
Szakdolgozat feladat

**Simon Zoltán**

BSc. mérnök informatikus hallgató részére

# Térfogatvizualizációs algoritmusok implementálása grafikus hardveren

A feladat olyan térfogatvizualizációs módszerek GPU-alapú implementálása, melyek lehetővé teszik a vizualizációs paraméterek (nézeti irány, átviteli függvény paraméterei) interaktív változtatását. A direkt térfogatvizualizációt két alapvetően különböző megközelítés szerint lehet implementálni. Az egyik a textúraszeletelés (texture slicing), a másik pedig a sugárkövetés (ray casting). A 3D voxeltömböt mindkét esetben feltöltjük a GPU textúramemóriájába, így azt a beépített trilineáris rekonstrukciós szűrővel lehet újramintavételezni. A textúraszeletelés esetén a befoglaló téglatest keresztmetszeti síkjaira illeszkedő háromszöghálók (proxy geometria) mentén mintavételezzük a 3D textúrát, míg a sugárkövetés esetén a képpontárnyaló (pixel shader) sorra egymásután vesz trilineáris mintákat a nézeti sugarak mentén.

**Elvégzendő részfeladatok:**

1. Textúraszeletelés implementálása: Proxy geometria kiszámítása a CPU keretprogrammal. Proxy geometria GPU-alapú megjelenítése 3D textúrázással és a keresztmetszeti szeletek kompozitálásával. Árnyékok számítása half-way slicing (a nézeti irány és a fényforrás irányának szögfelezőjére merőleges szeletelés) módszerrel.

2. Sugárkövetés implementálása: Belépési és kilépési pontok hatékony számítása a befoglaló téglatest lapjainak textúrába (Frame Buffer Object - FBO) történő renderelésével. A sugarak kiértékelése a képpontárnyalóval.

3. Magasabb rendű rekonstrukciós szűrők hatékony, GPU-alapú implementálása. A konvolúciós szűrés kiértékelése megfelelően pozícionált trilineáris minták súlyozott összegeként.

Budapest, 2022. október 5.

| | |
|---|---|
| …….. | …….. |
| Dr. Csébfalvi Balázs | Dr. Kiss Bálint |
| témavezető | tanszékvezető |
| egyetemi docens | egyetemi docens |

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai kar
Irányítástechnika és Informatika Tanszék

1117 Budapest, Magyar tudósok körútja 2. I. ép. III.em.
Postacím: 1521 Budapest, Pf.: 91.
Tel: 463-2699, Fax: 463-2204, http://www.iit.bme.hu


**Budapesti Műszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

# Térfogatvizualizációs algoritmusok implementálása grafikus hardveren

Szakdolgozat

*Készítette*
Simon Zoltán

*Konzulens*
Dr. Csébfalvi Balázs

2022. december 9.

# Tartalomjegyzék

## HALLGATÓI NYILATKOZAT

Alulírott *Simon Zoltán*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2022. december 9.

*Simon Zoltán*
hallgató

# Kivonat

Jelen dokumentum a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán mérnök informatikus alapszakon végző hallgató szakdolgozata. A dolgozat témája térfogatvizualizációs algoritmusok implementálása grafikus hardveren. Munkánkban foglalkozunk a sugárkövetés és a textúra szeletelés módszerével. Részletesen bemutatjuk a megjelenítéshez nélkülözhetetlen matematikai modellt. A dolgozat mellett egy térfogati adatok megjelenítésére alkalmas alkalmazást is készítettünk. Ennek architektúráját több absztrakciós szinten mutatjuk be. Szót ejtünk néhány optimalizációs lehetőségről. Kitérünk a térfogati adathalmaz mintavételezésének javítására. Szóba kerül a befoglaló geometria optimalizálása. Egy javított trilineáris szűrést alkalmazunk. Az írás bemutatja a területhez kacsolódó releváns és a munka során felhasznált szakirodalmat. Tárgyalja az algoritmusok implementálásának részleteit, és azok hatékonyságát. Elemzi az elkészült grafikus alkalmazást. Bemutatja és összeveti az ezzel előállított képeket. Végül értékeljük az elvégzett munkát.

# Abstract

This document is the BSc theses of a student at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. Topic of this thesis is implementation of volume rendering algorithms on the graphics hardware. We examine ray casting and texture slicing methods. We present the mathematical model necessary for volume rendering. Besides the thesis we have also created a rendering application capable of rendering volumetric data. We present the architecture of this software on different abstraction levels. We discuss some possible optimalisation techniques. We make some notes about the reduction of the bounding box. There is a part about bettering the volume sampling quality. For this purpose, we use trilinear interpolation with correction. We survey the relevant scientific literature. We write about the implementation details of the used algorithms. We analyze the complete program and showcase the rendered images. Finally, we discuss the results of the complete work.

# 1. fejezet

# Bevezetés

Térfogati adatok megjelenítésére számos területen szükség van. Az egyik fontos felhasználási terület az orvosi képalkotó eljárásokkal előállított adathalmazok vizualizációja. Ezen a területen a két meghatározó technológia a Komputertomográfia (CT) és Mágnesesrezonancia-képalkotás (MRI). Ezek statikus képet állítanak elő az élő szervezetről. Ezekhez szorosan kapcsolódik a Pozitronemissziós tomográfia (PET), ami időbeli változások, élettani folyamatok megfigyelésére alkalmas. Ebben a dolgozatban állóképek megjelenítésével foglalkozunk, és adatok időbeli változásával nem. A problémát más oldalról megközelítve ez a megkötés megengedi számításigényesebb szűrési módszerek és megjelenítési eljárások alkalmazását. Nem feltétlenül kell ragaszkodni ugyanis a magas képfrissítési rátához. Pontosítva, csak akkor szükséges az új képek gyors egymásutáni megjelenítése, amikor a felhasználó megváltoztat valamilyen paramétert. Talán a legkézenfekvőbb ilyen változtatás a nézeti irány módosítása. Mivel térbeli struktúrát vizsgálunk, ezért egy kamera állásból lehetetlen lenne alaposan szemügyre venni minden részletet. A kamera forgatása közben elvárjuk, hogy folyamatosan frissüljön a kép. Ez tehát megköveteli, hogy szükség esetén ‚valósidőben" (soft real-time) jelenítsünk meg új képeket egy kellően gyors a eljárással, viszont megengedi, hogy alkalomadtán hosszabb renderelés után részletesebb ábrát jelenítsünk meg.

A CT, MRI és egyéb technológiák intenzitások szabályos rácsba rendezett pontfelhőjét állítják elő. Mi kizárólag ezen pontfelhők megjelenítésével foglalkozunk, és ezek előállításának mikéntjéről nem ejtünk szót. Kétdimenziós képek képpontjaira pixelként szokás hivatkozni. Ennek megfelelően a háromdimenziós pontfelhők pontjait voxelnek nevezzük. Ez a térfogat (volume) és pixel szavak összevonásából származik. Egy térrészt véges sok voxellel írunk le. Ez fölveti a kérdést, hogy egy aránylag alacsony felbontású adathalmaz megjelenítésekor milyen módszerekkel állítsuk elő a rendelkezésre álló pontok közti köztes pontokhoz rendelendő intenzitást. Az egyik gyakran alkalmazott szűrési módszer a trilineáris interpoláció. Ez viszonylag kevés, 8 mintából állít elő egy interpolált értéket. A gyakorlatban gyakran jelent ideális kompromisszumot a képminőség és futási sebesség közt. Magasabb rendű közelítést jelent például a köbös B-Spline vagy Catmull-Rom spline [24]. Ezek sokkal több mintavételezést igényelnek, így jelentős a többletköltségük.

A térfogatvizualizációt két alapvetően eltérő megközelítés szerint lehet implementálni. Az egyik a sugárkövetés (ray casting), a másik a textúraszeletelés (texture slicing). Sugárkövetés során a képpontokhoz egy-egy sugarat rendelünk. A sugarak mentén lépkedve mintavételezzük a térfogatot. Az egyes lépések mintáit kombinálva áll elő a megjelenítendő szín. Textúraszeletelés során a térfogatot egymásra párhuzamos síkokkal metszük. A metszetként előálló legfeljebb hatszögek pontjaiban mintavételezünk és az így előálló szelet képét textúrába írjuk. A textúrák kombinációjával áll elő a végleges kép. Meg lehet úgy közelíteni a két módszer közti alapvető eltérést, hogy míg a sugárkövetés során egyszerre egy

sugáron haladunk végig, addig a textúraszeletelés során az összes sugár mentén egyszerre veszünk mintát. Mindkét módszernek megvannak az előnyei és hátrányai. A sugárkövetés egy általánosabb módszer. A textúra szeletelés lehetővé tesz speciális megoldásokat, mint a half-angle slicing. Lényege, hogy a szeletelő síkokat úgy választjuk, hogy azok felületi normál vektora megfelezze a térfogatból a néző és a fény irányába mutató irányvektorok által bezárt szöget. Így a szeletek megfelelő sorrendben való feldolgozása során mindig rendelkezésre áll az adott pontra beeső fényt árnyékoló pontok intenzitása. Ez lehetővé teszi realisztikus árnyékok megjelenítését. A módszer hátránya, hogy csak egy árnyékot vető fényforrást enged meg. Kihasználva a Grafikus Feldolgozó Egység (GPU) párhuzamosító képességét a sugárkövetést és a textúraszeletelést nagy hatékonysággal valósíthatjuk meg.

A dolgozat 2. fejezetében áttekintjük a témához kapcsolódó korábban publikált eredményeket, amelyekre ez a munka támaszkodik. Sorra vesszük a felhasznált eljárásokat. A 2.2. alfejezet részletesen bemutatja a sugárkövetés módszerét, kitérve a kamera irányába verődő fény kiszámításához szükséges matematikai modellre. Ezt követően a 2.3. alfejezetben részletezzük a half-angle slicing implementáció miként jét. Munkánk során nem csak algoritmusok kipróbálása volt a célunk. Szándékunkban állt egy teljes értékű megjelenítő szoftver fejlesztése is. A 3. fejezetben bemutatjuk az elkészült szoftver architektúráját. Itt részletes beszámolót adunk a fejlesztés során felmerült döntési helyzetekről. Megindokoljuk az egyes problémákra adott megoldások létjogosultságát. A 4. fejezetben értékeljük az alkalmazás teljesítményét. Bemutatjuk az megjelenített képeket. Szó lesz a képek megjelenítésének sebességéről. Megvizsgáljuk a program paramétereinek megváltoztatásával a megjelenített képeken bekövetkező változásokat. Megkíséreljük az ábrák minőségét kritikus szemmel vizsgálni. Azok erősségeit és gyengeségeit objektívan összehasonlítani. Az 5. zárófejezetben összefoglaljuk a elvégzett munka eredményeit. Megemlítünk további fejlesztési lehetőségeket, amelyekkel az itt bemutatott eredmények tovább javíthatóak, vagy egészen új irányba bővíthetőek.

# 2. fejezet

# Elméleti áttekintő

A fejezet célja, hogy részletesen bemutassa a felhasznált térfogatvizualizációs eljárásokat. Érdemesnek tartunk szót ejteni a mi megvalósításunktól eltérő módszerekről, és, hogy ezeket miért nem választottuk a saját implementáció során. Szilárd testekről készült térfogati adathalmazok megjelenítésekor gyakran alkalmaznak izofelületeket. Valamilyen átviteli függvény segítségével elhatárolják a térfogatban megjelenő különböző struktúrákat. Az ezek határát képző felületet poligonhálóval közelítik. Ez a módszer akkor alkalmazható hatékonyan, ha az egyes struktúráknak jól definiált határa van, valamint a struktúrák homogének. Ezzel szemben folyadék, füst, por és hasonló képlékeny struktúrák megjelenítésére sokkal alkalmasabb a direkt térfogati adatok mintavételezése és valamilyen optikai modell alkalmazása a közegben elnyelődő, illetve a néző irányába visszaverődő fény megállapítására. Ennek oka, hogy az ilyen közegek esetén nem tudunk pontos felülethatárt megállapítani. Hasonló a helyzet akkor, amikor nem szeretnénk, hogy az adathalmazban jelenlévő finom részletek – egy egyszerűsítés áldozatává válva – homogén közegként legyenek megjelenítve. Annak érdekében, hogy a térfogat apró inhomogenitásait is megjelenítsük izofelületek segítségével, nagy számú különböző struktúrát kell megkülönböztetnünk. Ezek mindegyikének határához konstruálnunk kell izofelületeket. Vegyük észre, hogy egy struktúra nem biztos, hogy összefüggő. Előfordulhatnak azonos csillapítási tényezőjű, de térben egymástól távol elhelyezkedő pontok. Ezek mind különböző izofelületet kellene, hogy kapjanak. Intuitívan érezhető tehát, hogy az izofelületes módszernek nem erőssége a gazdag részletességű, inhomogén térfogat megjelenítése. Munkánk során szerettünk volna az utóbbi leírásnak megfelelő adatokat megjeleníteni. Ezért esett választásunk a térfogati adatok direkt mintavételezésével történő megjelenítésére. Ezt a módszert is több, eltérő megközelítés szerint valósíthatjuk meg.

## 2.1. Optikai modellek

A direkt térfogatmegjelenítési eljárások szükségessé teszik valamilyen modell alkalmazását, amely leírja, hogy a a közegben hogyan verődik vissza, szóródik, nyelődik el vagy emittálódik fény [34, 26].

A megjelenítendő térfogatot tekinthetjük egy $f(\vec{p})$ függvénynek, ami egy adott térbeli pozícióhoz megadja annak optikai tulajdonságait. Ilyen tulajdonság a visszavert fény színe és a beeső fény abszorbciójának mértéke. Ez a függvényérték a gyakorlatban szomszédos minták interpolációjával áll elő. Az $f(\vec{p})$ paraméterek felhasználásával valamilyen modell szerint kiszámoljuk az adott pont megjelenését. Felmerülhet a kérdés, hogy miért szűrjük először az optikai tulajdonságokat, ahelyett, hogy először a rendelkezésre álló diszkrét felbontású rács pontjaiban számítanánk ki a megjelenést. Ezek után ugyanúgy interpolálhatnánk a rácspontokban számított megjelenítési módot. A válasz erre, hogy

az utóbbi módon eljárva sokkal feltűnőbb vizuális artifaktumokat kapnánk. Hasonlóan a *Phong* árnyalás sokkal jobb eredményeket ad, mint a Gouraud módszer. Ezek között – a térfogamegjelenítési esettel analóg módon – a különbség, hogy a *Phong* árnyalás először a felületi normálokat interpolálja, és csak az interpolált normál vektort használja árnyalás számítására. A *Gouraud* árnyalás ezzel szemben a már kiszámolt színt interpolálja.

Az általunk használt optikai modell két fő komponensre bontható. Ezek a sugár mentén abszorbált és a kamera irányába emittált erősség. Ezek leírásához a közegben jelenlévő részecskék árnyékolásán és sugárzásán alapuló modellből indulunk ki.

### 2.1.1. Fény abszorbció

Elsőként feltételezzünk egy olyan közeget, amelyben a részecskék tökéletes feketetestként viselkednek. Azaz nem sugároznak, és minden beeső fotont visszaverés nélkül nyelnek el. Egyszerűség kedvéért feltehetjük, hogy a részecskék azonos $r$ sugarú gömbök. Ekkor a kamera szemszögéből nézve – ortografikus projekciót feltételezve – a vetületük területe $A = \pi r^2$. Legyen az egységtérfogatban előforduló részecskék száma $\rho$. Legyen egy hengertest formájú térrész alapterülete $E$, magassága $\Delta s$ úgy, hogy a fény terjedési iránya a henger alapjára merőleges. A térrész térfogata $E\Delta s$. Felhasználva a bevezetett sűrűségfogalmat az ebben a térrészben található részecskék száma $N = \rho E \Delta s$. Feltételezve, hogy $\Delta s$ elég kicsi ahhoz, hogy a részecskék elhanyagolható valószínűséggel kerülhessenek fedésbe, a teljes térrészben $NA = \rho AE\Delta s$ területet árnyékolnak a részecskék. Az térrészen átjutó fény és eredetileg besugárzott fény aránya ebből $\frac{\rho AE\Delta s}{E} = \rho A \Delta s$. Ahogyan $\Delta s$ közelíti a nullát 2.1 differenciálegyenletet írhatjuk fel.

$$\frac{dI}{ds} = -\rho(s)AI(s) = -\tau(s)I(s) \tag{2.1}$$

A $\tau(s) = \rho(s)A$ a kioltási tényező. Ez egy szeparábilis elsőrendű differenciálegyenlet, azaz $y' = f(x)g(y)$ alakú, a 2.2 megfeleltetéssel.

$$f(x) = -\tau(s), g(y) = I(s) \tag{2.2}$$

$I(s) = 0$ megoldása az egyenletnek. Ez az az eset, amikor a közeg nincs megvilágítva. Így az intenzitás konstans nulla. Vizsgáljuk a $I(s) \neq 0$ esetet[1]. Rendezzük a 2.1 kifejezést elosztva $I$-vel, és szorozva a $ds$ szimbolikus konstanssal.

$$\int \frac{1}{I} dI = \int -\tau(s) ds \tag{2.3}$$

Nevezzük át a jobb oldali paramétert $t$-re, és $t$ szerinti integrált elég 0 és $s$ közt végezni, mivel a sugárerősséget csak ebben a tartományban befolyásolja a közeg. Végezzük el a baloldali integrálást.

$$ln(I) = \int_0^s -\tau(t)dt + K \tag{2.4}$$

Alakítsuk a 2.4 kifejezést $e$ alapúvá.

$$I = Ce^{\int_0^s -\tau(t)dt} \tag{2.5}$$

[1]Ez egy homogén lineáris diff egyenlet is egyben, tehát elegendő az $I(s) \neq 0$ eset vizsgálata. Ennek oka, hogy az általános megoldások egydimenziós lineáris teret alkotnak, amik csak egy C konstans szorzóban térnek el.

$I_0$ a térfogatba belépő fény intenzitása. Tehát az általános megoldás után keressük az $I = I_0$ és $s = 0$ kezdeti feltételhez tartozó partikuláris megoldást.

$$I_0 = Ce^{\int_0^0 \tau(t)dt}$$
$$I_0 = Ce^0$$
$$I_0 = C$$

Az egyenletet tehát a 2.6 formában írhatjuk fel.

$$I(s) = I_0 exp\left(-\int_0^s \tau(t)dt\right) \tag{2.6}$$

Függetlenítve a kezdeti intenzitástól a közeg áttetszőségét kapjuk, amit a 2.7 egyenlet ír le.

$$T(s) = exp\left(-\int_0^s \tau(t)dt\right) \tag{2.7}$$

Áttetszőséget egyből kivonva kapjuk a közeg opacitását, amit a számítógépes grafika területén az alfa csatornában szokás eltárolni $\alpha = 1 - T(l)$. Hogyha $\tau$ konstans, akkor a 2.7 egyenlet a 2.8 formára egyszerűsödik.

$$T(s) = exp(-\tau s) \tag{2.8}$$

Hogyha $\tau$ kioltási tényező 0-hoz tart, akkor az áttetszőség 1-hez tart. Így a közeg $\alpha$ értéke 0-hoz közelít. Ellenkező esetben, ha $\tau \sim 1$, akkor $\alpha \sim 1$.

### 2.1.2. Fény emisszió

Ebben az alfejezetben továbbra is a korábban bevezetett részecskéken alapuló modellre támaszkodunk. Most a részecskéket tökéletesen átlátszónak tekintjük, és azt mondjuk, hogy egységfelületenként $C$ intenzitással sugároznak a kamera irányába. A következő fejezetben majd kombináljuk az abszorbciót és emissziót.

Korábban megmutattuk, hogy a részecskék $\rho AE\Delta s$ területet fednek le. Mostani modellünkben ez $C\rho AE\Delta s$ járulékos energia áramot eredményez. Ez motiválja a 2.9 differenciálegyenlet felírását.

$$\frac{dI}{ds} = C(s)\rho(s)A = C(s)\tau(s) = g(s) \tag{2.9}$$

Az egyenlet jobb oldalán g(s) a forrás-tag (*source term*). Későbbiekben ez nem csak a közeg emisszióját, hanem a néző irányába visszavert fényt is tartalmazni fogja. $I(s) = const$ függvény nem megoldása az egyenletnek, mert a jobb oldalon az $I(s)$-től függő szorzó tag konstans 1. Ennek fizikai magyarázata, hogy egy emittáló közeg mindig növeli a sugárerősséget. Vizsgáljuk az általános esetet. Integráljuk mindkét oldalt $s$ szerint.

$$\int \frac{dI}{ds} ds = \int g(s)ds \tag{2.10}$$

A 2.10 kifejezés baloldalát egyszerűsítve, a jobboldal integrálási változóját átnevezzük, és az integrálás tartományát 0 és $s$ közé szűkítjük. Végül megkeressük az $I(0) = I_0$ kezdeti feltételhez tartozó partikuláris megoldást. Így kapjuk a 2.11 alakot.

$$I(s) = I_0 + \int_0^s g(t)dt \tag{2.11}$$

Vegyük észre, hogy ennek az integrálnak nincs felső korlátja. Ez azt jelenti, hogy erősen emittáló közegben, vagy mély közegen keresztül előfordulhat, hogy túl intenzív fény jut a kamerába. Ezért a megjelenített kép minősége a megfelelő paraméterválasztástól függ.

### 2.1.3. Abszorbció és emisszió együtt

A korábbi modellekben a realisztikustól messzemaradó kikötéseket tettünk a közeget alkotó részecskékre nézve. A 2.1.1 alfejezetben feltettük, hogy a részecskék tökéletesen elnyelelik a rájuk eső fényt. A 2.1.2 alfejezetben megkötöttük, hogy a részecskék átlátszóak és csak hozzáadnak a sugárerősséghez. Most ezeket a megkötéseket feloldva – de a megkötések segítségével elért eredményeket felhasználva – általánosítsuk a modellt. A 2.12 inhomogén lineáris elsőrendű differenciálegyenlet tartalmazza az abszorbciós- és forrás-tagot is.

$$\frac{dI}{ds} = g(s) - \tau(s)I(s) \tag{2.12}$$

A megoldáshoz felhasználjuk az inhomogén egyenlethez tartozó a 2.1 homogén egyenlet 2.5 általános megoldását. Egy inhomogén lineáris elsőrendű differenciálegyenlet általános megoldását a hozzá tartozó homogén egyenlet általános megoldásának és az inhomogén egyenlet egy partikuláris megoldásának összegeként kaphatjuk meg. A partikuláris megoldást konstans variálással keressük 2.5 alakban a $C$ helyett $C(s)$-t írva.

$$I_{ip}(s) = C(s)exp\left(-\int_0^s \tau(t)dt\right) \tag{2.13}$$

2.14 ennek $s$-szerinti deriváltja.

$$\frac{dI_{ip}}{ds} = C'(s)exp\left(-\int_0^s \tau(t)dt\right) - \tau(s)C(s)exp\left(-\int_0^s \tau(t)dt\right) \tag{2.14}$$

A 2.13 és 2.14 kifejezéseket 2.12 egyenletbe helyettesítjük és rendezzük $C'(s)$-re.

$$C'(s) = \frac{g(s)}{exp\left(-\int_0^s \tau(t)dt\right)} \tag{2.15}$$

Ezt a folytonos függvényt $s$ szerint integráljuk.

$$C(s) = \int_0^s \frac{g(t)}{exp\left(-\int_0^t \tau(u)du\right)}dt \tag{2.16}$$

$$I_{ip}(s) = \int_0^s g(t)exp\left(-\int_t^s \tau(u)du\right)dt \tag{2.17}$$

A fent írtaknak megfelelően $I(s)$ általános megoldása $y_i = y_h + y_{ip}$ alakú. Kombináljuk a 2.5 és 2.17 kifejezéseket, valamint megoldjuk az $I(0) = I_0$ kezdetiérték problémát.

$$I(s) = I_0exp\left(-\int_0^s \tau(t)dt\right) + \int_0^s g(t)exp\left(-\int_t^s \tau(u)du\right)dt \tag{2.18}$$

A 2.18 kifejezés $I_0$-t tartalmazó tagja a közeg mögül a kamera irányába érkező fény sugárerősség csillapítását írja le. A második a közegben összegyűlő járulékos sugárerősség és annak folyamatos – a közeg okozta – csillapítását írja le.

### 2.1.4. Integrál számítása számítógéppel

Néhány speciális esettől eltekintve az integrálás programban való implementálása csak közelítő módszerekkel segítségével valósítható meg. $\int_a^b h(x)dx$ közelítésére a $\sum_{i=1}^{n} h(x_i)\Delta x$ Riemann összeg nyújt lehetőséget, ahol $\Delta x = (b-a)/n$ a diszkrét lépések hossza és $x_i = a + i\Delta x$ az aktuális lépés helyettesítési értéke.

Az előzőek alapján $exp\left(-\int_0^s \tau(t)dt\right)$ kifejezést a 2.19 alakban közelíthetünk.

$$exp\left(-\sum_{i=1}^{n} \tau(i\Delta x)\Delta x\right) = \prod_{i=1}^{n} exp\left(-\tau(i\Delta x)\Delta x\right) = \prod_{i=1}^{n} t_i \tag{2.19}$$

A 2.18 kifejezést átalakítva kapjuk a 2.20.

$$I = I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j \tag{2.20}$$

2.20 alakot a gyakorlatban $t = 1 - \alpha$ helyettesítéssel használjuk.

### 2.1.5. Szóródás és árnyalás

Az eddig tárgyalt modell már figyelembe veszi a közeg mögül érkező megvilágítást és ennek a közegben elszenvedett csillapítását, valamint a közeg emisszióját. Eddig nem foglalkoztunk a fény szóródásával. A szóródási jelenségek szimulációja azonban nagyban hozzájárul a realisztikus megjelenéshez.

Általánosan a 2.21 formulával leírható az $X$ ponton $\omega$ irányba szóródó sugárerősséget.

$$S(X,\omega) = r(X,\omega,\omega')i(X,\omega') \tag{2.21}$$

$i(X,\omega')$ az $\omega'$ irányból $X$ pontba eső sugársűrűség (másképpen radiancia) $[\frac{W}{m^2 sr}]$. Az $r(X,\omega,\omega')$ szorzótényező határozza tehát meg, hogy a beérkező fény milyen mértékben szóródik a kamera irányába. Erre a szakirodalom Bidirectional Reflection Distribution Function (BRDF) néven hivatkozik. Ennek megvalósítása rengeteg különböző módon történhet. A BRDF azon részét, amely megadja, hogy $\omega'$ irányból érkező egységnyi sugárerősségnek megfelelő fény hányadrésze szóródik $\omega$ irányba, fázis függvénynek nevezzük. A 2.22 a Henyey-Greenstein fázisfüggvény [11] egy klasszikus formula.

$$p(\omega,\omega') = \frac{1}{4\pi} \cdot \frac{1-c^2}{(1+c^2-2c\vec{\omega}\cdot\vec{\omega}')^{3/2}} \tag{2.22}$$

Itt $c$ egy $-1$ és 1 közti konstans, amely a szóródás excentricitása. Pozitív értékre a beeső fény nagyobb része a beeső fény irányával megegyező irányban halad tovább. Negatív értékre a beeső fény nagyobb része a beeső fény irányával ellentétes irányba verődik vissza. Zérus értékre a szóródás izotropikusan szóródik minden irányba $\frac{1}{4\pi}$ valószínűséggel, egyenletes eloszlással. A 2.22 fázisfüggvényt Cornette és Shanks [7] tökéletesítette tovább a 2.23 alakra.

$$p(\omega,\omega') = \frac{3(1-c^2)}{2(2+c^2)} \cdot \frac{1+(\vec{\omega}\cdot\vec{\omega}')^2}{(1+c^2-2c\vec{\omega}\cdot\vec{\omega}')^{3/2}} \tag{2.23}$$

Blinn [3] munkájában a 2.24 kifejezést alkalmazza. Ez a közeg részecskéit a fény hullámhosszánál nagyobb átmérőjű gömbszerű testeknek tekinti. Ezeken a gömbfelületeken a Lambert törvénynek megfelelően diffúz szóródik a fény.

$$p(\omega,\omega') = \frac{8\pi}{3}(|\vec{\omega}\times\vec{\omega}'| + (\pi - \arccos(\vec{\omega}\cdot\vec{\omega}'))\vec{\omega}\cdot\vec{\omega}') \tag{2.24}$$

Inkrementális képszintézis során a poligonhálóval közelített felületek árnyalásához felhasználjuk a felületi normálvektort. Ez a felület egy pontját jellemző egység hosszúságú vektor, amely merőleges a felület érintővektoraira. Másképpen két – egymással nem párhuzamos – érintővektor keresztszorzata. A térfogat felületének direkt rekonstrukciója nélkül ezek a vektorok nem állnak rendelkezésünkre. A normálvektorok becsélésére azonban felhasználhatjuk a térfogat gradiensvektorait. A térfogat gradiensvektorának komponenseit a térfogat koordinátarendszer bázisvektoraival párhuzamos differenciálásával kapjuk. A gradiens jelölésére a $\nabla$ (nabla) szimbólumot használjuk. Legyen $\vec{i}$, $\vec{j}$ és $\vec{k}$ a $V$ vektortér három bázisvektora, amelyek az $x$, $y$ és $z$ irányba mutatnak. Ekkor az $f : V \to \mathbb{R}$ függvény gradiensét a 2.25 alakban fejezhetjük ki.

$$\nabla f = \frac{df}{dx}\vec{i} + \frac{df}{dy}\vec{j} + \frac{df}{dz}\vec{k} \tag{2.25}$$

Abból megfontolásból kiindulva, hogy felületek olyan közeghatárok, amelyek mentén a térfogati adatok ugrásszerűen változnak, használhatjuk a következő normálvektor fogalmat.

$$\vec{n} \simeq -\frac{\nabla f}{|\nabla f|} \tag{2.26}$$

A negatív előjel annak köszönhető, hogy általában a környezet optikailag ritkább, mint a benne levő objektum, aminek a felületét vizsgáljuk. Így a felület mentén a gradiensek az objektum belsejébe mutatnak. A felületi normál konvencionálisan az objektumból kifelé mutat. A felülettől távol a gradiensvektor rövid, vagy zérus hosszúságú. A hosszinformációt figyelembe véve interpolálhatunk egy normálvektort alkalmazó lokális árnyalási modell és a korábban bemutatott – vagy azokhoz hasonló – szóródási modell közt.

Árnyalásra a klasszikus Blinn-Phong egyenlet kiválóan alkalmazható. Blinn [2] publikációjában kiegészíti Phong [30] korábbi függvényét. Ez a diffúz, enyhén csillogó felületekről való fényvisszaverődésnek egy durva közelítését adja.

$$d = max(0, \vec{n} \cdot \vec{\omega}') \tag{2.27}$$

$$s = (\vec{n} \cdot \vec{h})^{c_s} \tag{2.28}$$

$$i(\omega, \omega') = p_a + dp_d + sp_s \tag{2.29}$$

Itt $i$ a kamera irányába verődő fény radianciája. $d$ a diffúz visszaverődés. $\vec{h} = \frac{\vec{\omega}+\vec{\omega}'}{|\vec{\omega}+\vec{\omega}'|}$ az úgynevezett *halfway vector*, ami a nézeti irány és a fényforrás iránya közti szöget felezi. $s$ spekuláris visszaverődés. $p_s$ az anyagra jellemző spekuláris visszaverődés mértéke. $c_s$ a felület csillogásának mértéke.

A 2.29 árnyaló függvény jellemzően elmosott spekuláris fényfoltokat eredményez. Ez azonban előnyös lehet a gradiensekből rekonstruált normálvektorok használata mellett. Ilyenkor ugyanis a rendelkezésre álló normálok gyakran zajosak. Az irányuk szabálytalan ugrásokkal változik. Ennek mértéke a megjelenített objektum felületének érdességével változik. A Blinn-Phong modell eredményezte elmosott fényfoltok segítenek szűrni az egyébként zajos normálvektorokat.

Másfelől megközelítve, egy összetettebb modell alkalmazása redundáns lehet. Physically Based Rendering (PBR) [29] árnyalási modellek mikroszkopikus tükröző felületek sokaságaként modellezik a makroszkopikus felületet. Torrance és Sparrow [6] publikációjukban levezetnek egy függvényt, amely fizikailag megalapozott, realisztikus látványt biztosít a felületeknek. Cikkükben többek közt felhasználnak egy $G$ geometriai attenuációs faktort, amely modellezi a felület tökéletlenségei miatt megjelenő önárnyékoló jelenséget. Ezek a tökéletlenségek a gradiensekből előállított zajos normálvektorok esetén azonban implicit jelennek meg.

## 2.2. Sugárkövetés

A térfogati adatok megjelenítésének talán a legkézenfekvőbb megközelítése sugárkövetés. A módszer lényege, hogy minden képpontnak megfeleltetünk egy-egy sugarat, amely a kamerából indul és keresztül halad a megjelenítendő közegen. A sugár mentén lépkedve mintavételezünk a térbeli adathalmazból. Az olvasott adatok alapján kiszámolhatjuk, hogy egy adott ponton mennyi járulékos fény verődik a kamera irányába, és mennyi fény nyelődik el a közegben. Előnye, a flexibilitásban rejlik. A korszerű GPU-k segítségével hatékonyan implementálható. A számításigény csökkentése érdekében a sugarak menti lépkedést korán megállíthatjuk [23] és az üres térrészeket átugorhatjuk [40]. Nincs megkötés a fényforrások számára nézve. A 2.3.1. alfejezetből ki fog derülni, hogy az utóbbi állítás nem mondható el a half-angle slicing módszerről, ahol csak egy virtuális fényforrás lehetséges.

A 2.1. fejezetben áttekintettük a rendelkezésre álló matematikai eszközöket. A 2.1.4. alfejezetben azt is megvizsgáltuk, hogyan közelíthetőek a szükséges integrálok Riemann összeg segítségével. Eljutottunk a 2.20 kifejezésig, amely már könnyen implementálható iteratív eljárásként. Ehhez vizsgáljuk meg még egyszer a korábban levezett képletet. Ezúttal alkalmazzuk a $t = 1 - \alpha$ helyettesítést.

$$I = I_0 \prod_{i=1}^{n}(1 - \alpha_i) + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} (1 - \alpha_j) \tag{2.30}$$

### 2.2.1. Iteratív csillapítás számítás

Az $\prod_{i=1}^{n}(1 - \alpha_i)$ tényező azt fejezi ki, hogy háttér mennyire látszik a közegen keresztül. Ennek együtthatóját az iteráció során szeretnénk előállítani a korábbi részeredményekből, hogy az iteráció befejeztével a GPU által hardveresen támogatott *alpha blending* segítségével kompozitálhassuk a háttérről korábban született és az iterációval újonnan előállított, térfogatról készült képet. Konvencionálisan egy áttetsző kép esetén a kép $\alpha$ csatornájában a kép „átlátszatlanságát" tároljuk. Ezt a szokást fenntartva az sugárkövetés során a 2.31 értéket tároljuk az alfa csatornában, majd ezt a kompozitálás során a 2.30 kifejezésben $I = I_0(1 - A_n) + ...$ alakban alkalmazzuk.

$$A_n = 1 - \prod_{i=1}^{n}(1 - \alpha_i) \tag{2.31}$$

Az iteráció során az aktuális részeredményt az előzőből állítjuk elő. Az $\alpha_1 := A_1$ a közeg határáig összegyűjtött csillapítás. Gyakorlatban lehet $A_1 \simeq 0$. Fejezzük ki $A_2$-t!

$$A_2 = 1 - (1 - \alpha_1)(1 - \alpha_2) \tag{2.32}$$

A 2.32 kifejezést rendezzük.

$$\begin{aligned} A_2 &= 1 - (1 - \alpha_2 - \alpha_1 + \alpha_1\alpha_2) \\ &\downarrow \\ A_2 &= \alpha_1(1 - \alpha_2) + \alpha_2 \end{aligned} \tag{2.33}$$

Teljes indukcióval azt kell belátni, hogy $A_{n+1}$ értékét $A_n$-ből a 2.33 kifejezéssel analóg módon számolhatjuk. Ehhez először tegyük fel, hogy $A_n$-re teljesül a 2.34 összefüggés!

$$A_n = 1 - \prod_{i=1}^{n}(1-\alpha_i) = A_{n-1}(1-\alpha_n) + \alpha_n \tag{2.34}$$

Ekkor vizsgáljuk az $A_{n+1}$-et!

$$\begin{aligned}
1 - \prod_{i=1}^{n+1}(1-\alpha_i) &\stackrel{?}{=} A_n(1-\alpha_{n+1}) + \alpha_{n+1} \\
&\downarrow \\
1 - (1-\alpha_{n+1})\prod_{i=1}^{n}(1-\alpha_i) &\stackrel{?}{=} \left[1 - \prod_{i=1}^{n}(1-\alpha_i)\right](1-\alpha_{n+1}) + \alpha_{n+1} \\
&\downarrow \\
1 - (1-\alpha_{n+1})\prod_{i=1}^{n}(1-\alpha_i) &\stackrel{?}{=} (1-\alpha_{n+1}) - (1-\alpha_{n+1})\prod_{i=1}^{n}(1-\alpha_i) + \alpha_{n+1} \\
&\downarrow \\
1 - (1-\alpha_{n+1})\prod_{i=1}^{n}(1-\alpha_i) &= 1 - (1-\alpha_{n+1})\prod_{i=1}^{n}(1-\alpha_i) \checkmark
\end{aligned} \tag{2.35}$$

A 2.35 levezetéssel bebizonyítottuk, hogy $A_{n+1}$ számolható $A_n$-ből a 2.34 összefüggéssel. Ezt *over* operátornak nevezzük. Vegyük észre, hogy a 2.32 alakban felcserélhető $\alpha_1$ és $\alpha_2$. Ennek megfelelően a rekurzív képzési szabály levezethető a 2.36 alakban is.

$$A_n = A_{n-1} + \alpha_n(1 - A_{n-1}) \tag{2.36}$$

Ez az *under* operátor. Akkor alkalmazzuk, amikor a közeg kamerához közeli oldalán kezdjük a sugárkövetést, és folyamatosan távolodunk a kamerától.

### 2.2.2. Iteratív sugárerősség számítás

Az előző alfejezetben megvizsgáltuk, hogyan lehet iteratívan előállítani az $A_n$ csillapítást. Ez határozza meg, hogy a megjelenített térfogat mennyire takarja el a hátterét. Most a 2.30 kifejezés azon tagjával foglalkozunk, amely a közeg tényleges látványát szolgáltatja. Ezt a gyakorlatban párhuzamosan számoljuk a vörös, zöld és kék színcsatornákra, és ennek megfelelően a sugárkövetés során előállított kép $RGB$ csatornáiban tároljuk. A továbbiakban a 2.30 kifejezés ide vonatkozó részét $C_n$-el jelöljük.

$$C_n = \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n}(1-\alpha_j) \tag{2.37}$$

Fejtsük ki a szorzatot az első néhány $n$-re!

$$\begin{aligned} C_1 &= g_1 \\ C_2 &= g1(1-\alpha_2) + g_2 \\ C_3 &= (g1(1-\alpha_2) + g_2)(1-\alpha_3) + g_3 \\ &\dots \end{aligned}$$

Bizonyítsuk teljes indukcióval, hogy a 2.38 rekurzív képzési szabály tetszőleges $n$-re igaz.

$$C_n = C_{n-1}(1-\alpha_n) + g_n \tag{2.38}$$

Az első néhány elemre már láttuk, hogy megfelelnek a képzési szabálynak. Tegyük fel, hogy $n$-re igaz 2.38 azonosság. Vizsgáljuk $n+1$-et!

$$\begin{gathered} \sum_{i=1}^{n+1} g_i \prod_{j=i+1}^{n+1} (1-\alpha_j) \stackrel{?}{=} C_n(1-\alpha_{n+1}) + g_{n+1} \\ \downarrow \\ (1-\alpha_{n+1}) \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} (1-\alpha_j) + g_{n+1} \prod_{j=n+2}^{n+1} (1-\alpha_j) \stackrel{?}{=} C_n(1-\alpha_{n+1}) + g_{n+1} \\ \downarrow \\ C_n(1-\alpha_{n+1}) + g_{n+1} = C_n(1-\alpha_{n+1}) + g_{n+1} \checkmark \end{gathered} \tag{2.39}$$

A 2.39 levezetéssel befejeztük a 2.38 összefüggés bizonyítását tetszőleges $n$-re.

A most levezetett 2.38 kifejezés az *over* operátor a szín csatornákra. Ez abban az esetben érvényes, amikor a sugarakat a térfogat kamerától távoli oldalán kezdjük bejárni. Lépésenként haladunk a kamera irányába, miközben az egyes lépéseken gyűjtött sugárerősség egyre kevesebb csillapítást szenved. Fel lehet írni az összefüggéseket az ellenkező esetre is. Ilyenkor a térfogat kamerához közeli oldalán indulunk, és lépésekkel távolodunk a kamerától. Ehhez a 2.37 kifejezést kell megváltoztatni úgy, hogy $\prod_{j=i+1}^{n}(1-\alpha_j)$ helyett $\prod_{j=1}^{i-1}(1-\alpha_j)$-t írunk. Ebből teljes indukcióval lehet bebizonyítani a 2.40 *under* operátor helyességét.

$$C_n = C_{n-1} + g_n(1-A_{n-1}) \tag{2.40}$$

## 2.3. Textúra szeletelés

Textúra szeletelés során a megjelenítendő térfogatot párhuzamos síkokkal metsszük el. Hogyha fenntartjuk azt a munkahipotézist, hogy a megjelenítendő térfogat egy befoglaló téglatestben adott, akkor ezt tetszőleges síkkal elmetszve egy legkevesebb három- és legtöbb hatszöget kapunk. Ezeket a sík alakzatokat a továbbiakban *proxy geometriának* nevezzük. A proxy geometria pontjaiban mintavételezünk, és a számolt csillapítást és sugárerősséget egy textúrába írjuk. Az egymást követő síkokban előállított képet a 2.2.1. és 2.2.2. alfejezetekben bemutatott *over* és *under* operátorokkal kombináljuk. Végül az *over* operátor ismételt alkalmazásával kombináljuk a háttér és térfogat képét.

Az egyik felmerülő kérdés, hogy a metsző síkokat milyen tengelyre merőlegesen helyezzük el. Hogyha a térfogati adathalmaz eleve rétegenként adott – 2D textúrák sorozataként –, akkor adódik a lehetőség, hogy a mintavételező síkok legyenek párhuzamosak az

adathalmaz rétegeivel. Ez azonban önmagában nem elégséges megoldás, mivel elképzelhető olyan nézeti irány, amikor valamelyik proxy geometria vetülete a képsíkon egy szakasz. Így az iteráció során lehetetlen lenne előállítani a képet, noha alkalmasan választott szeletelő tengellyel ennek nem lenne akadálya. Erre egy megoldás, ha három különböző szeletelést végzünk a koordináta-rendszer három bázisvektorával párhuzamos tengely mentén. A megoldás előnye, hogy feltételezve, hogy a térfogat befoglaló téglatestének élei a megfelelő koordináta tengelyekkel párhuzamosak, a proxy geometriák mind téglalapok lesznek. Ezeket a proxy geometriákat elegendő egyszer előállítani. Az előfeldolgozás során Rezk-Salama és társai [32] tökéletesítették az alapötletet Lacroute és társai [19] és mások publikációira támaszkodva. Cikkükben az eljárás számításigényének csökkentésére és a megjelenített kép minőségének javítására koncentrálnak. Megjelenítés során a három különböző szeletelés közül azt választják, amely tengelye a legkisebb szöget zárja be a nézeti iránnyal. Ez a technika a sugárkövetéshez képest torzított megjelenést nyújt. A vizuális pontatlanságot a különböző irányú szeletelések közti váltás eredményezi. Krüger [17] cikkében bemutat egy eljárást, amely a tengelyekkel párhuzamos szeletelést még hatékonyabban és jóval kevésbé torzító módon valósítja meg. Módszerük lényege az a felismerés, hogy sugárkövetés során, hogyha a mintavételezést a térfogat határán indítjuk, majd azonos delta lépésekkel haladunk a sugarak mentén, akkor a mintavételezett pontok a befoglaló téglatest alakját követik. Hogyha alkotunk egy felületet, hogy az az összes sugár mentén az $n$. lépésekor elért pontokat tartalmazza, akkor az a felület közelítőleg olyan, mintha a befoglaló téglatest kamera felé néző oldalait az oldalakra merőlegesen betoltuk volna a téglatestbe. Ez a megfontolás motiválta módszerük, amely a három különböző tengelyre merőleges szeletelést egyszerre alkalmazza. A három különböző irányú proxy geometriát elvágjuk megfelelően választott síkokkal. A vágósíkok a befoglaló geometria kamera irányában levő éleit és a kamera pozícióját tartalmazzák. Az így kialakuló proxy geometria front jól közelíti a korábban említett, sugárkövetés során előálló frontot. Ezen kívül a különböző irányú levágott proxy geometriák nem fognak fedésbe kerülni. A cikk felhívja a figyelmet arra, hogy az imént leírt eljárással kamera pozíciójából indított összes sugár mentén nem egyenlő a távolság az egyes proxy geometria rétegek közt. Ennek korrigálására a fény számításokban megszorozzák a szomszédos proxy geometriák közti távolságot a proxy geometriák normál vektora és a kamerából érkező sugár által bezárt szög szekánsával.

$$\Delta = sec(\theta)d = \frac{d}{cos(\theta)} \tag{2.41}$$

Ezt a $\Delta$ értéket használják az integrálok Riemann összeggel való közelítésekor, mint diszkrét lépéshossz.

### 2.3.1. Áttetsző közeg megjelenítése half-angle slicing eljárással

A textúra szeletelés egy másik formája, amikor a proxy geometriák elhelyezését a nézeti iránytól teszik függővé. Ennek egyszerű formája, amikor a proxy geometriákra merőleges tengely párhuzamos a nézeti iránnyal. Ez a megoldás azonban nem nyújt igazán jelentés előnyt az előzőekben bemutatott, koordináta-tengelyekre merőleges szeleteléshez képest. A tetszőleges irányú szeletek miatt a 3D textúrák címzése drága művelet. Előnye, hogy nem kell erőfeszítést tenni a különböző irányú szeletelések közti váltás okozta vizuális inkonzisztenciák elfedésére.

Kniss és társai [15] cikkükben *half-angle slicing* eljárást alkalmaznak, amely a korábbi megközelítésekkel ellentétben részletes árnyék számítást tesz lehetővé interaktív frissítési ráta mellett. Megközelítésük empirikus alapokon nyugszik. Az általános *light transport* modell egy gyakorlatban jó eredményeket nyújtó közelítését adják. Munkájuk központjában áttetsző, inhomogén közegek valósághű megjelenítése áll. Kiegészítik a korábban

bemutatott 2.18 függvényt egy árnyékot leíró tényezővel.

$$I(s) = I_0 exp\left(-\int_0^s \tau(t)dt\right) + \int_0^s g(t) exp\left(-\int_t^s \tau(u)du\right) I_l(t)dt \tag{2.42}$$

$$I_l(t) = I_l(0) exp\left(-\int_t^l \tau(u)du\right) \tag{2.43}$$

$I_l(t)$ a közegbeli pontból a fényforrás irányába indított sugáron összegyűjtött csillapítás. A cikkben bevezetnek egy elkent indirekt fény kontribúciót leíró tagot. Ez közelíti a közegben fellépő szóródást.

$$\begin{aligned} I(s) &= I_0 exp\left(-\int_0^s \tau(t)dt\right) + \int_0^s C(t) exp\left(-\int_t^s \tau(u)du\right) I_l(t)dt \\ C(t) &= g(t)((1 - S(s)) + f_s(t)S(t)) \\ I_l(t) &= I_l(0) exp\left(-\int_t^l \tau(u)du\right) + I_l(0) exp\left(-\int_t^l \tau_i(u)du\right) Blur(\theta) \end{aligned} \tag{2.44}$$

Itt $\tau_i(t)$ egy indirekt kioltási tényező. $C(t)$ a $t$ mintához tartozó reflektív szín. $g(t)$ a közeg emissziója. $f_s(t)$ a Blinn-Phong árnyalás értéke, amit normalizált gradiensvektor segítségével állítunk elő. $S(t)$ a felület árnyalási paramétere. $I_l(t)$ a direkt és indirekt fény kontribúció összege. A fény szóródását az indirekt tag blúrozásával közelítjük. Ez egy fényforrás irányába nyíló, $\theta$ fél nyílásszögű kúpon belüli átlagolással valósítható meg. Az $S(t)$ nulla és egy közti érték, amely meghatározza a felületi árnyalás mértékét. Egy korábbi publikációban Kniss és társai [14] már szót ejtettek a gradiensvektor hosszának felhasználásáról az $S(t)$ meghatározására. Ezt a megoldást a felületekhez tartozó nagyobb gradiens érték motiválta. Annak érdekében, hogy a közegben szóródó fény elszíneződését modellezzük, az indirekt fény csillapítását egy spektrális $\alpha_i$ érték írja le. Az $RGB$ csatornákra külön adjuk meg a csillapítást.

Az eddig leírt számítások elvégzéséhez nem csak a kamerából kiinduló sugarak mentén kell iterálni, hanem a sugár menti pontokból a fényforrás irányába is. Ez olyan mennyiségű mintavételezést jelentene, amely nem megengedhető valósidejű megjelenítés során. Kniss és társai [14] half-angle slicing eljárást alkalmaznak. Ez lehetővé teszi, hogy kihasználjuk a GPU párhuzamosító képességét és a két irányban az iterációt valós időben elvégezzük. A proxy geometriák normálvektora a közeg középpontjából a nézeti irány és a fényforrás iránya által bezárt szöget felezi. Abban az esetben, ha a nézeti irány és a fényforrás iránya tompaszöget zár be, a normálvektor a nézeti irány negáltja és a fényforrás iránya által bezárt szöget felezi. Így a szeletelés végezhető a nézeti kamera és a fényforrásba helyezett kamera perspektívájából is. A szeletelés sorrendjét mindig úgy választjuk, hogy a fényforrás felől nézve *front to back* sorrendben haladjunk. Ez azt jelenti, hogy a nézeti irányból projektálva a szeletelés sorrendje *front to back*, ha a nézeti irány és a fényforrás irányvektora hegyesszöget zár be. *back to front*, ha a két vektor tompaszöget zár be. A proxy geometriák ilyen elhelyezése és a helyes sorrendű iteráció biztosítja, hogy az aktuálisan feldolgozott szelet egy tetszőleges pontjának kiértékelésekor már korábban ki lettek értékelve azok a pontok, amelyek a direkt és indirekt fény kontribúció számításához szükségesek. Három kép buffert alkalmazunk. Ezek közül kettőt a fényforrás irányában adódó csillapítás tárolására használunk. A harmadik buffer kamera irányában adódó színt tárolja. Az iteráció során először a kamera szemszögéből rajzoljuk ki a soron következő proxy geometriát. A képpont árnyaló programban kiértékeljük a átviteli függvényt és elvégezzük a Blinn-Phong árnyalást. Az adódó $C(t)$ értéket beszorozzuk az egy mínusz direkt csillapítás és indirekt csillapítás összegével. Ezeket a csillapítás értékeket az erre a célra fenntartott két buffer közül az aktuálisan olvasásra kijelöltből olvassuk. Az olvasás címe az a textúra pozíció,

ahová a proxy geometria aktuálisan feldolgozott pontja vetítődött a fényforrás irányából nézve. A képpontárnyaló kimenetét *front to back* sorrend esetén az *under*, *back to front* sorrend esetén az *over* operátorral vegyítjük az iteráció során korábban előállított képbe a szín bufferben. Egy proxy geometria feldolgozása itt nem ér véget.

Miután a szín buffert frissítettük, másodszor is kirajzoljuk ugyanezt a geometriát. Ezúttal a fényforrás irányába vetítve. Itt egy olyan képpontárnyalót alkalmazunk, amely kimenete nem egy szín, hanem a $\vec{a} = (\alpha_{iR}, \alpha_{iG}, \alpha_{iB}, \alpha)$ vektor. A vektor az indirekt csillapítást a három színcsatornára és az egy direkt csillapítás tartalmazza. A csillapítást az átviteli függvény határozza meg. A direkt csillapítást az *under* operátorral kombináljuk a korábbi rétegekben felhalmozott csillapítással, kihasználva a GPU *blend* funkcióját. Az indirekt csillapítást az *under* operátorral több korábbi érték konvolúciójával kombináljuk. Ezt a GPU már nem tudja megvalósítani egyszerű csatorna keveréssel. Ezért van szükség két csillapítást tároló bufferre. Amíg az egyik a képpontárnyaló kimenetéül szolgál, addig a másikat az árnyaló program bemeneteként használjuk, és textúraként olvassuk. Az olvasott bufferből az aktuálisan számolt pozíció vetülete körüli $offset \leq d\tan(\frac{\theta}{2})$ környezetben mintavételezünk. Itt $d$ a távolság két proxy geometria közt. Kniss és társai négy mintát használtak. Az offset irányát és nagyságát randomizálták az egyes olvasásokhoz, hogy maszkolják az alulmintavételezés okozta mintázatokat. Az olvasott mintákat Gauss-szűrővel konvolválják. Ezt a blúrozott értéket az *under* operátorral kombinálják az átviteli függvénnyel meghatározott indirekt csillapítással. A következő rajzoláskor szerepet cserél a két buffer.

Az referált cikkben a szerzők nem térnek ki a arra, hogy ugyan a rétegek közti normálirányú távolság az iteráció során konstans, a perspektív vetítés miatt a sugarak mentén két proxy geometria közt eltérő a távolság. Ennek orvoslására alkalmazhatjuk a 2.41 formulát. További nem említett részlet, hogy a távolabbi rétegekben ugyanazt a távolságot kisebb szög alatt látjuk. A indirekt csillapítás konvolúciója során textúra olvasásnál használt offseteket a textúra normalizált koordináta-rendszerében adjuk meg. Ehhez azonban a távolabbi rétegek esetén nagyobb tényleges $\theta'$ fél nyílásszög társul. Hogyha szeretnénk, hogy a fény szóródása konzisztens $\theta$ szögben történjen a rétegek közt, akkor az offsetelt pontokat először a világ koordináta-rendszerben kell megadni. Ezeket vetíteni kell a fényforrásnál elhelyezett képsíkra. Az így előállított normalizált eszköz koordináta-rendszerben értelmezett koordinátákat skálázás és eltolás egymásutánjával alakítjuk textúra koordinátákká. Az ezekkel járó többlet számítások azonban nem feltétlenül térülnek meg. Csak textúra koordináta-rendszerben számolni még előnyösnek is tekinthető. Empirikus úton úgy érződik, hogy nagyobb $\theta'$ szögek azt a jelenséget közelítik, hogy a közeg fényforráshoz közeli oldalán még kevésbé szórt a belépő fény. A közegen áthaladva egyre nagyobb szögben érkezik a fény egy adott pontra.

## 2.4. Befoglaló geometria optimalizálása

A megjelenítendő térfogat egy befoglaló téglatestben helyezkedik el. Az esetek többségében ennek a térfogatnak a jelentős része optikailag üres. Vegyünk például egy emberi fejről készített CT felvételt, amely formáját inkább egy kapszula formában lehetne optimálisan tárolni. A befoglaló téglatestben mintavételezve rengeteg olyan mintát értékelünk ki, amely összesen annyi információt nyújt, hogy ott nincs lényeges adat. Adódik tehát az ötlet, hogy az üres térrészeket ugorjuk át [22]. Egy lehetséges megközelítés, hogy a teret uniform téglatestekre osztjuk. A téglatestekben kiszámoljuk az optikai kitöltöttséget az adathalmazt mintavételezve, ás az átvitei függvényt kiértékelve. Választunk egy kis $\epsilon$ küszöbértéket, amely feletti kitöltöttséggel rendelkező téglatesteket már meg szeretnénk jeleníteni. Ennek megválasztása kritikus pontja a optimalizáció működésének, mivel túl kis

értékek esetén számunkra haszontalan, zajjal terhelt térrészek is kiválasztódnak. Túl magas küszöbérték az olyan térrészeket is el fogja vetni, amelyek egyébként lényeges részét tartalmazzák a halmaznak. Ez különösen jellemző a térfogatban levő objektum felülete mentén. Itt gyakran a térfogat alig lóg bele egy téglatestbe. Így annak kitöltöttsége $\epsilon$ alatti lesz és eldobásra kerül.

A kiválasztott téglatestekből összeállítunk egy poligonhálót. Szeretnénk ismerni, hogy a kamerából indított sugarak hol lépnek be a közeget körülhatároló geometriába és hol hagyják el azt. A poligonhálót kétszer rajzoljuk ki két különböző bufferbe. Az első kirajzoláskor a GPU mélység tesztelő függvényét úgy állítjuk be, hogy a kamerához közelebbi oldalak felülírják a távoliakat. A mélység buffert rajzolás előtt a maximális távolságot jelentő, egyes értékekkel inicializáljuk. A második kirajzoláskor a mélység tesztelő függvényt úgy állítjuk be, hogy a kamerától távoli oldalak felülírják a közelieket. Ezt megelőzően a mélység buffert nullákkal inicializáljuk. Az első rajzolás során a szín csatornákban az adott képponthoz tartozó sugár geometriába való belépő pontját tároljuk el. A második rajzolás során a szín csatornákban az adott képponthoz tartozó sugár kilépő pontját tároljuk el. A nézeti kamera elmozdulásakor újra kell rajzolni a belépő és kilépő pontokat.

A térfogat rajzolásakor a sugárkövetést a belépő ponttól indítjuk, és a kilépő pontig haladunk. Annak érdekében, hogy akkor is értelmes pozíciók legyenek a belépési pozíciót tároló bufferben, ha a kamera a befoglaló geometria belselyében helyezkedik el, a belépési pontokat tároló buffert előzőleg a kamera aktuális pozíciójával inicializáljuk. Így ilyenkor rögtön a kamerából indul a sugárkövetés. Azokban a pontokban, ahová nem projektálódott befoglaló geometria, egyáltalán nem szeretnénk mintavételezni, mivel ezeket korábban optikailag üres területeknek nyilvánítottuk. Ennek érdekében a kilépési pontokat tároló buffert a pozíciók írása előtt szintén a kamera aktuális pozíciójával inicializálunk. Ekkor arról ismerjük fel azokat a képpontokat, ahol nem kell sugárkövetést végeznünk, hogy a sugár belépési és kilépési pontja megegyezik.

## 2.5. Átviteli függvény kialakítása

Az átviteli függvény (*transfer function*) meghatározza, hogy a térfogat egy adott pontján milyen szín és csillapítás értékek hatályosak [39, 26, 20].

A legegyszerűbb átviteli függvények egyváltozósak. A térfogati adathalmazból kiolvasott intenzitás értékhez rendelik a kimenetet. Az egydimenziós átviteli függvények hátránya, hogy nem tudnak megkülönböztetni azonos intenzitású mintákat. Ezen javítanak a kétváltozós átviteli függvények, amelyek az intenzitás mellett az adott pontban számított gradiens hosszát is figyelembe veszik.

$$f : (I(x), \|\nabla I(x)\|) \rightarrow (c_r, c_g, c_b, \alpha) \tag{2.45}$$

Így lehetőség nyílik az azonos intenzitású, de eltérő gradiens hosszú pontok színnel és csillapítással való megkülönböztetésére. A egy és kétdimenziós átviteli függvények hatékonyan implementálhatók textúrák segítségével. A textúrát egy *lookup* táblaként értelmezzük. A textúra koordináta-rendszer $x$ tengelyét az intenzitásnak, az $y$ tengelyt a gradiens hossznak feleltethetjük meg. Egy előfeldolgozó lépésként kiértékeljük a függvényt minden intenzitás és gradiens hosszra, és a kimenetet a textúra megfelelő pozíciójába írjuk. Ennek előnye, hogy innentől kezdve nincs szükség egy zárt alakú függvényre. Készíthetünk úgy átviteli függvényt, hogy kiszínezünk egy textúrát. Úgy ítéljük azonban, hogy továbbra is könnyebb függvényekről beszélni.

Az átviteli függvényt szeretnénk úgy definiálni, hogy az adathalmaz számunkra érdekes területeit emelje ki. Célunk, hogy a különböző struktúrákat eltérő színnel és csillapítással jelenítse meg, és a lényeges struktúrák környezete legyen átlátszó. A függvény

tervezése a gyakorlatban egy nehéz, iteratív folyamat. Kniss és társai [14] számos többdimenziós átviteli függvény tervezését segítő eljárást vizsgálnak.

Vannak eredmények az automatizált átviteli függvény tervezés területén is. Roettger és társai [33] egy klasszifikáló algoritmust javasolnak a kétváltozós átviteli függvények automatikus előállítására. Módszerük osztályokba sorolja a különböző intenzitás és gradiens hossz párokat. Munkájukban megfogalmazzák azt a megfigyelést, hogy a kétváltozós átviteli függvényekben az azonos struktúrákhoz tartozó intenzitás-gradiens értékpárok konvex ívekként jelennek meg. Ezek az ívek gyakran átfedésben állnak. Hogyha az intenzitás-gradiens hossz kétdimenziós hisztogramon átfedés van a struktúrák közt – egy intenzitás-gradiens páros több struktúrához sorolható–, akkor nem lehet egyértelműen eldönteni az átfedésben levő értékekről, hogy melyik struktúrához tartoznak. A cikkben prezentált módszer lényege, hogy mivel a hisztogram alapú klasszifikáció gyakran nem egyértelmű, ezért érdemes a intenzitás és gradiens hosszal rendelkező értékek térbeli elhelyezkedését is figyelembe venni. Megoldásukra Spatial Transfer Function (STF) névvel hivatkoznak. A csillapítás meghatározására a 2.46 átviteli függvényt javasolják, de összetettebb függvények is alkalmazhatóak.

$$f(I(x), \|\nabla I(x)\|) = \|\nabla I(x)\| c_\alpha \tag{2.46}$$

Itt $c_\alpha$ egy globális paraméter. Ez után következik a színek meghatározása. Szeretnénk elérni, hogy a térben egymáshoz közeli pontok intenzitás és gradiens értékéhez azonos szín rendelődjön. Legyen $H(I, \nabla) = n$ a térfogat hisztogramja, ahol az adott $I$ intenzitás és $\nabla$ gradiens hossz a térfogat pontosan $n$ különböző pontjához rendelődik. Legyenek $\vec{p_i}(I, \nabla)$, $i \in [1..n]$ a térfogat azon $n$ darab pontjának normalizált koordinátái, amelyek a $H(I, \nabla) = n$ hisztogramértékhez hozzáadtak. $\vec{b}(I, \nabla) = \frac{1}{n} \sum_{i=1}^{n} \vec{p_i}(I, \nabla)$ a $\vec{p_i}(I, \nabla)$ pontok baricentruma. $v(I, \nabla) = \frac{1}{n} \sum_{i=1}^{n} \|\vec{p_i}(I, \nabla) - \vec{b}(I, \nabla)\|$ a térbeli variancia. Legyen $N(T, T_0) = \|b(T) - b(T_0)\| + |v(T_0) - v(T)|$ a távolság norma. Definiáljunk egy $r$ maximális sugarat, amely alatti $N(T, T_0)$ esetén a $T$ és $T_0$ intenzitás-gradiens hossz párokat egy osztályba sorolhatjuk.

Az algoritmus első lépése a baricentrum és variancia kiszámítása minden pároshoz. Ezek után klasszifikáljuk a párokat. Egy-egy osztály kialakításához kiválasztunk az átviteli függvényből egy $T_0$ referencia párt. Ehhez hasonlítjuk a korábban nem klasszifikált $T$ párokat. Az $N(T, T_0) < r$ feltételt teljesítő párokat beválasztjuk az osztályba. A cikkben a legnagyobb hisztogram értékű párt választják referenciának. A különböző osztályok véletlenszerűen választott színezést kapnak. Nem szándékozunk realisztikus színeket választani. Elegendő, hogyha elkülönülnek a struktúrák a felhasználó számára. Később lehetőséget lehet adni a színek manuális változtatására.

Az elkészült osztályozás zajmentesítésére Gauss-szűrést és zárást végzünk az átviteli függvényt tároló textúrán.

## 2.6. Térfogati adathalmaz szűrése

A direkt térfogat megjelenítés során egy diszkrét felbontású rácshálóba rendezett adathalmazt mintavételezünk. Ezt az adathalmazt valamilyen képalkotó eljárással hozták létre, aminek a bemenete valamilyen fizikai valósággal bíró, analóg jel. Az objektum, amiről a felvétel készül, modellezhető egy folytonos, $f : \mathbb{R}^3 \to \mathbb{R}$ függvénnyel. Ez a függvény térbeli pozícióhoz egy skalár intenzitás értéket rendel. Azt, hogy a fizikai valóság hogyan képződik egy intenzitássá, ebben a munkában nem vizsgáljuk. Modellünk egyszerűsítése érdekében nem teszünk különbséget fizikai valóság és a megjelenítés bemenetéül szolgáló adathalmaz közt azokban a pontokban, amelyek az adathalmazban explicit megtalálhatóak. Általánosságban azonban a diszkrét rácsháló pontjai között nem tudjuk tökéletesen visszaállítani az eredeti folytonos függvényt. Annak érdekében, hogy mégis javítsuk esélyeinket a tökéletes

visszaállításra, megkötéseket tehetünk a rekonstruálni kívánt térfogatra nézve. A Harry Nyquistről elnevezett Nyquist frekvencia az a maximális frekvencia, amilyen komponenst egy folytonos jel tartalmazhat, hogy egy $f_e$ mintavételezési frekvencia mellett a mintákból tökéletesen visszaállíthassuk a jelet [35]. Ez a mintavételezési frekvencia fele.

$$F_e = \frac{f_e}{2} \tag{2.47}$$

A gyakorlatban ennek teljesülését nem követeljük meg. A jel helyreállítására is csak egy „elég jó" közelítést adó szűrőt alkalmazunk. A gyakorlatban a trilineáris interpoláció elfogadható eredményeket szokott eredményezni. Használatát indokolja az alacsony szükséges mintaigény. A korszerű GPU-k a térbeli textúrák olvasásakor hardveresen is támogatják ezt a szűrési módszert [5, 10, 18]. Az egydimenziós lineáris interpoláció emelése két dimenziós vektortérre. A trilineáris interpoláció ezek természetesen adódó emelése a $\mathbb{R}^3$ vektortérre. Mindegyik esetben lineáris függvényekkel közelítjük a az eredeti jelet. Az interpolációhoz a rácsháló azon nyolc pontjára van szükség, amelyek a közelítendő pontot tartalmazó voxel nyolc csúcspontja. Első lépésben a kocka két szemközti oldalának éllel összekötött csúcspontjai között végzünk lineáris interpolációt. Az interpoláció paramétere a helyreállítandó pont távolsága a kocka két oldalától. A következő lépésben az így előállított négy darab interpolált érték között páronként interpolálunk úgy, hogy azok kerülnek párba, amelyek olyan pontokból álltak elő, amik között a kockában páronként futott él. Az így előállított két interpolált érték között végül ismét interpolálunk. így kapjuk a helyreállítandó pont trilineáris interpolációval való közelítését. Ez egy alacsony számításigényű eljárás, ami a lineáris közelítés miatt gyenge eredményeket produkál magas frekvenciás részletek megjelenítésekor. Magasabb rendű közelítéssel kecsegtet például a triköbös B-spline görbeillesztés [25]. Hasonlóan jó illesztéstre képes a triköbös Catmull-Rom spline [13]. Ezek azonban jelentősen több mintát használnk. A GPU használatánál –ugyanúgy, ahogyan a központi CPU esetén is– a memória hozzáférések drága műveletek. A térfogati textúrák olvasása különösen erőforrás-igényes művelet. Ezért az olvasások számát lehetőség szerint szeretnénk csökkenteni. Korábban már szó esett arról, hogy az árnyaláshoz, és többdimenziós átviteli függvényhez szükség van a térfogat gradiens vektorára. Ezt a gyakorlatban centrális differenciálással állíthatjuk elő. Ez az eddig tárgyalt interpolált érték körül további hat textúra olvasást tesz szükségessé. Csébfalvi [8] munkájában rámutat a lehetőségre, hogy a centrális differenciálhoz szükséges minták felhasználhatóak a trilineáris interpoláció eredményének korrigálására. Így az eredeti függvény magasabb rendű közelítését kaphatjuk az alacsony költségű, hardveresen támogatott trilineáris interpoláció és az egyébként is szükséges differenciálás összköltségéért cserébe.

A következőkben bemutatjuk a módszert. Először foglalkozzunk az egydimenziós esettel! Szeretnénk rekonstruálni az $f(x)$ egyváltozós, folytonos függvényt. A rendelkezésre álló diszkrét felbontású adathalmaz az egész $x$ értékekre adja meg a függvény értékét. Ekkor $f(x)$ lineáris interpolációval történő közelítése az adathalmaz mintavételezésével a 2.48 alakban adható meg.

$$f_{lin}(x) = f[\lfloor x \rfloor] \cdot (1 - u) + f[\lfloor x \rfloor + 1] \cdot u \tag{2.48}$$

Itt $u = x - \lfloor x \rfloor$. A függvény első deriváltja $f_{lin}(x)$ centrális differenciálásával közelíthető [31].

$$d(x) = \frac{f_{lin}(x + 1) - f_{lin}(x - 1)}{2} \tag{2.49}$$

A 2.50 Laplace operátor a folytonos egyváltozós, skalár értékű függvények második deriváltjának általánosítása három változós, skalár értékű folytonos függvények esetére [36].

$$\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2} + \frac{\delta^2 f}{\delta z^2} \tag{2.50}$$

A 2.51 diszkrét Laplace operátor – most egyváltozós függvényekre – a folytonos függvény második deriváltjának közelítése [31].

$$s(x) = f_{lin}(x-1) - 2f_{lin}(x) + f_{lin}(x+1) \tag{2.51}$$

Negatív második derivált konvex görbére, pozitív pedig konkáv görbére utal. Hogyha az első derivált nem nulla és a második derivált nulla, akkor ez a görbe inflexiós pontja. Ezt a tudást felhasználva javítjuk az $f_{lin}(x)$ értéket, amely az adathalmazból olvasott két szomszédos értéket egy szakasszal köti össze. A korrekció értéke pozitív, ha a második derivált negatív, és negatív, ha a második derivált pozitív. A javasolt javított interpoláció egyváltozós esetben a 2.52 alakban írható le.

$$\tilde{f}(x) = f_{lin}(x) + s(x) \cdot (\alpha^2 - \alpha) \tag{2.52}$$

Itt $x = i + \alpha$, ahol $i \in \mathbb{Z}$ és $\alpha \in [0, 1)$ $x$ egész és tört része. A 2.52 egyenlet azonos eredményt ad az egyváltozós Catmull-Rom spline-nal.

Három változós esetben először kiszámoljuk a 2.53 alakban felírható trilineáris interpolációt.

$$\begin{aligned}
&f_{trilin}(x,y,z) = \\
&f[\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor] \cdot (1-u) \cdot (1-v) \cdot (1-w) + \\
&f[\lfloor x \rfloor + 1, \lfloor y \rfloor, \lfloor z \rfloor] \cdot u \cdot (1-v) \cdot (1-w) + \\
&f[\lfloor x \rfloor, \lfloor y \rfloor + 1, \lfloor z \rfloor] \cdot (1-u) \cdot v \cdot (1-w) + \\
&f[\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, \lfloor z \rfloor] \cdot u \cdot v \cdot (1-w) + \\
&f[\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor + 1] \cdot (1-u) \cdot (1-v) \cdot w + \\
&f[\lfloor x \rfloor + 1, \lfloor y \rfloor, \lfloor z \rfloor + 1] \cdot u \cdot (1-v) \cdot w + \\
&f[\lfloor x \rfloor, \lfloor y \rfloor + 1, \lfloor z \rfloor + 1] \cdot (1-u) \cdot v \cdot w + \\
&f[\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, \lfloor z \rfloor + 1] \cdot u \cdot v \cdot w
\end{aligned} \tag{2.53}$$

Itt $u = x - \lfloor x \rfloor$, $v = y - \lfloor y \rfloor$ és $w = z - \lfloor z \rfloor$. A gradiens vektor a 2.54 egyenlettel közelíthető [31].

$$\nabla f(x,y,z) = \begin{bmatrix} \frac{\delta}{\delta x} f(x,y,z) \\ \frac{\delta}{\delta y} f(x,y,z) \\ \frac{\delta}{\delta z} f(x,y,z) \end{bmatrix} \approx \begin{bmatrix} [f_{trilin}(x+1,y,z) - f_{trilin}(x-1,y,z)]/2 \\ [f_{trilin}(x,y+1,z) - f_{trilin}(x,y-1,z)]/2 \\ [f_{trilin}(x,y,z+1) - f_{trilin}(x,y,z-1)]/2 \end{bmatrix} \tag{2.54}$$

A javított interpolációt három változós esetben a 2.55 kifejezés írja le.

$$\begin{aligned}
\tilde{f}(x,y,z) = f_{trilin}(x,y,z) &+ s_x(x,y,z) \cdot (\alpha^2 - \alpha)/2 \\
&+ s_y(x,y,z) \cdot (\beta^2 - \beta)/2 \\
&+ s_z(x,y,z) \cdot (\gamma^2 - \gamma)/2
\end{aligned} \tag{2.55}$$

Itt $x = i + \alpha$, $y = j + \beta$ és $z = k + \gamma$, ahol $i, j, k \in \mathbb{Z}$ és $\alpha, \beta, \gamma \in [0, 1)$. A második deriváltat a három koordináta-tengellyel párhuzamosan alkalmazott diszkrét Laplace operátorok kombinációjával közelítjük. A 2.55 kifejezésben ugyanazt a hat további interpolált mintát alkalmazzuk, mint a 2.54 kifejezésben a gradiens centrális differenciálással való közelítésekor. Egy trilineáris interpoláció nyolc mintát használ, így összesen ötvenhat

mintával állítjuk elő $f(x, y, z)$ és $\nabla f(x, y, z)$ közelítését. Ahogyan korábban említettük, a $f_{trilin}(x, y, z)$ számítása a legtöbb korszerű GPU esetén hardveresen támogatott. Ennek fényében – a megfelelő beállítások elvégezte után – elegendő hét darab textúra hozzáférést kezdeményezni a képpont árnyalóban egy pont szűréséhez.

# 3. fejezet

# Szoftver architektúra

Az vizualizációs algoritmusok önmagukban nem elegendőek egy teljes értékű megjelenítő alkalmazás létrejöttéhez. Munkánk során fontos célnak tartottuk egy fenntartható és stabil rendszer kialakítását. Ennek elérése érdekében egy általános célú grafikus megjelenítő platformot implementáltunk, amely megadja a lehetőséget térfogati adatok direkt megjelenítésére, de ezen kívül poligonhálóval adott geometriák inkrementális megjelenítését is lehetővé teszi. Lehetőséget nyújt szöveg megjelenítésre és testre szabott menürendszer építésére. Kezeli a felhasználótól származó billentyűzet és egyéb perifériákon keresztül érkező bemenetet. Alkalmas környezetet biztosít akár dinamikus, időben változó jelenetek szimulációjához és ábrázolásához. A kép utófeldolgozása során különböző effektusokkal javítja a látványt.

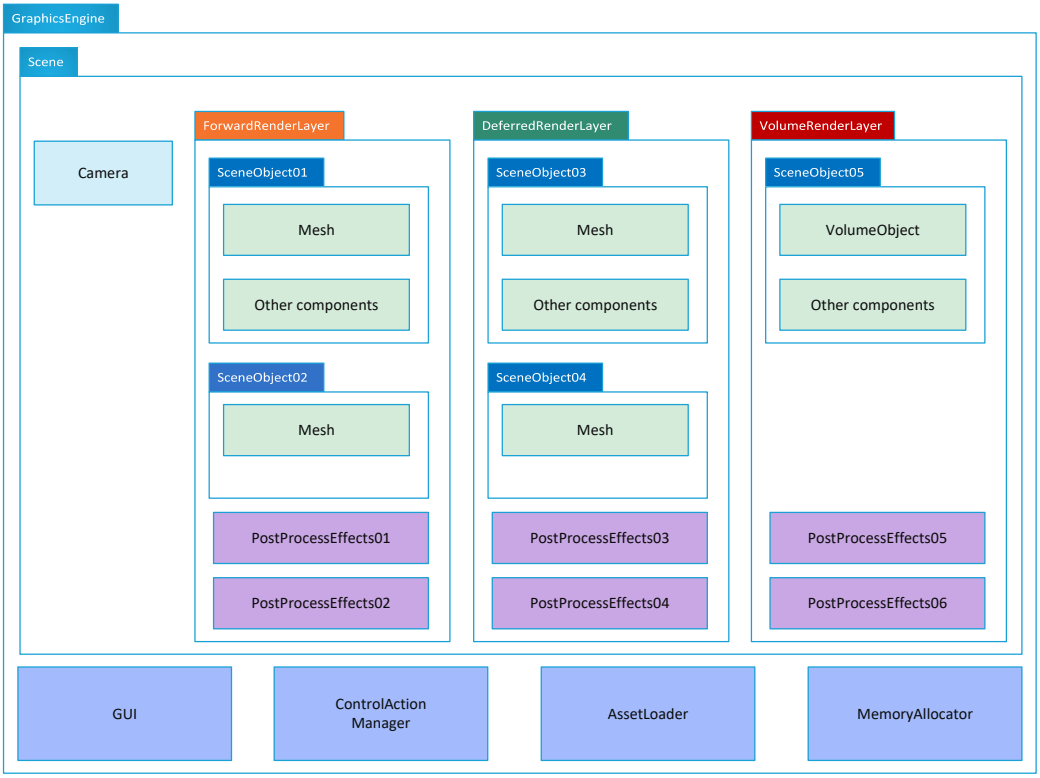
A funkcionális követelményeken felül megfogalmaztunk a szoftver minőségét növelő követelményeket is. Fontosnak tartottuk a könnyed és intuitív fejlesztés elősegítését. Ennek érdekében nagyfokú modularizációra törekedtünk. Hangsúlyt fektettünk a memória kezelés egyszerűsítésére a dinamikus allokáció alkalmazását érintő megkötések bevezetésével, és saját memória-allokátor implementálásával. A felhasználói bemenetek kezelését funkcionális minták alkalmazásával tettük könnyebbé.

Ebben a fejezetben megvizsgáljuk az elkészült szoftver architektúráját. Bemutatjuk a főbb komponenseket. Sorra vesszük a különböző mérnöki döntéshelyzeteket. Megkíséreljük megindokolni az egyes helyzetekben hozott döntések létjogosultságát. Említést teszünk a rendszer előnyeiről és eddig felismert hátrányairól.

## 3.1. Modulok kapcsolata magas absztrakció mellett

Ebben az alfejezetben magas absztrakciós szinten vizsgáljuk a program komponenseit. Bemutatjuk milyen főbb részekből épül fel az alkalmazás. Vázoljuk az egyes komponensek feladatát. Rámutatunk a részek közti kapcsolatokra. A 3.1 ábra szemlélteti programunk fontos komponenseit.

A megjelenítendő jelenetért a *Scene* modul felel. Minden képfrissítéskor aktualizálja a jelenetben résztvevő *SceneObject*-ek állapotát. Majd kirajzolja ezeket. Rendszerünk különböző megjelenítési eljárásokat támogat. Ezek kezelésére az objektumok megjelenítési rétegekbe lettek szervezve. A 3.1 ábrán szemléltetés céljából három réteget jelenítünk meg. Inkrementális megjelenítés esetén rétegenként állítható, hogy *forward* vagy *deferred* megjelenítést alkalmazunk-e, illetve, hogy igénybe vesszük-e a GPU *instanced rendering* technológiáját. A *forward* megjelenítés arra utal, hogy a geometriák megvilágítása a hozzájuk tartozó képpont árnyalóban egy kirajzolással rögtön kiszámítódik. Ezzel szemben a *deferred* megjelenítés során először a geometriák árnyalásához szükséges adatokat egy bufferbe írjuk, majd egy külön kirajzolás keretén belül kiszámítjuk az árnyalást. Az utób-




**3.1. ábra.** Magas absztrakciós szintű architektúra

bi megközelítés előnye, hogy a számításigényes megvilágítást, csak a ténylegesen látszódó pontokra számoljuk, hiszen a takarásban levő pontok az árnyalás számításakor már eldobásra kerültek. Ez a módszer akkor teljesít jól, hogyha sok a takarásban levő objektum, és költséges árnyalási modellt használunk (pl.: Torrance-Sparrow [37]). A kér fázisban történő kirajzolás, és az ehhez kapcsolódó textúra írás és olvasás jelentős hátulütő. A különböző objektumok eltérő árnyalási modellel történő kirajzolása is nehézkessé válik a fény számítások együttes kezelése miatt. Szeretnénk tehát meghagyni a lehetőséget, a *forward* és *deferred* megközelítés közti váltásra. A választás lehetőségénél egy fokkal nagyobb szabadságot ad, hogyha a jelenetünk tartalmazhat egyszerre *forward* és *deferred* technikával megjelenített objektumokat. Ez motiválta a megjelenítő rétegek fogalmának bevezetését. A különböző rétegekbe osztott objektumokat azonos virtuális kamera vetít a képernyőre. Előnyös továbbá, hogy a különböző rétegekben más és más képfeldolgozó eljárásokat applikálhatunk a kirajzolt ábrára. Ezek az utómunkálatok a rétegek között akkumulálódnak. Az először megjelenített réteg effektusai csak az első réteg képére vannak hatással. A későbbi rétegek effektusai már az összes korábbi réteg kompozícióját módosítják.

A jelenet objektumai további belső komponensekkel rendelkeznek. Ilyen komponens egy objektumhoz tartozó *mesh*, ami a geometriát és a felület anyagát tartalmazó adatszerkezet. A számunkra különösen fontos térfogati adathalmaz szintén egy komponensként adható meg. Jelenlegi megközelítésünkben a direkt térfogat kirajzolását sugárkövetéssel vagy textúraszeleteléssel megvalósító függvény az adattal együtt ebben a komponensben található. A megjelenítési rétegek szintjén a térfogat kirajzolása, már csak egy kamera felé néző két dimenziós ábra áttetsző megjelenítése. Más alkalmazások szükségessé tehetik a jelenet objektumainak dinamikai szimulációját. Ezt szintén az objektumokba ágyazható fizikai komponenssel tehetjük meg.

A Graphical user interface (GUI) egy lehetőséget ad arra, hogy a felhasználó, kapcsolatba lépjen a programmal. Ebben a kontextusban GUI alatt a program menürendszerét értjük.

A felhasználótól érkező billentyűzet és egér események feldolgozásáért a *control action manager* komponens felel. Lehetőség van a különböző bemeneti események kezelését végző *control action* objektumok létrehozására. Ezek összerendelik a billentyűzet egy adott betűjének lenyomását és a megfelelő függvényhívást. Az univerzalitás növelése érdekében az egyes események kezelését – funkcionális tervezési mintákat követve – tárolt lambda eljárások valósítja meg.

A program által használt forrásfájlokat a projekt könyvtárából olvassuk be. A különböző típusú fájlok, mint például textúrák, árnyaló programok forráskódja, vagy térfogati adathalmazok előre meghatározott alkönyvtárakban foglalnak helyet. Ezek a elérését egy *asset folder path manager* modul kontrollálja. Ennek megvalósítása a *singleton* tervezési mintát követi.

Egy összetett program architektúrájában fontos szerepet tölt be a megfelelő memória kezelés megvalósítása. A program komplexitásának növekedése mellett egyre nehezebbé válik a dinamikus memória foglalások számontartása. Ugyanazon területek megpróbáljuk többször felszabadítani. Korábban felszabadított területhez próbálunk hozzáférni. Az elhagyott felszabadítás miatt memóriaszivárgás jelenik meg. A felsorolt kihívások orvoslására általános megoldást kívánunk nyújtani. Bevezettünk egy saját memória allokátort, amely nyilvántartja a foglalásokat. Ezen keresztül foglalunk és szabadítunk fel memóriát. Ez jelentősen megkönnyíti a korábban említett gondok elkerülését.

A következő néhány alfejezetben részletesebben is bemutatjuk a program különböző részeit.

## 3.2. Scene osztály

Szoftverünk középpontjában a megjelenítendő jelenet áll. A jelenetet egy *Scene* nevű *C++* osztállyal reprezentáljuk. Ez magába foglalja a virtuális kamerát, a fényforrásokat és a jelenetben szereplő tárgyakat. A virtuális kamera leírja, hogy a felhasználó hol helyezkedik el a térben és merre néz [38]. Tartalmazza a nézeti és perspektivikus transzformációt. A kamera ezen kívül az ablak szélességének és magasságának arányát figyelembe véve úgy korrigálja a képet, hogy az ne szenvedjen torzítást (*aspect ratio*). A transzformációk mátrixok formájában adottak. A kamera paramétereit minden alkalommal frissíteni szükséges, amikor a felhasználó változtat a nézeten. Mivel a térfogati adatok megjelenítése költséges művelet, ezért eltároljuk a kamera állapotának megváltozását jelző eseményt. Később a térfogat kirajzolásakor, hogyha egyéb paraméterek változatlanok, elegendő a kamera változtatása után újrarajzolni a jelenetet. Az árnyaló programok a kamera adatait egy *uniform bufferben* kapják meg [12]. Így nincs szükség a transzformációs mátrixok minden programba történő külön bekötésére. Az árnyalók közös memóriaterületet használnak.

A jelenetben fényforrások vannak jelen. Ezek meghatározzák a tárgyakra eső fény irányát, színét és intenzitását. Programunkban pont- és irányfényforrásokat különböztetünk meg. Az előbbi egy pontban helyezkedik el. Onnan minden irányban azonos intenzitással sugároz. Az intenzitás a fényforrástól való távolság négyzetével arányosan csökken [28]. Az irányfényforrás nem rendelkezik pozícióval csak forrásiránnyal. Cserébe úgy tekintjük, hogy az általa kibocsátott fény nem szenved csillapítást. A térfogat megjelenítésekor egy pontfényforrást alkalmaztunk. Ennek intenzitásának négyzetes csökkenésén felül értelmezzük a közeg csillapítását. Ez azt jelenti, hogy teljesen áteresztő közegben is csak véges távolságban érzékelhető a pontfényforrás kontribúciója.

A jelenetben résztvevő tárgyakat valahogyan megjelenítjük. Ennek módja megadható egy poligonhálóval definiált geometriaként vagy térfogati adathalmazként. A poligonhálós megjelenítésnek is létjogosultsága van alkalmazásunkban, mivel szeretnénk például jelezni a felhasználó felé, hogy pontosan hol helyezkedik el a fényforrás. Erre a célra egy téglalapra rajzolt villanykörtét jelenítünk meg a fényforrás pozíciójában. Hasonló segítséget jelent a befoglaló téglatest körvonalainak megjelenítése, ami intuitívebbé teszi a test metszetének kiválasztását. Programunk lehetővé teszi különböző megjelenítési eljárások kombinációját. Ennek kezelésére bevezettük a *RenderLayer* osztályt. A rétegeket egymásután fűzhetjük. Az egyes rétegek listát tartanak számon az adott rétegben megjelenítendő tárgyakról. A rétegek tartalmazhatnak utófeldolgozó effektusokat, amelyek kimenete szintén láncolódik. A mélység buffer közös a rétegek közt. Így egy korábbi rétegben megjelenített – például inkrementális képszintézissel kirajzolt – test takarhatja a később megjelenített térfogatot. Utófeldolgozó effektusként egy *blur* hatást és egy Nagy dinamikatartomány (HDR)-ről Alacsony dinamikatartomány (SDR)-re konvertáló árnyalat megfeleltető eljárást alkalmaztunk. A *blur* az SDR tartományon kívül eső, nagy intenzitású pontokat szétfolyva jeleníti meg [4]. Ezzel utánozza az emberi szem és a mesterséges kamerák viselkedését, amikor azok erős fénybe néznek. A HDR tartomány leképezésére az Academy Color Encoding System (ACES)-t használtuk [1].

## 3.3. VolumeObject osztály

Alkalmazásunk középpontjában a térfogati adathalmazt becsomagoló *VolumeObject* osztály áll. Ennek felelőssége a térfogat megjelenítése sugárkövetés vagy *half-angle slicing* technikával. Az átviteli függvényt is ez az osztály menedzseli. Ezen kívül kezeli a sugárkövetés optimalizálását segítő befoglaló geometriát. Programunkban lehetőséget biztosítunk a térfogat metszeteinek vizsgálatára. Ehhez a befoglaló téglatest oldalait lehet rátolni a térfogatra. Az aktuális befoglaló téglatest méreteit szintén ez az osztály tárolja.

Annak érdekében, hogy a felhasználói felület a hosszabb időt igénybe vevő megjelenítések során is reszponzív maradjon, mindkét megjelenítési eljárást frame-ek közt elosztva végez a program. Sugárkövetés esetén a képernyőt kvadránsokra osztja. Egy frissítési ciklusban egy kvadránst jelenít meg. *Half-angle slicing* esetén adódik, hogy az egy frame alatt kirajzolt proxy geometriák számát korlátozzuk. A következő frame megjelenítésekor onnan folytatjuk a megjelenítést, ahol korábban abbahagytuk. Hogy ez ne okozzon villogást, a térfogatot először egy köztes bufferbe rajzoljuk, majd a teljes kép elkészólte után frissítjük vele a képernyőre rajzolt buffert. Egy másik optimalizációs technika, hogy a *Half-angle slicing* proxy geometriáinak számát első körben alacsonyra választjuk. Ez az alul-mintavételezés miatt Moiré mintákat eredményez. A gyenge minőségű képet azonban gyorsan meg tudjuk jeleníteni így nem okoz fennakadást például kamera forgatás közben. Miután a kamerát elengedtük, van idő számításigényes, nem feltétlenül valós idejű megjelenítés kiértékelésére. Az első gyors megjelenítés során egy egyszerűbb árnyalóprogramot is használunk, amely kevesebb mintát használ.

# 4. fejezet

# Eredmények bemutatása

Az elkövetkező fejezet célkitűzése, a program által előállított képek szemléltetése. Ezeket kritikusan összehasonlítjuk egymással.

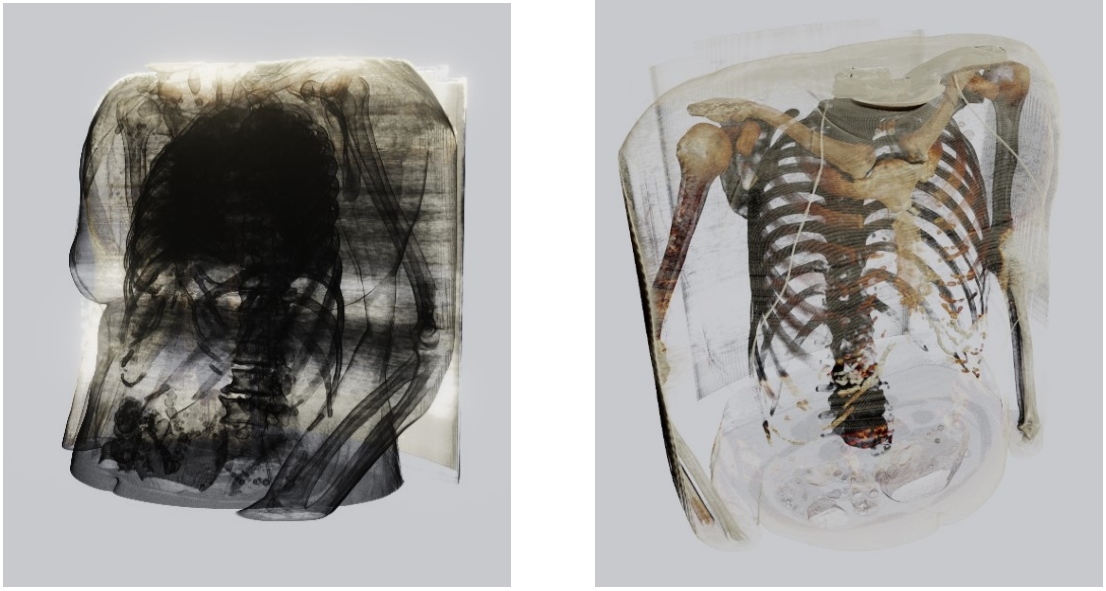
## 4.1. Felhasznált adathalmazok

A fejlesztés során több nyilvánosan hozzáférhető adathalmazt jelenítettünk meg. *The Stanford volume data archive* [21] egy jól bevált, klasszikus forrás. Alkalmazásunk prototípusa az itt elérhető *MRbrain* nevet viselő adathalmazt jelenítette meg először. Ennek érdekessége, hogy a felvételen a koponya nyitott. Az átviteli függvény manipulálása nélkül is rögtön szemügyre vehető. Később a fejlesztés jelentés részében a *CThead* nevű adathalmazzal teszteltük implementációnk. Ez utóbbi jó alapnak bizonyult az STF átviteli függvények tesztelésére. Különleges részlete a felvételnek, hogy a fogsor tömést tartalmazott. Hatására sugárszerű képhibák jelentek meg a felvételen. A halmaz gyenge pontja a mai szemmel viszonylag alacsony $256 \times 256 \times 99$-es felbontás. Így a rácsos szerkezet magasabb rendű interpolációs eljárások mellett is észlelhető marad.

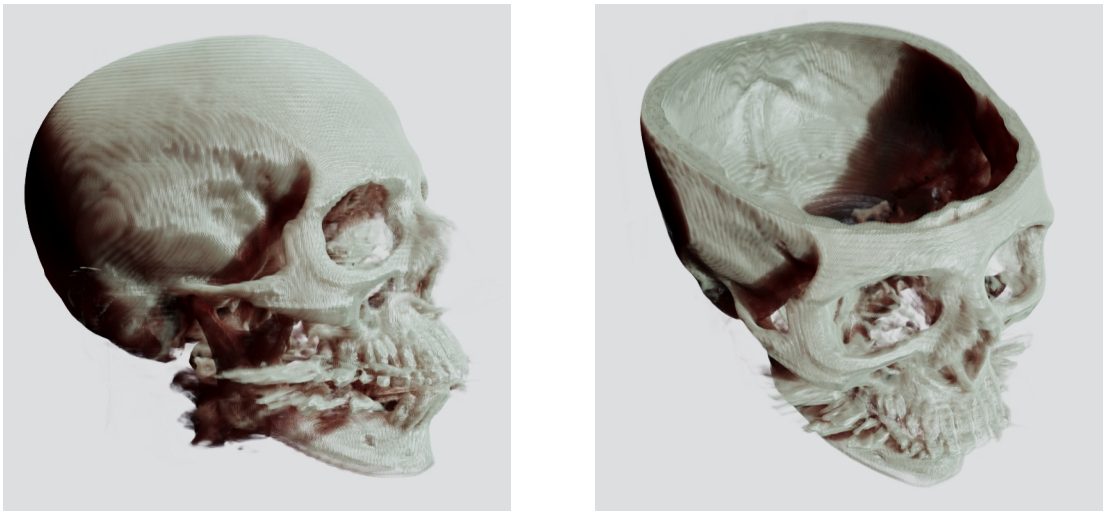
A fejlesztés második felében a *The Visible Human Project* címen elhíresült adatbázis részeit vizualizáltuk [27]. Ezt az amerikai egyesült államokbeli Bethesda városban működő *National Library of Medicine* gondozza. Az adatbázis 2019-óta a nyilvánosság számára szabadon érhető el az interneten. Részletgazdag, magas felbontású felvételeket tartalmaz az emberi szervezetről. Az adathalmazok több – részben átfedő – szeletre osztják a testet.

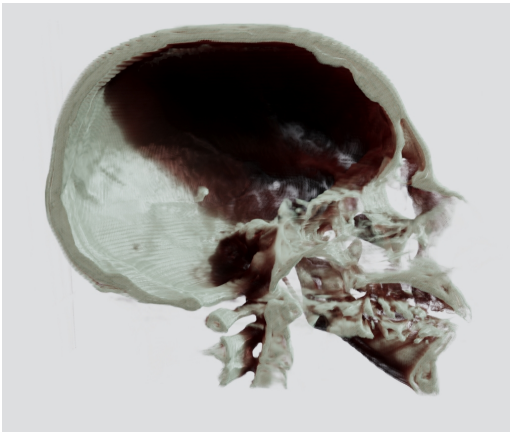
## 4.2. Ábrák bemutatása

A 4.1 ábrán megfigyelhető fény elszíneződése az áttetsző közegen keresztül. A 4.2 ábrákon demonstráljuk a program metszet készítő képességét. A 4.3 ábrák a gerincoszlopot és a tüdő lebenyek egy részét ábrázolja. A 4.4 ábrán egy részletesebb átviteli függvényt használtunk. A 4.5 ábrán látszik, hogy a javított trilineáris szűrés sem képes az alacsony felbontású adathalmaz teljesen simított megjelenítésére. Itt már láthatóvá válik a rácsos szerkezet, amire ráerősít a felületi árnyalás.



**4.1. ábra.** Áttetsző torzó - VHB adatbázis

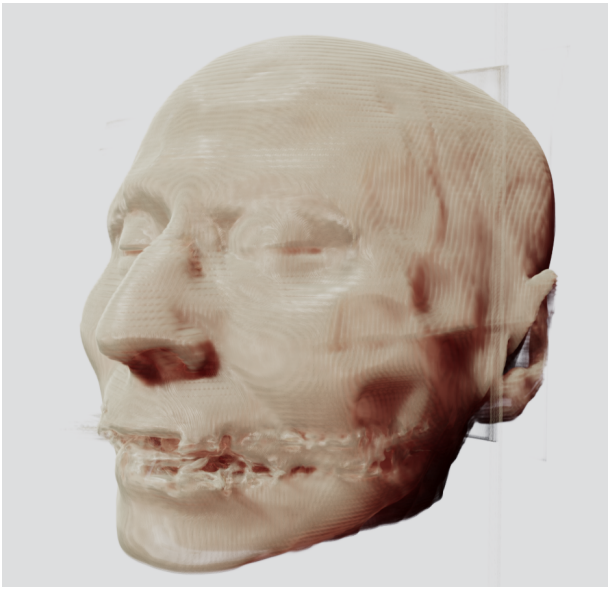




**4.2. ábra.** Koponya különböző vágásokkal - *CThead* adathalmaz

**4.3. ábra.** Gerincoszlop - *Shoulders* VHB adatbázis

**4.4. ábra.** Női alsótest színezett átviteli függvénnyel



**4.5. ábra.** Fej - *CThead* adathalmaz

# 5. fejezet

# Összefoglaló

## 5.1. Eredmények

Munkánkban térfogati adathalmazok direkt megjelenítésével foglalkoztunk. Megvizsgáltunk különböző megjelenítési eljárásokat. *Half-angle slicing* módszerrel áttetsző közegeket jelenítettünk meg realisztikus árnyékokkal. Az ehhez szükséges proxy geometriák előállítását Központi Vezérlőegység (CPU)-n végeztük. A proxy geometriák megjelenítését a GPU-n implementáltuk. A háromdimenziós textúrában tárolt adathalmaz mintavételezésekor javított trilineáris szűrést alkalmaztunk. Ezzel egy ponthoz tartozó intenzitás és gradiens párost összesen ötvenhat mintával számoltuk. Beépített trilineáris interpolációt használva az árnyaló programban ezt hét textúra olvasással valósítottuk meg. Mindez a nagyobb mintaigényű *Catmull-Rom spline*-al vetekedő minőségű rekonstrukciót eredményezett. Az eljárásokat egy saját fejlesztésű megjelenítő programban ötvöztük. Alkalmazásunk elsősorban a *half-angle slicing* metódust alkalmazva jelenít meg voxel tömböket. Így részletes árnyékvetést szimulálunk, ami jelentésen növeli az ábrák realisztikus mivoltát. Különböző eszközöket biztosítunk az átviteli függvény testreszabására. A függvényt reprezentáló textúrába rajzolni engedünk. Így az átviteli függvény intuitívan változtatható. A manuális szerkesztést könnyíti a STF. Ez a klasszifikációs eljárás intenzitások térbeli elhelyezkedését felhasználva alakít ki régiókat az átviteli függvényben. A térfogat különböző metszeteit a befoglaló téglatest falainak betolásával tárhatjuk fel. A megjelenítés során optimalizációs technikákat alkalmaztunk. A sugárkövetést egy kis téglatestekből felépülő befoglaló geometriával tettük gyorsabbá, amely szorosabban körülöleli a megjelenítendő térrészt. Így csökkentettük a fölösleges, optikailag üres közegben kiértékelt minták számát. Alkalmazásunk *The Stanford volume data archive* és  adatbázisok adathalmazaival teszteltük. Munkánkban ezen adatok vizualizációját közöltük.

## 5.2. Fejlesztési lehetőségek

Alkalmazásunk fejlesztése során felmerültek olyan fejlesztési irányok, amelyek ugyan eddig nem kerültek megvalósításra, de a jövőben érdemesnek találjuk ezeket közelebbről vizsgálni.

Eddigi munkánk során statikus felvételeket jelenítettünk meg. A jövőben érdemesnek tartjuk időben változó adatok megjelenítését vizsgálni. Ez alatt érthetjük az egyik statikus adathalmaz egy másik adathalmazzá történő metamorfózisát. Az átalakulás miként jére számtalan lehetőség kínálkozik. Folyadék szimulációs eljárások használhatók az átmenet vezénylésére. De Witt és társai [9] Laplace saját függvényeket felhasználva szimulálnak hatékonyan divergenciamentes folyadékokat. Megközelítésük jól alkalmazható az anyag kezdeti és végállapot közti áramoltatására.

A még realisztikusabb megjelenés érdekében érdemes megvizsgálni a Thomas Kroes és társai [16] által bemutatott *ray tracing*-en alapuló eljárást.

# Köszönetnyilvánítás

Szeretnék köszönetet nyilvánítani minden hallgatótársamnak, akivel munkám során az egyes részeredményeket megoszthattam és értékes hozzászólásaikkal elősegítették a programom fejlesztését. Különös köszönet konzulensemnek, Dr. Csébfalvi Balázsnak, amiért nem csak a konkrét feladatban nyújtott szakmai segítséget, de igyekezett felvillantani az akadémiai élet nyújtotta lehetőségeket. Ezzel mindig a jobb és több irányába terelgetett.

# Rövidítések

# Irodalomjegyzék


[1] Academy color encoding system, 2022.
URL https://www.oscars.org/science-technology/sci-tech-projects/aces. látogatva 2022. 12. 4. 12:00.

[2] James F. Blinn: Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77 konferenciasorozat. New York, NY, USA, 1977, Association for Computing Machinery, 192–198. p. ISBN 9781450373555.
URL https://doi.org/10.1145/563858.563893. 7 p.

[3] James F. Blinn: Light reflection functions for simulation of clouds and dusty surfaces. In *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '82 konferenciasorozat. New York, NY, USA, 1982, Association for Computing Machinery, 21–29. p. ISBN 0897910761.
URL https://doi.org/10.1145/800064.801255. 9 p.

[4] Activision Blizzard: Next generation post processing in call of duty: Advanced warfare. *Advances in Real-Time Rendering in Games course SIGGRAPH*, 2014.
URL https://advances.realtimerendering.com/s2014/.

[5] Brian Cabral – Nancy Cam – Jim Foran: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 Symposium on Volume Visualization*, VVS '94 konferenciasorozat. New York, NY, USA, 1994, Association for Computing Machinery, 91–98. p. ISBN 0897917413. URL https://doi.org/10.1145/197938.197972. 8 p.

[6] Robert L. Cook – Kenneth E. Torrance: A reflectance model for computer graphics. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81 konferenciasorozat. New York, NY, USA, 1981, Association for Computing Machinery, 307–316. p. ISBN 0897910451.
URL https://doi.org/10.1145/800224.806819. 10 p.

[7] William M. Cornette – Joseph G. Shanks: Physically reasonable analytic expression for the single-scattering phase function. *Appl. Opt.*, 31. évf. (1992. Jun) 16. sz., 3152–3160. p. URL https://opg.optica.org/ao/abstract.cfm?URI=ao-31-16-3152.

[8] Balázs Csébfalvi: Beyond trilinear interpolation: Higher quality for free. *ACM Trans. Graph.*, 38. évf. (2019. jul) 4. sz. ISSN 0730-0301.
URL https://doi.org/10.1145/3306346.3323032. 8 p.

[9] Tyler De Witt – Christian Lessig – Eugene Fiume: Fluid simulation using laplacian eigenfunctions. *ACM Trans. Graph.*, 31. évf. (2012. feb) 1. sz. ISSN 0730-0301. URL https://doi.org/10.1145/2077341.2077351. 11 p.

[10] Klaus Engel – Markus Hadwiger – Joe M. Kniss – Aaron E. Lefohn – Christof Rezk Salama – Daniel Weiskopf: Real-time volume graphics. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04 konferenciasorozat. New York, NY, USA, 2004, Association for Computing Machinery, 29–es. p. ISBN 9781450378017.
URL `https://doi.org/10.1145/1103900.1103929`.

[11] Louis G Henyey – Jesse Leonard Greenstein: Diffuse radiation in the galaxy. *The Astrophysical Journal*, 93. évf. (1941), 70–83. p.

[12] The Khronos Group Inc: OpenGL Wiki uniform buffer object, 2017.
URL `https://www.khronos.org/opengl/wiki_opengl/index.php?title=Uniform_Buffer_Object&oldid=13827`. látogatva 2022. 12. 4. 13:00.

[13] R. Keys: Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29. évf. (1981) 6. sz., 1153–1160. p.

[14] J. Kniss – G. Kindlmann – C. Hansen: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8. évf. (2002) 3. sz.

[15] Joe Kniss – Simon Premoze – Charles Hansen – David Ebert: Interactive translucent volume rendering and procedural modeling. In *Proceedings of the Conference on Visualization '02*, VIS '02 konferenciasorozat. USA, 2002, IEEE Computer Society, 109–116. p. ISBN 0780374983. 8 p.

[16] Thomas Kroes – Frits Post – Charl Botha: Exposure render: An interactive photo-realistic volume rendering framework. *PloS one*, 7. évf. (2012. 07), e38586. p.

[17] J. Krüger: A new sampling scheme for slice based volume rendering. In *Proceedings of the 8th IEEE/EG International Conference on Volume Graphics*, VG'10 konferenciasorozat. Goslar, DEU, 2010, Eurographics Association, 1–4. p. ISBN 9783905674231. 4 p.

[18] J. Kruger – R. Westermann: Acceleration techniques for gpu-based volume rendering. In *IEEE Visualization, 2003. VIS 2003.* (konferenciaanyag). 2003, 287–292. p.

[19] Philippe Lacroute – Marc Levoy: Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94 konferenciasorozat. New York, NY, USA, 1994, Association for Computing Machinery, 451–458. p. ISBN 0897916670. URL `https://doi.org/10.1145/192161.192283`. 8 p.

[20] M. Levoy: Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8. évf. (1988) 3. sz., 29–37. p.

[21] Marc Levoy: Stanford Computer Graphics Laboratory the stanford volume data archive, 2014. URL `https://graphics.stanford.edu/data/voldata/`. látogatva 2022. 12. 3. 11:00.

[22] Wei Li – K. Mueller – A. Kaufman: Empty space skipping and occlusion clipping for texture-based volume rendering. In *IEEE Visualization, 2003. VIS 2003.* (konferenciaanyag). 2003, 317–324. p.

[23] Lukas Marsalek – Armin Hauber – Philipp Slusallek: High-speed volume ray casting with cuda. In *2008 IEEE Symposium on Interactive Ray Tracing* (konferenciaanyag). 2008, 185–185. p.

[24] Stephen R. Marschner – Richard J. Lobb: An evaluation of reconstruction filters for volume rendering. In *Proceedings of the Conference on Visualization '94*, VIS '94 konferenciasorozat. Washington, DC, USA, 1994, IEEE Computer Society Press, 100–107. p. ISBN 0780325214. 8 p.

[25] Stephen R. Marschner – Richard J. Lobb: An evaluation of reconstruction filters for volume rendering. In *Proceedings of the Conference on Visualization '94*, VIS '94 konferenciasorozat. Washington, DC, USA, 1994, IEEE Computer Society Press, 100–107. p. ISBN 0780325214. 8 p.

[26] N. Max: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1. évf. (1995) 2. sz., 99–108. p.

[27] National Library of Medicine the visible human project, 2019.
URL `https://www.nlm.nih.gov/research/visible/visible_human.html`. látogatva 2022. 12. 3. 12:00.

[28] Pangratios Papacosta – Nathan Linscheid: The confirmation of the inverse square law using diffraction gratings. *The Physics Teacher*, 52. évf. (2014) 4. sz., 243–245. p. URL `https://doi.org/10.1119/1.4868944`.

[29] Matt Pharr – Wenzel Jakob – Greg Humphreys: *Physically based rendering: From theory to implementation.* 2016, Morgan Kaufmann.

[30] Bui Tuong Phong: Illumination for computer generated pictures. *Commun. ACM*, 18. évf. (1975. jun) 6. sz., 311–317. p. ISSN 0001-0782.
URL `https://doi.org/10.1145/360825.360839`. 7 p.

[31] W. K. Pratt: *Digital image processing : PIKS inside (3rd ed.).* 2001, John Wiley & Sons, Inc. ISBN 0471221325.

[32] C. Rezk-Salama – K. Engel – M. Bauer – G. Greiner – T. Ertl: Interactive volume on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, HWWS '00 konferenciasorozat. New York, NY, USA, 2000, Association for Computing Machinery, 109–118. p. ISBN 1581132573.
URL `https://doi.org/10.1145/346876.348238`. 10 p.

[33] Stefan Roettger – Michael Bauer – Marc Stamminger: Spatialized transfer functions. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'05 konferenciasorozat. Goslar, DEU, 2005, Eurographics Association, 271–278. p. ISBN 3905673193. 8 p.

[34] Paolo Sabella: A rendering algorithm for visualizing 3d scalar fields. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88 konferenciasorozat. New York, NY, USA, 1988, Association for Computing Machinery, 51–58. p. ISBN 0897912756.
URL `https://doi.org/10.1145/54852.378476`. 8 p.

[35] C.E. Shannon: Communication in the presence of noise. *Proceedings of the IRE*, 37. évf. (1949. jan) 1. sz., 10–21. p.
URL `https://doi.org/10.1109/jrproc.1949.232969`.

[36] G. Thomas – J. Hass – M.D. Weir: *Thomas' Calculus.* Always learning sorozat. 2014, Pearson Education. ISBN 9780321878960.
URL `https://books.google.hu/books?id=XXqrngEACAAJ`.

[37] K. E. Torrance – E. M. Sparrow: Theory for off-specular reflection from roughened surfaces∗. *J. Opt. Soc. Am.*, 57. évf. (1967. Sep) 9. sz., 1105–1114. p.
URL https://opg.optica.org/abstract.cfm?URI=josa-57-9-1105.

[38] Colin Ware: Chapter ten - interacting with visualizations. In Colin Ware (szerk.): *Information Visualization (Fourth Edition)*. Interactive Technologies sorozat. Fourth edition. kiad. 2021, Morgan Kaufmann, 359–392. p. ISBN 978-0-12-812875-6. URL https://www.sciencedirect.com/science/article/pii/B9780128128756000104.

[39] Peter L. Williams – Nelson Max: A volume density optical model. In *Proceedings of the 1992 Workshop on Volume Visualization*, VVS '92 konferenciasorozat. New York, NY, USA, 1992, Association for Computing Machinery, 61–68. p. ISBN 0897915275. URL https://doi.org/10.1145/147130.147151. 8 p.

[40] Roni Yagel – Zhouhong Shi: Accelerating volume animation by space-leaping. In *Proceedings of the 4th Conference on Visualization '93*, VIS '93 konferenciasorozat. USA, 1993, IEEE Computer Society, 62–69. p. ISBN 0818639407. 8 p.