VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE OBJEMOVÝCH DAT POMOCÍ VOLUME RENDERINGU

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE MAREK VARGA
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE OBJEMOVÝCH DAT POMOCÍ VOLUME RENDERINGU

3D VOLUME RENDERING DATA VISUALIZATION

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE MAREK VARGA
AUTHOR

VEDOUCÍ PRÁCE ING. PŘEMYSL KRŠEK PH.D.
SUPERVISOR

BRNO 2007

Strana pro zadání bakalářské práce:

První strana licenční smlouvy:

Druhá strana licenční smlouvy:

Abstrakt

Tato práce se zabývá využitím teoretických poznatků z oblasti počítačové grafiky a vektorové matematiky při zpracování a zobrazení objemových dat. Popisuje více způsobů vizualizace objemových dat a jejich optimalizaci. K problematice objemových dat spadají matematické principy a algoritmy, které zajistí získání dat, jejich správnou interpretaci, filtraci a odstranění chyb a vlastní zobrazení. Práce pojednává o využití a praktických možnostech metody vizualizace objemových dat pomocí voxel splattingu. Součástí práce je implementace demonstračního programu který používá tuto metodu. V programu je použito několik urychlovacích technik procesu vykreslení včetně přeskakování prázdných úseků a efektivní adresování v objemových datech a jejich vlastnostech. Systém umožňuje interaktivní zobrazení a manipulaci s objemovými daty.

Klíčová slova

Volume rendering, voxel splatting, sobelův operátor, konvoluce, voxel

Abstract

This thesis deals with using theoretical knowledge of computer graphic and vector mathematics for volume data construction and vizualization. This work provides more methods volume data vizualization and optimization. Algorithms and mathematical principles are necessary for data obtaining, correct interpretation, filtration and error correction and vizualization. This work describes possibilities of method voxel splatting. Model which implements this method is a part of this work and it contains some techniques for drawing process which include skipping empty sections and effective addressing of properties of volume data. System also provides interactive rendering and volume data manipulation.

Keywords

Volume rendering, voxel splatting, sobel edge detector, convolution, voxel

Citace

Marek Varga: Vizualizace objemových dat pomocí volume renderingu, bakalářská práce, Brno, FIT VUT v Brně, 2007

Vizualizace objemových dat pomocí volume renderingu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Přemysla Krška Ph.D. Všechny zdroje, prameny a literaturu, kterou jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

……
Marek Varga
1.5.2007

Poděkování

Děkuji tímto vedoucímu své bakalářské práce, panu **Ing. Přemyslovi Krškovi Ph.D**., za vedení, cenné rady a podnětné připomínky při tvorbě této práce a panu Slavomíru Hudákovi za poskytnutí mnoha cenných materiálů, které jsou základem této práce .

© Marek Varga, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah....1
Seznam obrázků....2
1 Úvod....3
1.1 Oblast problematiky....3
1.2 Motivace a cíl práce....3
1.3 Interaktivní vizualizace prostorových dat....4
1.4 Charakteristika dat....4
1.5 Předzpracování dat....5
1.6 Problémy při vizualizaci....5
2 Volumetrická vizualizace....6
2.1 Vizualizační metody....6
2.1.1 Algoritmy zobrazující povrchy....6
2.1.2 Objemové algoritmy....6
2.1.3 Porovnání vizualizačních metod....8
2.2 Základy....8
2.2.1 Vícerozměrná reprezentace dat....8
2.2.2 Klasifikace....10
2.2.3 Průchod daty....10
2.2.4 Stínování....10
2.2.5 Předvzorkování a interpolace....12
2.2.6 Kompozice....12
2.3 Voxel splatting....13
2.4 Zvyšování kvality a rychlosti....13
2.4.1 Urychlení voxel splattingu....13
2.4.2 Zvyšování kvality obrazu....14
3 Renderování....15
3.1 Dekompozice....15
3.1.1 Řazení voxelů....15
3.1.2 Výpočet průmětu voxelů....16
3.1.3 Stínování....17
3.1.4 Předzpracování....22
4 Implementace a specifikace....23
4.1 Návrh....23
4.2 Dostupný hardware a software....23
4.3 Implementace....23
4.4 Řízení....23
4.5 Třídy....24
4.6 Adresování....24
4.7 Softwarová omezení....24
5 Výsledky....25
5.1 Přechodová funkce....25
5.2 Optimalizace paměťových operací....25
5.3 Výkonnost algoritmu....25
6 Hodnocení a závěr....27
6.1 Modifikace práce....27
6.2 Další vývoj....27
6.3 Hodnocení výkonu a kvality....27
6.4 Závěr....28
Literatura....29
Seznam příloh....30

Seznam obrázků

2.2.1.1 Reprezentace objemových dat.....8
2.2.1.2 Třídy soustav.....9
2.2.1.3 Schéma vykreslovacího procesu.....9
2.3.1 Řazení voxelů.....13
3.1.1.1 Pohled pozorovatele.....15
3.1.1.2 Optimalizované řazení voxelů.....16
3.1.3.1 Schéma konvoluce.....18
3.1.3.2 Histogram signálu po konvoluci.....20
3.1.3.3 Vertikální detekce hran.....20
3.1.3.4 Horizontální detekce hran.....20
3.1.3.5 Hloubková detekce hran.....21
5.1.1 Výkon v závislosti na přechodové funkci.....25
5.2.1 Výkon v závislosti na použitém paměťovém modelu.....25
5.3.1 Výkonnost algoritmů.....26

1 Úvod

Objemová vizualizace je dobře známá věta počítačové grafiky, která umožňuje zkoumání vnitřních struktur a komplexního chování trojrozměrných objemových objektů. Především díky problémům medicíny, korektně a rychle zobrazovat orgány lidského těla na běžném počítači, se objemová vizualizace rozvíjí tak rychle. Nové techniky a dostupný hardware dnes umožňují lékařům vizualizovat objemové data dostatečně interaktivně. Výsledky dlouhodobých výzkumů vylepšily jak softwarovou, tak hardwarovou stránku problematiky. Jsou známé a dobře prozkoumané techniky zobrazující komplexní datové struktury získané z různých vědních oborů. Proti medicíně jsou struktury na mikroskopické úrovni ještě složitější a mnoho z nic je stále neprozkoumaných [4].

Vícerozměrná vizualizace se v posledních letech značně rozrostla. Mnohé techniky byly úspěšně aplikovány v oblasti magnetické rezonance nebo počítačové tomografie. Příchodem těchto technologií přinesl revoluční možnosti zobrazování, které vedli k rapidnímu nárustu využití vizuálních technik pro nejrůznější typy vzorků.

V první kapitole uvedeme problém interaktivní vizualizace a charakterizujeme objemová data. Druhá kapitola nabídne přehled metod objemové vizualizace, detailně popisuje princip fungování algoritmu voxel splattingu. První i druhá kapitola plně staví na poznatcích [4] kde je tato problematika velice detailně probrána. Třetí kapitola uvádí čitatele do problematiky voxel splattingu, popisuje možnosti přístupu k objemové vizualizaci a prezentuje více druhů optimalizačních metod popsaných v předcházejících kapitolách. Kapitola čtyři obsahuje všechny detaily návrhu a implementace programu, zmiňuje se o problémech, se kterými jsme se během vývoje setkali. Výsledky práce prezentujeme v kapitole pět. Shrnutí obsahu práce a její další možný vývoj jsou obsahem šesté kapitoly. Reference a přílohy uzavírají tuto práci.

1.1 Oblast problematiky

Trojrozměrné zpracování obrazu je část počítačové grafiky, která na sebe v posledních letech poutá víc a víc pozornosti. Hlavním důvodem je prudký nárust výkonu počítačů a dostupnost nových technologií, díky kterým jsme schopní mnohem rychleji zpracovávat obraz. Její využití postřehneme ve vícerých oblastech života. Velmi důležitou úlohu sehrává v medicíně, kde pomáhá lékařům analyzovat údaje ze zařízení jako magnetická rezonance (MRI) nebo počítačová tomografie (CT). Její další využití můžeme vidět například v geografii a geologii (zobrazování modelů ložisek ropy, plynů nebo modelů vodních nádrži získaných ze seismických dat), fyzice (vizualizace výsledků simulací fyzikálních procesů), průmyslu (analýza fraktur a chyb částí materiálu), námořnictví (zobrazování signálů ze sonaru), nebo i v biologii (rekonstrukce trojrozměrného modelu objektů ze série řezů získaných z konfokálních mikroskopů). V této práci se nebudeme omezovat na data z jednotlivých oborů, ale pojmeme problematiku komplexně, protože algoritmy použité v této práci jsou použitelné obecně a nelimituje je typ údajů.

1.2 Motivace a cíl práce

Cílem bakalářské práce bylo studium, porovnání a výběr nejvhodnějších metod pro vizualizaci objemových dat. Navrhli a vytvořily jsme nástroje pro vizualizaci objemových dat. Hlavním cílem použití použití vizuálních technik je příprava jednoduše interpretovatelných, a pokud možno, i realistických obrazů, bez artefaktů a s možností zvýraznění vybraných typů objektů zájmu prostřednictvím segmentačních a klasifikačních metod. Přestože získání dat je již jednoduché, vlastní vizualizace výsledných dat je docela složitý proces. Pro dosáhnutí tohoto cíle bylo potřeba detailně nastudovat problematiku objemové vizualizace (kapitola 2). V těchto kapitolách jsou uvedeny důležité techniky, jejich vzájemné porovnání, ale i problémy, které s nimi souvisí. Objemové renderování je jen jedna z metod vizualizace vícerozměrných dat. Porovnáváme výhody a nevýhody

volumetrického zobrazování. Detailně popisujeme nepřímou metodu voxel splatting, jako i tvorbu programu a těžkosti s tím spojené (kapitola 3). V této kapitole popisujeme i konstrukci a zobrazení obrazu. Pro zvýšení celkové rychlosti programu jsme vybrali více druhů optimalizace softwarového voxel splattingu, jako přeskakování neviditelných dat, efektivní schéma adresování, vyhledávací tabulky homogenních buněk a vynechání opakujících se informací. Obsahuje také popis návrhu systému, implementaci algoritmu a jejich urychlení. V kapitole 4. najdeme přesnou specifikaci hardwarových a softwarových možností, detaily implementace jednotlivých částí programu a problémy spojené s vývojem softwaru. Výsledky ve formě obrázků a grafů, rychlostní a kvalitativní porovnání se nachází v kapitole 5. Uvádíme porovnání základního a optimalizovaného algoritmu. Hlavní přínos práce spočívá ve snaze vytvořit program používající metodu voxel splattingu. Návrh jednotlivých součástí programu se snaží být co nejobecnější ve smyslu použití na jakékoli typy dat. Implementace tříd a algoritmů zase co nejportabilnější a nejefektivnější. Výsledný čas zobrazování obrázků není na rozměrných datech interaktivní, tj. jednotky snímků za sekundu. Práce nabízí přehled objemové vizualizace.

1.3 Interaktivní vizualizace prostorových dat

Ještě před analýzou samotné interaktivní vizualizace se zaměřme na význam některých pojmů. Vizualizaci můžeme chápat jako grafickou reprezentaci abstraktních údajů uloženou většinou ve formě čísel nebo textu. Tabulka s *n* vzorky a jim příslušícími hodnotami nám na první pohled hodně napoví. Jestliže z těchto hodnot vytvoříme graf, ztratíme sice přesnost v podobě čísel, ale dokážeme říct mnohem víc o rozměrech a charakteristice údajů. Volumetrickou (objemovou) vizualizaci pak můžeme jako proces projekce trojrozměrných dat na dvojrozměrnou rovinu za účelem pochopení struktury objektů obsažených v datech. V literatuře se také můžeme setkat s pojmenováním volumetrické renderování. Pod slovy interaktivní vizualizace rozumíme vizualizaci dat, které parametry mění uživatel v reálném čase. Požadavek na interaktivní vizualizaci je pro nás částečně kritická, protože zobrazujeme objekty, o jejichž struktuře často nemáme žádné informace. Změny by se měli projevit okamžitě, tj. bez povšimnutí uživatele. Pro plynulé zobrazování scény potřebujeme, aby náš počítač program dokázal vykreslovat obraz rychlostí alespoň 25 snímků za sekundu(což v praxi dosahujeme velice těžko). Jak uvidíme později, volumetrická vizualizace v reálném čase je výpočtově velice náročná a klasický přístup zde selhává. Proto je potřebné používat speciální techniky a optimalizace při řešení tohoto problému. Volumetrická vizualizace v reálném čase má proti klasické statické vizualizaci několik výhod: [4]

Změna pohledu: Libovolný pohled na scénu (pozice kamery). Uživatel může libovolně nastavit pozici kamery a její vzdálenost od zobrazovaných objektů. Rotací a posuny lépe pochopíme tvar objektů ve scéně.

Vyřezávání: Nebudeme zobrazovat jistou část dat. Díky tomu jsme schopni zobrazit vnitřek objektů, respektive odstranit ty oblasti, které zakrývají objekty našeho zájmu.

Modifikování vizualizační metody: Uživatel může během programu parametry projekční metody.

1.4 Charakteristika dat

Cílem vizualizace biologických údajů je zobrazení preparátu jeho vnitřní struktury. Data dostáváme ve většině případů jako sérii dvojrozměrných obrazů reprezentujících jednotlivých řezy materiálem. Obrazy jsou většinou zarovnány podél os x, y a z. Barevná hloubka je většinou 8 nebo 12 bitů. Velikost dat je většinou 128*128, 256*256,512*512 nebo 1024*1024 8 nebo 12 bitových obrazů. Tyto nastavení ovlivňují celkovou velikost série dat(Například série dat 1024 * 1024 12-bitových obrázků zabere na diskovém poli přibližně 50 MB místa). Hodnota pro daný pixel reprezentuje

úroveň jasu nebo hustoty tj. neobsahuje žádné informace o barevných složkách (například pro barevný model RGB). V praxi však pracujeme s mnohem většími daty jejichž velikost nezřídka přesahuje 500 MB. Obrázky jsou hrubá, nezpracovaná data a jako taková jsou vstupem pro náš program.

1.5 Předzpracování dat

Ještě před samotnou vizualizací dat je důležité jejich předzpracování. Data mohou být upravena pro zlepšení kvality obrázků. Jedna z nejrozšířenějších metod je dekonvoluce. Jejich aplikací dostáváme obrázky s ostřejší definicí, lepším kontrastem a vylepšeným poměrem signál/šum. Dekonvoluce je základem vysokých detailů pro zobrazení struktur. Dekonvoluční algoritmy se používají v tradiční fluorescenční mikroskopii. Její univerzální použití je stále náročné kvůli určení strojové funkce a obnovy obrázku pomocí získané strojové funkce. V neposlední řadě musíme brát v úvahu velké množství dat (512 * 512 * 16). Můžeme použít různé filtry na celou množinu dat s cílem odstranit artefakty, vyhladit nebo zaostřit obraz, nebo upravit jas a kontrast. Je důležité si uvědomit, že aplikace těchto filtrů radikálně ovlivní prostorovou rekonstrukci. Proto se v některých případech používají jen za účelem zobrazení a výpočty (kvalitativní měření) jsou aplikovány na nezpracované data. Ještě před trojrozměrnou rekonstrukcí vysegmentovat objekty našeho zkoumání. Identifikovat zajímavé voxely, které později zobrazíme. Při běžné barevné hloubce 12 bitů můžeme zbylé čtyři bity z celkově šestnácti bitů používat na identifikaci až 16 objektů v prostoru. Tuto dodatečnou informaci můžeme použít během generování výsledného obrazu. Různým objektům přiřadíme různou barvu, abychom zdůraznily oblasti našeho zájmu. V praxi je ovšem velice obtížné stanovit hranice mezi jednotlivými objekty, takže metodu dekonvoluce nepoužijeme pro filtraci dat, ale pouze pro stanovení hranic objektů (orientace v prostoru).

1.6 Problémy při vizualizaci

Během vizualizace objemových dat se mohou vyskytnout tyto základní problémy.

1. Velikost dat. Vzhledem k velkým rozměrům dat je nutné použít efektivní metody pro zpracování takových údajů.
2. Nízký kontrast, malé změny intenzit, špatný poměr mezi signálem a šumem. Kontrast se může snižovat se zvětšujícím se počtem prvků promítnutých do jednoho místa. Přímočará segmentace objektu našeho zájmu je prakticky nemožná, protože všechny techniky (prahování, rozdíly barev, homogennost regionů) jsou založeny na binárním rozhodnutí bez ohledu na to, jestli voxel do struktury patří nebo ne.
3. Různé rozlišení v rovině X, Y a Z. Vizualizační metoda musí pracovat s voxely obecných rozměrů. To může přinášet různé zobrazovací chyby a artefakty spojené s interpolací a zvětšením dat v paměti (tak, že je nedokážeme zpracovat na běžném počítači).
4. Zkoumáme neznámé struktury. Vizualizační metoda by měla potlačit artefakty v co největším rozsahu. Pozorovatel často nemá zkušenost s neznámou strukturou (co je a není správně zobrazeno), proto zobrazování artefaktů má velice špatný dopad na správné pochopení tvaru objektu. Pomocí vhodně zvoleného osvětlovacího modelu získáme jasnou představu o objektu. Zde se ovšem dostává do popředí otázka rychlosti vizualizace. Její parametry se musí zjišťovat interaktivně sérií pokusů. Tento postup může trvat velice dlouho, protože na zobrazení nového výsledku musí uživatel čekat i několik minut. Všeobecně prohlídka neznámého objektu(potenciálně velice komplikovaného) vyžaduje změny pozice a směru kamery, osvětlení a vizualizačních metod.

2 Volumetrická vizualizace

Volumeterická vizualizace je o pochopení komplexních vícerozměrných dat. Ve své práci rozděluji ve všeobecnosti proces volumetrické vizualizace de tří kroků. Prvním krok představuje předzpracování údajů. Druhý krok závisí na použité technice. Dvojrozměrné obrazy (řezy) jsou namapovány na trojrozměrnou mřížku. Tímto vytvoříme trojrozměrná data.

> Algoritmy zobrazující povrchy (surface based techniques) vytváří pomocnou geometrickou reprezentaci povrchu. V trojrozměrných datech se hledají hrany a body povrchu. Z nich se interpoluje povrch dvojrozměrnými záplatami. Jde o nepřímou metodu.
>
> Objemové algoritmy (volume based techniques) vykreslují objemová data přímo. Využívají plnou prostorovou informaci. Namísto pomocné geometrické reprezentace povrchu můžeme každému vzorku přiřadit barvu a průhlednost.

Třetím krokem je vygenerování výsledného obrazu. Pomocný povrch se zobrazuje tradičními metodami počítačové grafiky, zatímco reprezentace pomocných barev a neprůhlednost se zobrazuje pravým objemovým renderováním (například pomocí metody sledování paprsku).

2.1 Vizualizační metody

Existuje mnoho vizualizačních metod, které se v praxi používají s různým úspěchem. Můžeme je rozdělit podle více kritérií. Jedno dělení je ale společné, a to rozdělení na algoritmy zobrazující povrchy a objemové algoritmy.

2.1.1 Algoritmy zobrazující povrchy

Tyto metody se snaží z objemových dat dat aproximovat povrch pomocí geometrických primitiv a ty zobrazovat pomocí dobře známých metod počítačové grafiky. Nejznámější algoritmy jsou:

> sledování obrysů.
> kráčející krychle (marching cubes)
> marching tetrahedra
> rozdělení krychlí (dividing cubes)
> povrchové kostky (opaque cubes, cuberille)

Algoritmy vytvářejí pomocnou geometrickou reprezentaci povrchu, kterou potom umíme rychle, díky redukci dat zobrazit standardními grafickými akcelerátory. Informace o vnitřku objektu se však ztratí. Ve všeobecnosti se v těchto metodách dělá při každém vzorku test příslušnosti k povrchu objektu. Při špatně navzorkovaných datech nebo datech amorfní objekty (mlha a jiné) proto vznikají chyby. Ty se vyskytují buď ve formě děr, které neexistují v původním objektu, nebo ve formě falešných stěn.

2.1.2 Objemové algoritmy

Proti předcházejícím algoritmům, objemové algoritmy využívají plnou prostorovou informaci na vygenerování výsledného obrazu a nezávisí na složitosti scény. Každý vzorek má přiřazenu barvu a průhlednost. Tyto údaje jsou později složeny do výsledné barvy obrazu. Při výpočtu používají celou trojrozměrnou mřížku dat, jsou tedy velice náročné na velikost paměťi a výkon procesoru počítače. Každé renderování výsledného obrazu vžaduje kompletní průchod celým objemem dat. Dokážeme však zobrazit libovolný vnitřní detail a amorfní objekty. Celkově tyto algoritmy poskytují více informací jako algoritmy používající pomocnou reprezentaci povrchu.

Kvůli lepší interaktivitě byly představeny mnohé urychlení a optimalizace. Náhodné vzorkování dat nebo průchod v nízkém rozlišení se někdy používá pro rychlé vygenerování obrázků ve špatné kvalitě. Pokud uživatel nemění parametry zobrazovací metody, znovu vygenerujeme obraz, tentokrát ve vyšším rozlišení. Tento proces postupného zvyšování rozlišení a kvality výsledného obrazu se nazývá postupné zlepšování (progressive refinement). Jiná možnost je použití speciálního hardware. Přehled možných urychlení vzpomínáme v části 2.4 nebo konkrétně v kapitole 4.

Objemové algoritmy můžeme rozdělit podle funkce, kterou provádíme klasifikaci dat na:

binárně
pravděpododnostně

Binární algoritmy pokrývají každý voxel (nějakým objektem) buď úplně, nebo vůbec. Jsou to povrchově orientované algoritmy. Pravděpodobnostní algoritmy (také známé pod názvem poloprůhledné renderovací algoritmy) přiřazují voxelům procentuální podíl nějakého objektu. Jsou založeny na hromadění příspěvků ode všech vzorků podél paprsku do jednoho obrazového pixelu. Hodně autorů se při objemovém zobrazování omezuje jen na tuto skupinu algoritmů. Pravděpodobnostní algoritmy patří pod objemově orientované techniky.
Další důležité rozdělení je podle pořadí, v jakém jsou voxely postupně zpracovány při tvorbě výsledného obrazu. Jinak řečeno, je to dělení podle domény algoritmu.

Algoritmy pracující v obrazovém prostoru
Pro každý pixel výsledného obrazu se hledají voxely v objektovém prostoru, které přispívají do výsledné barvy bodu. Vypočtená pozice většinou padne mimo vrcholy mřížky, takže potřebnou hodnotu musíme zjišťovat interpolací. Do této skupiny můžeme zařadit následující algoritmy:

Sledování paprsku (ray-casting, ray-tracing)
Sábelova metoda

Algoritmy pracující v objektovém prostoru
Pro každý voxel objektového prostoru hledáme pixely výsledného obrazu, které daný voxel ovlivní. Splatting je technika procházející celý objektový prostor. Na každý voxel je aplikována trojrozměrná rekonstrukční funkce. Příspěvek jednotlivých voxelů se hromadí do obrazového prostoru. Fungování algoritmu si můžeme představit jako házení sněhových koulí na skleněnou plochu. Příspěvek intenzity ve středu voxelu bude nejvyšší a postupně bude klesat s rostoucí vzdáleností od místa dopadu. Algoritmy:

V-buffer
Splatting

Algoritmy pracující na hybridním principu
Existují i hybridní algoritmy, které kombinují výhody obou předcházejících přístupů. Příkladem je algoritmus Shear-warp tj. posun a pokryv, který je považován za nejrychlejší, čistě softwarový renderovací algoritmus. Myšlenka algoritmu spočívá v posunutí (shear) jednotlivých řezů v objektovém prostoru tak, aby jejich namapování (kompozice) na dvoudimenzionální zobrazovací prostor prostor bylo jednoduché a rychlé – vzorky podél paprsků budou ležet přesně v rovinách řezu. Průchod daty vyžaduje jednoduché adresování a 2D převzorkovací filtr. Po projekci zdeformovaný obraz upravíme zpět (warp). Algoritmus je velmi rychlý i na běžně dostupných počítačích. Na druhé straně však produkuje artefakty,je pohledově závislý a kvalita výsledného obrazu je nízká. Nižší kvalita výsledného obrazu je způsobena:

Algoritmus obsahuje až dva kroky převzorkování. Vícenásobné převzorkování může způsobit rozmazání a ztrátu detailních informací.
Rekonstrukční filtr je jen dvojrozměrný. V řezech se používá jen bilineární interpolace, mezi řezy jen interpolace s nejbližším sousedním voxelem. Tento bod je hlavní nevýhodou algoritm u shear-warp.

Počet paprsků je roven počtu voxelů jednoho řezu, algoritmus produkuje alias kvůli podvzorkování

Byly představeny některé optimalizace, ale výsledná kvalita obrazu stále zaostává za ostatními algoritmy např. ray-castingem.

2.1.3 Porovnání vizualizačních metod

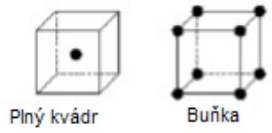
V dnešní době je snaha vyhnou se algoritmům zobrazujícím povrchy a používat objemové algoritmy. Tyto umíme vyladit, aby výsledek byl stejný, jako při algoritmech používajících pomocnou reprezentaci povrchu. Rychlé algoritmy jako Shear-warp produkují obraz v nízké kvalitě. Binární objemové algoritmy jsou snadno implementovatelné a velké množství paměti v porovnání s pravděpodobnostními. Trpí však nedostatky jako algoritmy zobrazující povrchy. Při binární klasifikaci, respektive testování, jestli vzorky patří povrchu, můžou vzniknout falešné povrchy, nebo neexistující díry. Ray-casting a splatting dávají stejně kvalitní výsledky. Čas potřebný pro vyrenderování výsledného obrazu závisí na typu dat a klasifikaci (přechodová funkce). Implementace voxel splattingu sice není přímočará, ale umožňuje jednodušší a kvalitnější úpravu dat a chyb v datech. Tato metoda umožňuje také poměrně kvalitní vyrenderování výsledného obrazu. Nevýhodou této metody je poměrně velká náročnost na paměti výkon procesoru. Ovšem relativní jednoduchost, schopnost zobrazovat celý objem dat, možnosti optimalizace, hardwarovou nezávislost a další důvody, které byly uvedeny dříve nás utvrdily v rozhodnutí implementovat tento algoritmus v našem programu. Dobré shrnutí a přehled algoritmů pro objemovou vizualizaci je možno najít v [1].

2.2 Základy

V této části jsou obsaženy některé informace týkající se souřadných systémů, klasifikace, stínování, principy převzorkování, interpolace a kompozice. Tyto techniky jsou základem metody voxel splattingu. Následující části vychází z [2].

2.2.1 Vícerozměrná reprezentace dat

Jak už bylo dříve řečeno, údaje (sada vzorků), se kterými pracujeme chápeme jako sérii řezů. Každý řez je ve skutečnosti dvojrozměrné pole čísel reprezentujících jistou hodnotu (např. Intenzitu). Celá série představuje trojrozměrné pole elementárních objemových jednotek, společně nazývaných voxel (voxel je objemový element, analogie dvojrozměrného pixelu). Má strukturovanou podobu a je reprezentovaný pravidelnou nebo nepravidelnou mřížkou.

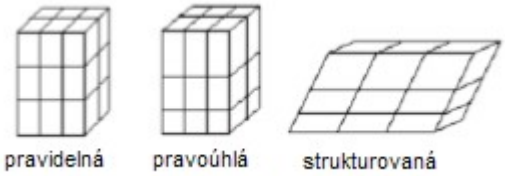


Obrázek 2.2.1.1: Reprezentace objemových dat

Jak interpretovat voxely?

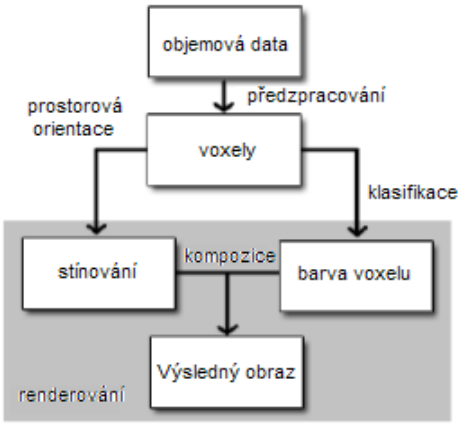
Můžeme je chápat jako plný kvádr konstantní hodnoty nebo bod v mřížce (rastru). V našem textu budeme používat první interpretaci, tj. voxel je kvádr konstantní hodnoty dané vlastnosti, i když obrázky vygenerované tímto způsobem nejsou většinou tak hladké jako v případě interpretace druhým způsobem tj. hodnota je známa pouze ve vrcholech. Ovšem při této reprezentaci máme hodnoty dané vlastnosti vždy pevně dány a nemusíme interpolací dopočítávat, jaké jsou požadované hodnoty mezi jednotlivými vrcholy. Námi použitá metoda je jednodušší a problémy a nepřesnosti spojené s tímto pojetím lze řešit.

Objemové data nemusí být nutně definovány jen v kartézské soustavě. Spearay a Kennon [3] rozdělili všechny mřížky podle jejich geometrických tvarů do sedmi tříd: *kartézská, pravidelná, pravoúhlá, strukturovaná, nestrukturovaná, blokově strukturovaná a hybridní mřížka.*



Obrázek 2.2.1.2: Třídy soustav

V dalším textu se budeme zabývat pouze pravidelnými mřížkami.
Na údaje v trojrozměrném prostoru a hodnoty pixelů na obrazovce budeme odkazovat pomocí souřadných systémů. Předpokládáme, že voxely leží ve vrcholech trojrozměrné mřížky v pravotočivé souřadné soustavě, označované jako objektový prostor. Voxel se nachází v souřadnicích (x, y, z). Souřadný systém obrazu, který budeme generovat, budeme označovat jako zobrazovací prostor. Pro lepší pochopení samotné vizualizace se podíváme trochu blíž na postup renderování. Zde je nutné poznamenat, že tento jednoduchý popis se vztahuje na algoritmus voxel splattingu, ale je platný i pro jiné vykreslovací algoritmy. Vstupem pro náš program je trojrozměrné pole voxelů, reprezentované trojrozměrnou mřížkou. Vzhledem k tomu, že poloha kamery může být ve kterémkoli bodě v prostoru je nutno provést jednu aproximaci. Voxel jež je definován pro určitý vrchol v trojrozměrné mřížce má jistou vlastnost např. intenzitu. Tato intenzita bude ovšem platit pro celou oblast prostoru, která je vymezena velikostí voxelu. Tato oblast může mít v každém směru jinou velikost. Pokud by jsme nevzali v potaz tento fakt, při pohybu kamery nebo pozorovatele by se ve výsledném obraze objevovali pouze body, nikoli však průměty několika voxelů ležících za sebou vzhledem ke směru pozorovatele. Po provedení této aproximace je možno provést převzorkování (vygenerování adresy a interpolaci). Převzorkováním můžeme částečně změnit vlastnosti zobrazovaného objemu. Po provedení této aproximace bude mít každý voxel určen svou polohu v prostoru a adresu. Vygenerovanou adresu zároveň chápeme jako určení pozice v objektovém prostoru námi zkoumaných dat. Během vykreslení přiřadíme každému voxelu jeho skutečnou pozici, barvu a průhlednost a přeneseme je podle pozice pozorovatele na výsledný obraz. Výsledný obraz je namapován na obrazovku.
Na obrázku je blokové schéma algoritmu voxel splatting. V dalších částech budeme podrobně mluvit o operacích jako klasifikace, mapování a stínování. Mimo toho se ještě vrátíme k počáteční úpravě dat a průchodu skrz objemová data. Předzpracování dat bylo zmíněno již v první kapitole.



Obrázek 2.2.1.3: Schéma vykreslovacího procesu

2.2.2 Klasifikace

Klasifikace je proces přiřazení barvy a průhlednosti zrekonstruovaným vzorkům. Pro tento účel se většinou používá vyhledávací funkce. Ve většině případů bývá implementována pomocí tabulek nebo např. Jako Beziérova křivka. Při interaktivní vizualizaci požadujeme možnost změny vyhledávací funkce během provádění programu. Pro její rozumné nastavení na vliv několik faktorů: uživatelova znalost materiálu, pozice a množství elementů v objektovém prostoru. Za nejtěžší jsou považována nastavení, které musí udělat uživatel. Pokud je obeznámen s materiálem, specifikace vyhledávací funkce je relativně jednoduchá a rychlá. V opačném případě tento krok zahrnuje rozsáhlé zkoumání materiálu, případně konzultace s osobou, která se v charakteristice materiálu vyzná.

Dodatečná klasifikace

Objemové algoritmy se také odlišují způsobem, jakým vyčíslují barvu a průhlednost vzorků. Hodnota vlastnosti vzorku je dána a výsledná barva je určena pomocí vyhledávací funkce. Pokud by toto určení nestačilo pro potřeby výpočtu, dodatečné hodnoty můžeme určit interpolací. Pořadí interpolace a aplikace vyhledávací funkce definujeme jako předběžnou a dodatečnou klasifikaci. Předběžná klasifikace je předpočítání hodnot před vlastní renderováním. Dodatečnou klasifikací rozumíme výpočet hodnoty ze sousedních voxelů až během renderování a to pouze pokud je to potřeba. Předběžná a dodatečná klasifikace ve většině případů vyprodukuje stejné výsledky. Rozdíl je pouze v rychlosti vykreslení a paměťovou náročností.

2.2.3 Průchod daty

Jak bylo dříve vzpomenuto, algoritmus voxel spalttin spadá do kategorie algoritmů pracujících v obrazovém prostoru. Postupně procházíme všechny voxely datového prostoru ve vhodném pořadí tak, aby se hodnoty ve zobrazovacím prostoru (obrazovce) naakumulovaly ve správném pořadí.

2.2.4 Stínování

Stínování je důležitá součást objemové vizualizace. Umožňuje vyprodukovat výrazově realistické výsledky. Světlo při cestě od zdroje mění své barevné složení (při odrazech a lomech). Po dopadu na povrch tělesa se světlo rozptýlí do všech směrů. Matematická funkce, která vyjadřuje intenzitu rozptýleného paprsku v závislosti na jeho směru, intenzity a vlnové délky dopadajícího světla se nazývá odrazová funkce a je základem osvětlovacích modelů. Čím lepší osvětlovací model máme k dispozici pro popis chování světla, tím realističtější výsledky dostaneme. V počítačové grafice se používají dva přístupy.

První přístup vychází z fyzikální podstaty šíření světla a tento model nazýváme globální osvětlovací model. Bere v úvahu vzájemné vztahy mezi objekty v prostoru a dává kvalitní výsledky. V počítačové grafice patří mezi výpočtově náročnější problémy.

Druhý přístup se snaží tento fyzikální děj obejít (ale kvalitou se přiblížit fyzikálnímu modelu) pomocí uměle vytvořených osvětlovacích modelů nesoucích společný název lokální osvětlovací modely. Tyto modely dávají dobré výsledky, které se přibližují skutečnému chování světla a jsou výpočtově mnohem jednodušší jak globální osvětlovací metody. Jejich výpočet je založen na normále povrchu v bodě dopadu světla, pozice světelného zdroje a pozorovatele. Ve většině algoritmů zobrazujících objem se nepoužívají globální osvětlovací modely. Hlavním důvodem je potřeba rychlého a interaktivního zobrazení zkoumaného objektu. Někdy použití globálního osvětlovacího modelu může mít negativní efekt ve formě nesprávné interpretace zobrazeného objektu. Proto se v této práci budeme dále zabývat pouze lokálními osvětlovacími modely.

Ve většině aplikací postačuje známý Phongův osvětlovací model [4].

Odražené světlo se skládá ze tří složek:

Ambientní složka I_A reprezentuje okolní světlo a dané oblasti přicházející ze všech směrů, které je příliš složité modelovat. Tato konstantní složka závisí na intenzitě okolního světla

I_A a konstanty k_A, která vyjadřuje schopnost povrchu odrážet okolní světlo ($0 < k_A < 1$).

Difůzní složka I_D reprezentuje odražené světlo. Nezávisí na poloze pozorovatele a pravděpodobnost nového směru paprsku je stejná pro všechny směry. Velikost difůzní složky závisí na úhlu dopadu a k_d koeficientu difůzního odrazu ($0 < k_d < 1$). Difůzní světlo přináší informace o barvě povrchu.

Zrcadlová složka I_s reprezentuje odlesk na povrchu tělesa. Závisí na poloze pozorovatele, vektoru R symetrického k vektoru dopadu podle normály a exponentu n, vyjadřujícího ostrost zrcadlového odrazu. Odlesk může mít jinou barvu, než povrch tělesa. Při nízké hodnotě exponentu n je možno použít jednoduchou aproximaci, při které je použito pouze sčítání, odčítání, násobení a dělení.

Celkově světlo vnímané pozorovatelem můžeme vyjádřit jako:

$$I = I_A + I_D + I_S$$

Mějme světlo I_l je jednobarevné světlo. Odražené světlo můžeme vypočítat upraveným Phongovým osvětlovacím modelem pomocí Schlickovy aproximace.

$$I = I_a k_a + I_l (K_d MAX(L*N, 0) + \frac{k_s * t}{n - nt + t})$$

Koeficient t vypočteme jako t = MAX(N*H, 0)

$$H = \frac{(L+V)}{|L+V|}$$

Z předcházejícího textu víme jak určit barvu bodu osvětlené plochy. Výpočet pro všechny body na ploše by byly časově velmin áročné, proto vzniky metody na urychlení tohoto výpočtu. Tyto metody se nazývají stínování. Pomocí osvětlovacího modelu určíme barvu jen v některých bodech plochy a zbytek odvodíme. V praxi se většinou používá Gouardovo a Phongovo stínování. Gouardovo stínování je založeno na interpolaci barev nebo jasu. Phong navrhl interpolovat normálové vektory. Toto stínování dává lepší výsledky jako Gouardovo stínování, je však výpočtově mnohem náročnější. Normálu imaginárního povrchu předcházejícím bodem aproximujeme pomocí gradientu.

$$N(x_i, y_j, z_k) = \frac{\nabla f(x_i, y_j, z_k)}{|\nabla f(x_i, y_j, z_k)|}$$

Výpočet gradientu $\nabla f(x_i, y_j, z_k)$ je časově velice náročná operace. Na jeho určení se často používá vzorec centrální diference:

$$\nabla f(x_i, y_j, z_k) \approx \frac{1}{2}$$
$$f(x_{i+1}, y_j, z_k) + f(x_{i-1}, y_j, z_k),$$
$$f(x_i, y_{j+1}, z_k) + f(x_i, y_{j-1}, z_k),$$
$$f(x_i, y_j, z_{k+1}) + f(x_i, y_j, z_{k-1})$$

Tento výpočet prvního stupně (6 sousedních prvků) je citlivý na šum a aliasy artefaktů. Použití výpočtu druhého stupně (26 sousedních prvků) dává lepší výsledky. S cílem dosahovat vykreslování

v reálném čase se používají i analytické metody určení gradientu, kde měníme kvalitu výsledku za rychlost výpočtu.

2.2.5 Předvzorkování a interpolace

Převzorkování (resampling) je proces transofmace diskrétního signálu (např. Obrázku) z jedné soustavy do druhé. Jeho hlavní použití je při navzorkování trojrozměrných dat za účelem jejich projekce do dvojrozměrné roviny. Převzorkování se skládá ze dvou kroků – vygenerování adresy a interpolace. Vygenerování adresy nám určuje pozici vzorku v mřížce (datech). Interpolace není nutná operace, protože pozice, odkud chceme získat hodnotu je vždy ve vrcholech mřížky (voxelů). Interpolaci provádíme pouze v případě, že chceme upravit vstupní objemová data do jiné mřížky. Při úpravě dat musíme z okolních hodnot dopočítat právě upravovaný vrchol. Takto upravená data ovšem můžou obsahovat chyby vzniklé při výpočtech. Při úpravě vzpomeneme jeden poznatek z teorie signálů. Frekvenčně omezená funkce, která splňuje Nyquistovo kritérium, může být přesně zrekonstruovaná použitím ideálního rekonstrukčního filtru sinc(x). Funkce sinc(x) je definována na nekonečném intervalu, proto potřebujeme nekonečné množství čísel na zrekonstruování mezivzorku. Obraz má ovšem konečné množství elementů, takže pro rekonstrukci musíme použít jiné funkce:

Nejbližší soused
Lineární interpolace
Interpolace vyššího stupně (kubické splajny)

Interpolace vyššího stupně dávají lepší výsledky za cenu pomalejšího výpočtu. Nejjednodušší interpolační funkce je interpolace nultého stupně, známou pod názvem interpolace hodnotou nejbližšího souseda. Hodnota vzorku je hodnotou nejbližšího souseda vzhledem k pozici aktuálního vzorku. Interpolace prvního stupně je trojrozměrná lineární interpolace (trilineární interpolace) při výpočtu bere v úvahu hodnotu všech okolních vzorků.

2.2.6 Kompozice

Nejpozději po převzorkování obsahují všechny vzorky svou vlastní hodnotu a dokážeme jí přiřadit barvu a průhlednost. Nízká hodnota průhlednosti patří objektům jasně viditelným, vysoká hodnota průhlednosti je přiřazena průhledným materiálům. V programu jsou tyto hodnoty invertovány z důvodu jednoduššího uživatelského nastavení. Kombinací těchto hodnot získá nakonec každý pixel svou výslednou barvu. Jinak řečeno, kompozicí vyprodukujeme dvojrozměrnou projekci těchto údajů. Bohužel díky zvolené metodě vykreslení je řešení lomů světla, odrazů a dalších vlastností osvětlení velice komplikovaná tématika a nebylo by možné objekt vyrenderovat interaktivně. Z těchto důvodů od řešení těchto vlastností osvětlení upouštíme.

Odzadu dopředu (back to front)

Z principu vlastní metody voxel splattingu vyplývá, že celá kompozice začíná nejvzdálenějšími vzorky a postupuje se směrem k pozorovateli. Předpokládejme, že máme propustnost objektů o intenzitě I_i a průhlednosti P_i označených od nejvzdálenějších objektů k nejbližším vzhledem k pozorovateli. Intenzita pozadí je $I_0 = N_0$. N_i reprezentuje nahromaděnou intenzitu během kompozice. Jeden krok kompozice v metodě odzadu dopředu můžeme zapsat jako:

$$N_i = P_i N_{i-1} + I_i$$

Hodnota N_n pixelu ve výsledném obraze je dána vztahem:

$$N_n = \sum_{k=0}^{n} I_k \prod_{l=k+1}^{n} P_l$$

Existuje i jiný náhled na kompozici odzadu dopředu, ale princip zůstává stejný.
našem programu. Dobré shrnutí a přehled algoritmů pro objemovou vizualizaci je možno najít v [1].

2.3 Voxel splatting

Voxel splatting nepatří k příliš používaným metodám. V našem programu používáme právě tento algoritmus pro generování výsledného obrazu.

Obrázek 2.3.1: Zobrazení voxelů

Každý voxel objektového prostoru promítáme na plochu zobrazovacího prostoru. Všechny voxely jsou postupně naneseny od nejvzdálenějšího k nejbližšímu vzhledem k pozici pozorovatele. Při operaci nanášení je každému voxelu přidělena barva a průhlednost pomocí klasifikace a také je plocha tohoto voxelu prostorově nasměrována. Tato orientace slouží pro stínování voxelu. Výsledná barva pixelů v zobrazovacím prostoru je tedy dána množinou voxelů, které jsou na něj promítnuty, jejich pořadím ve smyslu od nejvzdálenějšího k nejbližšímu a orientací každého z těchto voxelů. Voxel splatting zdědil většinu výhod i nevýhod objemových algoritmů. Je velice náročný na velikost paměti a rychlost procesoru počítače, ale obrázky jsou kvalitní a zobrazují celý objem dat, ne jen množinu tenkých povrchů jako v případě algoritmů zobrazujících povrchy (surface fitting algorithms). Tento algoritmus je možno také paralelizovat na úrovni voxelů (každý voxel je nezávislý) a dat (objekt je možno rozdělit na více částí, aplikovat algoritmus a výsledky zkombinovat). Je možné předpočítat gradienty pro jednotlivé voxely ve stádiu předzpracování. Prostorová orientace jednotlivých voxelů nezávisí na poloze pozorovatele a osvětlení, takže je platné neustále pro celý prostor objemových dat. Nevýhodou ovšem je, že metoda, která je již v základní podobě náročná na velikost paměti počítače spotřebuje další velké množství této paměti a to dokonce několikanásobně. Ovšem takto předpočítaná data se při použití této metody používají prakticky vždy, protože výpočet orientace pro každý voxel by byla časově velice náročná.
Tyto optimalizace jsou ovšem nutné v rámci požadavku na nutnost interaktivního ovládání a vykreslení objemových dat.

2.4 Zvyšování kvality a rychlosti

Základní přímočará implementace voxel splattingu je výpočetně velice náročná. Přestože je k dispozici stále rychlejší hardware a možnost paralelizace výpočtu, metoda je stále velice náročná a v základní podobě nepracuje interaktivně. V této části se seznámíme s několika technikami urychlujícími algoritmus.

2.4.1 Urychlení voxel splattingu

Strategie urychlující algoritmus voxel splatting jsou založeny například na předpočítání parametrů dat ještě před samotnou kompozicí, optimalizaci nanášení obrazů voxelů, vynechání nezobrazených

oblastí. Případně můžeme vyměnit kvalitu vygenerovaného obrazu za rychlost. V následujících bodech nastíníme nejdůležitější přístupy.

1. Předzpracování údajů: některé kroky renderovacího algoritmu můžou být vypočteny ve fázi předzpracování. Například stínování a prostorovou orientaci lze vypočítat ve fázi předzpracování. Takové předzpracování občas může způsobit drobné artefakty. Nejčastěji se týkají nesprávného stínování.

2. Optimalizace nanášení obrazů voxelů: Voxely jsou nanášeny od nejvzdálenějšího k nejbližšímu vzhledem k pozorovateli. Tento způsob předpokládá stav, ve kterém jsou voxely neustále seřazeny podle vzdálenosti od pozorovatele. Neustálé řazení všech voxelů je výpočetně velice náročné. Je možno rozdělit řešení algoritmu na tři stavy, ve kterých je pozice pozorovatele vždy nejvíce přikloněna k některé ze souřadných os. Takto upravené řešení algoritmu již umožňuje částečně vypustit řazení vzdáleností voxelů. Postačí seřadit jednotlivé vrstvy voxelů ležící v rovině kolmé k přikloněné souřadné ose. Toto řešení je mnohem rychlejší, ale pokud znovu aplikujeme možnost rozdělit řešení na několik stavů, vyhneme seřazení voxelů. Toto řešení je již rychlé a efektivní.

3. Vynechání nezobrazených oblastí: Během klasifikace se zjišťuje barva a průhlednost konkrétního voxelu. Dané vlastnosti se poté aplikují na daný voxel. Může nastat situace, že danému voxelu je během klasifikace není přidělena žádná barva. Pokud nastane tento stav, nemusí být voxel vyrenderován. Vynecháním každého takového vzorku se výsledný obraz nezmění, ale celkové vykreslení se značně urychlí.

4. Zjednodušení vizualizačního algoritmu: Toto urychlení zahrnuje zanedbání (např. stínování) nebo zjednodušení (trilineární interpolace nebo interpolace nejbližším sousedem) některých výpočtů. Patří sem i různé implementační "finty", mezi které patří použití vyhledávacích tabulek, při stínování použití celočíselných výpočtů místo výpočtů s reálnými čísly. Většina těchto technik je přizpůsobena na konkrétní architekturu počítače (bereme v úvahu množství dostupné paměti a cenu počítání s čísly s pohyblivou desetinnou čárkou).

5. Progresivní zlepšování kvality obrazu: Rychle vyrenderujeme obraz nižší kvality, pokud uživatel mění parametry zobrazovací metody. Jinak renderujeme obraz ve vyšší kvalitě. Obraz s vyšší kvalitou obsahuje obrazy voxelů definované jako plocha s různou sytostí barvy. Barva ve středu je nejintenzivnější a postupně směrem k okraji slábne. Intenzita barvy na okraji obrazu voxelu je nulová. Průběh intenzity barvy mezi středem a okrajem klesá po gaussove křivce. Každý voxel během klasifikace získá svou vlastní barvu a průhlednost. Tyto vlastnosti jsou zkombinovány s obecnou barvou voxelu. Takto zobrazený voxel je celý pokryt přidělenou barvou a zároveň tato barva směrem ze středu k okraji slábne. Takto zobrazený voxel odpovídá mnohem lépe skutečnému modelu voxelu.

2.4.2 Zvyšování kvality obrazu

Mimo již zmíněné progresivní techniky zvýšení kvality obrazu jsou použity některé další metody. Během vykreslení jsou všechny nanesené voxely stínovány. Pokud jsou během vykreslení měněny parametry zobrazovací metody, každému voxelu je přiřazena prostorová orientace pouze jako aproximace skutečné orientace. Pokud se vykresluje obraz ve vyšší kvalitě, voxelům je přiřazena směrová orientace s vyšší přesností.

3 Renderování

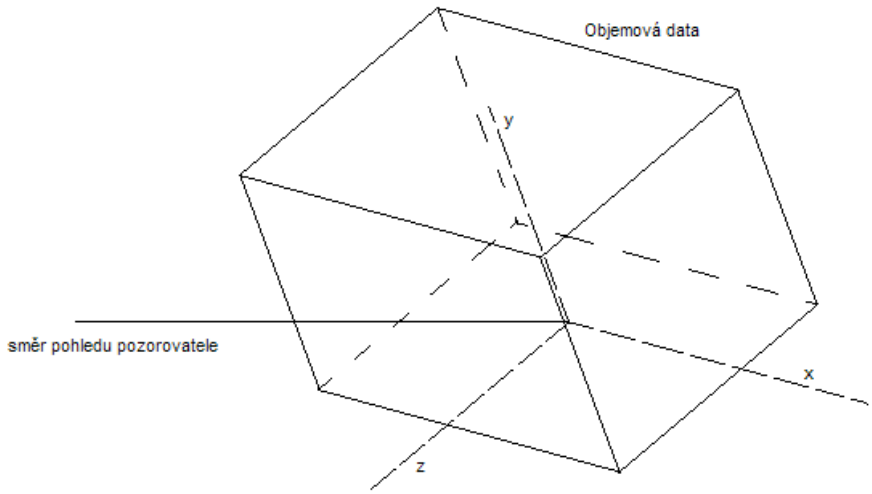
Vizualizace vícerozměrných dat je výpočetně velmi náročný proces. Pokud jsou vstupní objemová data velkých rozměrů, paměť a výpočetní výkon sekvenčních procesorů brzdí interaktivitu celého programu. V takových případech se velice často sahá k paralelizaci celé úlohy nebo alespoň paralelizaci vykreslení.

3.1 Dekompozice

Prvním krokem při tvorbě je rozdělit daný problém na menší podproblémy. Tyto samostatné celky jsou řešitelné samostatně. Mezi nejdůležitější bloky patří algoritmus řazení voxelů, výpočet průmětu voxelu, stínování a předzpracování dat. Systém vykreslení v našem programu je dekomponován na tyto základní části které zde důkladně vysvětlíme.

3.1.1 Řazení voxelů

Problematika řazení voxelů již byla nastíněna v předcházející kapitole. Pokud by jsme používali nejednodušší způsob tj. při každém vykreslení by jsme všechny voxely seřadily podle vzdálenosti od nejvzdálenějšího k nejbližšímu vzhledem k pozorovateli, vykreslení by nemohlo probíhat v reálném čase. Musíme tedy použít optimalizaci tohoto přístupu.

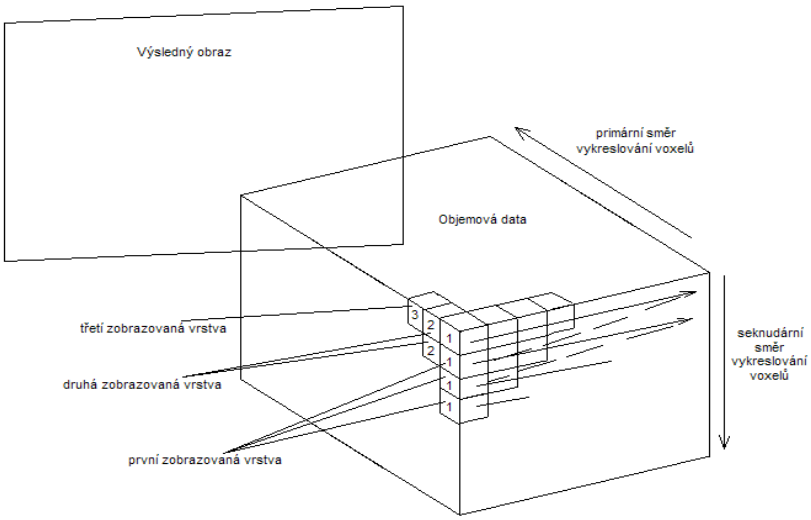


Obrázek 3.1.1.1: Pohled pozorovatele

V následujícím výpočtu je velice důležitý směr pohledu pozorovatele. Ten může být vyjádřen ve tvaru (x, y, z), kde x, y a z jsou souřadnice směrového vektoru jehož velikost je 1. Takto vyjádřený směr pohledu je téměř vždy více přikloněn k některé ze souřadných os. Výjimku tvoří stav, kdy souřadnice x, y i z nabývají stejných hodnot, avšak i tato výjimka je algoritmem řešitelná. S touto znalostí můžeme přistoupit k řešení vlastního problému. Nyní je možno nanášet voxely postupně ve vrstvách kolmých na souřadnou osu, ke které je směr pohledu pozorovatele nejvíce přikloněn. Vykreslování probíhá od nejvzdálenější vrstvy k nejbližší, přičemž takto upravený postup nanášení je plně korektní. Žádné voxely nejsou renderovány v nesprávném pořadí. Nevýhodou stále zůstává nutnost řadit jednotlivé voxely ve vrstvách. Abychom se vyhnuli potřebě řazení voxelů ve vrstvách, zjistíme ke které ze zbylých souřadných os je směr pohledu pozorovatele více přikloněn. Poté je již zřejmý způsob nanášení voxelů ve vrstvách. Daná vrstva je rozdělena na jednotlivé sloupce nebo řádky voxelů, které jsou v přímo za sebou vykreslovány.
Celý urychlený algoritmus probíhá následovně:

1. Nejprve získáme primární směr vykreslování voxelů. Ten je zjištěn ze souřadnic vektoru směru pohledu pozorovatele. Pořadí vykreslení vrstev je nyní dáno primárním směrem.

2. Následně určíme sekundární orientaci renderování voxelů, který opět získáme ze směru pohledu pozorovatele. Sekundární orientací je dokončen výpočet pomyslného řazení voxelů.
3. Každý voxel je vykreslen podle pořadí určeného primárním a sekundárním směrem. Pořadí renderování voxelů sice provedeno podle vzdálenosti od pozorovatele, ale jednotlivé pixely výsledného obrazu jsou překrývány ve správném pořadí.



Obrázek 3.1.1.2: Optimalizované vykreslení voxelů

Výhodou takto provedeného algoritmu je především mnohem vyšší rychlost vykreslení. Nelze ovšem pominout také korektnost vykreslení jednotlivých pixelů a odstranění potřebné struktury pro řazení. Jen malou nevýhodou upraveného řešení by se mohlo stát zvýšení složitosti algoritmu a jistá rozdrobenost řešení.

3.1.2 Výpočet průmětu voxelů

Každý renderovaný voxel je nanesen na výsledný obraz. Obraz voxelu představuje plochý průmět obecného elipsoidu na plochu výsledného obrazu. Provádět tento výpočet pro každý vzorek je výpočetně velice náročné. Velikost a tvar průmětu je možno předpočítat, nebo aproximovat bodem. Pokud je prostor objemových dat velký a velikost voxelu malá, aproximace bodem produkuje výsledky srovnatelné s vykreslováním průmětů voxelu. Pokud ovšem zobrazujeme malé množství dat nebo zobrazujeme pouze detail, je nutné vyřešit velikost a tvar obrazu voxelu.

1. Je nutné získat rovnice roviny zobrazovací plochy. Pro tento účel je možno použít možností zobrazovacího rozhraní, nebo inverzní transformací rotací provedených s objemovými daty. Obě metody jsou v podstatě stejné. Předpokládají znalost současného směru pohledu pozorovatele. Z něj je možno sestavit pohledovou matici M_m. Pokud provedeme inverzi matice M_m tak, že $M_i = M_m^{-1}$, potom z matice M_i je možno zpětně sestavit rovnice průmětné roviny.
2. Druhým krokem je úprava rozměrů voxelu. Velikost voxelu ve všech směrech musí být vyjádřena ve stejné soustavě jako je inverzní matice M_i.
3. Následující vztah definuje matici M_s, která již nese rozměry průmětu voxelu na výslednou plochu.

$$M_s = M_i M^1 M^T$$

   kde M_i je inverzní matice pohledové matice M_m, M^1 je jednotková matice a M^T je transponovaná pohledová matice M_m.
4. Z matice M_s vyjmeme matici S o rozměrech dvou řádků a dvou sloupců. Tato matice je získána ze sloupců a řádků vyjadřující osu x a y.

5. Prvky matice S jsou mocniny souřadnic směrů pomocných souřadných os obrazu voxelu vyjádřených v ploše obrazu. Pokud je tedy přeneseme na výsledný obraz, získáme osy a velikost plochého průmětu.
6. Tyto hodnoty jsou použity pro výpočet rozměrů trojúhelníku, který představuje obraz voxelu.

Tato úprava je provedena vždy při změně směru pohledu pozorovatele do prostoru objemových dat. Trojúhelník je zvolen pouze jako zástupce obrazu. Při vykreslování jsou na trojúhelníku provedeny další operace úpravy intenzity barvy směrem ze středu k okraji. Tímto problémem budeme řešit v následující kapitole.

3.1.3 Stínování

Proces stínování probíhá během vykreslování. Každému průmětu voxelu, bodu nebo trojúhelníku, je přiřazena barva a průhlednost. Během této operace je také získána prostorová orientace daného vzorku.

Princip detekce orientace objemových dat.
Prvním krokem k určení orientace jednotlivých prostorových objektů je detekce jejich hran. Tato problematika zahrnuje algoritmy, které jsou schopny nalézt hranici mezi jednotlivými typy dat v pravoúhlé oblasti původních dat.
Mějme tedy funkci f(x, y, z), kde x,y a z jsou prostorové koordináty a f je funkce intenzity hustoty prostředí v daném bodě. Tato funkce je pro tento bod vždy diskrétní za předpokladu, že (x ,y i z náleží N_0) kde N_0 je množina přirozených čísel včetně 0. Nyní definujeme operaci detekce hrany nebo filtr rozsahu jako matematickou transformaci funkce f.
Detekce orientace prostorových objektů v objemových datech se skládá ze série konvolučních operací. Po průchodu těchto filtrů jsou výsledky promítnuty ve směru souřadných os x,y, a z. Tyto projekce jsou použity při určení výsledné orientace prostorových objektů.

Detekce hran a rozsah filtrace
Pro účel filtrace objemových dat použijeme kruhovou konvoluci funkce f se specifickými maticemi m pro zjištění jednotlivých hran v prostoru.

$$f'(x,y,z)=f(x,y,z)*m[x,y,z]$$

$$f'(x,y,z)=\sum_{i=0}^{w-1}\sum_{j=0}^{h-1}\sum_{k=0}^{d-1} f(x,y,z)*m[mod_w(x-i), mod_h(y-j), mod_d(z-k)]$$

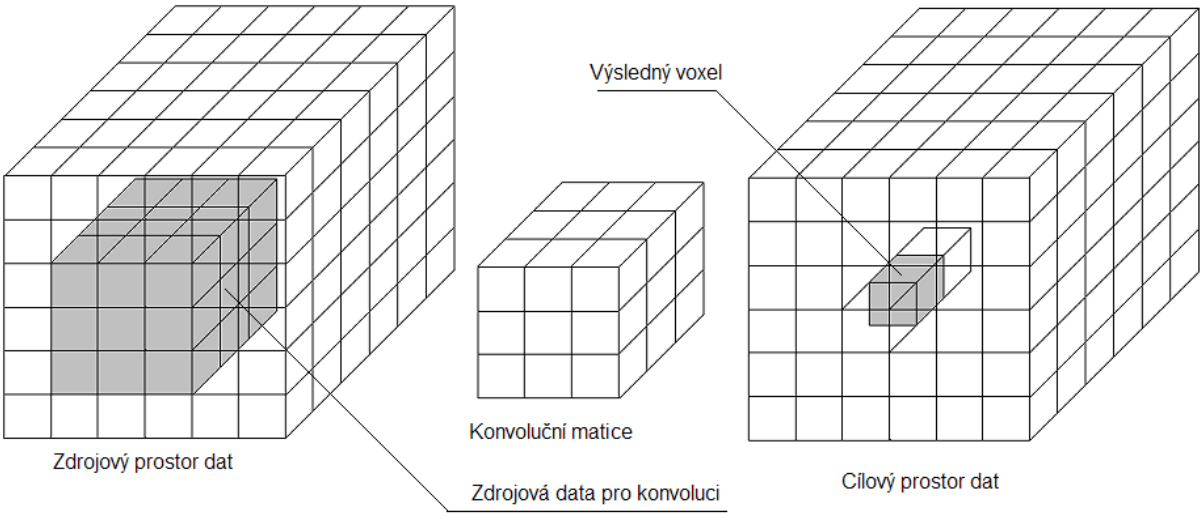
kde w je šířka, h je výška a d je hloubka objemových dat reprezentovaných funkcí f.

Konvoluční matice
Téměř každý filtr nad obrazovými daty je definován jako konvoluční matice. Konvoluční matice definuje jak je dotyčný segment prostorových dat ovlivněn sousedními segmenty v procesu konvoluce. Jednotlivé prvky v matici reprezentují vztah sousedních segmentů k segmentu urpostřed matice. Segment reprezentující prvek S v cílovém prostoru objemových dat je ovlivněn mimo prvku se stejnou polohou také všemi okolními prvky ze zdrojového prostoru objemových dat. Tento vztah je zapsán v následujícím vzorci:

S - výsledný segment ve výsledném prostoru objemových dat
m - konvoluční matice
x,y,z - koordináty pozice ve zdrojovém i výsledném prostoru objemových dat
data - zdrojový prostor objemových dat

$$S(x,y,z)=\sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2} m[k,j,i]*data[x-k-1, y-j-1, z-i-1]$$

Následující obrázek prezentuje, jak je segment ovlivněn všemi sousedními segmenty při konvoluci.



Obrázek 3.1.3.1: Schéma konvoluce

Horizontální, vertikální a hloubková detekce hran

Pro detekci vertikálních,horizontálních a hloubkových hran v objemových datech používáme při konvoluci matice m_x, m_y, m_z. Konvoluční matice jsou většinou mnohem menši než aktuální objemová data. Je ovšem možné použít mnohem větší matice. Velikost intenzity ovlivněného segmentu je rovna vzorci:

$$|S|=\sqrt{m_x^2+m_y^2+m_z^2}$$

V praxi je většinou použita jednoduchá aproximace této velikosti jako:

$$|S|=|m_x|+|m_y|+|m_z|$$

Toto zaokrouhlení ovšem přináší některé problémy, které budou probrány později.

$$m_{x-1}=\begin{pmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{pmatrix} \quad m_x=\begin{pmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{pmatrix} \quad m_{x+1}=\begin{pmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{pmatrix}$$

$$m_{y-1}=\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix} \quad m_y=\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix}$$

$$m_{z-1}=\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \quad m_z=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}-1 & -1 & -1\\ -1 & -1 & -1\\ -1 & -1 & -1\end{pmatrix}$$

Tyto konvoluční matice reprezentují funkci Perwittova filtru, jež může být použit ve fázi předzpracování.

Sobelova detekce hran
Sobelova detekce hran pro objemová data používá 3 matice o rozměrech 3*3*3 prvků. První z nich je určena pro detekci horizontálních hran, druhá pro zjištění vertikálních hran a poslední pro výpočet hloubkových hran.
Při výpočtech je možno použít tyto matice:

$$m_{x-1}=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} \quad m_x=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} \quad m_{x+1}=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix}$$

$$m_{y-1}=\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix} \quad m_y=\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix}$$

$$m_{z-1}=\begin{pmatrix}1 & 1 & 1\\ 2 & 2 & 2\\ 1 & 1 & 1\end{pmatrix} \quad m_z=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}-1 & -1 & -1\\ -2 & -2 & -2\\ -1 & -1 & -1\end{pmatrix}$$

nebo tyto matice:

$$m_{x-1}=\begin{pmatrix}-1 & 0 & 1\\ -3 & 0 & 3\\ -1 & 0 & 1\end{pmatrix} \quad m_x=\begin{pmatrix}-3 & 0 & 3\\ -6 & 0 & 6\\ -3 & 0 & 3\end{pmatrix} \quad m_{x+1}=\begin{pmatrix}-1 & 0 & 1\\ -3 & 0 & 3\\ -1 & 0 & 1\end{pmatrix}$$

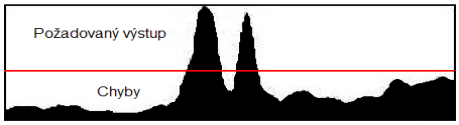
$$m_{y-1}=\begin{pmatrix}1 & 3 & 1\\ 0 & 0 & 0\\ -1 & -3 & -1\end{pmatrix} \quad m_y=\begin{pmatrix}3 & 6 & 3\\ 0 & 0 & 0\\ -3 & -6 & -3\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}1 & 3 & 1\\ 0 & 0 & 0\\ -1 & -3 & -1\end{pmatrix}$$

$$m_{z-1}=\begin{pmatrix}1 & 3 & 1\\ 3 & 6 & 3\\ 1 & 3 & 1\end{pmatrix} \quad m_z=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \quad m_{y+1}=\begin{pmatrix}-1 & -3 & -1\\ -3 & -6 & -3\\ -1 & -3 & -1\end{pmatrix}$$

Dříve zmiňované matice sice provádí filtraci objemových dat a zjišťují hrany v tomto prostoru, ale zároveň mají některé problémy, které druhý případ matic bez problémů řeší.

Filtrace dat
Konvoluční matice při průchodu skrz objemová data filtrují intenzitu hustoty z okolí. Pro tento filtr jsou data do něj přiváděná chápána jako signál se vším, co k této problematice patří. V ideálním případě by takový filtr měl potlačit chyby a různé šumy ve vstupním signálu nebo datech a zároveň by měl na výstupu zesílit požadované výsledky. Při použití sobelova filtru ovšem tento stav nenastává, tedy musíme řešit problém chyb. Filtrace pomocí matic z prvního případu se chová jako velice tvrdý kmitočtový filtr. Propustí pouze velice rychlou a výraznou změnu. Naopak matice z druhého případu se chovají jako mnohem měkčí filtr, ale ve výsledných datech je více chyb. Z těchto možností jsem vybral filtraci s větším množstvím chyb, protože výsledky byly mnohem lépe zpracovatelné a zároveň škála výstupních dat je mnohem širší.

Úprava signálu
Při průchodu jednotlivých filtrů prostředím se v každém směru vyskytují chyby. Tyto chyby je třeba potlačit. Charakteristika vstupního signálu na obrázku vypovídá, že tyto chyby mají velký odstup od signálu. Tohoto lze s výhodou použít při jejich eliminaci. V následující obrázek představuje histogram žádaného výstupu v poměru k chybám. Tento histogram je průměr několika experimentálních měření chyb ve filtrovaném signálu.

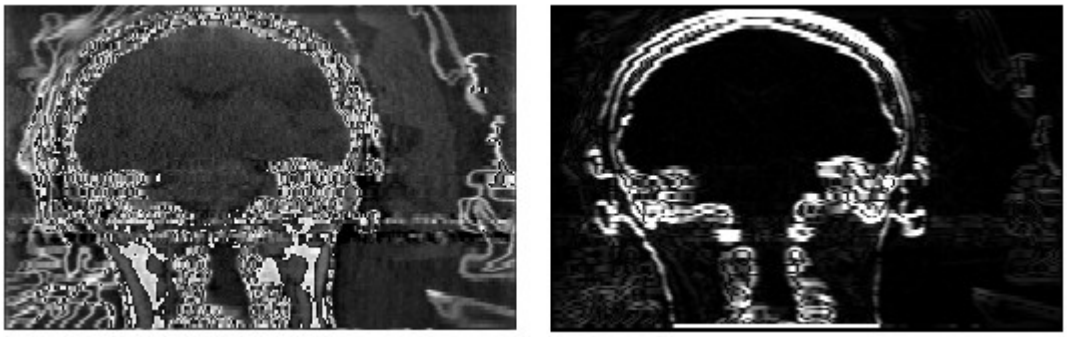


Obrázek 3.1.3.2: Histogram signálu po konvoluci

S ohledem na změřené vlastnosti signálu jsem zvolil jako nejvhodnější možnost úpravy signálu snížení hladiny citlivosti na výstupu filtru. Tato úprava se projeví výrazným snížením množství chyb. Výsledný signál se poté pouze zesílí na původní úroveň, čímž jsou data na výstupu již upravena do požadované podoby.

Vertikální detekce hran

Při průchodu filtru vertikálních hran skrze data jsou zjišťovány vertikální změny hustoty v datech. Následující obrázek představuje originální obraz dat ze vstupu a výsledná data na výstupu.

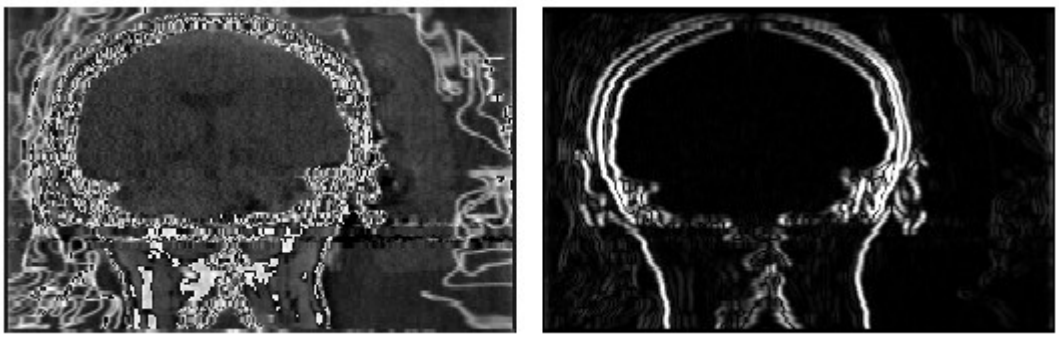


Obrázek 3.1.3.3: Vertikální detekce hran

První část obrázku jsou pouze zdrojová data. Druhá část obrázku jsou výsledná data po provedení filtrace vertikálních hran. Ve výsledných datech se také nachází určité množství chyb, ale množství těchto chyb je zanedbatelné.

Horizontální detekce hran

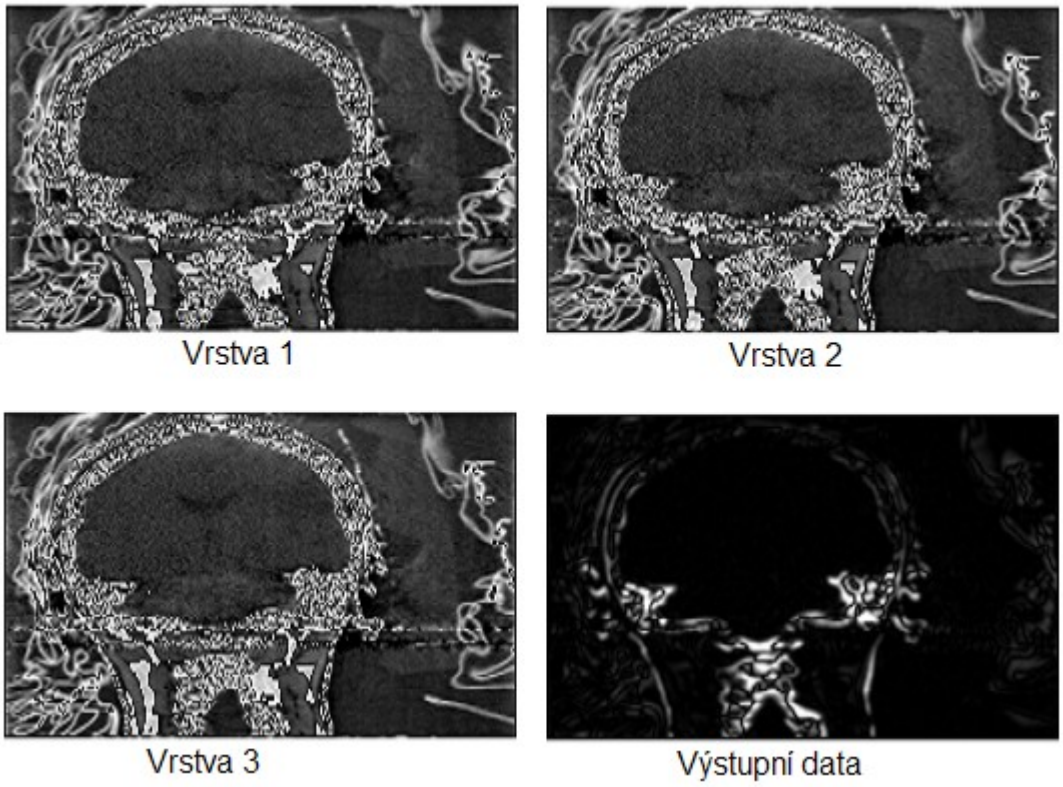
Při průchodu filtru horizontálních hran skrze data je již dobře viditelná změna vlastností výsledného signálu. V následujícím obrázku je opět originální obraz dat a obraz upravených dat na výstupu.



Obrázek 3.1.3.4: Horizontální detekce hran

První část obrázku jsou opět pouze vstupní data. Další část obrázku představuje upravená vstupní data po filtraci horizontálních hran. Pokud srovnáme předchozí skupinu obrázků se současnou, jsou mezi nimi vidět zřetelné rozdíly.

Hloubková detekce hran
Změny hustoty materiálu ve smyslu osy z jsou stejnou aplikací konvolučního filtru, pouze s jinou konvoluční maticí. Pro dobrou představu jak tyto gradienty vypadají je nutno pochopit, že upravujeme trojrozměrná data. V následujícím obrázku je série tří po sobě jdoucích vrstev zdrojových dat a jedna vrstva výsledných dat.



Obrázek 3.1.3.5: Hloubková detekce hran

Oblasti označené jako vrstvy jedna až tři jsou vstupní data pro filtr hloubkových hran. Poslední oblast jsou upravená výstupní data. Přestože tři vrstvy vstupních dat vypadají velice podobně, filtr v těchto vrstvách nalezl změny.

Výstupy filtrace
Při průchodu konvolučních filtrů skrze data jsou výsledná data uložena. Jejich šířka,výška i hloubka je ovšem menší, než rozměry původních dat o jeden segment z každé strany. Tento problém se může zdát nepodstatný, ale pokud budeme chtít z jakýchkoli důvodů zobrazit i tyto segmenty může nastat problém. Proto během výpočtu zavedeme možnost provést filtraci neúplně i v okrajových částech, kde nejsou k dispozici všechny potřebné voxely. Výpočet je prakticky stejný, pouze chybějící voxely jsou nahrazeny vypočtenou hodnotou hustoty. Tato hodnota je před započetím filtrace vypočtena tak, aby o okrajových částech nezpůsobovala vznik chyb.

Výpočet orientace voxelů v prostoru
Po provedení filtrace jsou změny stále jen hranice hustot materiálu pro jednotlivé voxely. Pomocí těchto hodnot je nutno stanovit orientaci daného voxelu. Mějme tedy funkci $n(g_x,g_y,g_z)$, kde g_x,g_y a g_z jsou gradienty v daném umístění a funkce n je funkcí směru orientace gradientu v daném umístění. Tato funkce je vždy diskrétní pro toto umístění, za předpokladu, že gradienty g_x,g_y a g_z jsou získány z umístění x,y a z pro které platí, že x,y i z náleží N_0. N_0 je množina všech přirozených čísel včetně 0. Funkce n je dána vztahem:

$$n(g_x, g_y, g_z) = \frac{\frac{g_x}{g_{max}}}{\sqrt{\frac{g_x^2}{g_{max}^2} + \frac{g_y^2}{g_{max}^2} + \frac{g_z^2}{g_{max}^2}}}, \frac{\frac{g_y}{g_{max}}}{\sqrt{\frac{g_x^2}{g_{max}^2} + \frac{g_y^2}{g_{max}^2} + \frac{g_z^2}{g_{max}^2}}}, \frac{\frac{g_z}{g_{max}}}{\sqrt{\frac{g_x^2}{g_{max}^2} + \frac{g_y^2}{g_{max}^2} + \frac{g_z^2}{g_{max}^2}}};$$

kde g_{max} je maximální gradient.

Funkce n zachovává směr, poměr velikostí i orientaci původních gradientů, ale změní je na orientaci v prostoru. Výsledkem funkce n je směrový vektor roviny zastupující voxel daný souřadnicemi x,y a z.
Výsledkem výpočtu je směrový vektor daného voxelu, jež umožňuje stínovat každý objekt. Takto předpočítaná orientace pro každý voxel je paměťově velice náročná. Již pro malý objem dat je předpočítáno velké množství směrových vektorů. Zavedeme tedy navíc operaci vyhledání směrového vektoru.
Během výpočtu jsme upravovali gradienty jako celočíselné hodnoty. Z tohoto faktu plyne možnost jednotlivé složky vektoru uložit pouze jako adresy. Adresování je jednodušší a ušetří velké množství paměti.
Celá operace výpočtu směrového vektoru je velice náročná, proto se provádí během procesu předzpracování dat.

3.1.4 Předzpracování

Fáze předzpracování proběhnou ještě před prvním vykreslením. Do této kategorie částečně patří výpočet směrových vektorů voxelů. V programu je ve fázi předzpracování provedena už jen úprava logických adres voxelů a prostor hodnot směrových vektorů.

4 Implementace a specifikace

Vývoj programu pro vizualizaci objemových dat pomocí volume renderingu prošel několika etapami. Program se skládá z několika samostatných modulů (programů, tříd). Museli jsme mít na paměti propojení vstupů a výstupů a funkcí, jak bude dále patrno.

4.1 Návrh

Klientská část programu je námi naprogramovaná aplikace. Server představuje část ovladačů grafického hardware zastřešenou knihovnou OpenGL. Jednotlivé části programu a fáze fungování budou detailně popsány v dalších částech.

4.2 Dostupný hardware a software

Při vývoji programu jsme počítali s počítačem běžných parametrů, tj. jednoprocesorová stanice s grafickou kartou podporující standart OpenGL 1.0. Na daném počítači je předinstalován operační systém Mictosoft Windows. Program byl kompilován a linkován pomocí kompilátoru GNU g++ verze 3.2.2.

4.3 Implementace

Třídy, funkce a moduly jsme vytvářeli prakticky od začátku. K tomu nás vedlo několik skutečností. Existuje velké množství prácí a experimentů na poli vizualizace objemových dat, ale jen velice malé množství má přístupné zdrojové kódy svých implementací. Pokud jsou k dispozici, přicházejí do cesty problémy s licencí (na rozdíl od open source projektů). Nebo jsou zdrojové kódy velice svázané s danou aplikací (studium a úpravy by byly časově náročnější než tvorba nového kódu). Výsledek naší implementace jsou třídy a funkce, které jsou jednoduché a dostatečně silné na tvorbu kompletního programu.
Bohužel nelze vytvořit moduly a třídy elegantně a zároveň rychle. Proto jsme se rozhodli použít při implementaci programu objektově orientovaný jazyk. Náš program se snaží při řešení vyhnout nákladům spojeným s objektovým návrhem v kritických místech programu. Všechny třídy použité v blocích programu, které v reálném čase, obsahují jen datové typy a přetížené operátory. Tento návrh je sice obtížnější rozšířit, ale rychlost programu je vysoká. V kritických částech také nepoužíváme žádné techniky OOP návrhu programů (zapouzdření, polymorfizmus, dědění). Takto optimalizovaný kód je navíc zbaven množství aritmeticky náročných operací. Práce překladače a linkeru byla optimalizována volitelné parametry pro dosažení lepších výsledků.
Program byl implementován v jazyce C++. Vývoj probíhal pod operačním systémem Windows za použití programu Dev-Cpp. Velká část programu je psána tak, aby byla použitelná pro více operačních systémů.

4.4 Řízení

Ovládání obsahuje jednoduché rozhraní a nenáročné ovládání. Uživatel pohybem ukazovacího zařízení rotuje kameru jako po sféře kolem objektu, jehož střed se nachází v počátku soustavy souřadnic nebo se může posunovat v samotných datech. Pohyb kamery je omezen na rotaci vzhledem k souřadným osám. Přiblížení a oddálení objektu je možno provádět pomocí klávesnice. Zobrazení je řešeno pomocí knihovny OpenGL. V aplikaci je vytvořen editor přechodové funkce, jež umožňuje uživateli interaktivně měnit její vlastnosti. Přechodová funkce je implementována ve formě vyhledávací tabulky (look-up table). Uživatel může měnit barvu jednotlivých skupin voxelů. Výška křivky v daném bodě určuje neprůhlednost pro daný bod nebo rozsah.

4.5 Třídy

TVector3f	Operace s trojrozměrným vektorem
Tmatrix4x4f	Operace s maticemi o rozměrech 4 x 4
TTexture	Operace s texturováním
TwinCntrl	Ovládání vlastností okna aplikace
TVolume	Operace s objemovými daty
TPoseControl	Pohyb kamery
TOpenGL	Řízení OpenGL
TMouse	Ovládání myši
TMenu	Menu programu
TMaterial	Vlastnosti materiálu
TMap	Mapování voxelů
TLight	Vlastnosti jednoho světla
TLights	Řízení všech světel
TKeyBoard	Vlastnosti klávesnice
TData	Základní vlastnosti voxelu

4.6 Adresování

Během řešení algoritmu jsme neustále naráželi na neoptimální způsob uložení dat. Vzhledem v velikosti paměti potřebné pro uložení jsme se rozhodli implementovat rychlejší a úspornější mechanizmus. Původní řešení vyžadovalo několik relativně velkých vyhledávacích tabulek. Pozdější provedení je mnohem efektivnější a úspornější. Úložný prostor vyhledávacích tabulek se zmenšil na konstantní velikost 17 kB. Zefektivnilo se také schéma práce s objemovými daty a jejich směrovými vektory. Paměťová náročnost sice roste v obou případech kubicky, ale v případě efektivního režimu je použito pouze 19% původní kapacity. Tohoto jsme dosáhli zavedením indexace obsahu a zjištění pozice voxelu na základě jeho rozměrů. Výsledný program již podporuje pouze efektivní paměťový režim.

4.7 Softwarová omezení

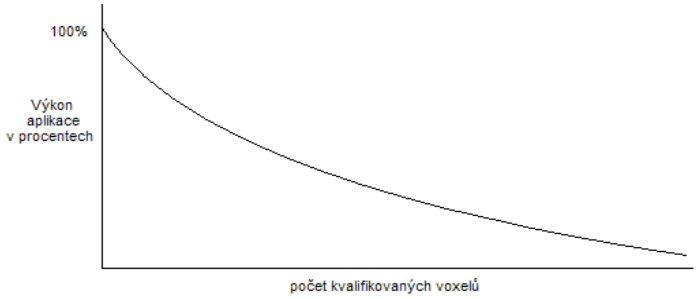
Program i jeho nástroje jsou primárně implementovaný na počítači s 32 bitovou adresovou sběrnicí. Pokud budeme program používat, nesmíme teoreticky použít rozměrnější data než 37837^3. Tento limit je způsoben maximálním možným adresovatelným prvkem na této architektuře. Program je vytvořen s ohledem na možný přechod na počítač s jinými limity velikosti paměti. Při přechodu na s jinou šířkou adresové sběrnice je nutné program znovu zkompilovat.

5 Výsledky

Porovnání výsledků je důležitá součást práce. Kvalitně provedenou optimalizací algoritmu jsme dosáhli výrazně vyššího výkonu. Překvapivě velký význam má i optimalizace kompilátoru jež se velice pozitivně projevila na výkonu aplikace. Testy probíhali jak na reálných tak na syntetických datech.

5.1 Přechodová funkce

Kvalifikační funkce určuje barvu a průhlednost jednotlivých voxelů. Následující graf představuje závislost barvy voxelu na výkonu aplikace. Horizontální osa zastupuje množství voxelů jež během operace kvalifikace nalezne svou výslednou barvu a průhlednost.

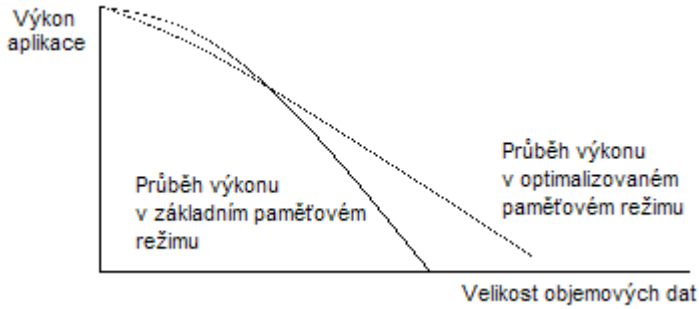


Obrázek 5.1.1: Výkon v závislosti na přechodové funkci

Z grafu je jasně patrná závislost výkonu aplikace na počtu voxelů, jež během operace kvalifikace získají barvu a průhlednost.

5.2 Optimalizace paměťových operací

Během vývoje programu se ukázalo jako klíčové zvolit vhodné schéma adresace. Srovnání paměťové náročnosti a celkového výkonu aplikace prezentují následující grafy.

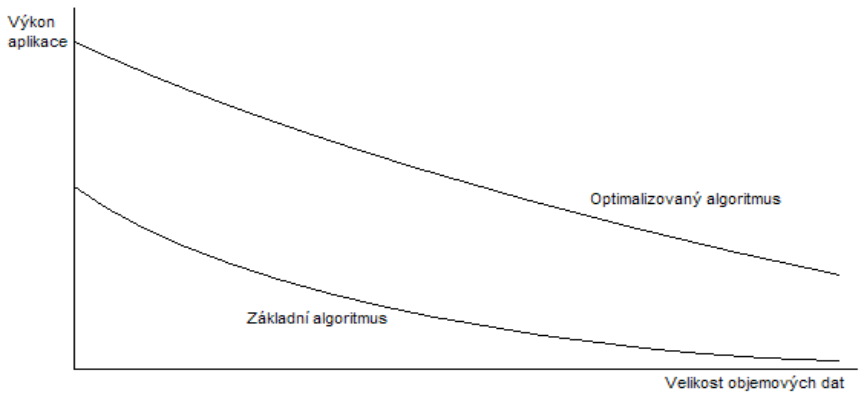


Obrázek 5.2.1: Výkon v závislosti na použitém paměťovém modelu

Výkon aplikace v optimalizovaném režimu je v první části sice nižší, ale na daném rozsahu je zároveň její výkon dost vysoký. Ve zbytku rozsahu je její výkon značně vyšší než výkon základního paměťového modelu.

5.3 Výkonnost algoritmu

Jeden z nejdůležitějších ukazatelů výkonnosti a optimalizace je rychlost vykreslení. Uvádíme výkon základního algoritmu a jeho optimalizované verze.



Obrázek 5.3.1: Výkonnost algoritmů

Zde je jasně zřetelné, že výkon optimalizovaného algoritmu je mnohem vyšší než jeho základní verze. Výkonnost ještě mnohonásobně vzrostla použitím vylepšeného paměťového modelu především na datech vyššího rozměru. Zde je nutno poznamenat další pozitivní dopad urychlení paměťového modelu. Pokud počítač nemá dostatek paměti, použije systém stránkování, což je naprosto nevhodné, protože všechny informace musí být neustále k dispozici. Jestliže data nejsou k dispozici, systém je umístí na pozici v paměti, kde jsou umístěny jiné také potřebné informace. Následně bude probíhat neustálá výměna potřebných dat jež celý výpočet téměř zastaví. Tento stav systému se nazývá trashing. Optimalizovaná podoba algoritmu podobnou situaci odsouvá na čtyřnásobek velikosti potřebné paměti.

6 Hodnocení a závěr

Prostudovali jsme různé metody pro objemovou vizualizaci a vybrali jsme poměrně neznámou metodu. Práce nabízí detailní pohled na metodu voxel splatting a probírá jednotlivé techniky a algoritmy. Návrh se snaží být co nejobecnější, zdrojové kódy portabilní. Výsledkem našeho úsilí je program vykreslující objemová data. S cílem dosáhnout co nejvyšší rychlost jsme implementovali více optimalizací vykreslovacího algoritmu. Provedli jsme sérii testů s různými druhy optimalizace a daty různých rozměrů. Program nevykresluje data v reálném čase. Zpoždění při vykreslování je závislé na rozměrech dat a nastavení přechodové funkce.

6.1 Modifikace práce

Je implementováno více algoritmů a tříd, které tvoří výsledný program. Oblast výpočtů je dobře prostudována a nabízí mnoho urychlovacích metod. Uvnitř kódu jsou označeny segmenty, které obsahují jiné metody zpracování daného problému mezi něž patří paměťový model, řazení voxelů, výpočet orientace v prostoru a předzpracovámí. Výsledný program používá co nejoptimálnější verze algoritmů.

6.2 Další vývoj

Metoda voxel splatting je natolik výpočetně náročná, že vykreslení i s použitými optimalizacemi neprobíhá plně v reálném čase. Nabízí se tedy možnost provést další urychlení, jež lze rozdělit do dvou skupin:

Paralelní výpočet
Optimalizace na příslušný hardware.

Paralelní výpočet je velice široká kapitola. Úprava algoritmu pro paralelní zpracování by předpokládala zvládnutí dekompozice problému pro skupinu více počítačů nebo procesorů a následné spojení výsledků od každého z nich. Výhodou takového řešení by byla hardwarová nezávislost. Takto upravený algoritmus ovšem nemusí mít využití, protože jen málo uživatelů má k dispozici několik velice výkonných pracovních stanic propojených rychlou sítí.

Optimalizace na příslušný hardware je postup opačným směrem. Moderní grafický hardware umožňuje přímé programování jednotek nazvaných shadery v graficém čipu. Tyto jednotky jsou schopny provádět velice rychle výpočty nutné pro vykreslení grafických objektů. Rychlost vykreslení by velice vzrostla. Nevýhodou je nutnost vlastnit moderní hardware. Ovšem stále není nutno pro vlastní vykreslení vlastnit skupinu počítačů, ale stačí jeden.

6.3 Hodnocení výkonu a kvality

Vlastní algoritmus voxel splatting patří mezi metody pracující v prostoru objemových dat. Z toho plynou nároky na velikost paměti, výkon procesoru a grafického hardware. Kapacita paměti potřebná pro uložení dat a řídících indexů je i přes různé vylepšení stále nepřiměřeně vysoká. Navíc její spotřeba roste kubicky v závislosti na rozměrech vstupních dat. Během vývoje programu se podařilo snížit nároky na výkon procesoru natolik, že nebyl již limitujícím faktorem ani na nevýkonné stanici. Bohužel výkon grafického subsystému je stále úzké místo implementace. I při použití veškerých zmíněných technik se nepodařilo snížit nároky na tuto oblast. Pravděpodobným řešením by mohlo být použití některé z optimalizací zmíněných v předcházející kapitole.

Renderování objemových dat se stalo hlavním způsobem testování výsledného výkonu. Rychlost vykreslení je velice závislá na rozměrech vstupních dat. Pokud jsou data rozměrů 256*256*256 voxelů, tak již vykreslování neprobíhá v reálném čase, i když zpoždění je stále minimální. Rozměry nižší než předcházející hodnoty jsou renderovány v reálném čase bez jakýchkoli problémů. Naopak rozměrnější data jsou již zobrazena jako snímky. Vlastní zobrazení výsledného obrazu pak proběhne po jedné až třech sekundách.
Díky metodě progresivního zlepšování kvality obrazu je možno produkovat výsledky různých kvalit. Program je schopen zobrazit voxely jako body nebo trojúhelníky. Při změnách úhlu pohledu pozorovatele jsou voxely zobrazeny jako plochy ve tvaru vybraného průmětu. Jakmile uživatel ukončí pohyb objektu, je vše vykresleno ve vyšší kvalitě. Pokud je voxel vykreslen jako trojúhelník, je na jeho plochu nanesena barva slábnoucí směrem k okrajům trojúhelníka. Takto zobrazená objemová data mnohem lépe aproximují skutečnou podobu reálného stavu zkoumaného objektu. Tento způsob vykreslení však ztratí svůj smysl při vykreslení příliš rozměrných dat. Jednotlivé průměty voxelů jsou potom tak malé, že bod i trojúhelník se jeví jako jeden obrazový pixel. Progresivní zlepšování kvality obrazu je tedy vhodné pouze pro zobrazení méně rozměrných dat.

6.4 Závěr

V této práci jsem popsal a vysvětlil základní metody vizualizace objemových data a jednu z nich jsem v programu implementoval. Během tvorby programu jsem prozkoumal některé zajímavé techniky optimalizace a zahrnul jsem je do programu.
Tato práce řeší vizualizaci objemových dat pomocí metody voxel splatting. Základní podoba algoritmu je pomalá a neefektivní. Po provedení různých změn a optimalizací se rychlost metody značně zvýšila. Přesto je rychlost velice závislá na rozměrech vstupních dat. Kvalita zobrazení je nezávisle na rozměrech vysoká. Ve výpočtu výsledného obrazu není zahrnut vztah mezi voxely. Výsledná barva tedy závisí pouze na řazení voxelů od nejvzdálenějšího k nejbližšímu vzhledem k pozorovateli. Nejsou zahrnuty situace vržení stínu voxelu, lomy a odrazy světla a vliv prostředí.
Dle mého názoru je tato metoda vhodná spíše jako doplňující k některé rychlejší metodě zobrazení. Algoritmus voxel splatting generuje výsledný obraz poměrně rychle i kvalitně, zároveň však nepatří k nejrychlejším ani vysoce kvalitním metodám.
Mnou implementovaný program není dokonale urychlen. Byl testován na několika počítačích ve snaze optimalizovat jej tak, aby pracoval na jakémkoli běžně dostupném počítači.

Literatura

[1] Martišek D.: Matematické principy grafických systémů ISBN 2002

[2] Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. Praha, Computer press 1998.

[3] Shreiner D., Mason W., Neider J., Davis T., OpenGL průvodce programátora. Praha Comuter press 2006

[4] Slavomír Hudák: Vizualizácia dát z konfokálného mikroskopu
www stránky: http://www.realtime.sk/~skip/dipl/index.html (12.4.2007)

Seznam příloh

Dodatek A. Manuál programu

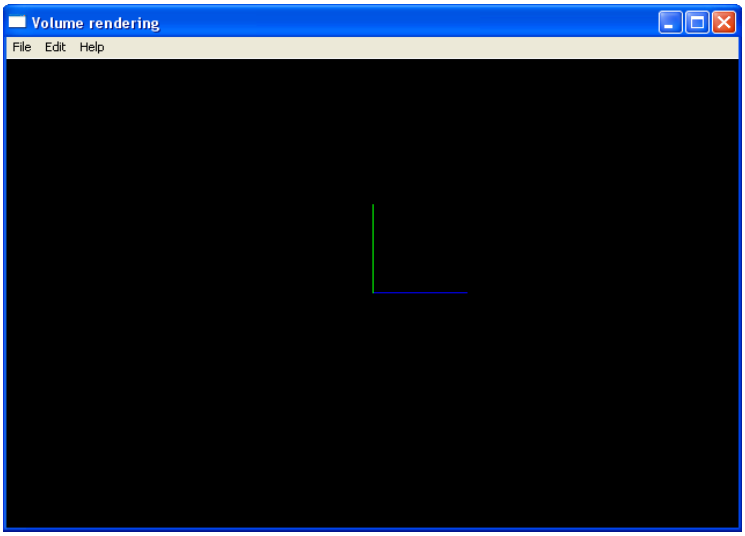
Dodatek B Obrázky generované programem

Manuál programu

Cílem následujícího výkladu je pomoci uživateli se základním ovládáním programu. Informace zde uvedené slouží pouze jako základní nastavení.

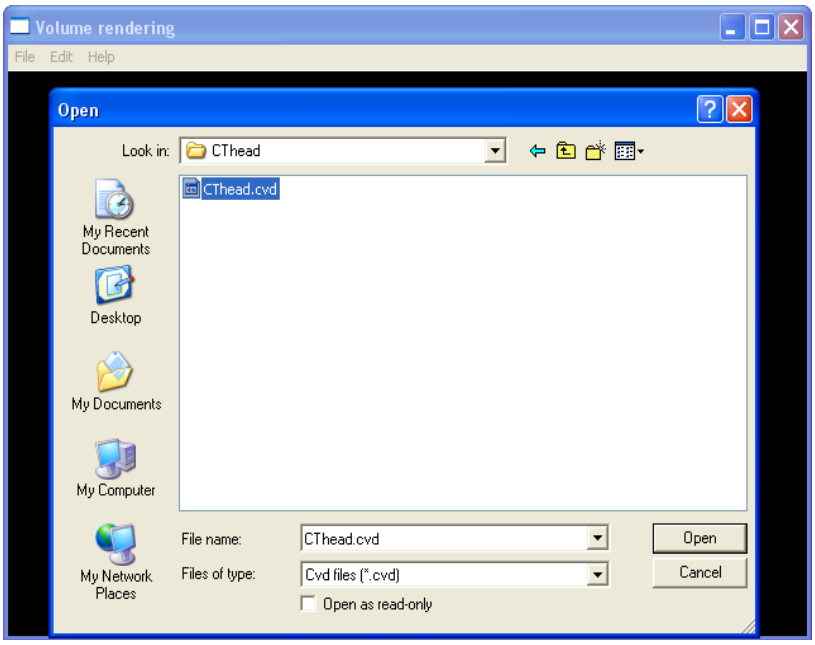
1. Spuštění programu: spustitelná aplikace se nachází na přiloženém CD v adresáři **program**. V adresáři je pouze jediný spustitelný soubor **Projekt1.exe**.

2. Po spuštění program zobrazí pouze černou plochu a základní systém souřadnic.

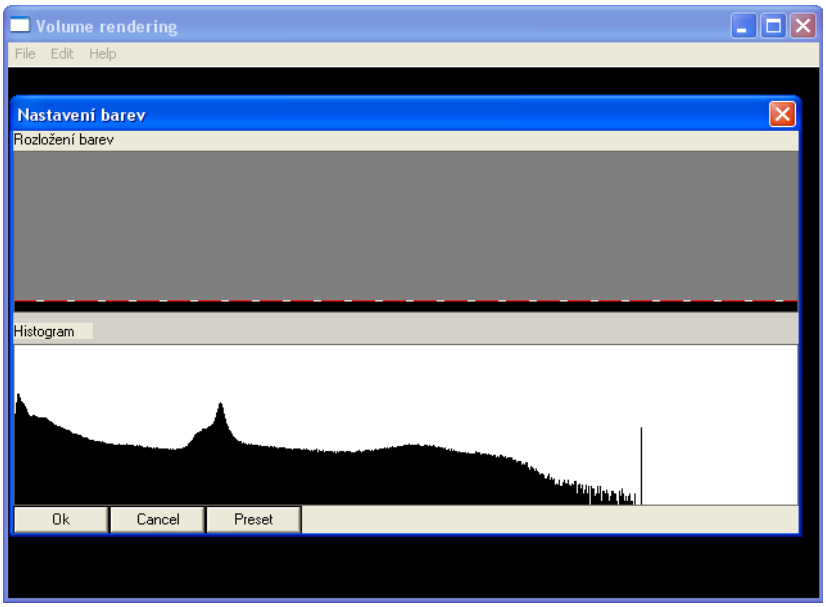


3. V tomto stavu je aktivní jen základní ovládání tj. Pomocí levého tlačítka myši je možné uchopit souřadnou soustavu a otáčet jí. Pravým tlačítkem je možno provádět posuny souřadné soustavy. Stisknutím šipky nahoru je model zvětšován, šipkou dolů zmenšován. Tyto manipulace s objektem je možno provádět vždy a s jakýmkoli objektem. Pokud jsou rozměry objemových dat velké, manipulace je pomalá. Velikost okna je možno standardním způsobem měnit.

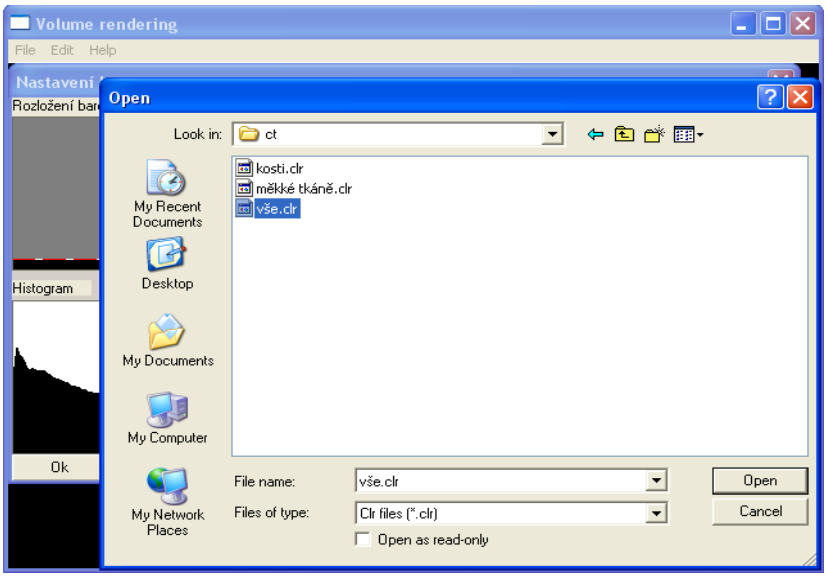
4. Pokud bude chtít uživatel otevřít libovolný soubor objemových dat, musí použít položku menu **File** a zvolit funkci **Open**. Zobrazí se standardní dialog pro otevření souborů typu **cvd**. Jakmile uživatel otevře některý soubor, program začne zpracovávat objemová data. Během zpracování nebudou viditelné souřadné osy a není možno s objektem manipulovat.



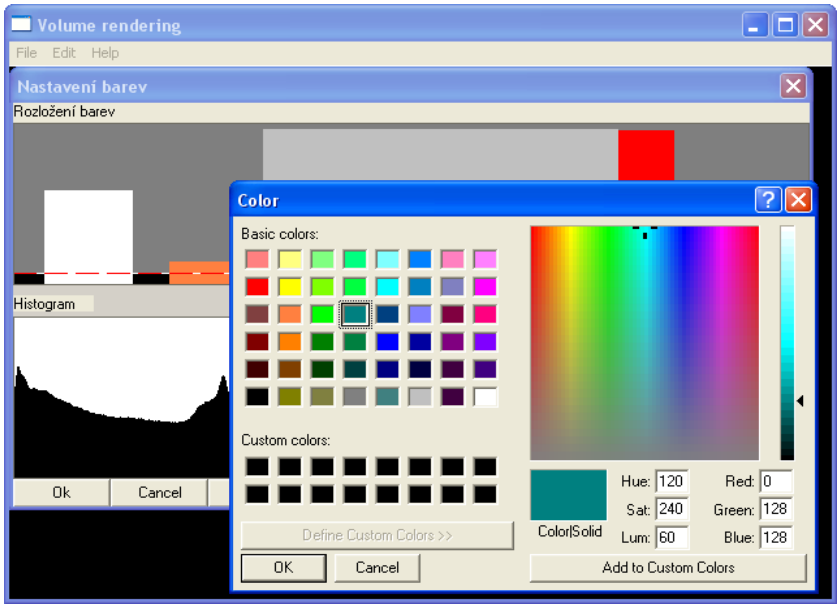
5. Jakmile jsou data zpracována, je znovu zobrazen souřadný systém. Zatím není viditelný žádný objekt, protože není nastavena klasifikační funkce, která přiděluje jednotlivým voxelům barvu a průhlednost. Jestliže uživatel chce upravit nastavení jednotlivých skupin voxelů, musí použít položku menu **Edit** s jedinou funkcí **Color**. Následně je zobrazen dialog pro úpravu barev a průhlednosti. Na formuláři dialogu je k dispozici také histogram představující počet voxelů v jednotlivých skupinách. Histogram je zobrazen v logaritmickém měřítku.



6. Výběr nastavení je možno provést manuálně, nebo načíst přednastavené hodnoty. Pokud uživatel nemá zkušenost s tímto nastavením, je lepší nejprve použít některé přednastavené schéma. Po aktivaci tlačítka **Preset** na formuláři nastavení barev se objeví standardní dialog pro otevření souboru **clr**. Poté je použito zvolené nastavení. Nakonec je nutno ukončit dialog tlačítkem **Ok**.



7. Pokud uživatel chce změnit nastavení manuálně, pak musí provést výběr v horní oblasti pomocí myši. Úroveň výběru mezi horní hranicí okna a přerušovanou čárou je stupeň neprůhlednosti, tj. čím výš bude horní okraj výběru, tím neprůhlednější dané skupiny voxelů budou. Po uvolnění tlačítka myši je vyvolán výběr barvy, čímž uživatel změní barvu dané skupiny voxelů. Dialog je opět ukončen tlačítkem **Ok**.

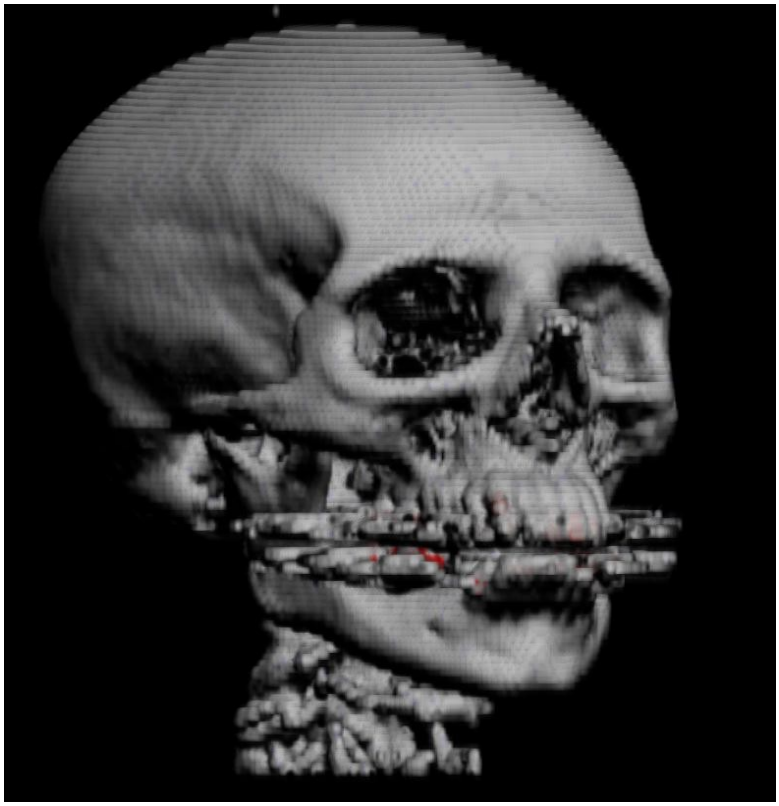


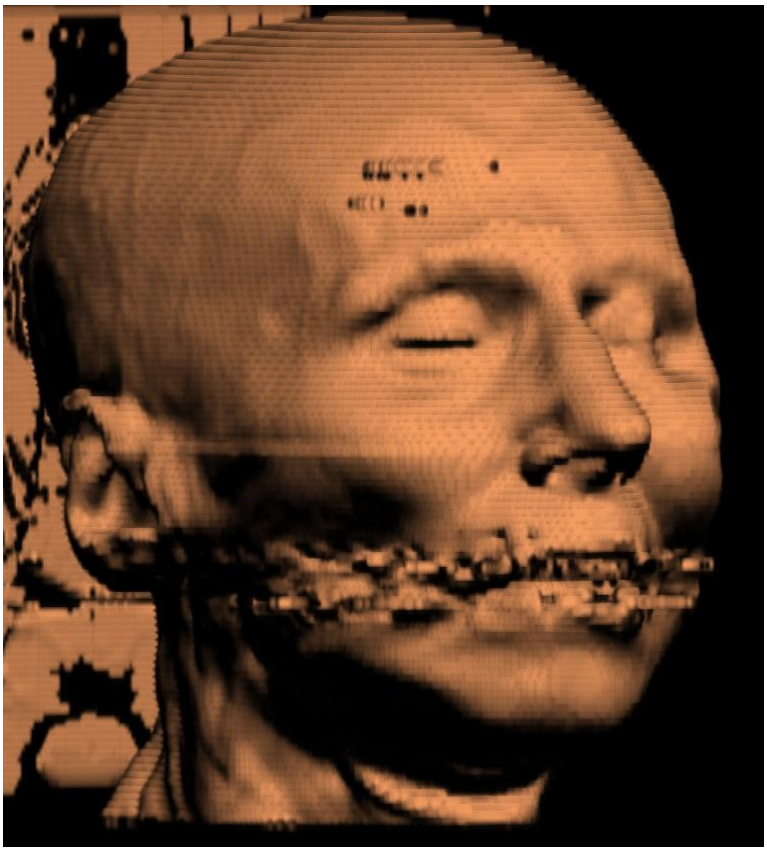
Aplikace je schopna zobrazit objemová data pomocí dvou základních grafických primitiv. Přepnutí mezi nimi se provádí stisknutím klávesy **T (tiangle)** pro zobrazení trojúhelníků a **P (point)** pro zobrazení bodů. Při otáčení a posunech je použit základní způsob zobrazení. Ve stavu kdy uživatel nemění velikost, úhel pohledu a pozici objemových dat je systém přepnut do režimu kvalitnějšího vykreslení. Jakmile uživatel opět začne měnit některou z uvedených vlastností, zobrazení se přepne do základního režimu. Přepínání mezi jednotlivými vykreslovacími metodami trvá nějaký čas v závislosti na velikosti objemových dat. Pokud uživatel pouze změní barvu některé skupiny voxelů, vždy musí myší aktivovat nové vykreslení.
Program je ukončen stisknutím klávesy **Esc** nebo položkou menu **Exit**.

Obrázky generované programem

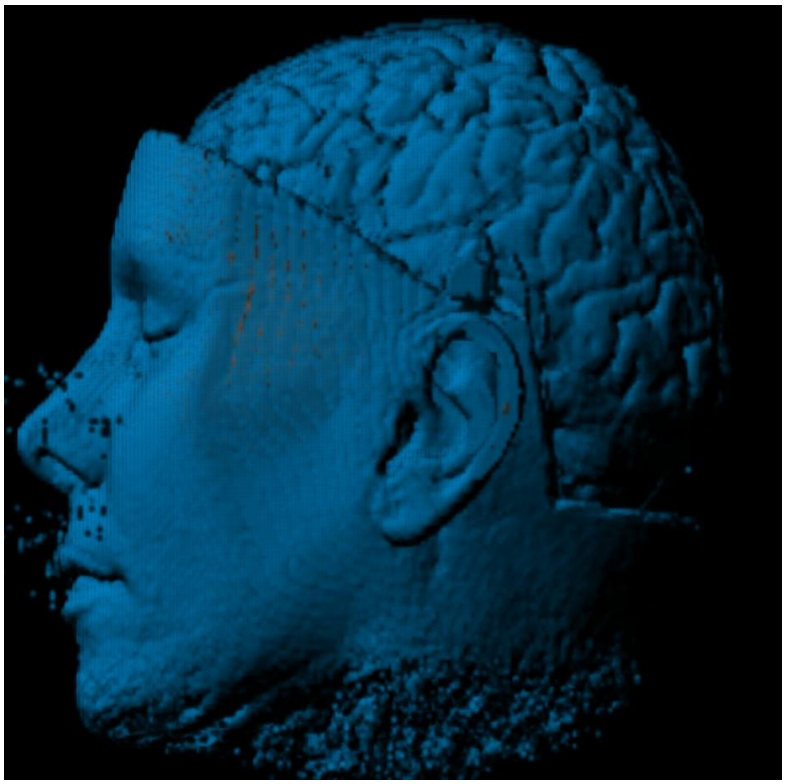
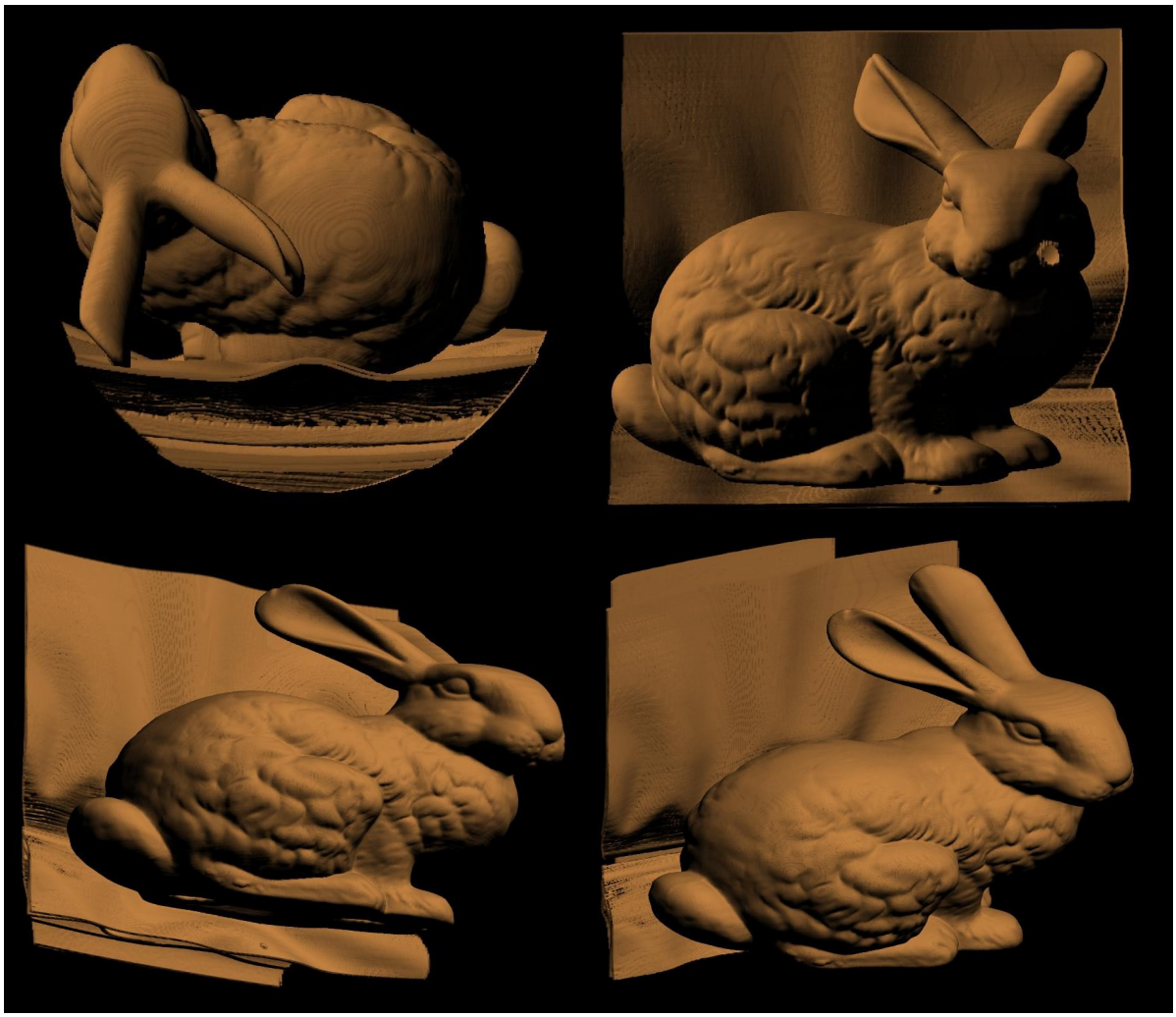
Zde je přiloženo několik obrázků vygenerovaných v tomto programu.

CT scan hlavy člověka: je pro názornost vždy zobrazen ze stejného úhlu pohledu.





MRI scan hlavy člověka:



CT scan zajíce:
Tento keramický zajíc je nasnímán z několika úhlů. Velikost objemových dat je 512 * 512 * 361 voxelů. První dva obrázky jsou zobrazeny pomocí trojúhelníků, následující pomocí bodů. Na obrázcích tento rozdíl již není patrný. Vykreslení pomocí trojúhelníků má výrazný smysl pouze u méně rozměrných objemových dat.