrálja a következő formulát adja:

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \left[f(a) + 4f\frac{a+b}{2} + f(b) \right] \quad (3.16)$$

Az $[a, b]$ intervallum itt is felbontható az

$$a = x_1 < x_2 < \dots < x_{n+1} = b \quad (3.17)$$

ekvidisztáns pontokkal n részintervallumra, ekkor megkapjuk az összetett Simpson-formulát, mely a következő alakban írható fel:

$$\int_a^b f(x)dx \approx S_n(f) = \frac{h}{6} \left[f(x_1) + 2 \sum_{i=2}^n f(x_i) + 4 \sum_{i=1}^n f(x_i + \frac{h}{2}) + f(x_{n+1}) \right] \quad (3.18)$$

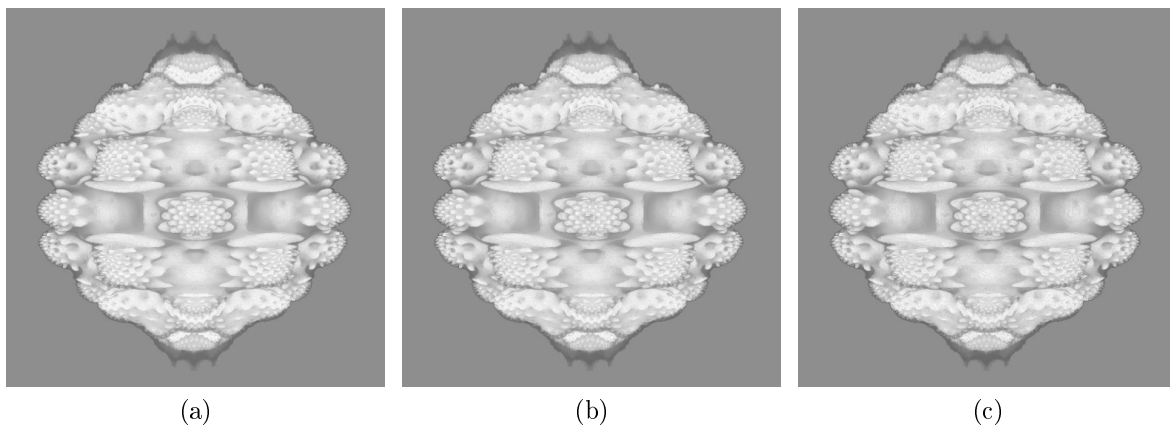
ahol $h = \frac{b-a}{n}$. Az összetett Simpson-formula hibájára fennáll, hogy

$$\left| \int_a^b f(x)dx - S_n(f) \right| \leq \frac{M_4(b-a)h^4}{2880} = \frac{M_4(b-a)^5}{2880n^4} \quad (3.19)$$

ahol M_4 az $f(x)$ függvény negyedik deriváltjának maximum értéke az $[a, b]$ intervallumon, , feltéve, hogy $f(x)$ legalább négyszer folytonosan differenciálható.

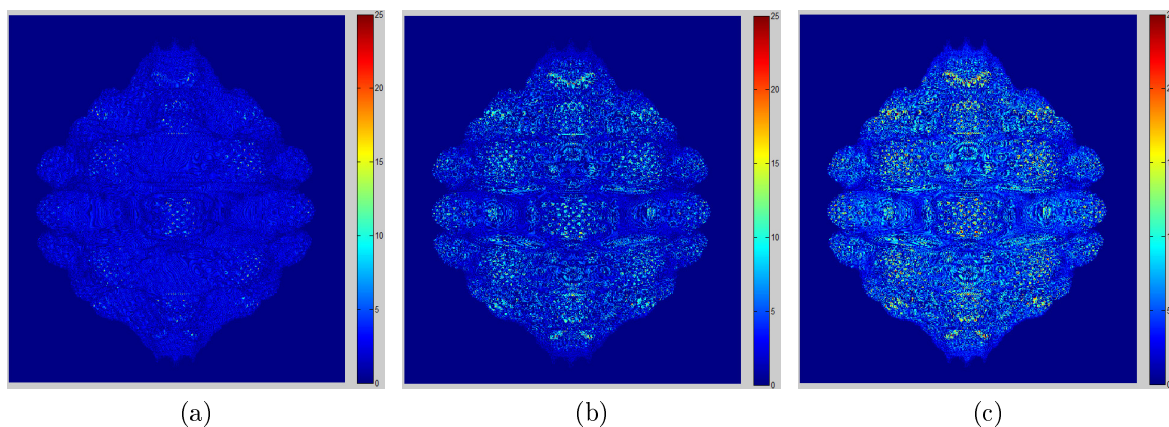
3.3.3. Megoldások összevetése

Most az előbbieken említett három numerikus megoldást vetjük össze a vizualizáció végeredményének szempontjából.



3.2. ábra: Térfogavizualizációs integrál kiszámítása különböző megoldásokkal. A képek a Mandelbrot-halmaz háromdimenziós változatáról, a nyolcadrendű Mandelbulb halmazról készültek. Az (a) ábrán az irodalomban használt eljárás, a (b) ábrán az összetett trapéz-formula által előállított kép, a (c) ábrán az összetett Simpson-formula segítségével létrehozott kép látható.

A három kép között szabad szemmel alig észrevehető a különbség, de a következő Matlab segítségével előállított ábrán jól láthatóak a képek közötti eltérések.



3.3. ábra: Az ábrán a 3.2-es ábra képeinek pixelenkénti intenzitáskülönbségei láthatóak. Az (a) ábrán a 3.2-es ábra (a) és (b), a (b) ábrán a (b) és (c), a (c) ábrán az (a) és (c) képekből számoltunk pixelenkénti intenzitáskülönbséget. A skála maximum 25 intenzitásértéknyi különbséget jelenít meg. Ha a két kép azonos helyen lévő pixelei között 0 az intenzitáskülönbség akkor azt kék szín jelzi, ha 25, akkor piros, a két érték között pedig átmenetes.

A 3.3-as ábra jól mutatja az eljárások által alkotott képek közötti különbséget. Az (a) képen látható, hogy az irodalmi megoldás és a trapéz-formula által alkotott kép között nincs nagy eltérés, viszont a (c) képen az irodalmi megoldás és a Simpson-formula eredménye között már jóval szembevetőbb a különbség. A megoldások shader-ként kerültek implementálásra a Voreen nyílt forráskódú háromdimenziós engine-be. A technikák hibája mellett a gyakorlati alkalmazás esetén a sebességük is fontos tényező, emiatt méréseket végeztem a különböző megoldások közötti sebesség eltérések megállapítására. Az irodalmi megoldás a 3.8-as formulára utal.

Mintavételi arány	Irodalmi megoldás	Trapéz-formula	Simpson-formula
0.3	14.9 fps	13.8 fps	14.4 fps
0.6	8.6 fps	7.9 fps	8.2 fps
0.9	6.1 fps	5.6 fps	5.8 fps
1.2	4.7 fps	4.3 fps	4.4 fps
1.5	3.8 fps	3.5 fps	3.6 fps

3.4. ábra: A táblázatban az egyes módszerek másodpercenkénti képfrissítéseinek száma (fps) láthatóak a mintavételi arány függvényében. A mintavételi arány növelésével a rendszer növeli az egységnyi szakaszhoz jutó minták számát, csökkentve ezáltal a mintavételi távolságot. A mérést egy GeForce 9600M GT GPU-val, 4GB memóriával és Intel Core2Duo P8400-as CPU-val rendelkező személyi számítógépen végeztem 512³ méretű adathalmazzal. Az adathalmazról 100 darab 512² felbontású képet számoltam ki ezalatt az adathalmazon 360 fokos forgatást végeztem. Ezután a 100-at elosztottam az eltelt idővel, megkapva ezzel a másodpercenkénti képfrissítések számát.

A táblázatban látható, hogy az összetett trapéz és összetett Simpson-formulás megoldásnál magasabb az fps érték, emiatt a megoldások jobbak, de adott mintavételi arány mellett a három megoldás között nincs gyakorlati szempontból számottevő eltérés, mivel a szemléltető a valósidejű renderelést figyelve ránézésre nem tudja megállapítani ezen

kicsi eltéréseket, valamint a mért eredmények különböző hardverek esetén eltérőek lesznek. A trapéz vagy Simpson-formulákat alkalmazva a megoldás hibája kisebb lesz az irodalmi megoldásától, és használatuk a sebességre sincs negatív hatással.

3.4. Megvilágítási modellek

Az árnyalás az a folyamat, melynek során egy fénymodell segítségével becsülhetjük a pixelek színét figyelembe véve a fényforrások valamint az anyagok tulajdonságait. A hangsúly a fényforrások figyelembevételén van, mivel a térfogatvizualizációs integrálban nem szerepelt a grafikai objektum és a fényforrások kapcsolatát leíró formula. Az árnyalást leíró matematikai modelleket nevezzük megvilágítási modelleknek, az irodalom hivatkozik rájuk árnyalási modellek néven is. A megvilágítási modellek két nagyobb kategóriára oszthatók: lokális és globális megvilágítási modellekre. A lokális megvilágítási modellek esetében egyetlen kölcsönhatás történik a fény és a grafikai objektum között amíg a fény eljut a fényforrásból a képsíkra, a globális megvilágítási modellek esetén pedig több kölcsönhatás is történhet. A globális modellek számításigényesebbek, viszont vannak olyan folyamatok (pl. anyag belsejében történő fénytörés), melyek modellezésére a lokális megvilágítási modellek nem alkalmasak. Erről részletesen a későbbiekben esik majd szó. A lokális megvilágítási modellek közül az egyik legnépszerűbb a Phong-féle model, mely a következő formában írható fel:

$$I_{phong} = I_{ambient} + I_{diffuse} + I_{specular} \quad (3.20)$$

Az ambient (környezeti fény) konstans, mely az indirekt fény modellezésére hivatott, a diffuse komponens szórt fényt modellez és függ a fényforrások helyétől, a specular komponens pedig a grafikai objektum fényességét modellezi és függ a nézőponttól, valamint a fényforrások pozíciójától. Felületek esetén a diffuse és specular komponensek kiszámításához felületi normálist használnak, térfogati modellek esetén a gradiens használható erre a célra, mivel skalármező adott pontjában vett gradiense merőleges a szintfelületre. $f(x, y, z)$ skalármező gradiensvektora a következő formában írható fel:

$$\nabla(f(x, y, z)) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix} \quad (3.21)$$

A gradiens becslésére véges differencia alapú módszerek alkalmazhatók. A becslésnél kapott eredményt normalizálni kell, hogy a modellben alkalmazható legyen. Ha az adott pontban vett gradiens nullvektor akkor a skalármező az adott pontban homogén, vagy szélsőértéke van. A gradiens vektorokat kiszámításuk után tárolhatjuk, és összeállításnál felhasználhatjuk. Másik lehetőség, hogy az összeállításnál valós időben számítjuk ki őket. A valós időben történő számítás kedvezőbb a memóriafelhasználás szempontjából, mivel nem kell az összes gradienst tárolni, viszont lassítja a megjelenítést. Mivel a memória térfogatvizualizációnál nagyon költséges erőforrásnak minősül, emiatt legtöbb esetben a valós idejű számítás kerül alkalmazásra. Fraktálok vizualizációja esetén akkor hasznos a Phong-model alkalmazása ha a térfogati függvénynek van jellegzetes részhalma, mivel ellenkező esetben a kép minőségén szinte alig vehető észre javulás.

3.5. Osztályozás

A osztályozás egy nagy jelentőséggel bíró folyamat a megjelenítés során, mivel ennek folytán dől el, hogy az egyes voxelek láthatóak lesznek-e a végleges képen, és ha láthatóak lesznek, akkor milyen színnel jelenjenek meg. A voxelek intenzitásértékeihez az ún. átviteli függvény rendeli hozzá a színt és az átlátszatlanságot. A osztályozás célja, hogy az adathalmazból kiemeljük a számunkra fontos információval rendelkező részhalmazokat és ezen részhalmazokhoz olyan színeket és átlátszatlanság értékeket rendeljük, melyek a későbbi megjelenítés során a lehető legtöbb vizuális információt nyújtják számunkra. Annak ellenére, hogy a osztályozás fontos szerepet játszik a vizualizációban nincs jól használható módszer az automatizálására, a felhasználónak kell beállítania az átviteli függvényt. A osztályozás folyamatot végrehajthatjuk az interpolációval történő mintavételezés előtt (pre osztályozás) vagy pedig utána (poszt osztályozás). A pre osztályozás hamis színeket és rossz minőségű képet eredményez, emiatt a gyakorlati implementációk esetén a poszt osztályozást használják, melynek nagyobb a számításigénye.

3.5.1. Átviteli függvények

Az átviteli függvény feladata, hogy az adathalmazban található intenzitás értékekhez szín, és átlátszatlanság értékeket rendeljen. Az átviteli függvények tervezése nem egyszerű feladat, mivel az optikai modell mellett ez határozza meg az adathalmaznak a folyamat végén kiszámolt képen történő megjelenését, és mivel az automatizálására nem létezik egységes, jól használható megoldás (csak speciális feladatok esetén), emiatt az esetek nagyrésztében a felhasználónak kell megadnia. Az átviteli függvénynek el kell különítenie az adathalmazból azt a részhalmazt, amit a felhasználó látni szeretne, mindemellett egyszerűnek is kell lennie, hogy a felhasználó könnyen megértse kezelését és egyszerűen változtathasson rajta.

A legegyszerűbb átviteli függvények az egydimenziós átviteli függvények. Az egydimenziós átviteli függvények csak a voxelek intenzitásértékét veszik figyelembe a szín és átlátszatlanság hozzárendelésnél, használatuk a felhasználó számára is könnyen megérthető. A hátránya, hogy az adathalmaz részhalmazainak elkülönítése csak a voxelek intenzitásértékét figyelembe véve sok esetben nehézkes, mivel lehetnek olyan a végső képen a felhasználó számára nem kívánatos részhalmazok, melyekben a voxelek ugyanazon intenzitásértékekkel rendelkeznek, mint a felhasználó által látni kívántak esetében.

Abban az esetben, ha az egydimenziós átviteli függvényeket alkalmazva az előbb említett hiba előjön, alkalmazhatunk többdimenziós átviteli függvényeket, melyekkel az egyes részhalmazok jobban elkülöníthetővé válnak, viszont ezen függvények a felhasználó számára már nehezebben kezelhetőek, átláthatóak. Magasabb dimenziós átviteli függvények esetében a voxelek intenzitásértéke mellett felhasználják a voxelhez tartozó gradiens nagyságát, melyből az élek hollétére lehet következtetni. Habár a többdimenziós átviteli függvények több paraméterrel rendelkeznek ezáltal jobban testreszabhatóak, jó beállításukhoz a felhasználónak ismernie kell az adathalmazt és némi felhasználói gyakorlattal kell rendelkeznie a többdimenziós átviteli függvények terén, hogy a kívánt végeredményt kapja a vizualizáció szempontjából.

3.6. Globális megvilágítási modellek

A 3.4-es fejezetben bemutatott Phong-féle megvilágítási modell és a térfogatvizualizációs integrál segítségével képesek vagyunk a voxelenkénti fénykibocsátás és elnyelés modellezésére a fényforrás figyelembevételével, de az anyagon belüli többszöri fénytörést nem tudjuk modellezni segítségükkel, ehhez globális megvilágítási modellekre van szükség. A anyagon belül történő többszöri fénytörés modellezése számításigényes, de sokkal szebbé, élethűbbé teszi a végleges képet. A nagy számításigénye miatt a többszöri fénytörés becslésére kisebb számításigényű módszerek alakultak ki, melyek fizikailag nem hűek ugyan, de ezt a szemlélő a végeredményt nézve nem tudja megállapítani. Ezen módszerek közé tartoznak az Ambient Occlusion technikák.

3.6.1. Ambient Occlusion

Az Ambient Occlusion azon globális megvilágítást szimuláló technikák gyűjtőneve melyek a grafikai alapelemekhez (polygon vertex, voxel) minden irányból beérkező fénysugarak segítségével becsülik a megvilágítást.

3.6.2. Lokális Ambient Occlusion

Hernell [11] kidolgozott egy megoldást, melynek során több irányba sugarakat indít az adott voxeltől és ezen sugarak segítségével állapít meg információt az adott voxel környezetéről. A sugarak végetérnek ha a sugarak mentén vett mintavételi pontok egy előre definiált távolságnál messzebbre kerülnek az adott térfogati elemtől, ez teszi a megoldást lokálissá. Minden voxel esetében a voxel környezetéből vett minták helykoordinátái egy adott sugarú $\Omega(x)$ gömb belsejében helyezkednek el melynek középpontja az adott voxel.

A k irányból az x pozíciójú voxelhez beérkező fény intenzitásértéke a következőképp írható fel:

$$I_k = \int_a^{R_{\Omega(x)}} \frac{w_k}{R_{\Omega(x)} - a} e^{-\int_a^s \tau(u) du} ds \quad (3.22)$$

Ahol a kezdeti eltolás a sugár mentén, R_{Ω} a gömb sugara, w_k súly, $\tau(u)$ az adott pontban vett abszorpciók együttható. A formula diszkrét alakban a következő:

$$I_k = \sum_{m=0}^M \frac{w_k}{M} \prod_{i=0}^{m-1} (1 - \alpha_i) \quad (3.23)$$

Ahol M a vett minták száma, α_i pedig az adott minta átlátszatlanság értéke. A környezet által az adott voxelhez közvetített intenzitás a következő alakban írható fe:

$$I = \frac{1}{K} \sum I_k, \quad (3.24)$$

ahol K az adott voxelből indított sugarak száma. A 3.22-es formula kibővíthető úgy, hogy a c_E fénykibocsátást is figyelembe vegye:

$$I_k = \int_a^{R_{\Omega(x)}} w_k \frac{1 + c_E(s)}{R_{\Omega(x)} - a} e^{-\int_a^s \tau(u) du} ds \quad (3.25)$$

Ennek a megoldásnak az a hátránya, hogy minden egyes képfrissítésnél az előbbiekben leírt számításokat el kell végezni az összes mintavételi pontban található voxel-re a összeállítás során, emiatt nagy a módszer számításigénye.

3.6.3. Dinamikus Ambient Occlusion

Az előző alfejezetben leírt megoldással szemben a Ropinski által publikált eljárásnál [12] nem szükséges minden képfrissítésnél elvégezni az előbbiekben leírt számításokat, nem függ az átviteli függvényről, és ambient occlusion mellett képes color bleeding hatás (a grafikai objektumok színezésénél figyelembe vesszük a grafikai objektumok környezetéből érkező szórt fényt) modellezésére is. A renderelési idő az előbbiekben leírt módszerhez képest alacsony, mindemellett dinamikusan változtathatjuk az átviteli függvényt és kamera paramétereket. A módszer lokális hisztogramok segítségével becsüli az adott voxel környezetéből érkező fény intenzitását.

Lokális hisztogramok

Az x voxel környezetéből érkező fény intenzitásának becsléséhez kiszámoljuk az x voxelhez tartozó lokális hisztogramot, melyet jelöljön $LH(x)$. Minden $\tilde{x}$ voxel amely benne van egy olyan adott sugarú $\Omega(x)$ gömbben melynek középpontja az adott voxel, szerepelni fog a lokális hisztogramban. Legyen $f(x) \in [0, 2^b]$, ahol $b \in \{8, 12, 16, 32\}$ amely az adathalmaz bitmélysége. A módszer minden voxelhez a következőképp rendeli hozzá a hisztogramot:

$$LH(x) = (LH_0(x), \dots, LH_{n-1}(x)), \quad (3.26)$$

$$LH_k(x) = \sum_{\tilde{x} \in \Omega(x), \tilde{x} \neq x} f_{dist} \left(\frac{|x - \tilde{x}|}{d_{min}} \right) g(f(\tilde{x}), k), \quad (3.27)$$

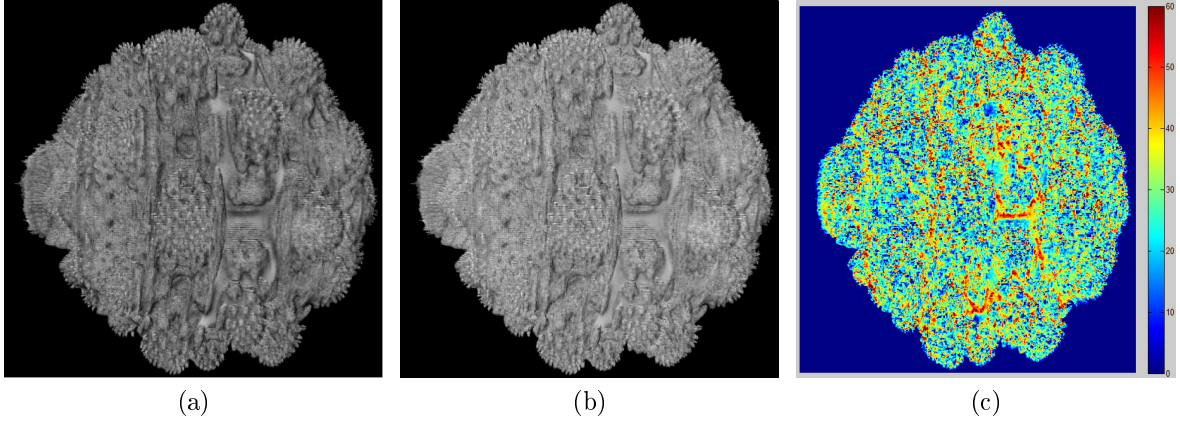
ahol d_{min} az adathalmaz bármely két voxele közötti távolságok közül a legkisebb, $f_{dist}(d) = \frac{1}{d^2}$, $g(i, k)$ alatt pedig a következő függvényt értjük:

$$g(i, k) = \begin{cases} 1 & \text{ha } i = k \\ 0 & \text{ha } i \neq k \end{cases} \quad (3.28)$$

Ezzel a módszerrel nem tudjuk becsülni a voxelek fényelnyelését, mivel a $\Omega(x)$ gömbön belül egyenletes eloszlás szerint választjuk ki a hisztogramhoz a voxeleket, következésképpen ezek nem egyenesek mentén helyezkednek el, emiatt került be a Ropinski által javasolt formulába a távolság-alapú súlyozás. A távolság alapú súlyozásnál kedvezőbb becslés adható a fényelnyelésre, ha az opacitást konstansnak tekintjük és a 3.3-as formulát alkalmazzuk a súlyozás helyett. Ekkor a következő formulát kapjuk, helyettesítve ezzel a 3.27-es egyenletet:

$$LH_k(x) = \sum_{\tilde{x} \in \Omega(x), \tilde{x} \neq x} e^{-\tau \frac{|x - \tilde{x}|}{d_{min}}} g(f(\tilde{x}), k) \quad (3.29)$$

A következő ábrán a két formula által alkotott képeket és a közöttük levő eltérést láthatjuk.



3.5. ábra: Az (a) képen a 3.27-as, a (b) képen a 3.29 as formula segítségével előállított képek láthatók a Mandelbulb halmazról, melyek a saját fejlesztésű Java alkalmazás segítségével készültek. Az adathalmaz mérete 512^3 . A (c) képen a két kép pixelenkénti intenzitáskülönbségét láthatjuk, mely jól mutatja a képek közötti eltérés nagyságát.

3.6.4. Az optimális mérték

Legyen adott $(\Theta, \mathcal{F})$ mérhető tér.

Definíció 3.1 Egy függvényhalmazt $p : \mathcal{F} \rightarrow [0, 1]$ optimális mértéknek nevezünk, ha teljesíti a következő három axiómát:

Axióma 1 $p(\Theta) = 1$ és $p(\emptyset) = 0$

Axióma 2 Minden B és E mérhető halmazra fennáll, hogy $p(B \cup E) = p(B) \vee p(E)$.

Axióma 3 Ha $(E_n) \subset \mathcal{F}$ csökkenő sorozat, akkor fennáll, hogy

$$p\left(\bigcap_{n=1}^{\infty} E_n\right) = \lim_{n \rightarrow \infty} p(E_n) = \bigwedge_{n=1}^{\infty} p(E_n).$$

Definíció 3.2 Az

$$A_{\Theta}(s) := \bigvee_{i=1}^n b_i p(B_i), \quad s = \sum_{i=1}^n b_i \chi_{B_i} \quad (3.30)$$

mennyiséget s optimális átlagának nevezzük, ahol $B_1, \dots, B_n$ teljes eseményrendszert alkot.

A [13] cikkben bemutatásra került egy módszer, mely az előbb definiált optimális átlagot használja arra, hogy a minták közül kiválassza a "legjobbat". Minden mintához tartozik egy átlag. A mintákhoz az optimális átlagok is kiszámításra kerülnek. Az optimális átlagok előállítása addig történik, míg a megadott kilépési feltétel nem teljesül. A kilépési feltétel során azt vizsgáljuk, hogy az optimális átlag mennyiben tér el az átlagtól, ha az eltérés egy előre definiált epszilon érték alatt van (legkisebb négyzetek módszere), leállítjuk az algoritmust. A módszer felhasználásra kerül a 3.6.2-es alfejezetben leírt lokális Ambient Occlusion algoritmus Java implementációjában.

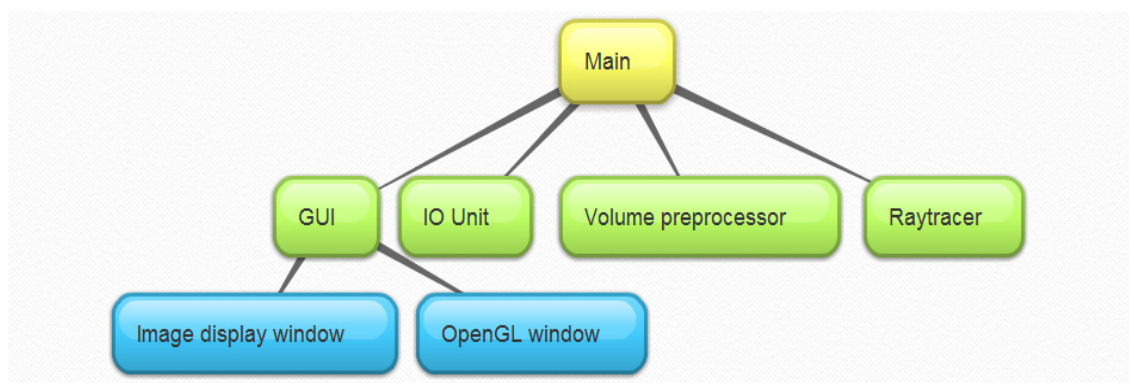
4. fejezet

Elkészült alkalmazások

Munkánk során elkészült egy Java alkalmazás mely bemutatja az elért eredményeket. Mivel az algoritmusok CPU-n történő számítása relatíve lassúnak bizonyult egy GPU-t használó CUDA alkalmazás is fejlesztésre került, valamint shader-ek a Voreen nyílt forráskódú vizualizációs engine-hez, és konzolos Matlab program, melynek segítségével háromdimenziós adathalmazt állíthatunk elő a Mandelbulb fraktálról. Most a Java és CUDA alkalmazás kerül bemutatásra.

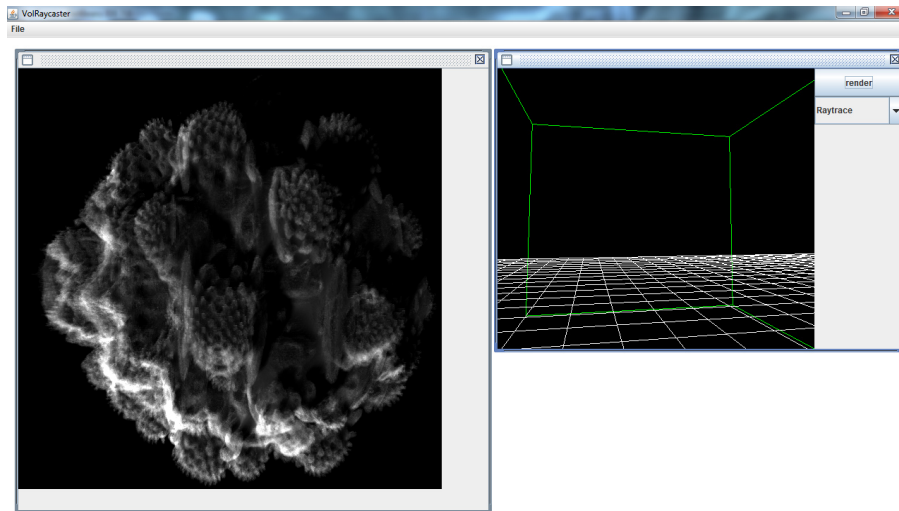
4.1. Java alkalmazás

A Java alkalmazás segítségével képesek vagyunk képeket előállítani volumetrikus adathalmazokról a 3. fejezetben leírt eljárások segítségével. Az alkalmazás interaktív grafikus beállító felületének kialakításához (mellyel a kamera paramétereit állíthatjuk be) a JOGL függvénykönyvtár került felhasználásra, mely az OpenGL Java implementációja. A következő ábrán az alkalmazás alrendszerei láthatók.



4.1. ábra: Java alkalmazás alrendszerei

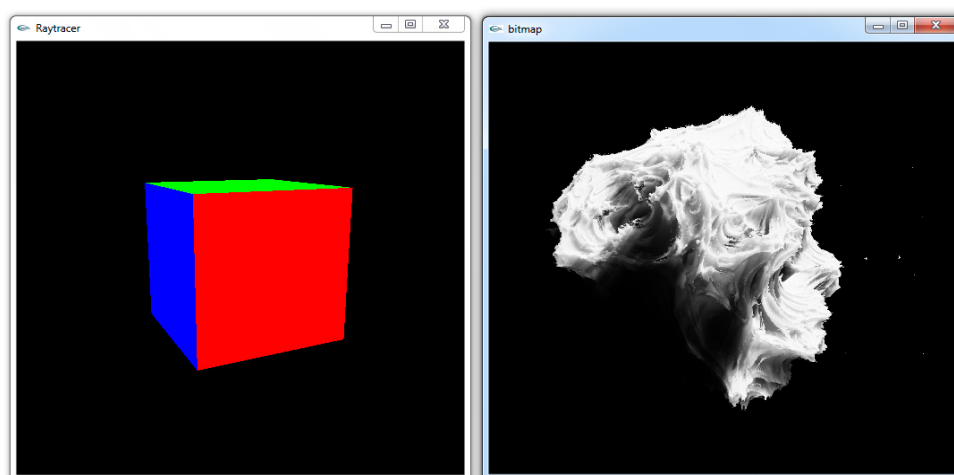
Az IO alrendszer végzi a program futásához szükséges input és output műveleteket, a volume preprocessor a voxelek előfeldolgozásáért felelős (árnyékolás és lokális hisztogramok kiszámítása), a grafikus felhasználói felületet és annak eseménykezelését a GUI alrendszer kezeli, a sugárkövető eljárások pedig a raytracer alrendszerben találhatóak.



4.2. ábra: Képernyőmentés a Java alkalmazásból. A ábra jobb oldalán látható a kamera paramétereit és a megjelenítő módszert beállító komponens, bal oldalon pedig egy sugárkövetéssel készült kép a Mandelbulb halmazról.

4.2. CUDA alkalmazás

A Java alkalmazás CPU-n számolja a sugárkövetést. Egy olyan árnyékolásos sugárkövetéssel készült kép kiszámítása melynél voxelenként egy árnyéksugarat indítunk körülbelül negyven percet vesz igénybe. Mivel a sugárkövetés jól párhuzamosítható a számítási sebesség növelése érdekében egy CUDA-t használó alkalmazás is kifejlesztésre került. A CUDA (Compute Unified Device Architecture) az Nvidia által kifejlesztett párhuzamos architektúra. Az elkészült alkalmazás képes Ambient Occlusion, és árnyékolásos sugárkövetés kiszámítására. Az előbbieken említett kép kiszámítása a CUDA programmal, GeForce 9600M GT GPU-n körülbelül negyven másodperc, modernebb GPU-n a számítási idő még kevesebb.



4.3. ábra: Képernyőmentés a CUDA alkalmazásból. A ábra bal oldalán látható a kamera paramétereit beállító komponens, a jobb oldalon pedig egy sugárkövetéssel készült kép egy kvaterniókon értelmezett Júlia-halmazról.

5. fejezet

Konklúzió

A fraktálokról viszonylag kevés ismeretanyaggal rendelkezik a matematika, mivel a terület eléggé újkeletű. Megismerésük egyik módja, hogy megjelenítjük őket. Dolgozatunkban a determinisztikus, hiperkomplex iterációval előállítható fraktálok megjelenítésére helyeztük a fő hangsúlyt. Ezen halmazok vizualizációja jóval összetettebb, mint a komplex számsíkon megadható fraktáloké, emiatt az algoritmusok hatékonysága sokkal nagyobb jelentőséggel bír. Munkánk során azt vizsgáltuk, hogy milyen eredmény érhető el a megjelenítés tekintetében, ha a kevésbé ismert térfogati megjelenítést alkalmazzuk a felületalapú megoldás helyett, eközben nagy figyelmet fordítottunk a térfogati megjelenítő algoritmusban felhasznált lokális megvilágítási modell megoldásának hibaanalízisére.

A hibaanalízis során bizonyítást nyert, hogy az általunk használt összetett Simpson-formula alkalmazása kedvezőbb a hiba szempontjából, mint az irodalomban használt megoldás és az algoritmus sebességére nincs negatív hatással. Globális megvilágítási modellek esetén a statikus Ambient Occlusion algoritmusban alkalmazásra került az optimális átlagok elmélete, dinamikus Ambient Occlusion esetén új megközelítés került bemutatásra a lokális hisztogramok előállításához, mely kedvezőbb az irodalomban használt megoldásnál, mivel levezetése az elméleti modellből történt. Munkánk során arra a következtetésre jutottunk, hogy a térfogatvizualizációs technikák jól alkalmazhatóak magasabb dimenziós fraktálok megjelenítésére. Abban az esetben, ha csak a fraktál alakját szeretnénk megfigyelni, elegendőek a lokális megvilágítási modellek, melyek a mai grafikus hardveren már valós időben számolhatóak. Realisztikus fény és árnyék eléréséhez globális megvilágítási modelleket kell alkalmaznunk, melyek számításigényesek.

Munkánk során elkészült egy Java alkalmazás, mely bemutatja az elért eredményeket. Mivel az algoritmusok CPU-n történő számítása relatíve lassúnak bizonyult, egy GPU-t használó CUDA alkalmazás is fejlesztésre került, valamint shader-ek a Voreen nyílt forráskódú vizualizációs engine-hez, és Matlab program, melynek segítségével háromdimenziós adathalmazt állíthatunk elő a Mandelbulb fraktálról. A Matlab kód megtalálható a függelékben.

A munkát több irányba is lehet folytatni. Kutatást lehet indítani nem fotorealistikus megjelenítés [15] irányába tovább növelve ezzel a megjelenítési lehetőségek számát, fotontérképek létrehozásával [14], melyek segítségével módunk nyílik volumetrikus kausztika modellezésére, nagyméretű adathalmazok tömörítésének az irányába is folytatható a munka [17], valamint különféle gyorsító technikákkal [16] növelhetjük a meglévő algoritmusok sebességét.

Irodalomjegyzék

- [1] Dr. Szirmay-Kalos László: *Számítógépes grafika*, Computerbooks, 1999.
- [2] Palle E.T Jorgensen: *Analysis and Probability Wavelets, Signals, Fractals* , Springer, 2006.
- [3] Daniel E. Lanier: *Fractal Geometry and the Escape-time algorithm as Graphic Art Tools*, presentation at SDSU Mathematics Summer Seminar, 2008.
- [4] Yumei Dang, Louis H. Kauffman, Dan Sandin: *Hypercomplex Iterations, Distance Estimation and Higher Dimensional Fractals*
- [5] Philipp Schlegel, Renato Pajarola: *Layered Volume Splatting*, IEEE Visualization Conference 2008.
- [6] Philipp G. Lacoute: *Fast volume rendering using a shear-warp factorization of the viewing transformation*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science Stanford University, 1995.
- [7] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, Daniel Weiskopf: *Real-Time Volume Graphics*, A K Peters, 2006.
- [8] James F. Blinn: *Light reflection functions for simulation of clouds and dusty surfaces*, ACM SIGGRAPH 82, 1982.
- [9] Kenneth Dean Moreland: *Fast High Accuracy Volume Rendering*, The University of New Mexico, 2004.
- [10] Galántai Aurél, Jeney András: *Numerikus módszerek*, Miskolci Egyetemi Kiadó, 2008.
- [11] Frida Hernell, Patric Ljung, Anders Ynnerman: *Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering*, Proceedings Eurographics/IEEEVGTC Symposium on Volume Graphics, Eurographics/IEEE, 2007.
- [12] Timo Ropinski, Jennis Meyer-Spradow, Stefan Diepenbrock, Jörg Mensmann, Klaus H. Hinrichs: *Interactive Volume Rendering with Dynamic Ambient Occlusion and Color Bleeding*, Computer Graphics Forum (Eurographics 2008), 27(2):567–576, 2008.
- [13] Nutefe Kwami Agbeko, Attila Házy: *An algorithmic determination of optimal measure form data and some applications*, Acta Mathematica Academiae Paedagogicae Nyíregyháziensis, Vol. 26, No. 1, pp. 99-111 (2010).

-
- [14] Henrik Wann Jensen: *Realistic Image Synthesis Using Photon Mapping*, A K Peters, 2001.
 - [15] Rezwan Sayeed, Toby Howard: *State of the Art Non-Photorealistic Rendering (NPR) Techniques*, EG UK Theory and Practice of Computer Graphics, pp. 1–10, 2006.
 - [16] Kristof Römisch: *Sparse Voxel Octree Ray Tracing on the GPU*, Master’s Thesis, Department of Computer Science, Aarhus University, 2009.
 - [17] Jörg Mensmann, Timo Ropinski, Klaus Hinrichs: *A GPU-Supported Lossless Compression Scheme for Rendering Time-Varying Volume Data*, IEEE/EG International Symposium on Volume Graphics, 2010.

A. függelék

Most az elkészült Matlab alkalmazás forráskódját nézheti meg az olvasó, mely az Escape-Time algoritmust használja adathalmaz előállítására a Mandelbulb halmazról.

```
%Függvény a+bi+cj alakban felírható
%hiperkomplex szám négyzetének
%kiszámítására. Az implementált formula
%a következő linken található meg:
%http://www.fractalforums.com/3d-fractal-generation
%/true-3d-mandlebrot-type-fractal/msg8680/#msg8680
%
%Használata: v = hypComplexSqr(a,b,c);
%
%Input adatok: '{a,b,c}' hiperkomplex szám
%
%Output adatok: 'v' vektor, mely tartalmazza a
%                hiperkomplex szám négyzetét.

function [v] = hypComplexSqr(a,b,c)
    v = [0,0,0];
    h = b*b+a*a;

    if(h==0)
        return;
    end;

    if(a==0&&b==0&&c==0)
        return;
    end

    h2 = 1-c*c/h;

    a1 = (a*a-b*b)*h2;
    b1 = 2*a*b*h2;
    c1 = 2*c*sqrt(h);

    v = [a1,b1,c1];
end
```

```

%Függvény metszet kiszámítására a
%Mandelbulb halmazból Escape-Time
%algoritmussal.
%
%
%Használata: img = calcMandelbulbSlice(width,height,
%           depth,z1, bailout, max_iterations);
%
%
%Input adatok: A 'width' a kimeneti kép szélessége.
%
%           A 'height' a kimeneti kép magassága.
%
%           A 'depth' pedig a maximális mélység érték.
%
%           A 'z1' a jelenlegi mélység érték.
%
%           A 'bailout' változó adja meg az Escape-Time
%           algoritmus kilépési feltételében lévő számot
%           (ami Mandelbulb halmaz esetén általában 8).
%
%           A 'max_iterations' változó adja meg az Escape-Time
%           algoritmus szereplő maximális iterációszámot.
%
%Output adatok: Az 'img' mátrix, mely tartalmazza a kiszámított képet.

function [img] = calcMandelbulbSlice(width,height,depth,z1, bailout, max_iterations)

    img = zeros(height,width);

    for(x1 = 1:width)
        for(y1 = 1:height)

            %skálázás
            x0 = (x1/(width/3))-1.5;
            y0 = (y1/(height/3))-1.5;
            z0 = (z1/(depth/3))-1.5;

            v1 = [0,0,0];
            iteration = 0;

            %Escape-Time algoritmus
            while ( (v1(1)*v1(1) + v1(2)*v1(2)+v1(3)*v1(3)) <= bailout
                    && iteration < max_iterations )

                %nyolcadik hatvány kiszámítása
                v1 = hypComplexSqr(v1(1),v1(2),v1(3));
                v1 = hypComplexSqr(v1(1),v1(2),v1(3));
                v1 = hypComplexSqr(v1(1),v1(2),v1(3));

```

```

        v2 = [x0,y0,z0];
        v1 = v1+v2;
        iteration = iteration + 1;
    end

    if ( iteration == max_iterations )
        img(y1,x1) = 0;
    else
        img(y1,x1) = iteration/max_iterations*255;
    end

end

end

end

% Függvény .raw kiterjesztésű adathalmaz előállítására a Mandelbulb
% halmazról.
%
% Használata: createMandelbulbDataset(size,bailout,max_iterations,filenév);
%
% Input adatok: A 'size' változó az adathalmaz mérete egy tengely mentén. Amiatt
%              kerül csak egyszeri megadásra, mivel az adathalmaz méretei mindhárom
%              tengely mentén megegyeznek. Ajánlott  $2^{2^n}$  méretet
%              választani (n pozitív egész szám), mivel a grafikus hardver
%              számára ez a méret kedvező a későbbi megjelenítés során.
%
%              A 'bailout' változó adja meg az Escape-Time algoritmus
%              kilépési feltételében lévő számot
%              (ami Mandelbulb halmaz esetén általában 8).
%
%              A 'max_iterations' változó adja meg az Escape-Time
%              algoritmus szereplő maximális iterációszámot.
%
%              A 'fileName' változó segítségével adhatjuk meg a kimeneti
%              file elérési útját, és nevét. Arra érdemes figyelni, hogy
%              az elérési útban szereplő könyvtárnak léteznie kell, mert
%              egyébként a Matlab nem tudja elmenteni az adathalmazt. Az
%              elmentett .raw kiterjesztésű file-t megnézhetjük a neten
%              ingyenesen elérhető ImageJ program segítségével, valamint a
%              Voreen nyílt forráskódú háromdimenziós engine is kezeli.

function createMandelbulbDataset(size, bailout, max_iterations, fileName)

%Kimeneti tömb inicializálása.
OUT = zeros(size,size,size);

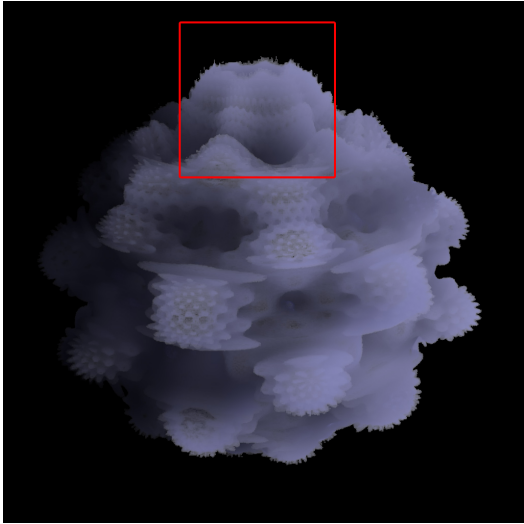
%Metszetek kiszámítása
for k = 1:size

```

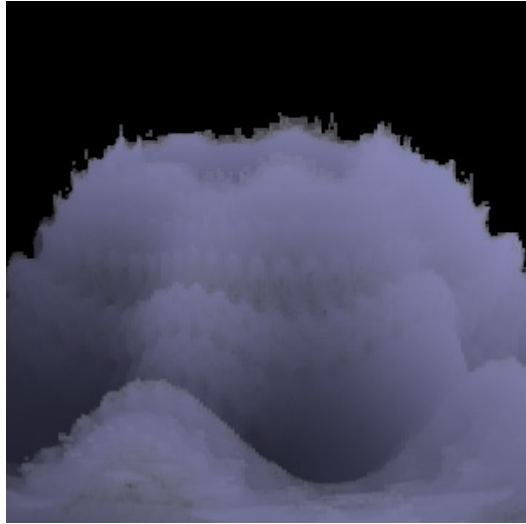
```
tic
%k-adik metszet kiszámítása
img = calcMandelbulbSlice(size,size,size,k, bailout, max_iterations);
toc
OUT(:, :, k) = img;
fprintf('%d metszet kiszámításra került, %d metszet
kiszámítása van még hátra. ', k, (size-k))
end

%Kimeneti tömb .raw file-ba mentése.
fid=fopen(fileName, 'w+');
fwrite(fid, OUT, 'uint8');
fclose(fid);
fprintf('A számítás befejeződött.')
```

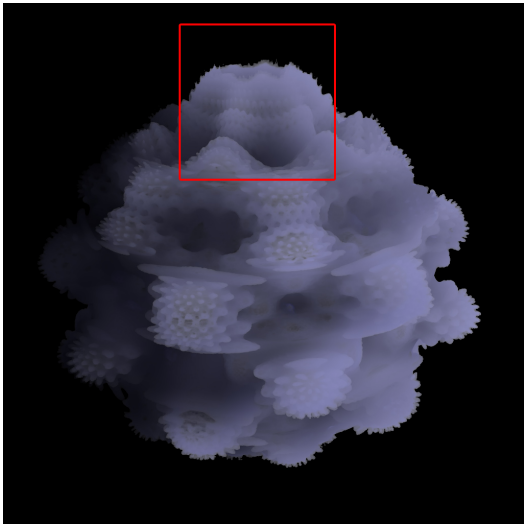
end



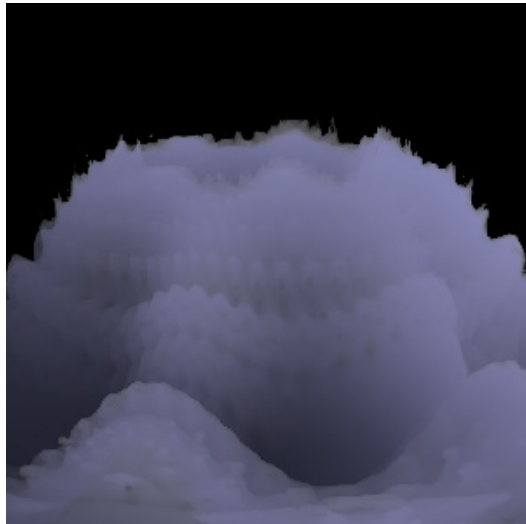
(a)



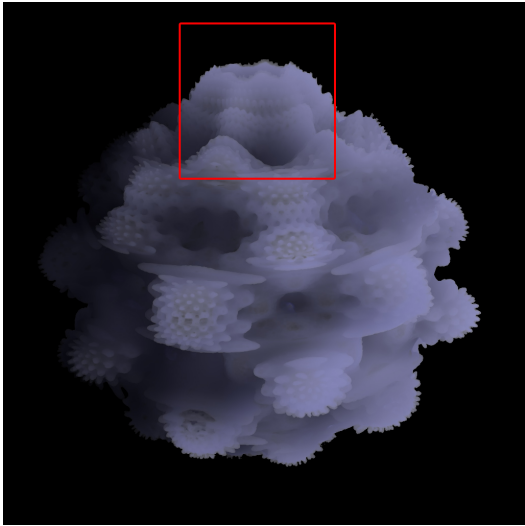
(b)



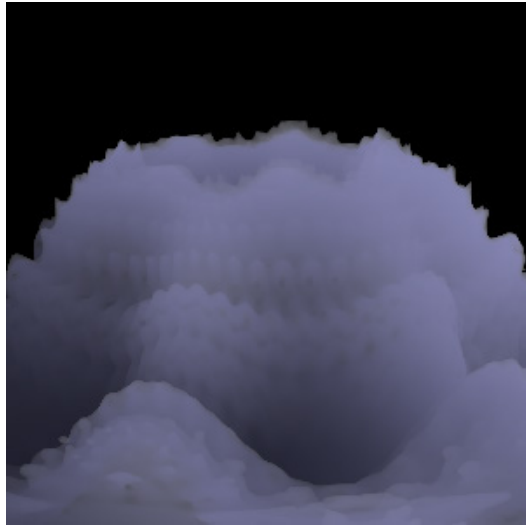
(c)



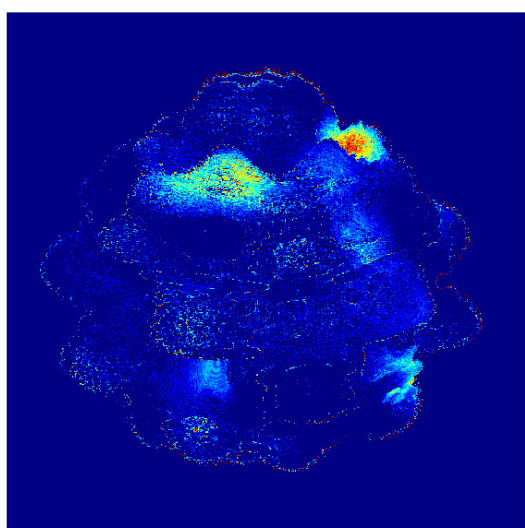
(d)



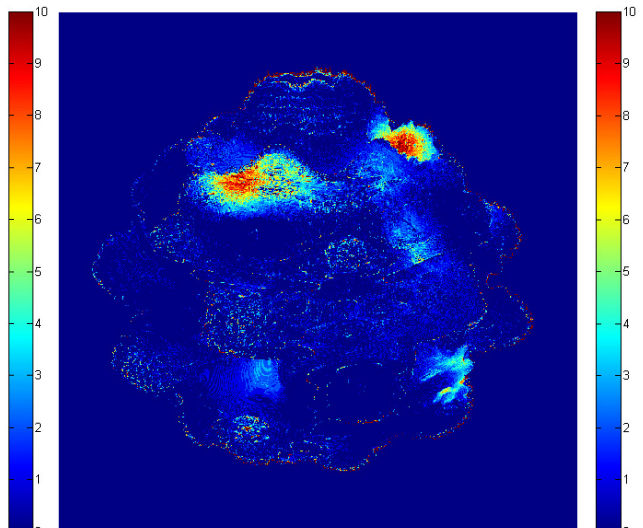
(e)



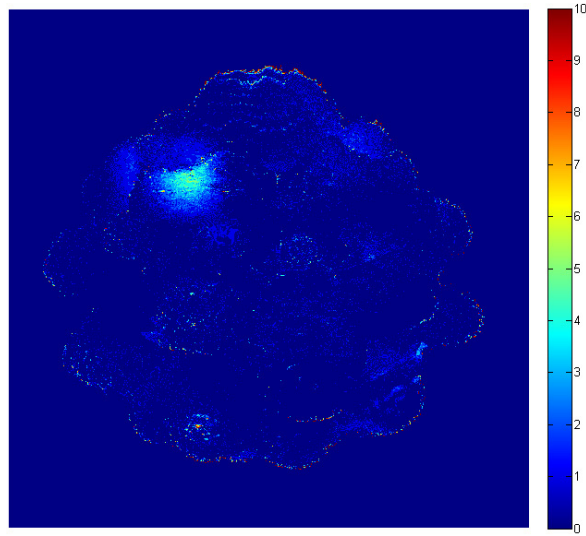
(f)



(g)

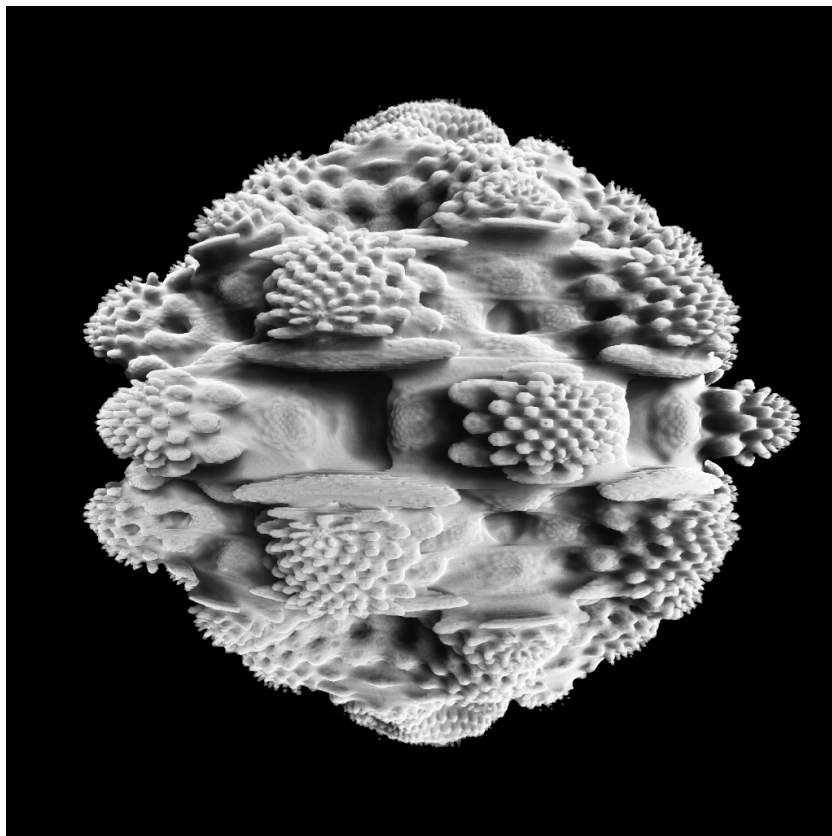


(h)

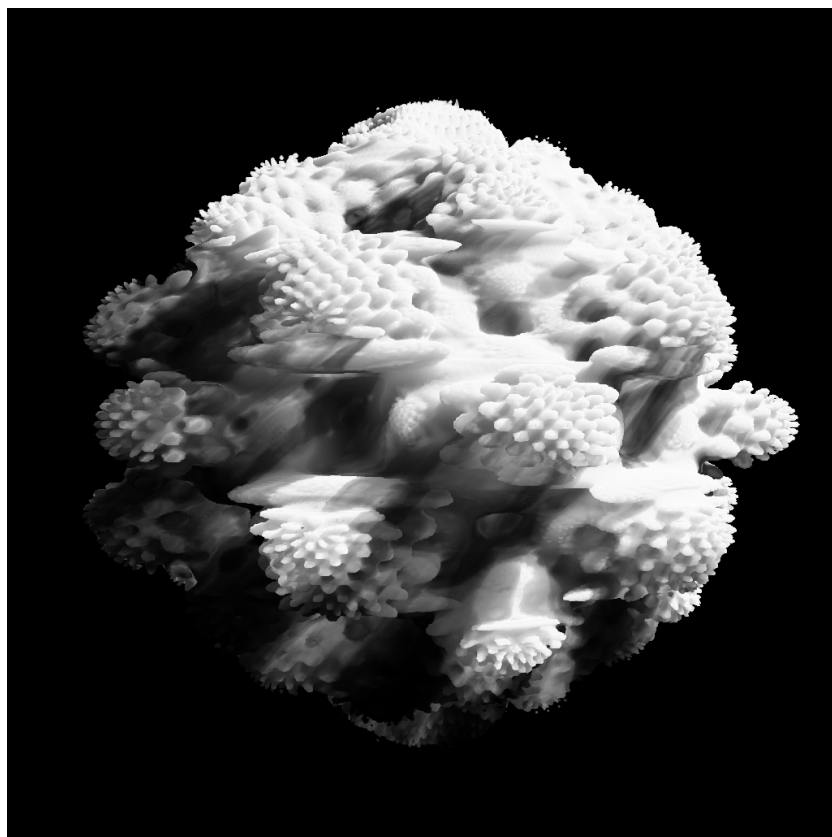


(i)

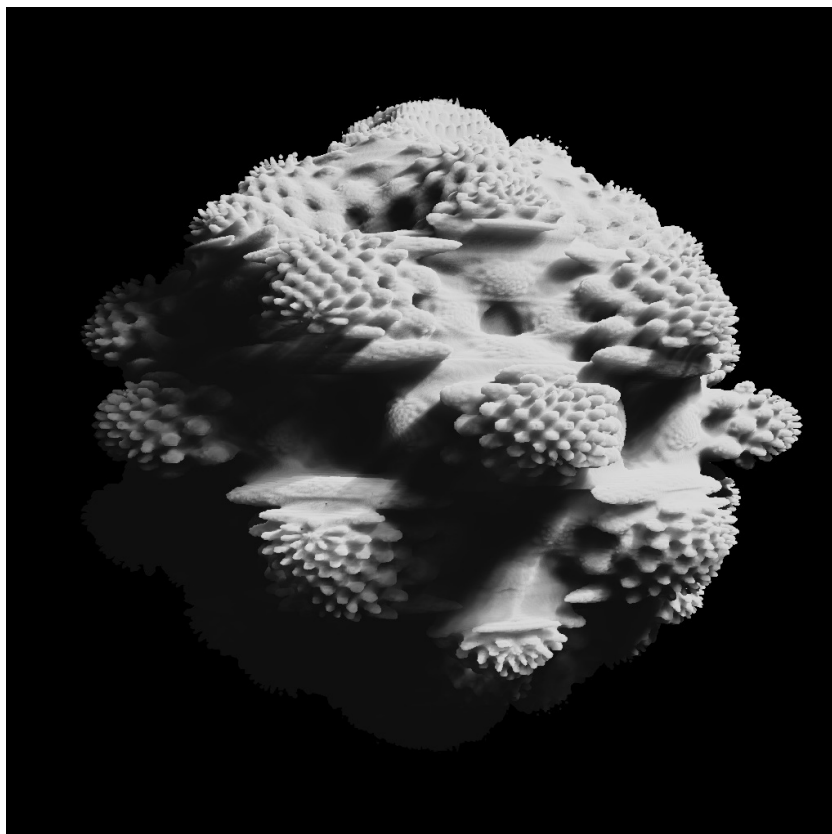
A.1. ábra: Az (a), (c) és (e) ábrákon a Mandelbulb halmazról Blender segítségével készült képeket láthatunk, melyek elkészítésénél a mintavételezéshez különböző interpolációs módszerek kerültek alkalmazásra. Az adathalmaz mérete 512^3 . Az (a) ábrán a nearest-neighbor (a mintavételi ponthoz legközelebbi ismert mintát választó), a (c) ábrán lineáris, az (e) ábrán köbös b-spline interpoláció került alkalmazásra. A (b), (d) és (f) ábrákon az (a), (c) és (e) ábrákon piros négyzettel jelölt részek nagyításai láthatóak. A nagyításokon jól észrevehetőek a képek közötti különbségek, szemmel jól látható eltérést az éleknél vehetünk észre. A legjobb élsimító hatással a köbös b-spline interpoláció rendelkezik, de ez a leginkább számításigényes. Valósídejű megjelenítésénél a lineáris interpoláció használata a leginkább elterjedt. A (g), (h) és (i) ábrákon az egyes képek pixelenkénti intenzitáskülönbségei láthatók. A (g) ábrán az (a) és (c), a (h) ábrán az (a) és (e), az (i) ábrán a (c) és (e) képek közötti intenzitáskülönbségeket vehetjük szemügyre. Az ábrákból kivehető, hogy a nearest-neighbor interpoláció és a másik két interpolációs módszer között jól látható eltérés adódik, míg a lineáris és a b-spline interpolációs módszer között kisebb.



(a)

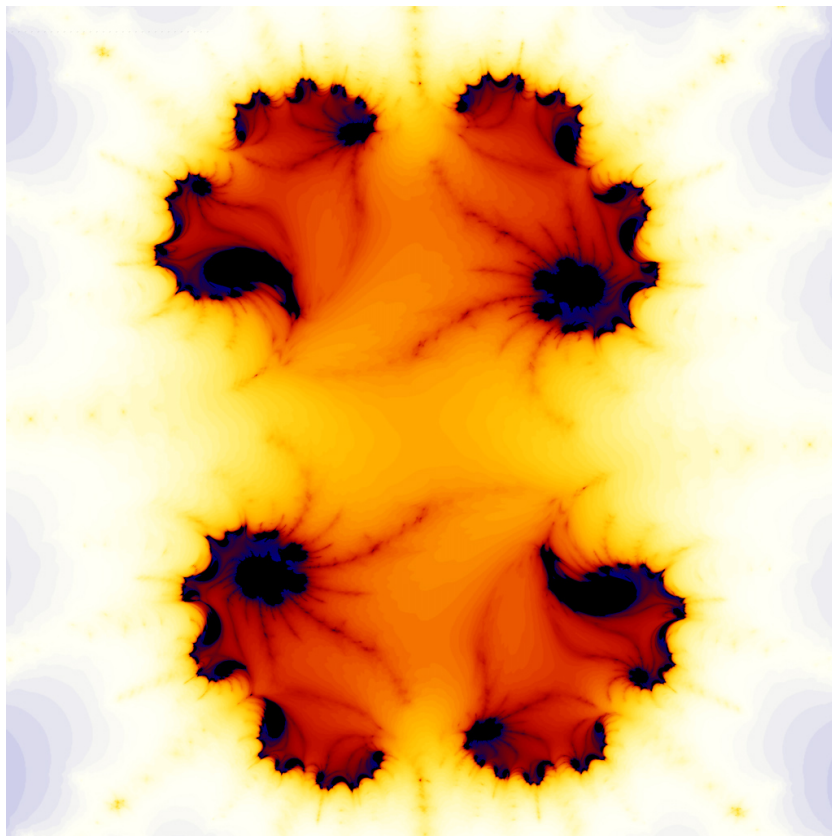


(b)

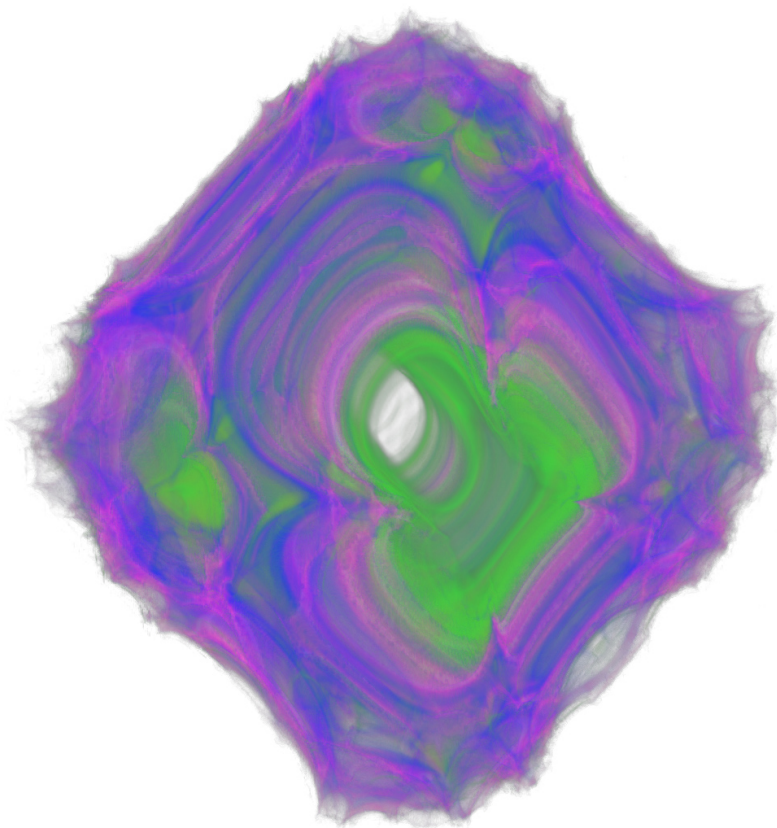


(c)

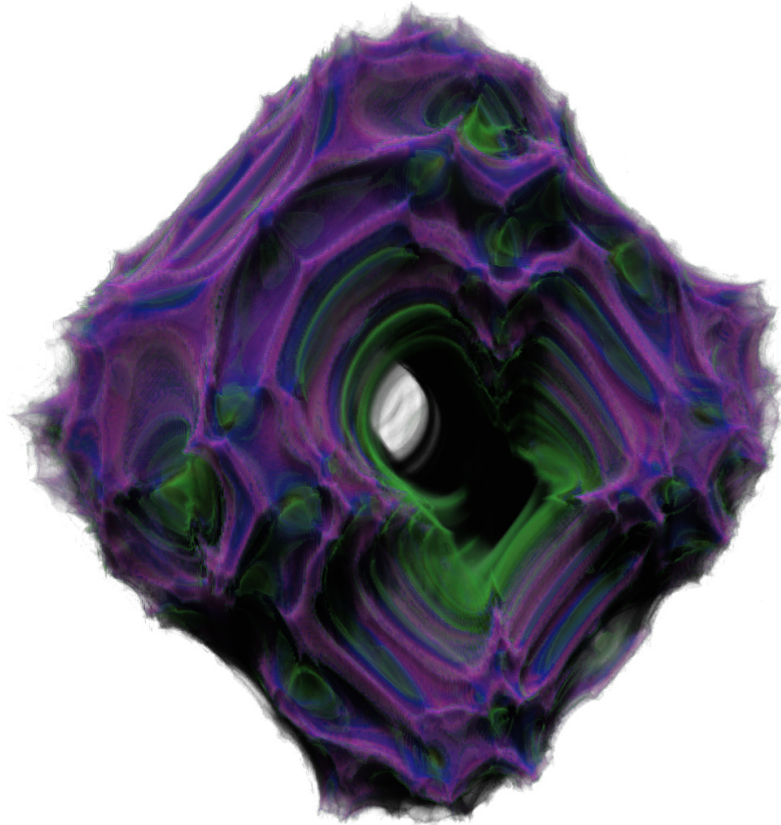
A.1. ábra: Az (a), (b) és (c) képeken az elkészült CUDA alkalmazás által előállított képek láthatók a Mandelbulb halmazról. Az adathalmaz mérete 512^3 . Az (a) képen a 3.6.2. alfejezetben ismertetett lokális ambient occlusion, a (b) és (c) képen a 3.2.4. alfejezetben ismertetett árnyékolási eljárás látható. Az árnyékoláshoz a (b) képen egy, míg a (c) képen több sugár került indításra a mintavételi pontokból a fényforrás irányába.



(a)

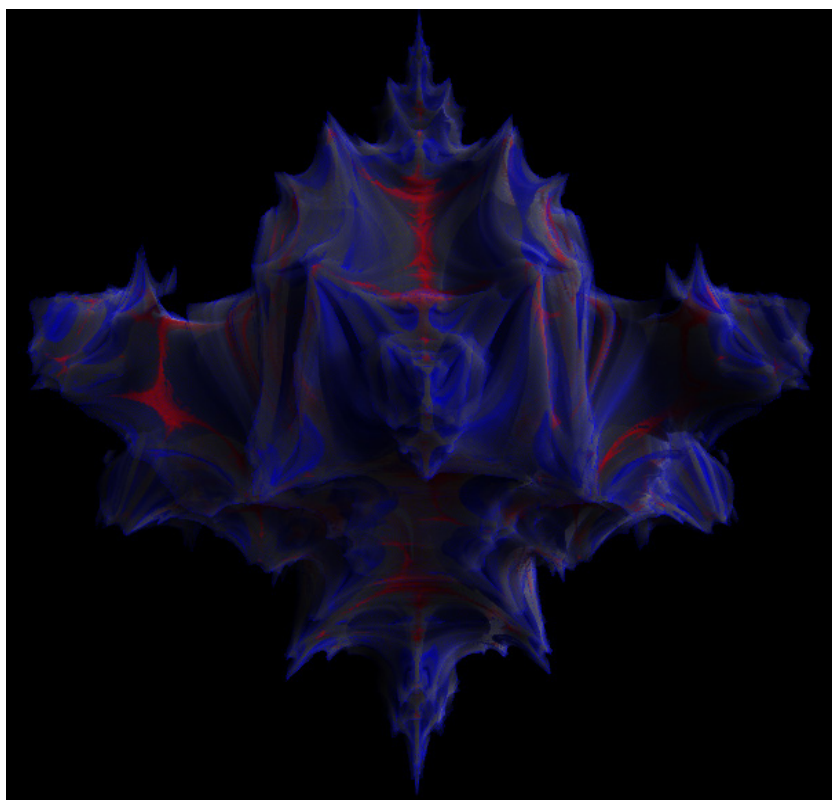


(b)

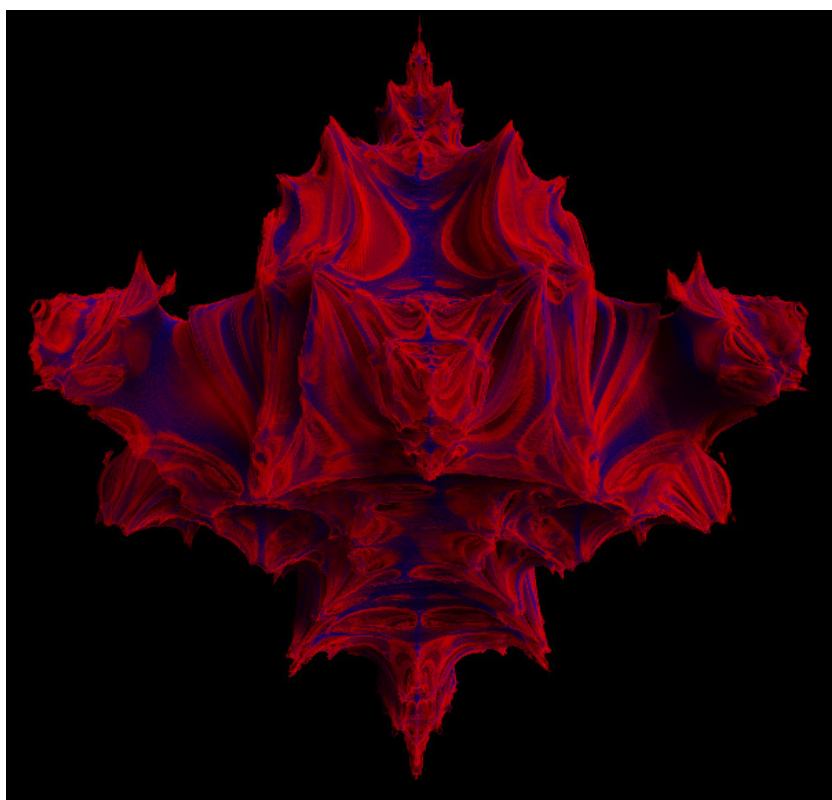


(c)

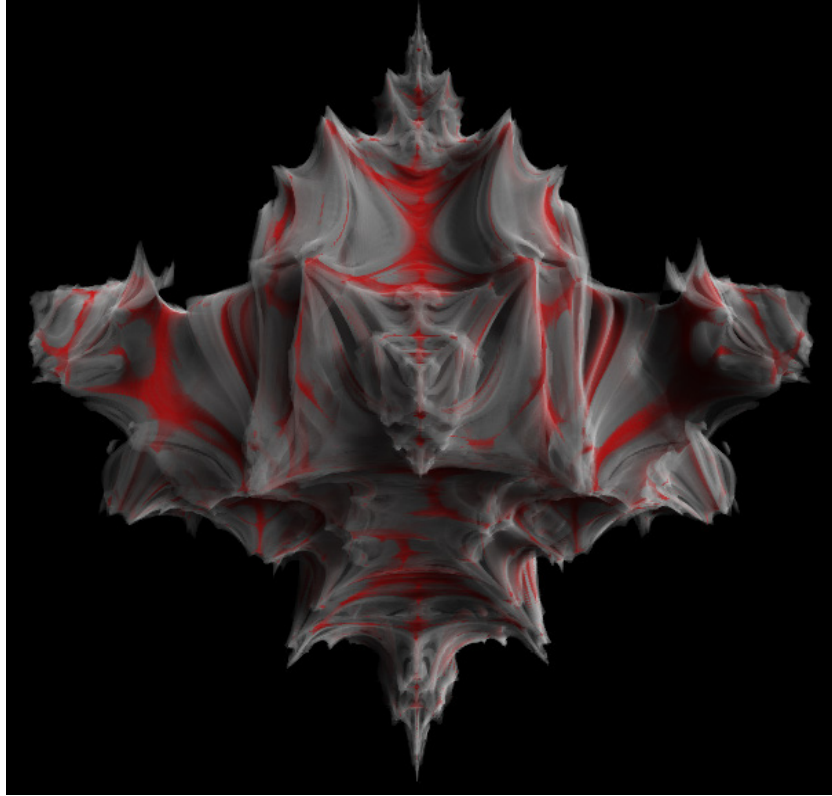
A.1. ábra: Az (a), (b) és (c) képek egy kvaterniókon értelmezett Julia-halmazról készültek. Ezen halmazok definíciója megegyezik a 2.2.1-es alfejezetben leírt definícióval, azzal a különbséggel, hogy komplex számok helyett kvaterniókat használunk. A képeken látható halmaz esetén $c = -1.474 + 0.0004i - 0.0004j$. Az adathalmaz mérete 512^3 . Az (a) képen egy metszet látható a fraktálból ($j = 0$ síkkal metszettük), a (b) képen a térfogatvizualizációs integrál segítségével előállított kép látható árnyékolás nélkül, a (c) képen pedig egy a problémát a grafikai objektumok szemszögéből megközelítő (object order) technika által készült képet láthatunk, a módszer neve Half-Angle Slicing, mellyel hasonló hatás érhető el, mint az árnyékolásos sugárkövetéssel. Az ábrán láthatjuk, hogy az árnyékolás segítségével a fraktál alakjára vonatkozóan több információt kapunk. A (b) és (c) képek a Voreen nyílt forráskódú háromdimenziós engine segítségével készültek.



(a)



(b)



(c)

A.1. ábra: Az (a), (b) és (c) képek egy kvaterniókon értelmezett Julia-halmazról készültek ($c = -0.3 - 0.01i + 0.5k$) az elkészült CUDA alkalmazás segítségével, lokális Ambient Occlusion technikával. Az adathalmaz, melynek mérete 512^3 , mindhárom esetben megegyezik, csak az átviteli függvény változik. A tapasztalat azt mutatja, hogy az Ambient Occlusion használata hiperkomplex fraktálok esetén célravezetőbb, mint az árnyékolásos sugárkövetés, mivel a sugárkövetéshez definiálnunk kell a fényforrás pozícióját. Abban az esetben, ha a fényforrás "nem jó" pozícióba kerül, az elkészült képen kevés információ válik láthatóvá. Ez a hátrány az Ambient Occlusion esetén nem áll fenn, és kevésbé számításigényes mint a több árnyéksugarat használó sugárkövetés.