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

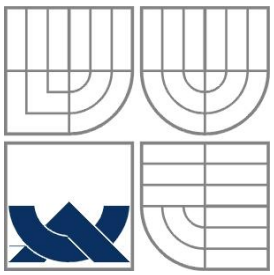
VYUŽITÍ VERTEX A PIXEL SHADERU V OPENGL PRO 3D ZOBRAZENÍ 3D OBRAZOVÝCH DAT V MEDICÍNĚ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

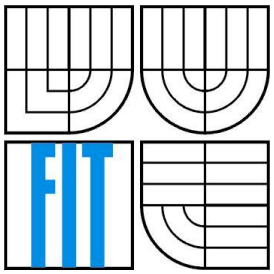
AUTOR PRÁCE
AUTHOR

BC. JIŘÍ VAĎURA

BRNO 2009



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

VYUŽITÍ VERTEX A PIXEL SHADERU V OPENGL PRO 3D ZOBRAZENÍ 3D OBRAZOVÝCH DAT V MEDICÍNĚ

VERTEX AND PIXEL SHADERS OPENGL VISUALISATION OF MEDICAL 3D IMAGE DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. JIŘÍ VAĎURA

VEDOUCÍ PRÁCE
SUPERVISOR

DOC. ING. PŘEMYSL KRŠEK, PH.D.

BRNO 2009

Abstrakt

Tato práce se zabývá akcelerovaným zobrazováním 3D medicínských dat, např. z počítačové tomografie, pomocí grafického procesoru a s použitím rozhraní OpenGL. Nezpracované řezy jsou načteny do grafické paměti a zobrazeny metodou ray-casting. Cílem je kvalitně zobrazit 3D data a zároveň umožnit plnou interakci. K dispozici je několik režimů zobrazení jako MIP, simulace rentgenového zobrazení a realistické stínování.

Abstract

This thesis deals with accelerated 3D rendering of medical data, e.g. computed tomography, using a graphics processor and OpenGL library. Raw data slices are send to graphic memory and rendered by a ray-casting algorithm. The goal of this project is high quality visual output and full user interaction at the same time. Multiple rendering modes are avaiable to the user: MIP, X-Ray simulation and realistic shading.

Klíčová slova

medicína, tomografie, grafický procesor, OpenGL, vrhání paprsků, shadery, volume rendering

Keywords

medical imaging, computed tomography, graphics processing unit, OpenGL, ray-casting, shaders, volume rendering

Využití Vertex a Pixel shaderu v OpenGL pro 3D zobrazení 3D obrazových dat v medicíně

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Přemysla Kršeka, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Vaďura
1.5.2009

Poděkování

Tímto bych chtěl poděkovat vedoucímu svého diplomového projektu, Doc. Přemyslu Kršekovi, za pomoc při tvorbě této práce.

Obsah

1 Úvod

S příchodem moderních diagnostických zařízení (CT, MRI) nastala i potřeba tato data vhodně interpretovat. Ve většině případů tyto přístroje generují řezy snímaným objektem. Přestože v řadě aplikací je možné analyzovat tyto snímky samostatně, podobně jako rentgenové snímky, v určitých případech je výhodné se na snímaný objekt dívat jako na celek.

Počítačové interpretaci (nejen) medicínských dat se říká volume rendering. Jeho úkolem je zobrazit trojrozměrná data v počítači na 2D zařízení (např. monitoru). Jedna z metod volume renderingu je vrhání paprsků. Tyto pojmy budou dále popsány v následujících kapitolách.

S postupným rozvojem grafických procesorů vzrostly možnosti, jak tyto vysoce specializované, ale výkonné, čipy využít pro jiné účely než hry. U poslední řady karet (2009) je grafické jádro dostatečně flexibilní k téměř jakýmkoliv obecným výpočtům. V této práci je využita síla grafického procesoru k akceleraci volume renderingu metodou vrhání parsků (ray casting). Použité grafické rozhraní je OpenGL a jeho nástavba Open Scene Graph .

Úkolem je vytvořit aplikaci, která zobrazí 3D objekt na monitoru, a umožní plnou interaktivitu. Ta zahrnuje rotaci objektu, posun a zvětšení. Možné režimy zobrazení budou: MIP, rentgen, segmentace a stínování. Hlavním cílem práce je kvalitní a přesné zobrazení 3D medicínských dat pro diagnostiku a plánování.

2 Teoretický rozbor problematiky

V této kapitole se čtenář seznámí s problematikou zobrazování objemových dat. Čeština si pro tuto problematiku osvojila pojem (převzatý z angličtiny) volume rendering, který bude používán dále v textu.

Tato práce se zaměřuje na použití volume renderingu pro zobrazování objemových dat v oblasti medicíny. Výsledný software je však schopen zobrazovat libovolná trojrozměrná data, která jsou uložena v počítači. Volume rendering se například používá pro vizualizaci archeologických nálezů, které byly trojrozměrně nasnímány do počítače, dále pak pro zobrazování částí motorů a jiných součástek. Nejširší oblast aplikací je ale právě v medicíně.

Postup pro zobrazení 3D dat je obecně známý, širokému nasazení až donedávna bránil nedostatečný výkon osobních počítačů, které nebyly schopny překreslovat obraz v reálném čase, nebo bylo nutné podstatně omezit kvalitu výsledného obrazu. Poslední část této kapitoly se proto zabývá možnostmi současné výpočetní techniky z hlediska akcelerace volume renderingu.

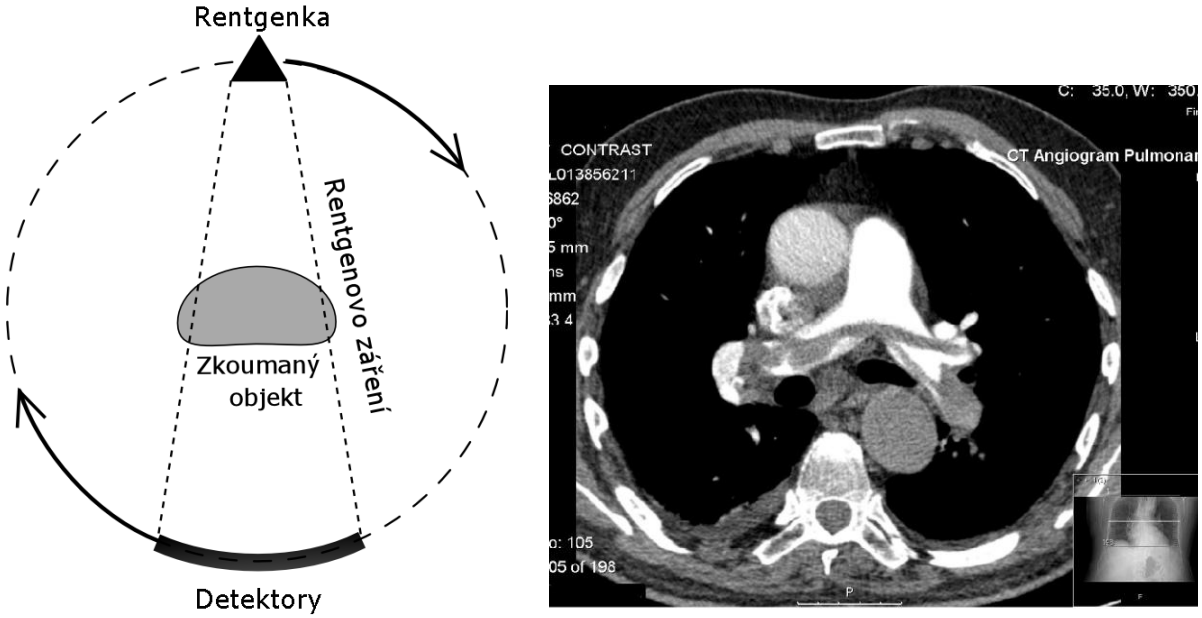
2.1 Zdroje dat

Tato práce se primárně věnuje zobrazení medicínských dat. Existuje celá řada technických způsobů, jak získat digitální reprezentaci snímaného objektu, tedy člověka. Všechny metody, které jsou rozebírány dále, produkují diskrétní 3D rastrová data. Obraz lze reprezentovat jako trojrozměrnou mřížku, kde každý bod v prostoru má přiřazenu takovou hodnotu, kterou v daném bodě nasnímal skener.

Ne všechny uvedené metody jsou však vhodné pro zobrazení na počítači. Pro kvalitní zobrazení je požadována vysoká přesnost snímání, nízký šum a zkreslení.

2.1.1 Počítačová tomografie

Často označována jako CT (Computed Tomography), případně CAT (Computed Axial Tomography) scan. Poprvé byla použita v sedmdesátých letech minulého století k vytvoření prostorového obrazu lidského těla. Základní princip spočívá v pořízení velkého množství dvourozměrných rentgenových snímků kolem jedné rotační osy. K vytvoření výsledné sady 2D řezů je potřeba snímaný obraz zpracovat komplexními algoritmy na počítači.



Obrázek 2-1 Princip činnosti počítačové tomografie a výsledný řez hrudníkem (převzato z [2]).

Výsledné řezy jsou obrázky v odstínech šedi, obvykle s 12 bitovou hloubkou (4096 odstínů šedé). Světlá barva reprezentuje oblasti s vysokou absorpcí rentgenového záření (kosti, krev, …), černá barva pak oblasti, kde nedochází k žádné absorpci. Použitá Hounsfieldova stupnice začíná na +3071 (nejvíce pohlcující) a končí na hodnotě -1024 (nejméně pohlcující).

Látka	**HU**
Vzduch	-1000
Tuk	-120
Voda	0
Svaly	40
Kontrast	130
Kost	+400 a víc

Tabulka 2-1 Příklady absorpce záření v Hounsfieldově stupnici.

Velmi husté části těla (lebeční kost), případně kovové implantáty, mohou způsobit chyby v obraze ve formě linek vycházejících z těchto částí. Náhlé změny mezi hustým a řídkým prostředím produkují hodnoty, které jsou mimo rozsah zpracovávající elektroniky a vytváří v obraze artefakty.

Jednotlivé řezy jsou od sebe vzdáleny řádově milimetry. Kompletní sada snímků hlavy může mít například až 400 řezů. Velikost obrázků závisí na kvalitě použitého skeneru a pohybuje se obvykle v rozmezí 256×256 až 1024×1024 bodů. Jeden řez tak může přesáhnout i velikost 1MB.

Vzhledem k tomu, že tato data obsahují pouze informaci o tom, jak dané místo v obrázku absorbuje rentgenové záření, je třeba tato data ještě dále transformovat pomocí vyhledávacích tabulek.

2.1.1.1 Použití v diagnostice

Vzhledem k tomu, že počítačová tomografie zachycuje všechny typy tkání v lidském těle, je její použití v diagnostice široké. Díky konstrukci CT skeneru je možné nasnímat obraz celého člověka, což dále rozšiřuje možná využití. Prostorové zobrazení není nezbytně nutné k identifikaci zranění a jiných chorob, přesto může být úspěšně použito k doplnění 2D řezů.

Hlava

Rentgenové záření snadno proniká lebeční kostí, což umožňuje diagnostiku řady onemocnění:

- krvácení, zranění mozku a fraktury lebky
- krevní sraženiny
- mozková mrtvice
- nádory na mozku
- poškození lebky
- infekce dutin a další

Hrudník a srdce

Pomocí CT skeneru je možné detekovat, pro klasický rentgen nedetekovatelné, změny ve struktuře plic a sledovat změny způsobené chronickým onemocněním.

Počítačová tomografie se také používá k pořízení velmi detailních snímků bijícího srdce, kdy jsou klasické řezy navíc doplněny o čtvrtý, časový, rozměr.

Pánevní oblast a břicho

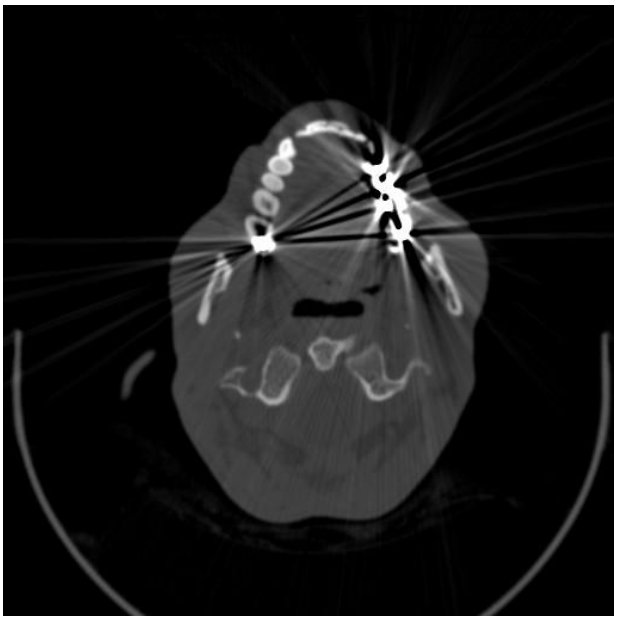
Zde se tomografie využívá k sledování stádia zhoubných onemocnění a k vysvětlení akutní bolesti v břišní oblasti. K běžně diagnostikovaným stavům patří: ledvinové kameny, zánět slepého střeva, aneurysma, zablokování střev, atd. Využívá se i ke snímkování pánve.

Končetiny

Složité zlomeniny blízko kloubů jsou často cílem CT snímkování, protože je možné prozkoumat postiženou oblast ve více prostorových rovinách.

2.1.1.2 Artefakty

Přestože je CT relativně přesný, mohou se v obraze objevit chyby.



Obrázek 2-2 Artefakty způsobené přítomností kovu (převzato z [6]).

Chyba zarovnání

Projevuje se temnými čarami vycházejícími z ostrých rohů snímaného objektu. Příčinou je neschopnost skeneru nasnímat tuto oblast z dostatečného množství úhlů. Chybně nastavený rentgenový zářič se může také projevit podobným způsobem.

Rozmazání

Tento druh chyby způsobuje rozmazání obrazu na ostrém přechodu mezi silně a slavě absorbujícím materiálem. Výpočetní procesor není schopen správně rozlišit tyto ostré přechody a provádí průměrování, které způsobuje ztrátu informace.

Prstencový artefakt

Je způsoben vadou na samotném detektoru a v datech tvoří prstence.

Šum

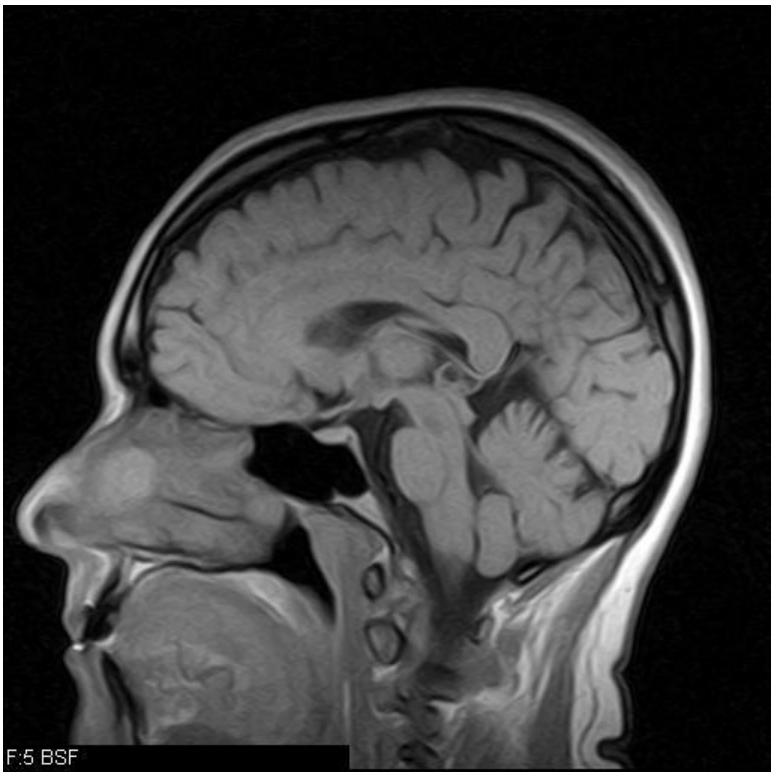
Při tenkých CT řezech dochází k snížení poměru signál-šum a v obraze vzniká rušivý šum.

Pohybové artefakty

Jsou způsobeny pohybem objektu, který je snímán.

2.1.2 Magnetická rezonance

Druhá nevyužívanější snímkovací metoda pro 3D zobrazení. MRI schopna dosáhnout daleko většího kontrastu u měkkých tkání v porovnání s CT. Z toho to důvodu se častěji používá na snímkování mozku, srdce a nádorů. MRI nevystavuje pacienta radiaci jako CT, vyšetření je proto zcela bezpečné.



Obrázek 2-3 Snímek z magnetické rezonance (převzato z [2]).

2.1.2.1 Princip MR

Lidské tělo se skládá především z vody. V každé molekule vody jsou dva atomy vodíku, které ve svém atomovém jádře ukrývají jeden proton. Právě atomy vodíku jsou klíčové pro MR, protože mají velký magnetický moment. To znamená, že atom vodíku, který je vystavený silnému magnetickému poli, se otočí ve směru tohoto pole. Magnetické pole uvnitř skeneru je orientované podél pacienta. Každý atom vodíku se natočí směrem k hlavě nebo nohám v okamžiku zapnutí magnetického pole.

Přístroj následně vyšle puls rádiového záření do vyšetřované oblasti. Frekvence vln je zvolena tak, aby je zachytil pouze vodík. Atomy vodíku pohltí tento rádiový puls a začnou „rezonovat“ na Larmourově frekvenci. Tato frekvence je různá pro různé typy tkání a závisí i na síle magnetického pole. Po vypnutí magnetického pole se atomy relativně pomalu vrací do své původní orientace a vyzařují při tom energii, která je detekována cívkami kolem snímané oblasti.

Aby z těchto dat mohl vzniknout skutečný obraz tkání, je nutné výsledky nejprve zpracovat na počítači. K převedení naměřených dat do obrazové podoby se používá Fourierova transformace.

2.1.2.2 Srovnání s CT

Magnetická rezonance dokáže velmi dobře odlišit i malé rozdíly ve struktuře tkání, které leží těsně u sebe. CT umožňuje dosáhnout podobného prostorového rozlišení, ale s menším kontrastem u měkkých tkání.

U MR existuje celá řada vstupních parametrů, které lze nastavit tak, aby co nejvíce vyhovovaly prováděnému vyšetření a zájmové oblasti. Klasická tomografie je závislá pouze na rentgenové absorpci záření a další parametrizace není možná.

2.1.3 Lékařská ultrasonografie

Lékařská ultrasonografie je diagnostická technika založená na odrazu ultrazvukového signálu od tkání. Frekvence signálu se pohybuje od 8 do 20 MHz, což mnohanásobně převyšuje schopnosti vnímání lidského ucha.

2.1.3.1 Použití

Ultrazvuková vyšetření se provádí ruční sondou, která se pohybuje na kůži nad vyšetřovanou oblastí. Sonografie dosahuje nejlepších výsledků u měkkých tkání. Čím hlouběji je vyšetření prováděno, tím horšího rozlišení sonografie dosahuje.

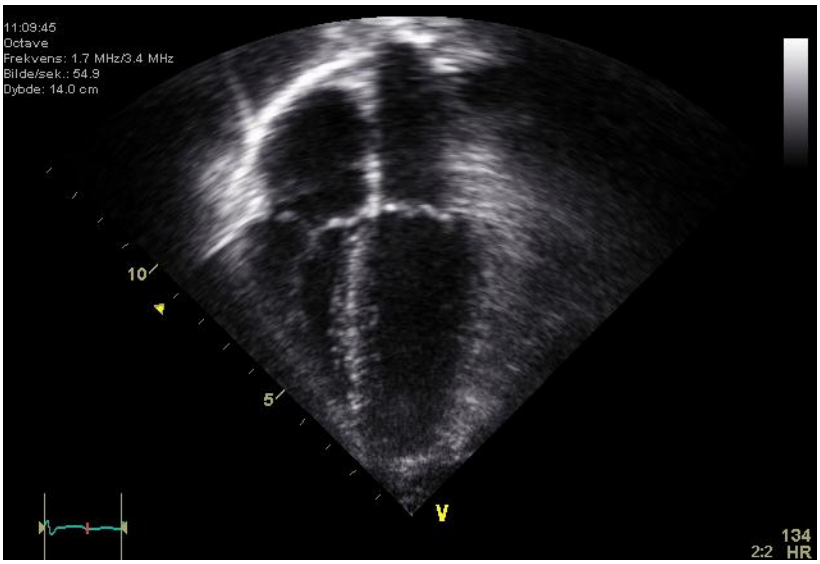
2.1.3.2 Vlastnosti

Kladné vlastnosti sonografie zahrnují:

- velmi dobré zobrazení měkkých tkání a povrchu kostí
- obraz je získáván v reálném čase
- zobrazuje strukturu vnitřních orgánů
- dobré prostorové rozlišení

Ultrazvuk však přináší i celou řadu nevýhod:

- ultrazvuk neproniká kostí
- selhává, pokud je v cestě vzduchová kapsa
- ultrazvuk neproniká hluboko do těla
- výsledná kvalita silně závisí na zkušenostech operátora



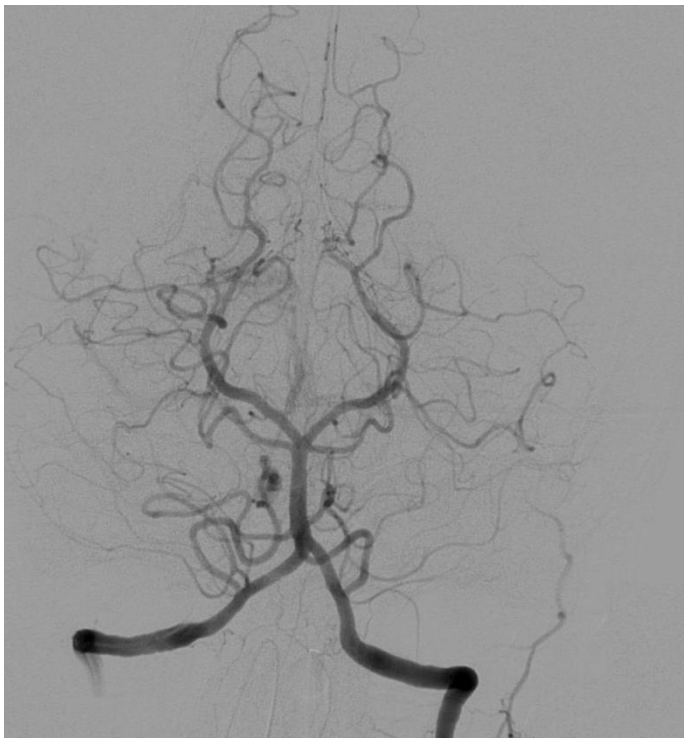
Obrázek 2-4 Ukázka sonografie (převzato z [2]).

Výsledná kvalita ultrazvuku však není prakticky použitelná pro trojrozměrné zobrazování. Trojrozměrná vizualizace ultrazvuku se používá v porodnictví, kde postačuje základní zobrazení.

Obrázek 2-5 Trojrozměrná vizualizace sonografie (převzato z [2]).

2.1.4 Angiografie

Angiografie je technika, která zobrazuje zvýrazněné cévy v těle, převážně tepny, žíly, srdce a vnitřní orgány. Tohoto se běžně dosahuje vstříknutím kontrastní látky do těla a snímkování tradičním rentgenem či CT.



Obrázek 2-6 Angiografie (převzato z [2]).

2.1.4.1 CT Angiografie

Tato metoda vyšetření se používá k zobrazení tepen v plicích, ledvinách, nohách atd. Je vhodná i ke studiu aneurysma větších cév a jiných defektů.

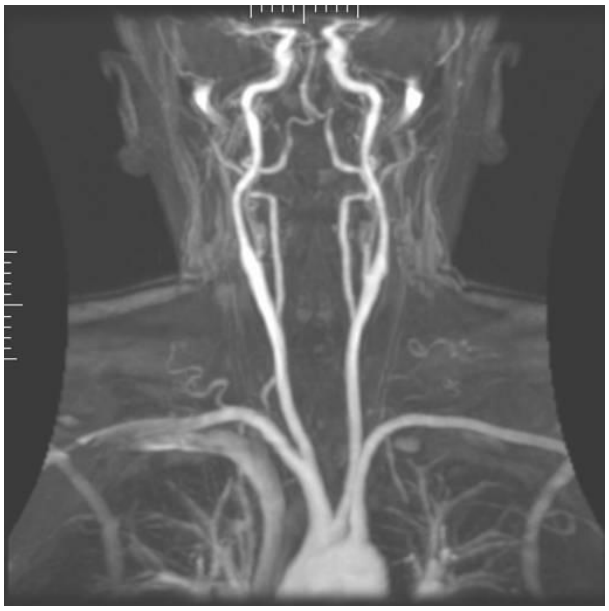
Základní přednosti CTA je možnost sledování jakéhokoliv cévního systému v těle. Detaily cév jsou lepší než při použití MR či ultrazvuku. CTA je neinvazivní technika, která je šetrnější k pacientovi než zavádění katétrů.

Někteří lidé mají alergii na jód, který je obsažený v kontrastní látce. Následná alergická reakce může být velmi vážná. CTA také není použitelná u lidí s nemocnými ledvinami, u kterých hrozí další poškození kontrastní látkou.

2.1.4.2 MR Angiografie

Podobná technika jako CT Angiografie. Kontrastní látka je vstříknuta do žíly a při svém prvním průchodu snímanou oblastí jsou pořízeny snímky. Při správném načasování jsou výsledné obrázky velmi kvalitní. Druhá možnost je vstříknutí kontrastní látky, která zůstává aktivní až jednu hodinu. Nevýhoda déle působící kontrastní látky je zvýraznění jak tepen, tak i žil.

Další techniky pro pořízení MRA spočívají ve zvýraznění rychle se pohybující krve. Části cév, kde krev proudí relativně pomalu (aneurysma), jsou ovšem zobrazena s nízkou kvalitou.



Obrázek 2-7 MR Angiografie (převzato z [2]).

Při zkoumání žil a tepen je klasické 2D zobrazení řezů již nedostatečné. Zde se s výhodou uplatňuje některá z technik volume renderingu.

2.2 Metody 3D zobrazení

Tato podkapitola popisuje nejběžnější způsoby reprezentace objemových dat na PC. Tyto metody lze rozdělit na dva rozdílné přístupy, a to: vektorizaci dat a volume rendering.

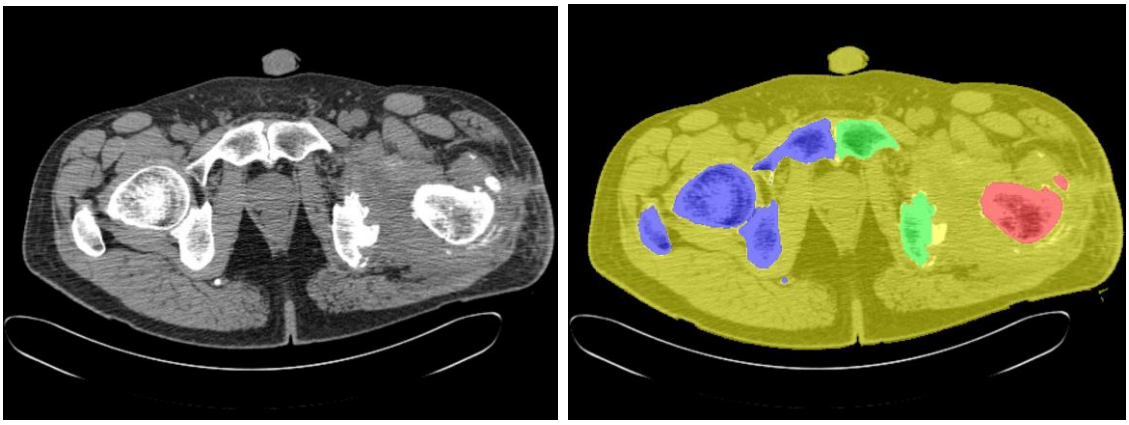
2.2.1 3D vektorizace

Geometrická vektorizace je založena na vytvoření 3D geometrického modelu z CT/MR dat. Model popisuje pouze povrch vybraného objektu v těle.

2.2.1.1 Segmentace dat

Jednotlivé tkáně v CT/MR snímcích jsou zobrazeny jako oblast s určitou vlastností (pohlcování rentgenových paprsků). Cílem segmentace je odlišit jednotlivé tkáně v datech a označit je. Existuje řada různých postupů pro segmentaci.

Jednou z nich je využití histogramu obrázku. Histogram je vypočítám ze všech pixelů v obrázku a jednotlivé špičky a prohlubně jsou použity k detekci oblastí. Je možné využít jak intenzitu, tak i barvu v obrázku.



Obrázek 2-8 Vstupní snímek z CT (vlevo) a výsledek po segmentaci (vpravo) (převzato z [6]).

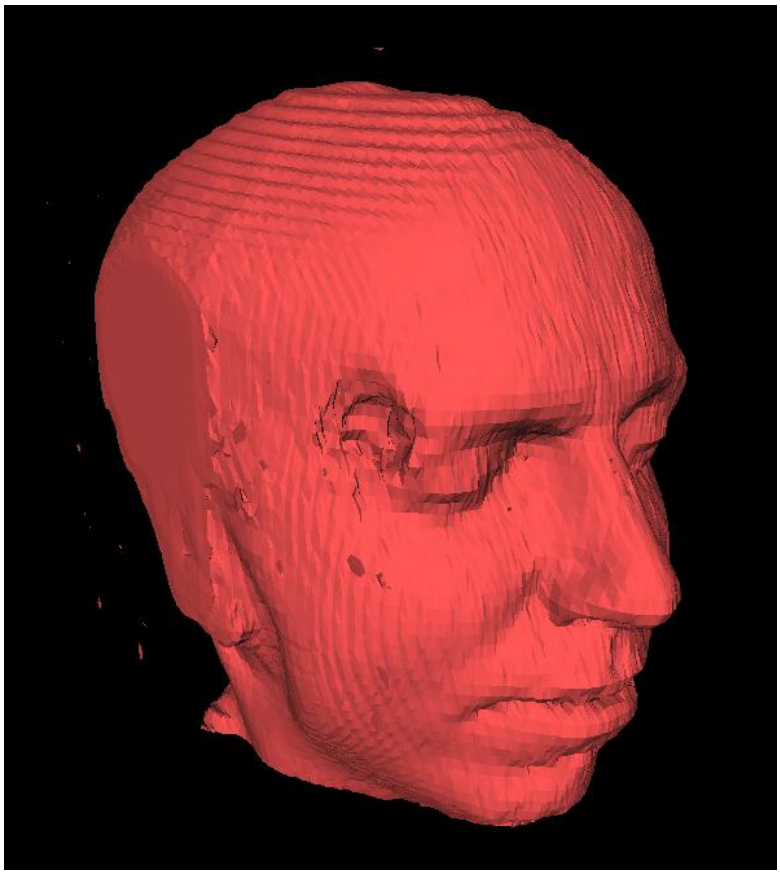
Další variantou je detekce hran. U CT/MR dat je většinou dobře zřetelný přechod mezi různými tkáněmi. Na tomto rozhraní dochází k výrazné změně jasu obrázku. Tato změna je dobře detekovatelná a je možné ji použít ke stanovení hranic.

Poslední technika zkouší „zaplavovat“ oblasti stejných vlastností. Do obrázku jsou zasazena „semínka“, která se postupně rozrůstají a zaplavují oblasti stejných vlastností. Kritérium pro zahrnutí sousedního bodu do rozrůstající oblasti je dosavadní průměr všech bodů v již zaplavené oblasti. Pixel s nejmenším rozdílem oproti průměru je zahrnut. Rozrůstaní pokračuje, dokud všechny obrazové body nejsou součástí některého segmentovaného regionu [6].

2.2.1.2 Vektorizace

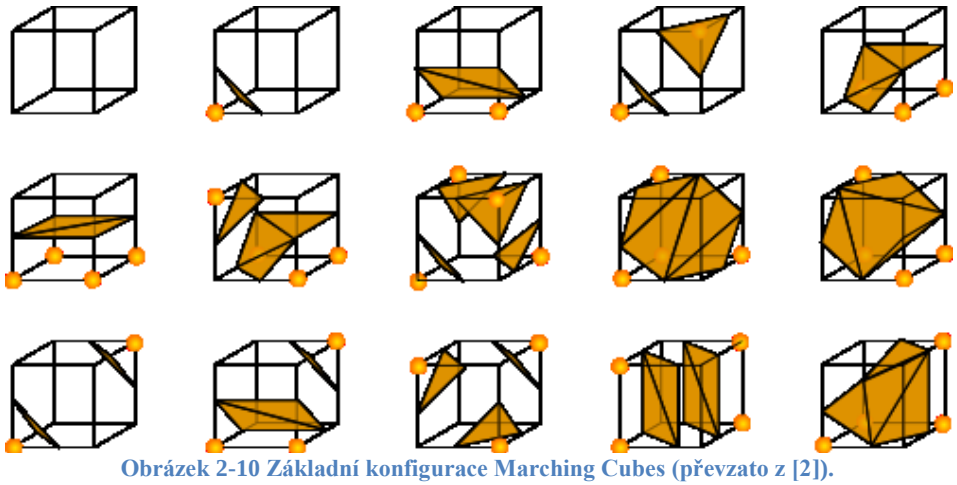
Výstupem segmentace jsou stále rastrová 3D data reprezentovaná voxely. Jednotlivé tkáně je potřeba dále převést na geometrii povrchu. Geometrie se vytváří z hraničních oblastí, které určila segmentace.

Nejběžněji používaným algoritmem pro vektorizaci je metoda „Marching Cubes“ (MC). Marching Cubes automaticky vytvoří geometrii pro libovolný rastrový vstup. Výstupem je síť trojúhelníků, která je ovšem vrstvená po vzoru vstupních dat.



Obrázek 2-9 Geometrie vygenerovaná metodou Marching Cubes (převzato z [2]).

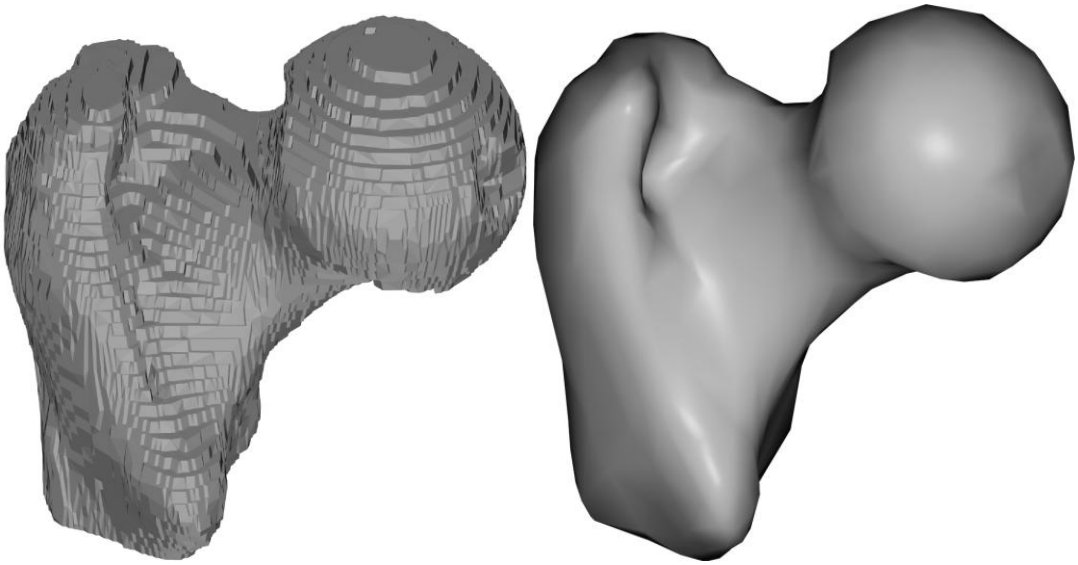
Algoritmus prochází 3D daty a analyzuje vždy 8 okolních bodů v prostoru. Podle rozložení informace v těchto bodech se vybere takový polygon, který nejlépe reprezentuje danou konfiguraci. Na konci jsou všechny polygony spojeny do jednoho objektu. Všech 8 rohů posouvající se krychle tvoří dohromady 8 bitů informace. Bit daného rohu je nastaven na jedna, pokud je bod uvnitř vektorizovaného povrchu, případně na nulu, pokud je vně povrchu. Výsledné osmibitové číslo je indexem do 256 různých konfigurací polygonů. Základních tvarů je pouze 15 a všechny zbylé konfigurace jsou pouze otočené.



Obrázek 2-10 Základní konfigurace Marching Cubes (převzato z [2]).

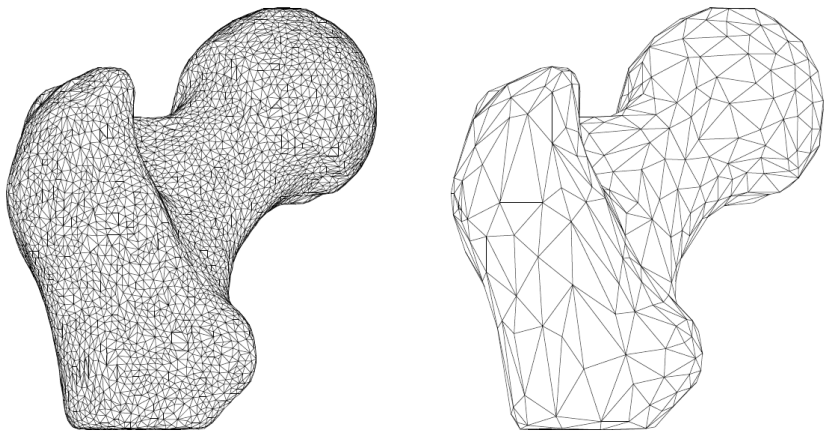
2.2.1.3 Vyhlazení a decimace modelu

Po aplikaci Marching Cubes není geometrie vyhlazená a obsahuje viditelné vrstvení způsobené rozlišením obrazu a artefakty. Pro vyhlazení povrchu se často využívá Laplaceova vyhlazení, které průměruje sousední vrcholy tak, aby se povrch vyhladil.



Obrázek 2-11 Vyhlazení modelu kloubu získaného metodou Marching Cubes (převzato z [6]).

Protože metoda MC pracuje na úrovni voxelů, obsahuje model velké množství trojúhelníků. Pracovat s takto velkým množstvím vrcholů je nepraktické a náročné na výpočetní výkon a paměť. Závěrečná úprava geometrického modelu je redukce počtu polygonů (decimace) modelu. Požadavkem je, aby geometrie zůstala zachována, ale počet trojúhelníků byl redukován na minimum.



Obrázek 2-12 Vyhlazený model (vlevo) a decimovaný model (vpravo) (převzato z [6]).

2.2.2 Volume rendering

Volume rendering zobrazuje přímo 3D data získaná z CT/MR. U volume renderingu (VR) není potřeba segmentace ani žádné jiné předzpracování dat. Jediný požadavek je namapování každého vzorku dat na hodnotu barvy a průhlednosti.

Data získaná z CT/MR, i jiných technik, jsou reprezentovány jednou hodnotou. Tato hodnota je závislá na použité technice snímání. Pro CT se jedná o reprezentaci Hounsfieldovy stupnice, která

udává absorpci rentgenového záření. Pro potřeby volume renderingu je potřeba převést tuto hodnotu na reprezentující barvu. Průhlednost je nutné dodat, aby bylo možné zobrazit průhledné struktury.

V této podkapitole budou dále rozebrány nejběžnější režimy, které se v medicíně při volume renderingu používají. Dále význam přenosových funkcí, předintegrace a konečně i významné techniky volume renderingu.

2.2.2.1 Režimy zobrazení

Na rozdíl od 3D vektorizace, volume rendering umožňuje uživateli dívat se na nasnímaná data různými způsoby. Při pohledu na prostorový objekt musíme pohled upravit tak, aby jej bylo možné zobrazit na 2D monitoru. Při této redukci je třeba nastavit správný úhel pohledu a zobrazit tento pohled jako jediný 2D obraz. Technik pro takové zobrazení je v medicíně několik. Následuje několik nejběžněji používaných metod.

Dále v textu bude používán pojem „paprsek“. Výsledný 2D obraz volume renderingu se skládá z konečného počtu obrazových bodů. Každý bod vznikl průchodem pomyslného paprsku od pozorovatele skrz prostorová data. Paprsek je možné si představit jako algoritmus, který vybírá data pro finální zobrazení na monitoru.

MIP

Zkratka pro Maximum Intensity Projection. Z celé dráhy paprsku se zobrazí pouze nejvyšší hodnota intenzity, kterou paprsek po své cestě 3D texturou navzorkoval. Vzhledem k tomu, že je nutné pouze porovnávat aktuální hodnotu s maximem, je tato metoda nejrychlejší.

Zobrazení MIP je velmi rychlé, ale nezachovává prostorovou hloubku v datech. Výsledný obraz působí plochým dojmem, který je možné vylepšit rotací objektu. Rotace pomáhá nalezení relativní polohy objektu v těle. Jelikož se jedná o ortografické zobrazení, není možné rozlišit pohled zepředu – zezadu a zleva – zprava.

V medicíně se MIP používá například při hledání plicních nádorů, jelikož je dokáže jasně odlišit od okolní tkáně.

Obrázek 2-13 Maximum Intensity Projection.

Simulace rentgenu

O něco složitější než MIP. Zatímco MIP vyhledá pouze nejvyšší hodnotu po cestě paprsku, rentgenové zobrazení sčítá veškerá data při své cestě. Kosti a jiné struktury s vysokou absorpcí rentgenových paprsků jsou v datech zvýrazněny tak, že hodnota navzorkovaných dat se umocní na číslo *n*. Tento exponent je závislý na charakteru dat a není předem dán. Teoretický rozsah hodnot pro celkový součet je podstatně větší než u MIP.

Aby bylo možné simulovat velmi vysoký dynamický rozsah skutečného rentgenu, je třeba zobrazovat data v logaritmické škále.

$$Výstup = \frac{\log \sum_{i=0}^{n} s(i)^4}{\log n}, n = \text{počet vzorků po cestě paprsku}$$

Rovnice 2-1 Simulace rentgenu.

Obrázek 2-14 Simulace rentgenu.

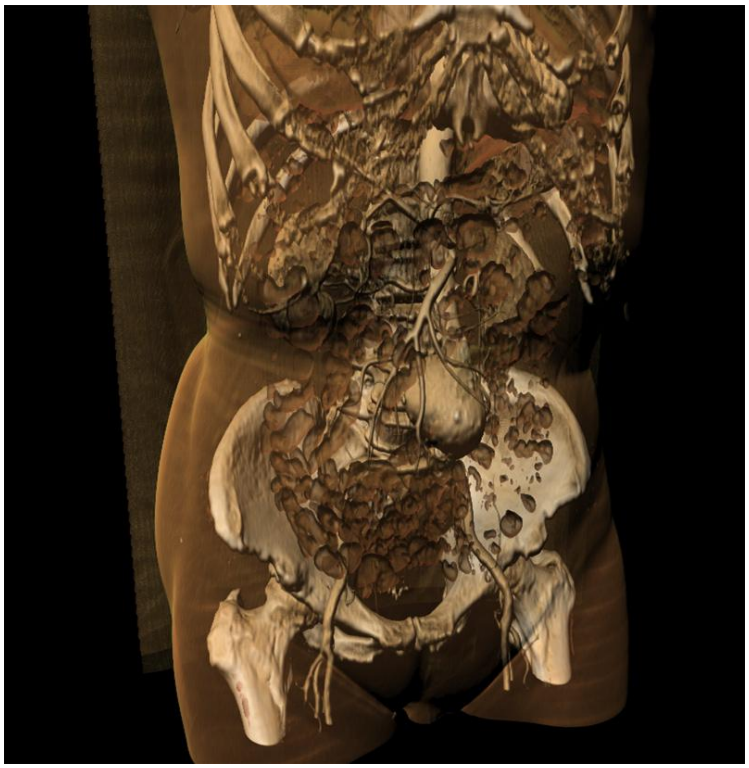
Uvedený vzorec vznikl experimentováním na reálných CT datech. Výsledný obraz je přirozenější než v případě MIP, ale nese méně detailů.

Stínování

Simulace stínování nemá zásadní diagnostický význam. Je však vizuálně nejzajímavější, protože rekonstruuje prostorovou strukturu zkoumaného objektu. Zásadní význam má přenosová funkce (viz následující kapitola) s barvami a průhlednostmi pro jednotlivé hodnoty vstupní intenzity. Pomocí této tabulky lze určit, které hustoty budou zobrazeny určitou barvou a také jejich průhlednost.

Při stínování se, kromě stanovení barvy bodu pomocí přenosové funkce, počítá směr normálového vektoru a následně Phongovo stínování. Pro výpočet stínování je třeba navíc znát směr světla. Při vhodně zvolené tabulce barev a průhledností je možné dosáhnout téměř fotorealistické kvality zobrazení.

Zatímco předchozí režimy neumožňují žádné širší nastavení parametrů zobrazení, u stínování je jich možné nastavit celou řadu. Mezi ty důležitější se řadí způsob výpočtu normálových vektorů a osvětlovací model.



Obrázek 2-15 Režim stínování.

Segmentace

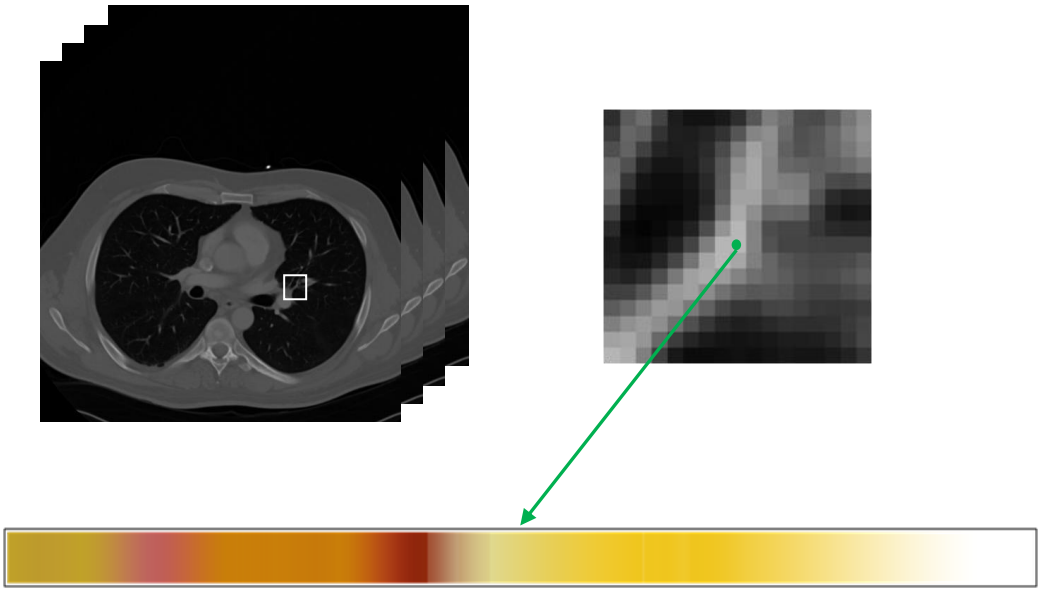
Jedná se o modifikaci metody stínování. Druhá 3D textura, obsahující jednobitové informace o hranách, je zapojena do výpočtu barvy pixelu. Tyto doplňující informace můžou pocházet ze segmentace dat. Segmentační algoritmus najde hrany v datech a ty se poté během průchodu paprsku zvýrazní. Tento postup není omezen pouze na zvýraznění hran, může také sloužit k označení oblasti zájmu, která může být zvýrazněna oproti okolí. Další výhodou značkování je snížení výpočetní náročnosti algoritmu, protože komplexní počítání normál a osvětlení může být ve většině nezajímavých dat zcela vypnuto.

2.2.2.2 Look-up tabulky

Tyto vyhledávací tabulky je možné použít ve všech režimech zobrazení. Největší dopad mají ovšem při stínování. Převádí 12 bitovou informaci o míře pohlcování rentgenových paprsků (u CT) na odpovídající optickou barvu a průhlednost. Tyto hodnoty nemusí odpovídat realistickým barvám.

1D tabulky

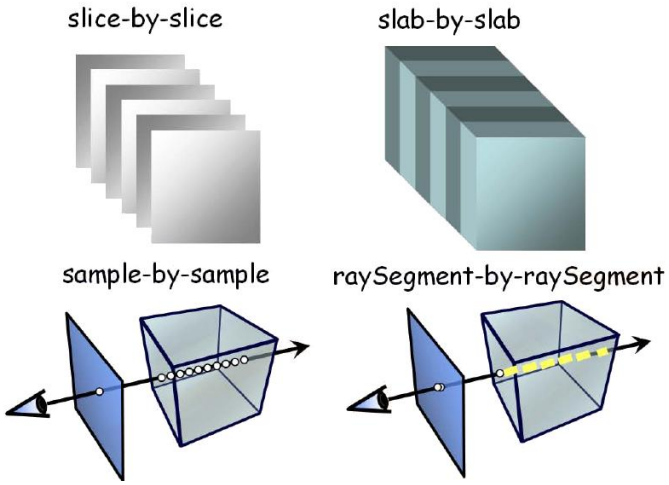
Jednorozměrné tabulky mají na starost jednoduchý převod intenzity na barvu a průhlednost. Každé možné hodnotě vstupních dat je v tabulce přiřazena odpovídající hodnota barvy a průhlednosti.



Obrázek 2-16 Look-up tabulka slouží k převodu hodnoty density z CT na reprezentující barvu.

2D předintegrované tabulky

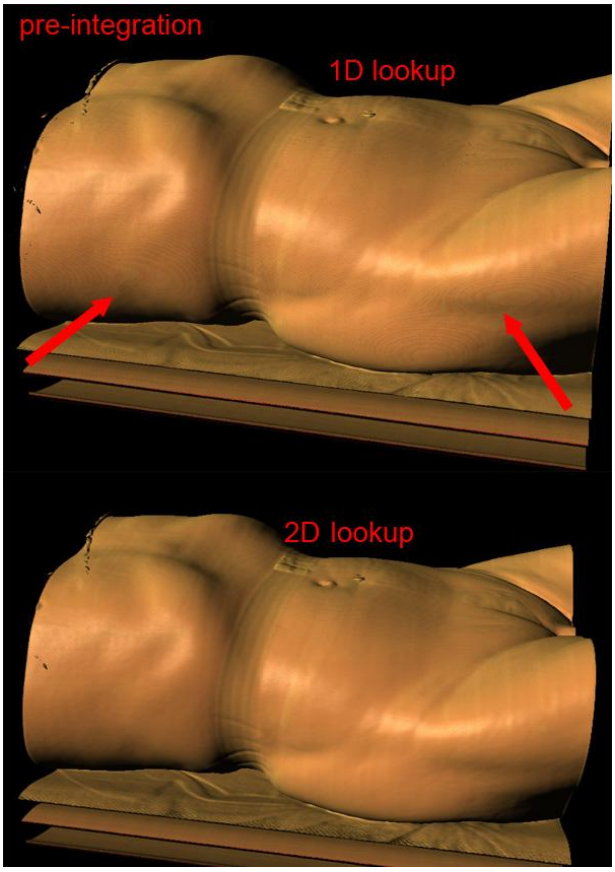
Při průchodu paprsku objemem dochází nevyhnutelně k vzorkování dat. Při přechodu paprsku z průhledné části textury do neprůhledné dochází na hranici k nežádoucím barevným přechodům. Tyto přechody jsou způsobeny nedostatečně jemným vzorkováním dat. Zmenšení vzorkovacího kroku ale přináší citelné zpomalení výpočtu. Řešením je použití předintegrované vyhledávací tabulky (pre-integrated look-up).



Obrázek 2-17 Předintegrované tabulky umožňují sčítat celé segmenty objemu a zabránit tak artefaktům (převzato z [5]).

Paprsek si po své dráze vždy pamatuje hodnotu předchozího vzorku. Společně s aktuálním vzorkem tato dvojice tvoří dvě souřadnice do 2D vyhledávací tabulky reprezentované 2D texturou. Zatímco 1D LUT přičítá při vzorkování pouze barevné hodnoty v jednom bodu, kde došlo k navzorkování 3D textury, pomocí předintegrované tabulky je možné přičítat celý úsek, který byl

paprskem přeskočen mezi vzorkovanými body. Na pozici, kde se setkají dva indexy v tabulce, je možné nalézt integraci všech hodnot, které v 1D tabulce leží mezi těmito dvěma indexy. V diagonále 2D předintegrované tabulky je uložena původní 1D LUT.



Obrázek 2-18 Artefakty způsobené diskrétním vzorkováním 3D dat a jejich potlačení.

Tato předintegrovaná LUT je schopná pracovat korektně pouze s jednou velikostí vzorkovacího kroku. Pro více délek kroku je třeba použít 3D předintegrovanou tabulku, kde ve třetí dimenzi je právě délka kroku.

$$A_{preint}\left(s_f, s_b, d\right) = \frac{d}{s_b - s_f}\Big(T(s_b) - T\big(s_f\big)\Big),$$

$$T(s) = \int_0^s \alpha(s)ds, \qquad \alpha = \text{1D alfa LUT}$$

Rovnice 2.2 Výpočet alfa kanálu předintegrované 2D tabulky, s_f – přední vzorek, s_b – zadní vzorek [3].

$$RGB_{preint}\left(s_f, s_b, d\right) = \frac{d}{s_b - s_f}\Big(K(s_b) - K\big(s_f\big)\Big),$$

$$K(s) = \int_0^s \alpha(s)\delta(s)ds, \qquad \alpha = \text{1D alfa LUT a } \delta = \text{1D rgb LUT}$$

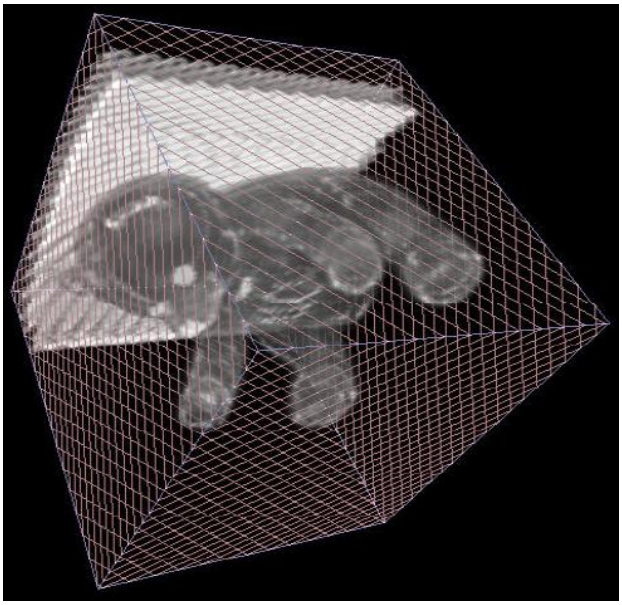
Rovnice 2.3 Výpočet barvy předintegrované 2D tabulky, barevná složka se zde násobí průhledností [3].

2.2.2.3 Metody

Tato kapitola popisuje technickou realizaci volume renderingu. Metoda vrhání paprsku (ray casting) je teoreticky velmi přirozený způsob zobrazení. Je ovšem také nejnáročnější na výpočetní výkon. Existuje řada zjednodušení, jejichž jediným účelem je zrychlení zobrazení na úkol vizuální kvality.

Texturování (blending)

Starší metoda volume renderingu. Anglický pojem blending znamená míchání, v tomto případě dochází k postupnému skládání řezů objemem s použitím průhlednosti. Nejprve se vytvoří sada polygonů, které odpovídají řezům objemem kolmo k pozorovateli (kameře). Na jednotlivé body polygonu se nanesou texturovací souřadnice tak, aby přímo odpovídaly poloze řezu v 3D textuře. Každý bod na polygonu dostane při vykreslování přiřazenu interpolovanou hodnotu intenzity z 3D textury. Z této informace je možné zjistit výslednou barvu pixelu a průhlednost pomocí vyhledávací tabulky. Tyto řezové polygony se v požadovaném směru postupně vykreslují do paměti snímku se zapnutým blendingem. Výsledkem je akumulace informace ze všech vykreslovaných řezů a tedy zobrazení celého objemu.



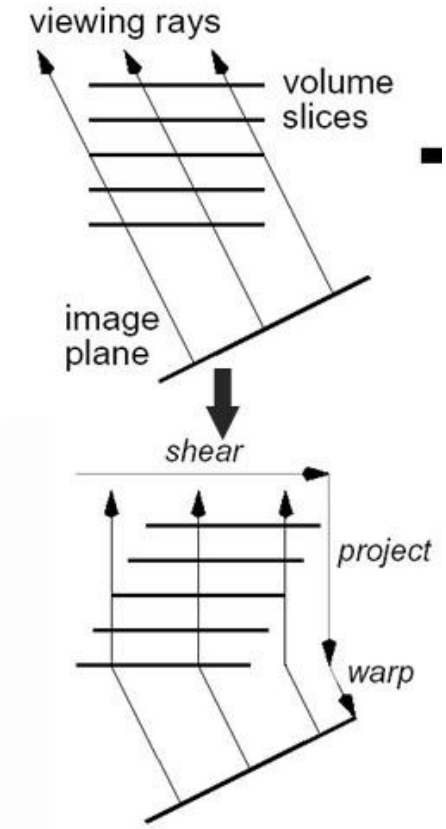
Obrázek 2-19 Při texturování je obraz tvořen kompozicí mnoha kolmých řezů (převzato z [7])

Tato technika je výhodná z hlediska požadavků na hardware. Je ji možné realizovat téměř na všech grafických akcelerátorech. Popsaný blending je relativně obecná metoda, která vyžaduje práci s celým 3D objemem najednou a podporu grafického hardwaru.

Shear-warp

Zjednodušená varianta texturování s vyšší rychlostí výpočtu. Základní výhoda shear-warp metody je její paměťová nenáročnost. Při bočním pohledu na objekt se data vždy prochází zepředu dozadu po jednotlivých rovinách. Boční pohled se simuluje tak, že se řezy před průchodem paprsku navzájem

posunou. Posunutí je dáno úhlem pohledu (shear). Vzniklý obraz je ale skosený a je třeba ho vyrovnat (warp).



Obrázek 2-20 Shear-warp ilustrace (převzato z [10]).

Splatting

Splatting je možné přirovnat k házení sněhovými koulemi na zeď. Celý objem je zobrazen jako velké množství rozmazaných plošek, obvykle eliptického tvaru. Velikost a průhlednost plošek se liší podle vzdálenosti a vlastnosti té části dat, které reprezentují. Praktická realizace spočívá ve vykreslování plošek, které svoji barvou odpovídají části zobrazovaných dat.

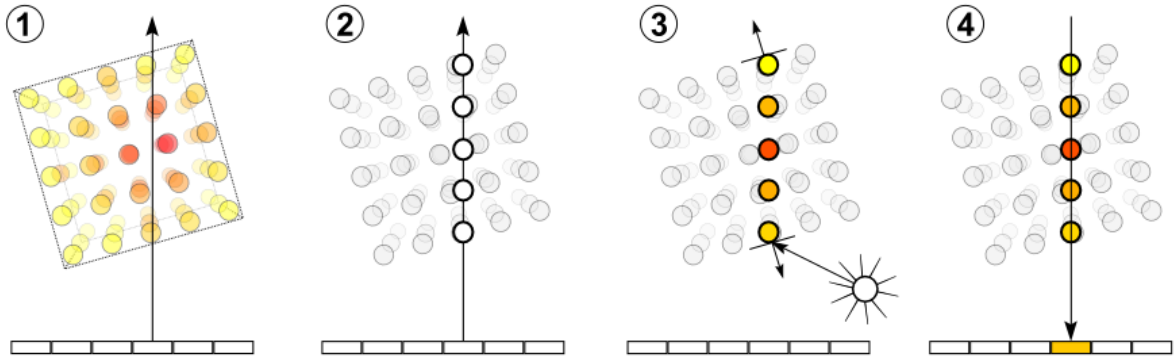
Obrázek 2-21 3D obraz vzniklý metodou voxel splatting (převzato z [7]).

Při „vržení“ dostatečně velkého množství těchto neostrých bodů se postupně začne rýsovat zobrazovaný objem. Tato technika je velmi rychlá, ale nedosahuje kvalit ray castingu.

Ray casting

Nejkvalitnější a nejpomalejší metoda. Na rozdíl od blendingu a splattingu, umožňuje ray casting zobrazení skutečného stínování, stínů a odlesků. Zobrazení objemu se provede tak, že z každého bodu obrazovky (případně okna, do kterého vykreslujeme) se sleduje paprsek, který prochází tělesem reprezentovaným 3D texturou. Všechny paprsky jsou vyslány rovnoběžně, pokud se jedná o paralelní projekci, a různoběžně v případě perspektivní projekce.

Paprsek po své cestě vzorkuje 3D texturu a akumuluje barvu. Akumulace barvy je větší u méně průhledných částí a naopak. Paprsek (a tím i přičítání nových barev) je tlumen při postupu objemem – opticky neprůhledné části tlumí paprsek více než např. vzduch (např. kolem snímaného člověka na CT skeneru). Paprsek je ukončen v případě, že je zcela utlumen (narazil na neprůhledný objekt apod.) nebo pokud opustil 3D texturu.



Obrázek 2-22 1. Výpočet dráhy paprsku, 2. Určení bodů pro vzorkování objemu, 3. V těchto bodech se spočítají normálové vektory a osvětlení, 4. Všechny vzorky se sečtou do výsledného bodu na obrazovce (převzato z [2]).

Vrhání parsku je vysoce adaptibilní metoda, kterou je možné zobrazovat data téměř libovolných způsobem. Implementace všech režimů z kapitoly 2.2.2.1 je možná.

2.3 Hardwarová akcelerace

Vzhledem k velkému objemu zpracovávaných dat a faktu, že je tato data nutné při každém zobrazení výsledného snímku znovu celá zpracovat, vzniká potřeba urychlení výpočtu. V běžném počítači jsou pouze dvě možnosti: hlavní procesor (CPU) a grafická karta (GPU). Výkon grafických karet ve specifických případech až stonásobně převyšuje možnosti CPU.

2.3.1 Grafické procesory

Algoritmus vrhání paprsků, který je obecně nejsložitější, vyžaduje použití cyklu, ve kterém paprsek cestuje 3D texturou z jedné strany na druhou. Při rozlišení dat 512×512 bodů a faktem, že je potřeba

každý voxel vzorkovat alespoň dvakrát [5], je zřejmé, že takový cyklus musí být schopný iterovat přibližně tisíckrát.

Vývoj programovatelných grafických procesorů byl velmi rychlý a první sériově vyráběný procesor, který byl schopný vrhání paprsků zvládnout, byl NV40 od firmy NVIDIA. Tento procesor zavedl nový Shader Model verze 3. Klíčové vlastnosti byly:

- cyklus omezen na 255 průchodů, ale s možností vnoření
- možnost předčasně ukončit cyklus (příkazy break a continue v jazyce C)
- podpora 3D textur do velikosti 512^3
- shader může mít délku až 512 instrukcí
- celkový počet provedených instrukcí max. 65535
- počet obecných registrů: 32

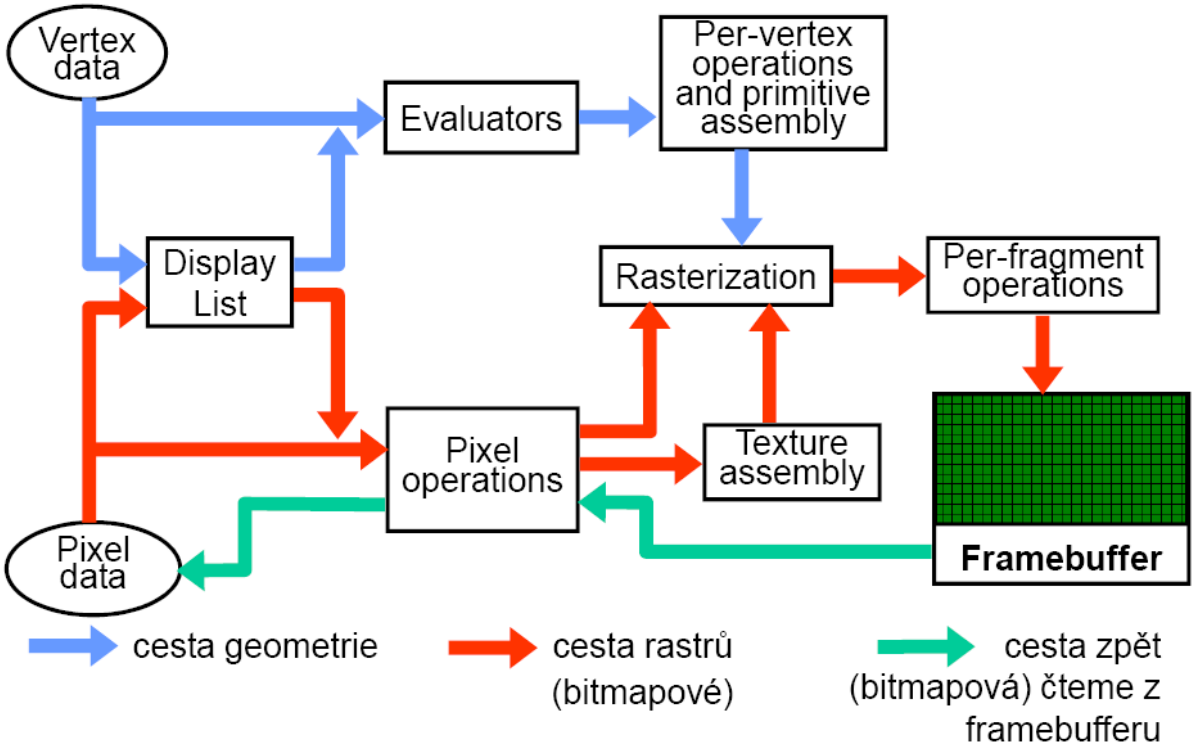
Právě tyto vlastnosti umožnily implementaci skutečného vrhání paprsků 3D texturou. I když bylo zrychlení oproti CPU velké, čip stále algoritmus zásadně omezoval. Většina limitů (viz výše) byla stále ještě malá, aby umožnily plnohodnotnou a efektivní implementaci. Další problém představoval způsob, jakým jsou jednotlivé paralelní jednotky spuštěny. Grafické karty byly určeny převážně pro akceleraci her, kde se předpokládá, že sousední body budou podmíněné výrazy a cykly procházet stejně. Blok několika desítek pixelů bude vždy provádět úplně stejný kód. Musí se tedy přizpůsobit „nejpomalejšímu“ pixelu a snižuje tak efektivitu právě u výpočtu jakým je vrhání paprsků. Grafický procesor je navíc vysoce paralelní a každá jednotka pracuje vždy se čtyřmi čísly zároveň. Tento fakt je výhodný pro zpracovávání barevných textur, ale u vrhání paprsků zůstává část výkonu nevyužita.

S příchodem čipu G80 (NVIDIA) byla většina těchto problémů odstraněna. Všechny limity vzrostly vysoko nad potřeby volume renderingu a výkon hardwaru je možné plně využít. Při porovnání podobných grafických karet série GeForce 6600 a GeForce 8600, dochází k až desetinásobnému nárůstu výkonu ve prospěch novější karty.

Karty na bázi G80 přinesly další novinku – celočíselné operace. Doposud bylo možné v GPU pracovat pouze s čísly typu float (32bitů).

2.3.1.1 Architektura grafických procesorů

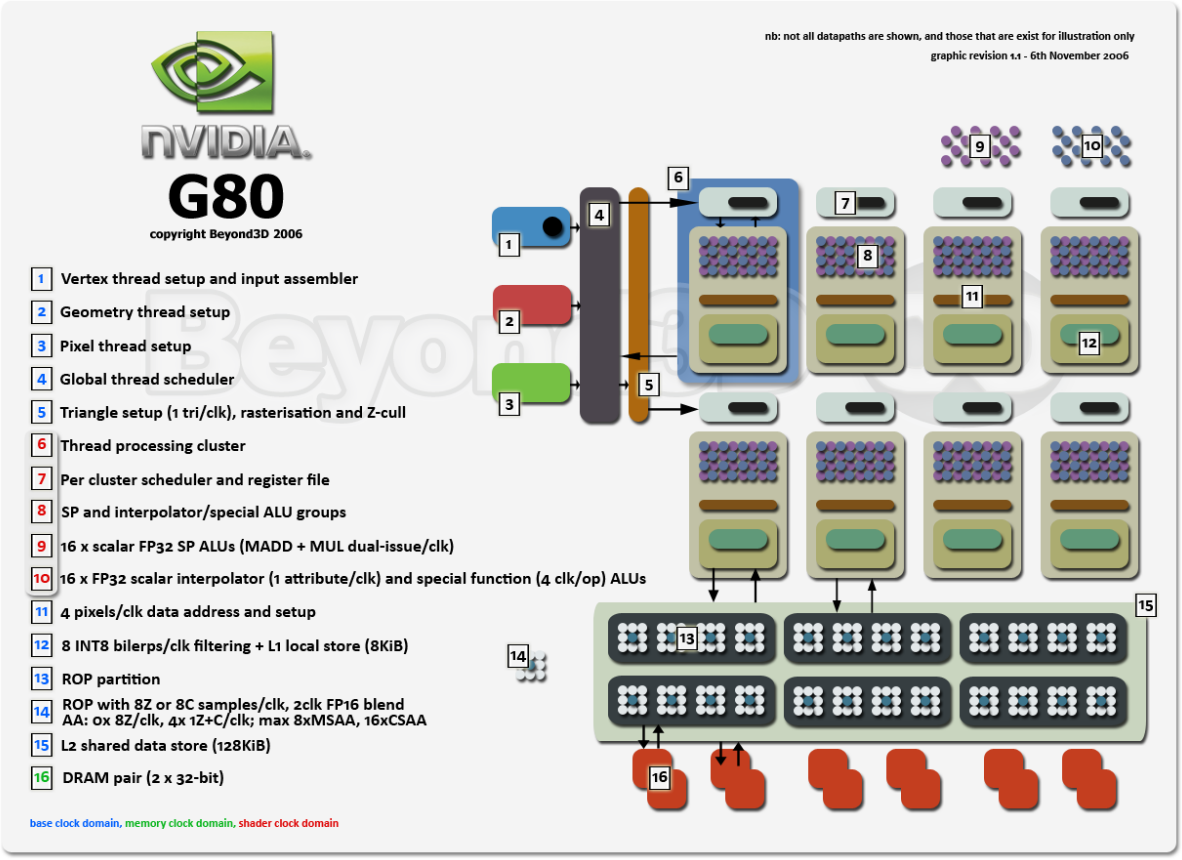
Moderní grafické procesory se podstatně liší od grafických karet první generace. Starší grafické procesory obsahovaly pouze jednotky dedikované pro jednotlivé kroky zobrazení 3D scény. Zobrazení volume renderingu bylo možné pouze pomocí technik jako Shear-warp. Nejnovější generace procesorů mají značně zjednodušenou architekturu, kterou je možné využít k nejrůznějším úlohám.



Obrázek 2-23 Řetězec vykreslování v OpenGL; většina výkonu GPU je soustředěna v per-fragment operacích (pixel shadery) a per-vertex operacích (vertex shadery) (převzato z [8]).

Následující diagram zobrazuje schéma grafického procesoru G80. Malá část obvodů se stará o přípravu dat, zobrazování na monitor a koordinaci. Většina grafického čipu je tvořena menšími procesory, které jsou schopny vykonávat libovolný program v rámci jejich instrukční sady. Uživatel má možnost napsat malé programy, které se na těchto procesorech spustí při vykreslování scény.

Obecné procesory, které jsou schopné zpracovávat libovolné úlohy, jsou součástí poslední generace grafických čipů. Předchozí generace se dělily na procesory pro zpracování vertexů a fragmentů (pixelů). Díky své vysoké specializaci nebyly tyto čipy vhodné pro obecný volume rendering (např. ray casting) a nejlepších výsledků se dosahovalo metodami jako např. Shear-warp.



Obrázek 2-24 Architektura procesoru G80 (převzato z [9]).

2.3.1.2 Stream procesory

Současné grafické čipy obsahují až několik set jednoduchých procesorů. Tyto procesory jsou napojeny na hlavní paměť GPU, vyrovnávací paměti L1 a L2, sady registrů a pomocné jednotky. Konkrétní vnitřní architektura procesorů a jejich uspořádání se mění z generace na generaci a zásadní rozdíly jsou i mezi jednotlivými výrobci grafických čipů. Schopnosti těchto procesorů jsou však dány verzí podporované grafické knihovny.

Výhody GPU procesorů

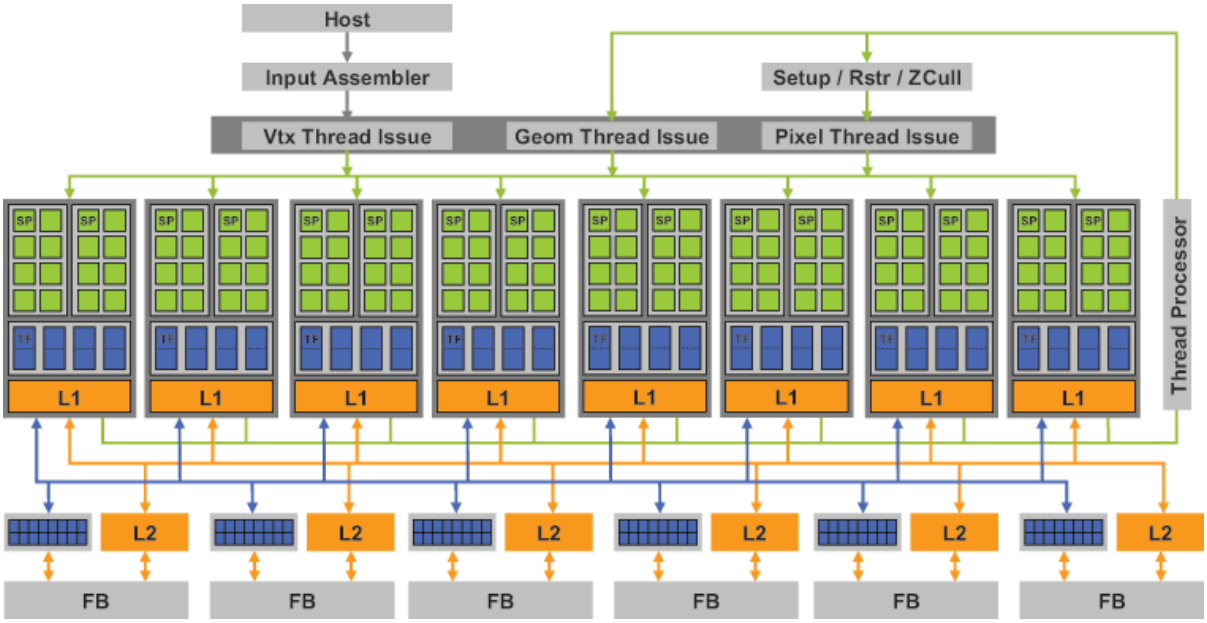
- velké množství vyrovnávacích pamětí urychluje přístup k datům
- specializované jednotky: lineární/bilineární/trilineární interpolace, trigonometrické funkce
- množství instrukcí pro grafiku: minimum, maximum, osvětlení, saturace, MADD, …
- paralelní zpracování vektorů o 4 složkách
- velmi rychlé a efektivní operace nad 32 bitovými čísly float a 24 bitovými čísly integer
- dostatek registrů pro všeobecné použití (více než 4000)

Nevýhody GPU procesorů

- větvení kódu znatelně zpomaluje výpočet
- omezená délka programu

- omezený počet vykonatelných instrukcí
- omezení počtu opakování cyklu
- chybějící podpora pro datové typy nad 32 bitů
- nemožnost komunikace mezi vlákny

Přestože jsou grafické procesory navrženy především pro náročné hry, jejich architektura je vhodná k akceleraci 3D medicínských dat prakticky všemi metodami. V případě metody vrhání parsků lze celou dráhu paprsku simulovat v jednom procesoru.



Obrázek 2-25 Diagram stream processorů G80 (převzato z [9]).

Samotné čipy umožňují zpracování několika instrukcí najednou – například instrukce ADD a MADD je možné za určitých okolností počítat současně. Složité instrukce jako sin, cos, sqrt, dot jsou urychleny vyhledávacími tabulkami a jejich výpočet je relativně rychlý (jednotky taktů). I tyto instrukce pracují paralelně s jednotkami pro základní operace. Další výhodou grafických procesů je inteligentní načítání dat z texturovací paměti, kde se při čekání na data vykonávají jiné instrukce dopředu. Tyto vlastnosti se podobají schopnostem klasického CPU, s tím rozdílem, že GPU je zaměřeno na práci v plovoucí desetinné čárce, obsahuje až stovky procesorů a implementuje hardwarově složitější funkce, které jsou v grafice potřebné.

2.3.2 Shadery

Shadery jsou malé programy psané v asembleru nebo jazyce C. Po zavedení do grafické karty se stejný program spouští paralelně pro každý pixel. Omezení paralelizace vychází z počtu jednotek, které jsou na grafickém čipu k dispozici. Algoritmus vrhání paprsků je tak psán pouze pro jediný paprsek a grafický čip tento program spustí pro každý výstupní obrazový bod.

2.3.2.1 Dostupné knihovny

V současné době existují tři různé knihovny, pomocí kterých lze programovat grafické procesory. Výkon shaderů je nezávislý na použité knihovně, ta může ovšem omezovat přenositelnost mezi operačními systémy a různými výrobci grafických čipů.

Direct 3D

Grafická knihovna od firmy Microsoft. Svoji komplexností se hodí na větší a složitější projekty. Jednotlivé verze velmi rychle zastarávají, což je důvod, proč se Direct 3D nepoužívá v profesionálních aplikacích. Jistou překážkou může být i nepřenositelnost na jiné systémy.

Pro programování shaderů používá Direct 3D jazyk HLSL (High Level Shading Language). Programy v HLSL jsou distribuovány ve zdrojovém kódu, který se podobá jazyku C a překládány za běhu aplikace ovladačem grafické karty. Následuje ukázka shaderu psaného v HLSL:

```
sampler2D g_samSrcColor;

float4 MyShader( float2 Tex : TEXCOORD0 ) : COLOR0
{
    float4 Color;

    Color = tex2D( g_samSrcColor, Tex.xy);
    return Color;
}



technique PostProcess
{
    pass p1
    {
        VertexShader = null;
        PixelShader = compile ps_2_0 MyShader();
    }

}
```

OpenGL

Konkurenční knihovna Direct 3D. Knihovna je přenositelná mezi platformami, podporuje celou řadu programovacích jazyků a sestává z více než 250 funkcí. Na rozdíl od Direct 3D, OpenGL odstiňuje

samotný hardware a všechny funkce jsou dostupné na každé grafické kartě. V případě potřeby je použita softwarová emulace. Nejnovější funkce GPU jsou v OpenGL zpřístupněny pomocí rozšíření (extensions).

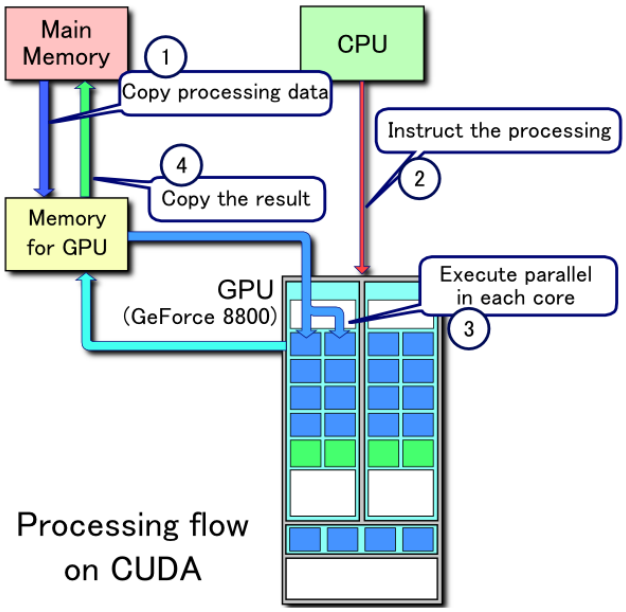
Psaní shaderů je od verze OpenGL 2.0 realizováno v jazyce GLSL (GL Shading Language). Při použití tohoto jazyka není nutná žádná další knihovna – překlad probíhá v ovladači grafické karty. GLSL je podobný jazyku Cg od NVIDIA, který ale vyžaduje použití další knihovny. Jazyk Cg však nabízí některé funkce (rychlejší datové typy, specializované funkce), které GLSL postrádá. Ovladače NVIDIA umožňují použití těchto Cg datových typů a funkcí v rámci jazyka GLSL. Pro karty jiných výrobců je možné tyto chybějící funkce emulovat pomocí vestavěných GLSL typů s použitím preprocesoru jazyka C. Následuje opět příklad shaderu v jazyce GLSL.

```
void main()
{
      vec4 v = vec4(gl_Vertex);
      v.y = sin(0.5 * v.x);

      gl_Position = gl_ModelViewProjectionMatrix * v;
      gl_FrontColor.rgb = vec3(0.0, 0.0, 1.0);
}
```

CUDA

Na rozdíl od předchozích knihoven není CUDA grafická knihovna, ale pouze rozhraní od firmy NVIDIA se kterým lze přistupovat k procesorům na grafické kartě. Zdrojový kód se i v CUDA podobá jazyku C. CUDA je nástroj se kterým zle využít sílu grafického procesoru i pro úlohy, které nesouvisí s grafikou.



Obrázek 2-26 Schematický diagram CUDA (převzato z [9]).

Výhody CUDA proti grafickým shaderům:

- náhodný přístup do celého paměťového prostoru
- sdílená paměť – malá část paměti (16KB) je sdílená pro všechna spuštěná vlákna
- rychlejší kopírování dat z hlavní paměti
- podpora celočíselných datových typů

Nevýhody proti klasickým výpočtům na CPU:

- není možná rekurze
- omezená propustnost sběrnice mezi CPU a GPU
- větvení nezpomaluje výpočet, pokud celý blok vláken větví stejnou cestou
- podpora pouze pro karty NVIDIA

2.3.2.2 Vertex Shadery

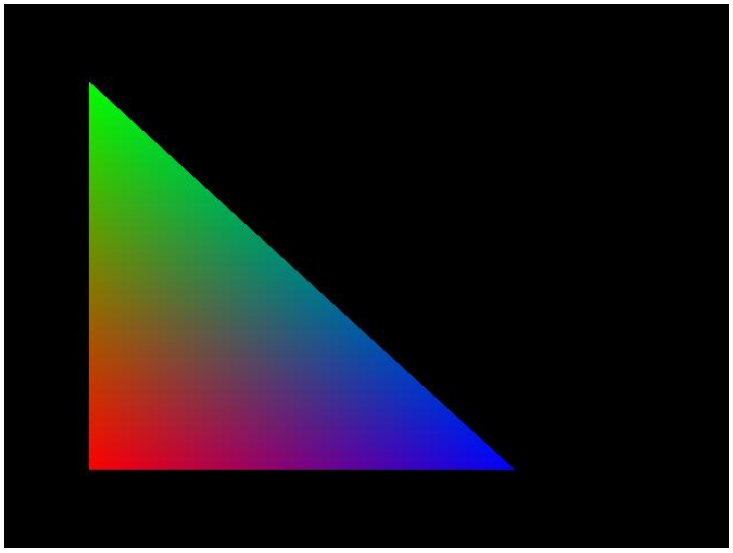
Nahrazují klasickou piperine, která se stará o transformaci vertexů, osvětlení, mapování textur a jiné pomocné výpočty na úrovni vrcholů. Vertex shadery (VS) zcela nahrazují původní zpracování vrcholů, proto je nutné při vytváření shaderu provést explicitně transformaci vrcholů, osvětlení apod. K těmto základním operacím je možné přidat téměř libovolné množství uživatelského kódu, který provede další transformace.

Přestože je v shaderu možné provádět libovolné operace, vstupy a výstupy jsou částečně omezené. Vstupní parametry VS jsou:

1. souřadnice vrcholu ve scéně
2. texturovací souřadnice zadané pro tento vrchol
3. další uživatelem definované atributy vrcholu

4. parametry předané všem spuštěným vertex shaderům pro všechny vrcholy
5. uživatelem definované textury

Vertex shader nepracuje s celými trojúhelníky a jeho výstupem je opět souřadnice vrcholu, nyní již transformovaná. Shader je také zodpovědný za přípravu hodnot pro interpolační jednotky, které tyto údaje používají při rasterizaci k rozložení hodnot přes celý trojúhelník. Typický příklad je definice barvy. Další možné využití je pro náročné operace, které stačí provést na úrovni vrcholů a pro jednotlivé pixely je pouze interpolovat. Tato technika se běžně využívá pro výpočet osvětlení.



Obrázek 2-27 Barvy definované ve VS jsou později automaticky interpolovány v celém trojúhelníku.

U zobrazování 3D medicínských dat se VS uplatní pouze u metod, které využívají velké množství vertexů. Sem spadá například 3D vektorizace dat a jejich zobrazování jako geometrie. Z metod volume renderingu je možné zmínit voxel splatting.

2.3.2.3 Geometry Shadery

Pokud je tento shader aktivován, bere na svém vstupu data přímo z vertex shaderu. V geometry shaderu (GS) se ovšem pracuje s celým grafickým primitivem: body, úsečky a trojúhelníky. Ve skutečnosti tedy geometry shader přebírá jeden až několik jednotlivých výstupů VS najednou.

Smyslem GS je generování dalších primitiv. Z jednoho bodu může vzniknout celá řada trojúhelníků, jiných bodů, atd. Běžné použití GS je při generování nových primitiv tak, aby se model více vyhladil. Této technice se říká tesselace. Geomerty shader snižuje zatížení sběrnice tím, že potřebnou geometrii vytváří až těsně před rasterizací.

V medicíně je GS vhodné využít u technik, které zobrazují objem pomocí velkého množství malých geometrických útvarů (elipsy, čtyřstěny, …). Z jediného vstupního bodu, který nese atributy o barvě a průhlednosti, lze vygenerovat složitější geometrii, která se následně zobrazí.

2.3.2.4 Pixel Shadery

Do pixel shaderu (PS) vstupují fragmenty vytvořené při rasterizaci trojúhelníků. Fragmenty jsou body, které vyplňují prostor mezi vrcholy rasterizovaného trojúhelníku. Nejedná se ovšem o výstupní obrazové body (pixely), protože například při zapnutém vyhlazování (antialiasing) se generuje podstatně více fragmentů než pixelů. Právě pixel shadery jsou finální krok zpracování geometrie před uložením do paměti snímku. Výstupem PS je barva fragmentu, průhlednost a hloubka. Toto platí při zobrazování na monitoru. Existuje volba, která přesměruje výstup pixel shaderu do textury v paměti grafické karty. Tento výstup není omezen na jednu texturu, moderní čipy dovolují zápis do až 16 různých textur najednou. Tímto způsobem lze výsledné textury znovu použít v dalším renderovacím kroku jako vstup. V herních aplikacích se podobné techniky využívají při rozmazání obrazu, efektu nočního vidění, apod.

Podobně jako u VS, i PS mají na vstupu konstanty a textury, které byly definovány před vykreslováním. Další vstupní data jsou: souřadnice fragmentu v okně, interpolované hodnoty z vertex shaderů a interpolované texturovací souřadnice. Z těchto informací PS běžně vzorkuje dostupné textury, dopočítává osvětlení, bump mapping, displacement mapping, průhlednost, per-pixel osvětlení atd. Pixel shader má ze všech shaderů nejširší výpočetní možnosti a zpravidla spotřebuje většinu času při vykreslování snímku (u běžných aplikací jako jsou např. hry).

Pixel shader se uplatní téměř u všech metod volume renderingu, kde jednodušším technikám jako texturování a shear-warp stačí kratší shader. Naopak u ray castingu bývají programy delší a výrazně komplikovanější.

2.3.2.5 Compute Shadery

V současné době (počátek roku 2009) jsou compute shadery (CS) pouze součástí specifikace nového rozhraní Direct X 11. Compute shadery kombinují výhody architektury CUDA a klasické grafické operace. Compute shader bude spouštět vlákna s malou sdílenou pamětí a náhodným přístupem do paměti. Cílem CS je poskytnout grafickým aplikacím výhody obecných výpočetních vláken.

2.3.2.6 Provázanost s GPU

Nejmodernější grafické procesory obsahují unifikované shadery – obecné procesory, které se při vykreslování snímku postupně rekonfigurují na roli vertex, geometry a nakonec pixel shaderu, a plní per-vertex a per-pixel operace (viz obrázek 2-21). Protože jsou shadery ve většině případů psány ve vyšším programovacím jazyce, je od překladače požadována maximální efektivita. Tento požadavek je ještě umocněn faktem, že jednoduché procesory na GPU neobsahují žádné jednotky pro dynamické přeskládávání instrukcí takové, jaké jsou běžně u CPU.

3 Návrh a implementace

3.1 Výběr technologie

Základní požadavek na výsledný program je možnost plně interaktivního zobrazení zvoleného 3D objektu. Při prohlížení objemových dat musí být umožněna rotace, posun a zoom načtené části lidského těla. Další požadavky zahrnují změnu zobrazovacího algoritmu v reálném čase, adaptivní nastavení kvality výsledného zobrazení, možnost dodatečné segmentace a minimalizaci vizuálních artefaktů, které snižují kvalitu renderingu. Program je zároveň potřeba navrhnout tak, aby korektně fungoval na většině grafických akcelerátorů.

3.1.1 Metoda zobrazení

Vzhledem k pokroku ve schopnostech GPU zpracovávat složitější algoritmy, jsem zvolil ray casting jako základ celé práce. Objemová data budou v nezměněné podobě přenesena do paměti grafické karty. Jednotlivé režimy zobrazení mohou být reprezentovány pixel shadery. Pro každý vykreslený bod volume renderingu se spustí jeden pixel shader, který simuluje dráhu paprsku daty.

3.1.2 Knihovny

OpenGL se používá v řadě profesionálních aplikací, má širokou podporu napříč platformami a obsahuje vlastní jazyk pro psaní shaderů. Nad OpenGL je postavena knihovna Open Scene Graph (OSG), která využívá graf scény a odstiňuje nízkoúrovňové příkazy OpenGL. Právě tyto knihovny jsem zvolil pro realizaci volume renderingu.

3.1.3 Optimalizace

Ruční optimalizace kódu shaderů v asembleru jsou nepraktické, proto se optimalizace zaměří na maximální efektivitu zdrojového kódu shaderů a různé stupně kvality výstupu.

Použitý algoritmus ray casting má tu vlastnost, že pro každý výsledný bod na obrazovce se sleduje jeden paprsek, který prochází objemovou texturou. Velikost vstupních dat bývá 256^3 – 512^3. Pokud by okno ray castingu značně převyšovalo tuto velikost, počet snímků zobrazených za sekundu by začal rychle klesat, ovšem bez citelného nárůstu vizuální kvality. Z toho důvodu jsem zvolil postup, kdy je 3D objem vždy vykreslován do textury pevné velikosti, která se na konci přepočítá na velikost okna.

3.2 Návrh aplikace

Celý postup se dá rozložit na čtyři významné fáze. Nejprve se při startu programu vytvoří a naplní všechny potřebné datové struktury. Ty zahrnují 3D data, příslušné look-up tabulky, pomocné textury a jiné. Fázi vykreslování je možné rozložit na tři samostatné části.

3.2.1 Vstupní data

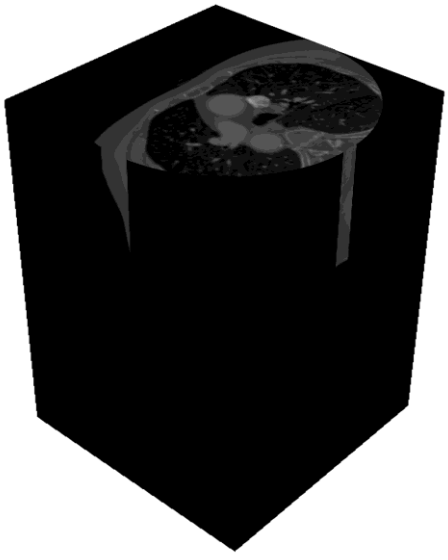
Maximální možný rozměr a bitová hloubka vstupních dat je limitována pouze aktuálně použitým hardwarem. Běžné rozměry jsou do 512^3 s 16 bitovou hloubkou (pouze prvních 12 bitů je použito). U karet řady G80 je rozměrový limit 2048^3 a hloubka je omezena na 32 bitů.

3.2.2 Look-up tabulky

Tyto vyhledávací tabulky je možné použít ve všech režimech zobrazení. Největší dopad mají ovšem při stínování. Převádí 12 bitovou informaci o míře pohlcování rentgenových paprsků na odpovídající optickou barvu a průhlednost. Tyto hodnoty nemusí odpovídat realistickým barvám.

3.2.3 Příprava pohledu

Celá data jsou mapována na kvádr ve 3D prostoru tak, že jeho stěny jsou totožné s okraji objemových dat. Pokud bychom na tento kvádr namapovali odpovídající část dat, viděli bychom například první a poslední snímek CT skenu jako jeho horní a spodní stranu. Tímto způsobem tedy vidíme pouze dva krajní řezy a ze všech ostatních jen pouhé okraje. Ray casting (viz další podkapitola) vrhá paprsek z každého bodu kvádru, jehož část je vidět při aktuální transformaci.



Obrázek 3-1 Kvádr s namapovanou 3D texturou. Na horní stěně je vidět první řez hrudníkem. Body na povrchu kvádru jsou počáteční body pro vrhané paprsky v shaderu.

Vzniká ovšem problém, jak určit směr cestování paprsku a jeho koncový bod. Toto je možné vyřešit tak, že paprsek, při své cestě 3D texturou zná nejen svůj počáteční bod, ale i koncový bod na kvádru. Pro každý zobrazený snímek se tedy nejprve vykreslí zadní (neviditelné) strany a pro každý bod se uloží jeho poloha v rámci 3D dat. Z těch dat bude možná v další kroku dopočítat dráhu paprsku při libovolném nastavení kamery.

3.2.4 Ray casting

V tomto kroku se vykreslují přední strany kvádru a počítá se dráha paprsku objemem. Poté se pro každý bod spouští algoritmus ray casting. Tento proces je zcela v režii grafického čipu a je podrobněji popsán dále. Výsledný obraz se ukládá do paměti, jejíž velikost je předem stanovená uživatelem a přímo ovlivňuje konečnou kvalitu.

3.2.5 Přizpůsobení oknu

V této fázi se vyrenderovaný obraz prezentuje uživateli na obrazovku. Je téměř jisté, že okno uživatelské aplikace bude mít jinou velikost než vzniklý obraz volume renderingu. Pro potřeby libovolného zvětšování a zmenšování obrazu bude použit samostatný program pro pixel shader, který provede změnu velikosti v maximální možné kvalitě.

3.3 Implementace

3.3.1 Možnosti programu

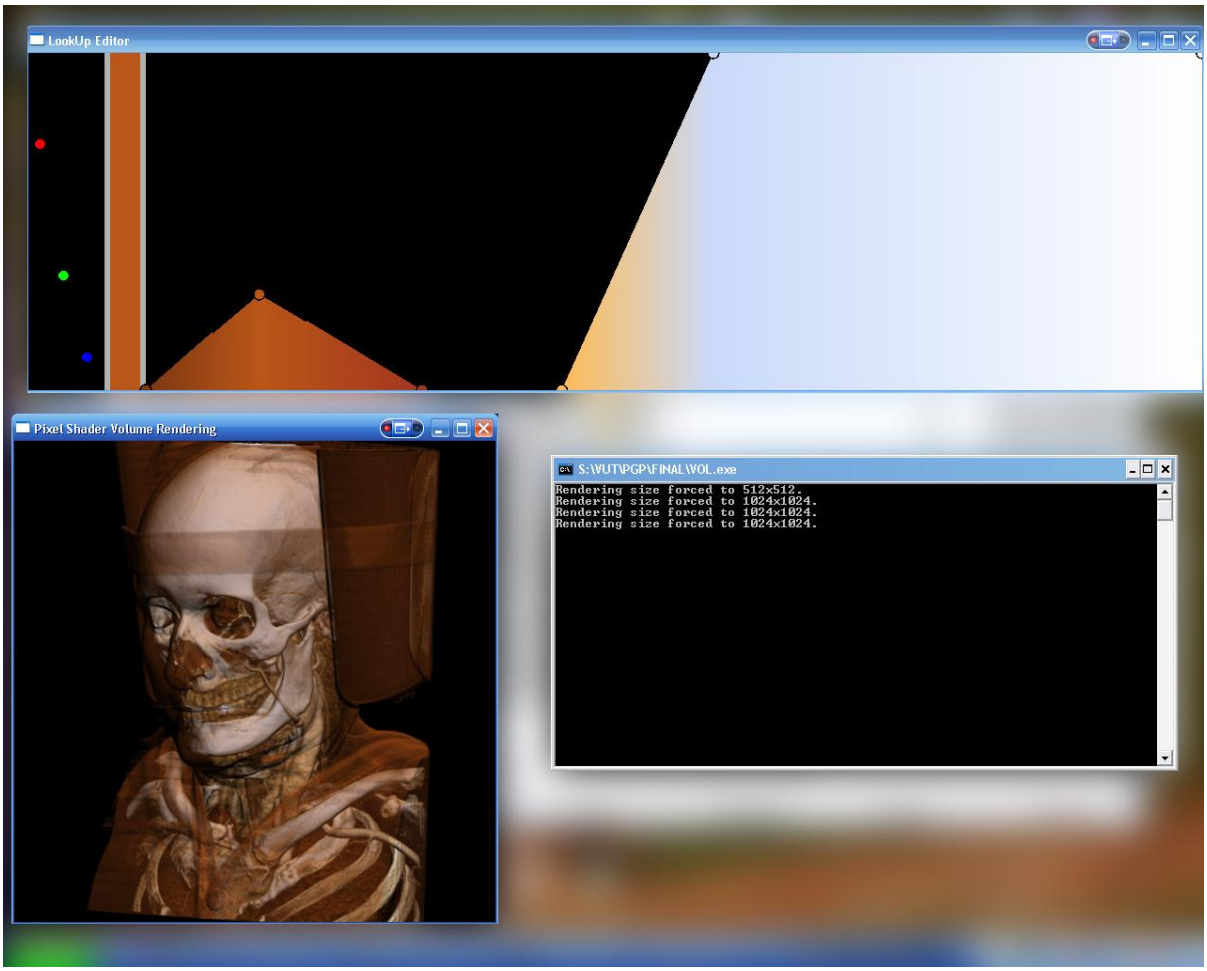
Uživatel má možnost provádět libovolné transformace objemu jako například otáčení, zoom a posun. Dále je možné vybrat si jednu z přednastavených look-up tabulek nebo vytvořit tabulku vlastní pomocí editoru.

Kromě stínovacích režimů je implementován mód MIP a simulace rentgenu.

3.3.1.1 Editor look-up tabulky

Součástí programu je interaktivní editor look-up tabulek. Vzhledem k tomu, že CT snímky jsou v grafické paměti uloženy ve své původní netransformované podobě, je možné měnit použitou tabulku v reálném čase. Tato možnost znatelně zrychluje rychlost vytvoření nové tabulky.

Do okna editoru je možné vkládat libovolné množství řídících bodů a každému nastavit vlastní barvu a průhlednost. Z této grafické reprezentace se dopočítá celý rozsah hodnot tabulky a pole se přenese do grafické karty při každé změně.



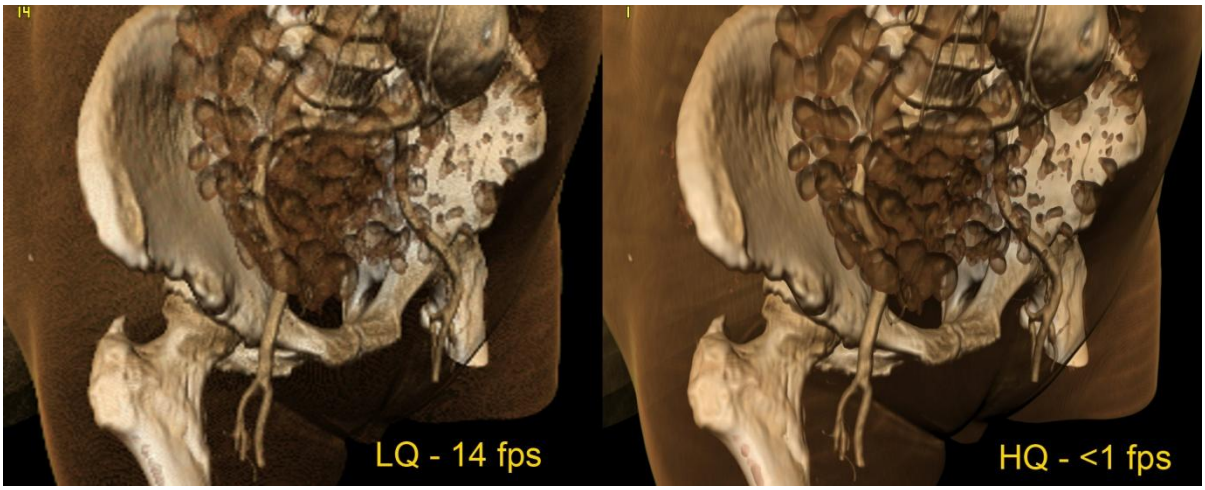
Obrázek 3-2 Editor look-up tabulky a průběžný výsledek. Výška bodu v editoru reprezentuje požadovanou průhlednost v daném místě tabulky. Každý kontrolní bod v tabulce je ovládán myší a definuje barvu a průhlednost. Hodnoty mezi kontrolními body se lineárně interpolují, a celá tabulka se odesílá do GPU.

Editor zároveň umožňuje vytvoření dvourozměrné předintegrované look-up tabulky. Tato operace však trvá několik minut a není tedy možné s 2D tabulkou pracovat interaktivně.

3.3.1.2 Režimy zobrazení

Protože plná interaktivita a maximální vizuální kvalita jdou proti sobě, rozhodl jsem se implementovat dva režimy zobrazení – rychlý náhledový a pomalý kvalitní.

Při extrémním zvětšení zkoumaného objemu ve velkém okně se zbytečně v každém bodě vrhá paprsek, přestože by stačil pouze zlomek těchto paprsků k dosažení stejného výsledku. Z tohoto důvodu jsem omezil velikost výstupu renderingu. Výsledek ray castingu se zapisuje do textury o zadané velikosti a teprve ta je roztažena přes výstupní okno.



Obrázek 4-1 Při velkém zvětšení je výhodnější rychlý režim (výstup přes celou obrazovku).

Režim maximální rychlosti

- výstupní rozlišení je omezeno na 256×256 případně 512×512 bodů
- výpočet normál je proveden pouze z šestice okolních bodů
- nepočítá se spekulární složka Phongova stínování
- používá se pouze 1D LUT
- vzorkovací krok je prodloužen o 50%

Režim vysoké kvality

- výstupní rozlišení je typicky 1024×1024 bodů
- výpočet normál 3D Sobelovým operátorem
- plné Phongovo stínování
- předintegrovaná 2D LUT
- vzorkovací krok 2× pro každý bod v textuře (tedy přes 1000 kroků u tex. 512^3)

3.3.2 Shadery

Všechny použité pixel shadery sdílejí stejnou základní kostru. Rozdíly jsou pouze v tom, jakým způsobem se zpracovává navzorkovaná 3D textura.

3.3.2.1 Struktura shaderu

Preprocesor

Protože se v shaderech používají funkce a datové typy jazyka Cg, je nutné v preprocesoru ošetřit jejich možné použití na hardwaru jiného výrobce než NVIDIA. S použitím příkazu

```
#ifndef __GLSL_CG_DATA_TYPES
```

je možné názvy těchto funkcí a datových typů předefinovat na hardwaru jiných výrobců.

Druhé použití preprocesoru je v detekci Shader Model 4.

```
#ifndef GL_EXT_gpu_shader4
```

Tento příkaz je použit při definici funkce, která čte bit z druhé 3D textury obsahující informace o segmentaci. Zatímco Shader Model 3 musí tento bit získat složitým výpočtem s čísly typu float, Shader Model 4 a bitové instrukce AND, OR, SHIFT stejnou práci vykonají mnohonásobně rychleji. Díky preprocesoru je možné mít obě verze v jednom souboru a jejich výběr se provádí až při překladu v ovladači grafické karty.

Vstupní textury

- ***Volumetrická data***

Trojrozměrná trilineárně interpolovaná textura obsahující vykreslované objekty.

- ***Look-up tabulka***

1D nebo 2D textura používaná k indexaci navzorkovanou hodnotou.

- ***Souřadnice zadních polygonů***

2D textura obsahující souřadnice určující místo ukončení paprsku a hloubku deph bufferu v tomto místě.

- ***Textura se šumem***

2D textura o velikosti 1024×1024 obsahující osmibitový šum. Náhodná informace se používá pro malé posunutí začátku paprsku, kterým lze předejít artefaktům v obraze.

- ***Segmentační informace***

3D textura s rozměry 8× menšími než samotná data. Pro každý voxel existuje v textuře jeden bit, který udává přídavné vlastnosti pro tento bod. Textura je tedy vždy 16× menší než původní data a je u ní vypnutá interpolace.

Vstupní konstanty

Informace, které je nutné předat všem paprskům, zahrnují rozměry datové textury, rozměry okna, podmínky ukončení paprsku apod.

Výpočet délky kroku

Délka vzorkovacího kroku by měla být z každého pohledu vždy stejná – zhruba dva vzorky na každý voxel. Počet řezů např. CT skenu je většinou jiný než hodnota šířky jednoho řezu. Přesto je každý rozměr v 3D textuře indexován čísly od 0 do 1. Z toho vyplývá, že z různých pohledů bude vzdálenost mezi jednotlivými voxely v textuře různá. Aby byl zajištěn vždy stejně dlouhý skok, je třeba jej vypočítat individuálně pro každý paprsek.

Pro tento účel v shaderu existuje funkce `float StepLength(vec3 path)`. Je tedy znám směr paprsku, ale není známa vzdálenost (v souřadném systému texturovacích souřadnic 3D textury) od jednoho voxelu ke druhému v tomto směru. Zmíněná funkce počítá průsečík s rovinami kolmými

k osám. Každá rovina je vzdálena od počátku o 1/N, kde N je rozlišení textury v dané ose. Z průsečíků se vybere minimum, které indikuje první – a tedy správný – průsečík. Vzdálenost od počátku k průsečíku je přibližně vzdálenost dvou voxelů ve směru paprsku.

Obrázek 3-3 Hloubková mapa zobrazovaného objektu odpovídá délce dráhy paprsku v objemu; tuto informaci lze využít k interakci s jinými objekty ve scéně.

Zarovnání všech paprsků vzhledem k pozorovateli

Další technika, která redukuje nepříjemné artefakty způsobené tím, že paprsky začínají na stěnách obalového tělesa. Stěny na sebe navazují v ostrých úhlech a při dlouhém vzorkovacím kroku dochází k jejich zvýraznění. Funkce provádějící zarovnání není prozatím aktivní, protože se ukazuje, že její použití není zásadní. V současném stavu projektu není možné tuto funkci aktivovat v perspektivní projekci kvůli nelineárnosti paměti hloubky v tomto zobrazení.

Hlavní smyčka

Z délky dráhy paprsku a velikosti kroku je možné vypočítat počet skoků v textuře. I přesto je tento údaj použit pouze k předčasnému ukončení cyklu, jehož počet opakování je pevně nastaven na 2048. Tento postup umožňuje vytvořit jednoduchý cyklus o délce větší než 255 (maximální počet iterací cyklu v Shader Model 3). Překladač může sám rozhodnout, zda použije dva vnořené cykly (SM3) nebo pouze jeden (SM4). Pomocná proměnná udržuje během cyklu pozici uvnitř textury a pomocí této hodnoty je textura při každém vzorkování (průchodu cyklem) indexována.

3.3.2.2 Stínování

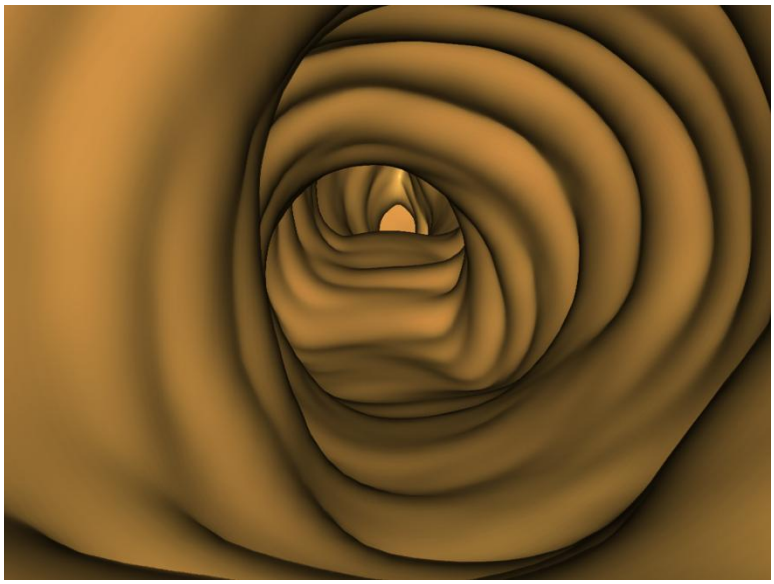
Při režimu stínování se v každém kroku navzorkuje 3D textura trilineární interpolací. Tato hodnota se použije jako index do look-up textury. Pokud je alfa (průhlednost) v tomto místě velmi malá, stínování se přeskočí a cyklus (paprsek) pokračuje dál. Pokud je v tento okamžik paprsek již téměř utlumen (tedy že nasbíral dostatečné množství barvy), cyklus se ukončí.

V opačném případě se pokračuje odhadem normálového vektoru. Podle aktivního režimu se navzorkuje potřebný počet okolních voxelů a opačné strany se odečtou. Normálový vektor se dále normalizuje a je připraven k výpočtu osvětlení.

Obrázek 3-4 Ukázka interakce volume renderingu s klasickým geometrickým polygonálním objektem a využití průhlednosti k zakomponování do existující scény.

Následuje Phongovo stínování. Malá modifikace spočívá v násobení alfa hodnoty spekulární složkou, čímž se dosáhne zesílení odlesků na „pevnějších“ površích. Vektor pohledu kamery je stejný jako vektor samotného paprsku. Vektor světla je možné libovolně měnit.

Inverzní hodnota průhlednosti se poté vynásobí s proměnnou útlumu paprsku (tato proměnná začíná na hodnotě 1 a je postupně násobením snižována až do nuly). Aktuální pozice v textuře se posune o délku jednoho kroku a cyklus začíná znovu.



Obrázek 3-5 Stínované zobrazení vnitřní stěny jícnu. Obrázek vznikl z CT hlavy a horní části hrudníku (viz např. obr. 3-3). Tento obrázek demonstruje jedno z možných praktických využití tohoto režimu.

3.3.2.3 Segmentace

S použitím další 3D textury s jednobitovými informacemi bude možné dramaticky zvýšit výkon a zvýraznit vybrané části objemu (např. lidského těla) přidáním jednoduché podmínky do těla shaderu. Data pro segmentaci by měla být generována pomoci segmentačního toolkitu.

3.3.2.4 MIP a Rentgen

Při zobrazení MIP se každý vzorek dat, který byl nalezen v objemu, porovnává s maximální doposud nalezenou hodnotou. Při dosažení konce paprsku se zobrazí nejvyšší hodnota, která byla nalezena. K tomuto účelu se využívá funkce `max`, která reprezentuje hardwarovou instrukci stejného jména. Výpočet MIP je velmi rychlý.

Prakticky stejných výsledků dosahuje rentgenové zobrazení, kde se používají pouze instrukce pro násobení a sčítání. Pouze na konci se provádí operace logaritmus, která je ovšem v HW realizována také velmi efektivně. Vynechávání výpočtu normálových vektorů dělá z těchto dvou režimů nejrychlejší dostupné zobrazení.

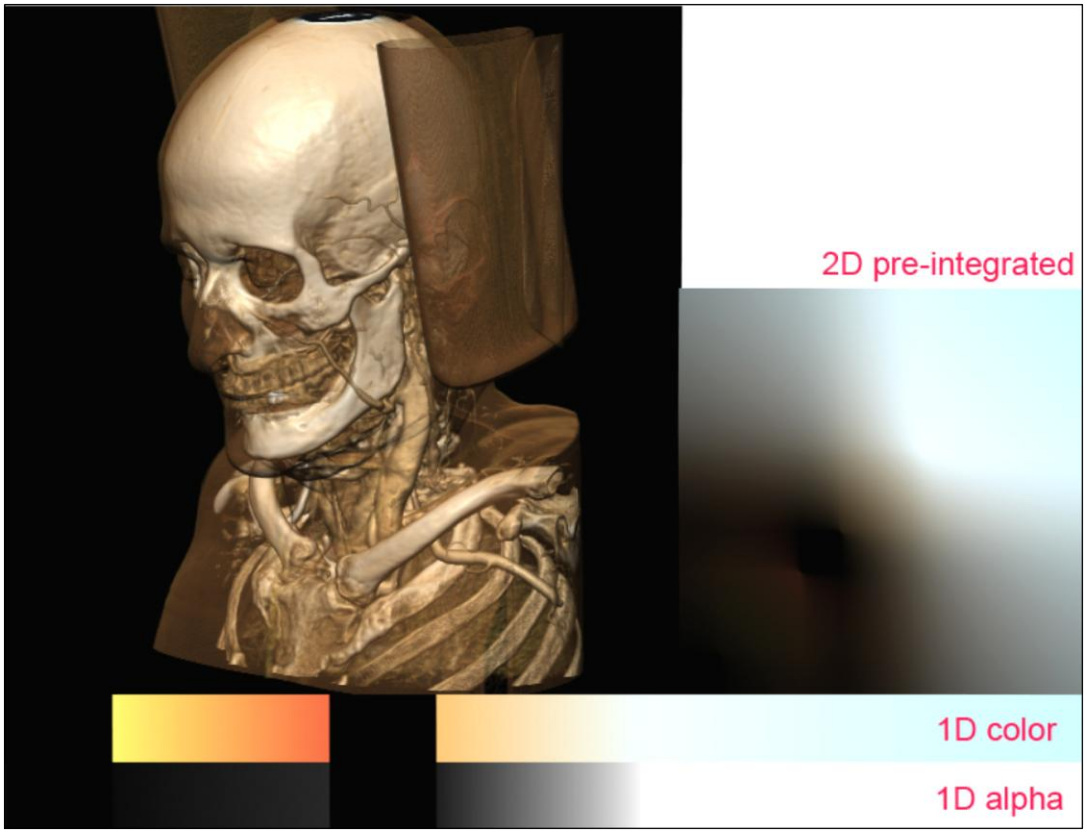
3.3.2.5 Převzorkování

Konečné přepočítání vyrenderovaného obrazu je provedeno bikubickou interpolací provedenou za pomoci samostatného shaderu. Tento způsob je podstatně kvalitnější než lineární interpolace, která je použita při běžném zvětšování textur v grafickém procesoru. Ztráta výkonu způsobená kvalitním nevzorkováním dat je zanedbatelná.

Vedle barevné složky se interpoluje i uložená průhlednost a hloubka v depth bufferu. Tak lze volume rendering zakomponovat do existující scény.

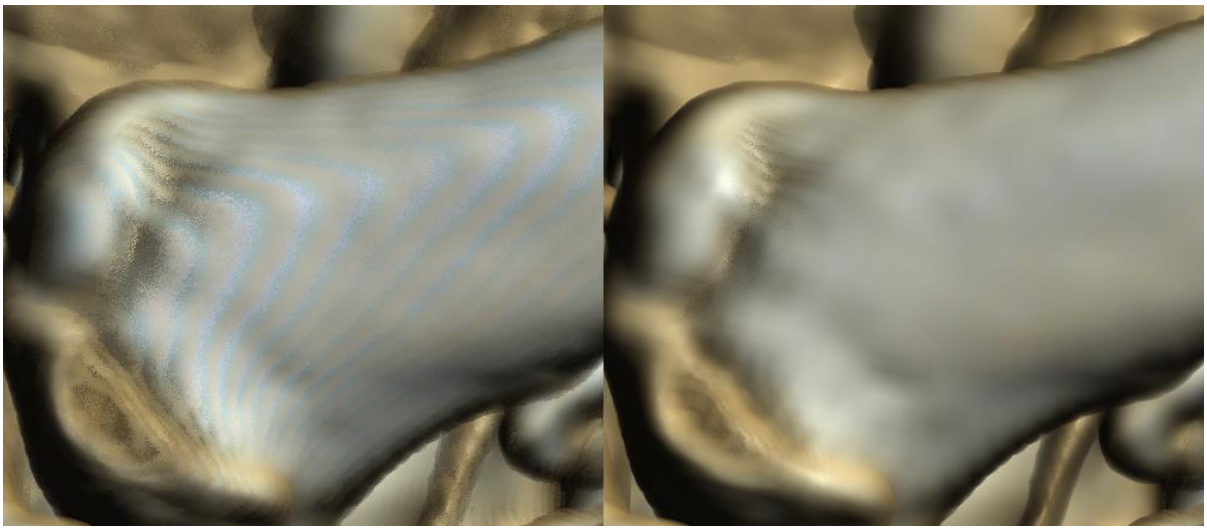
3.3.3 Textury

U 1D tabulek se každé hodnotě z 12 bitového rozsahu je přiřazena právě jedna barva. Celá tabulka je uložena v grafické paměti jako 1D textura o délce 4096. Každá položka tabulky se skládá ze čtyř 16 bitových hodnot RGBA.



Obrázek 3-6 Ukázka 1D LUT, 2D LUT a výsledného výstupu.

Použité 2D tabulky v tomto projektu mají velikost 2048×2048, RGBA, 16 bit na kanál, zapnutá lineární interpolace. Tato velikost je zvolena jako kompromis mezi maximální kvalitou výstupu a velikostí textury (32MB v uvedeném případě).

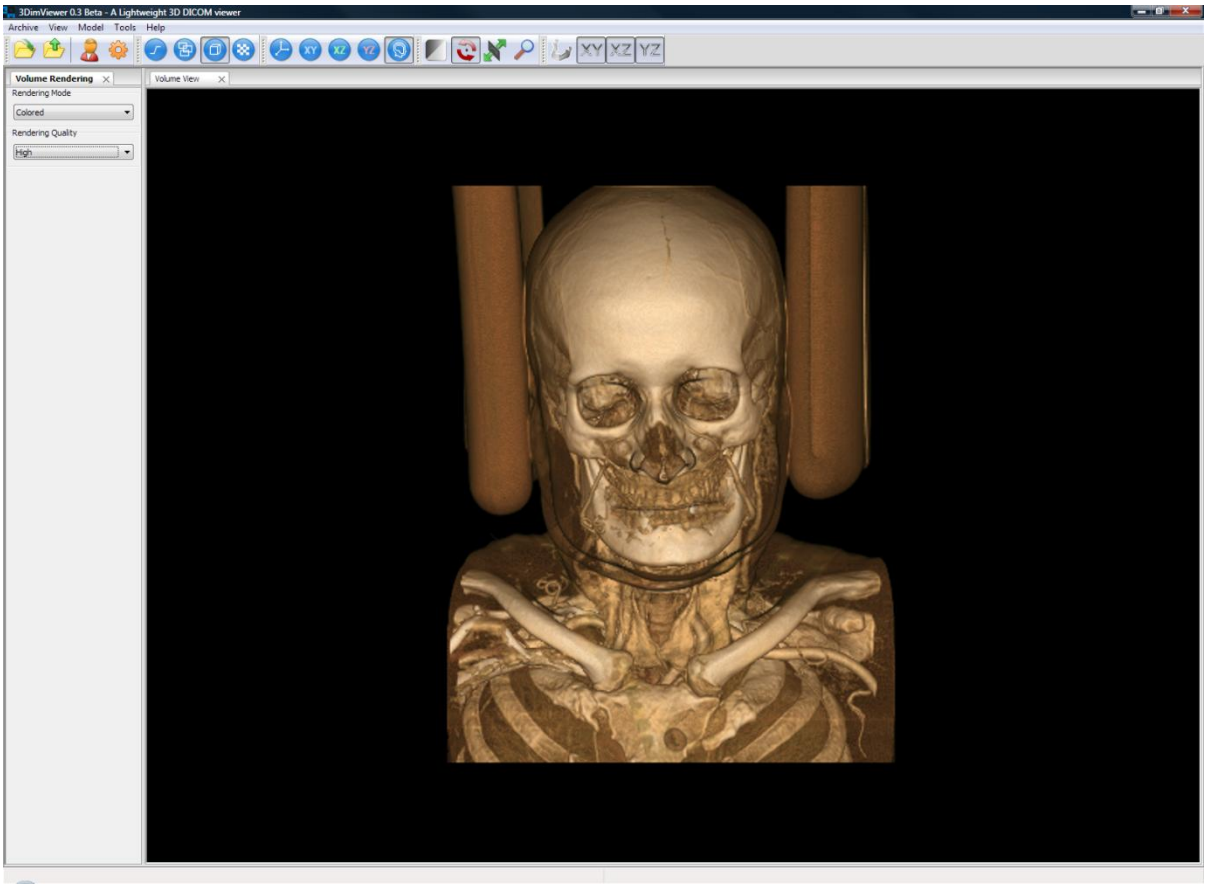


Obrázek 3-7 Srovnání 1D LUT a 2D předintegrované LUT.

Všechny řezy objemu jsou beze změny sestaveny do jediné 3D textury, která je celá uložena v grafické paměti. Nad 3D texturou je zapnuto trilineární filtrování, které zajistí hardwarovou interpolaci dat a zvýšení vizuální kvality. Dopad na výkon je minimální za předpokladu, že jsou použity dostatečně rychlé DDR paměti na grafické kartě.

3.4 Návaznost na další aplikace

Algoritmus volume renderingu z tohoto projektu byl použit pro 3D zobrazení v programu 3DimViewer firmy 3Dim Laboratory s.r.o. Tento program umožňuje načítání medicínckých 3D dat ve formátu DICOM, ořezání, převzorkování a další vlastnosti, které demonstrační aplikace tohoto projektu neumožňuje.

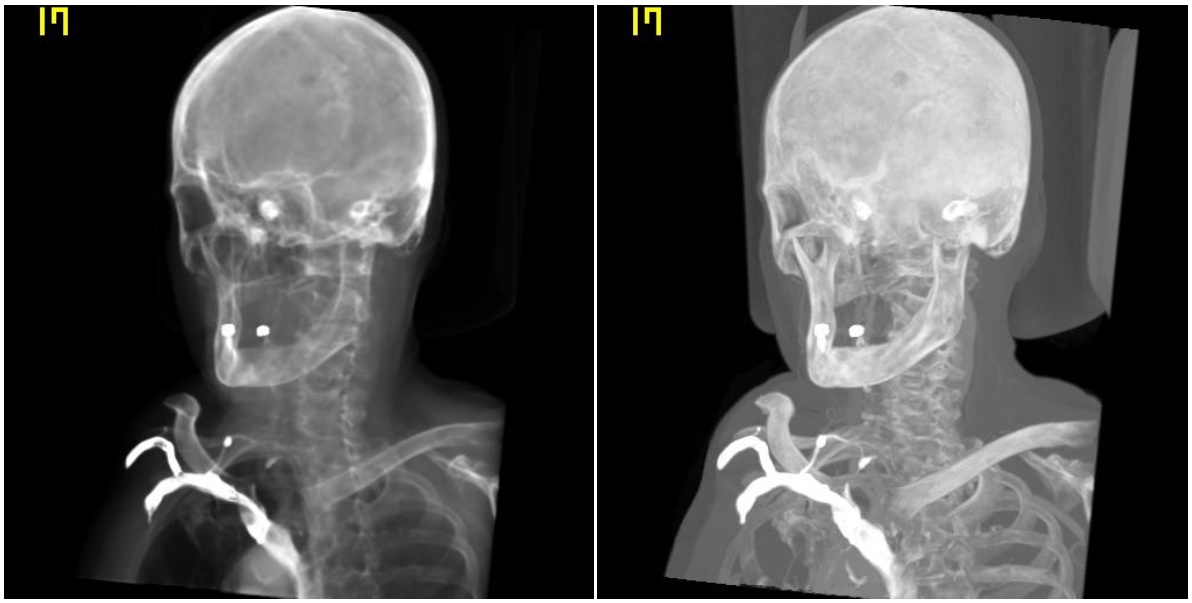


Obrázek 3-8 Volume rendering v programu 3DimViewer.

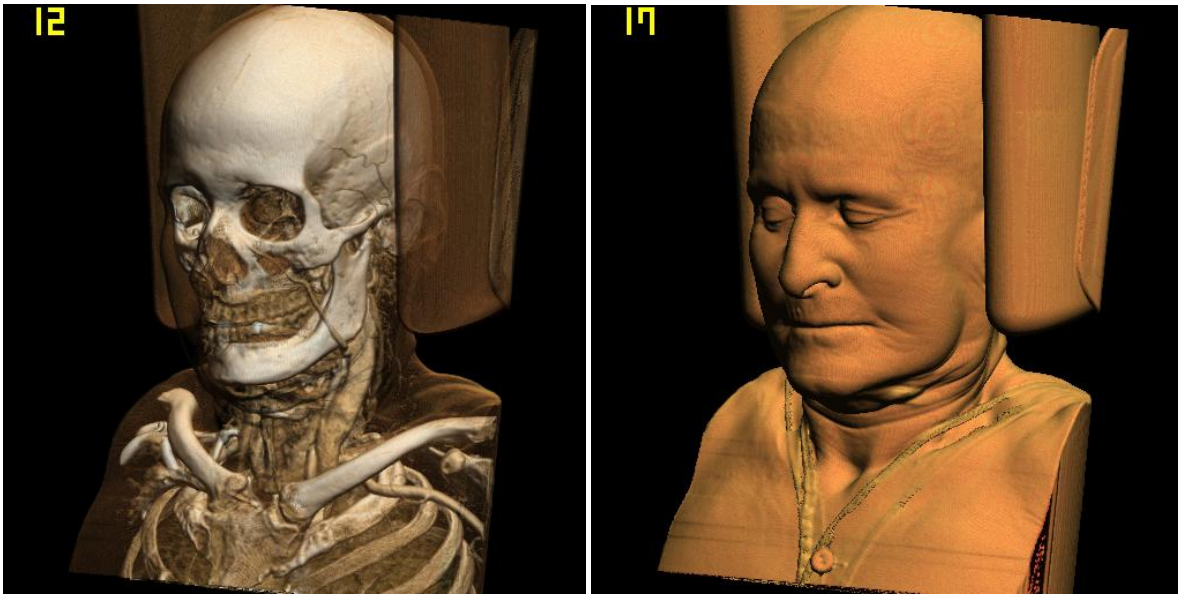
4 Výsledky

4.1 Rychlost zobrazování

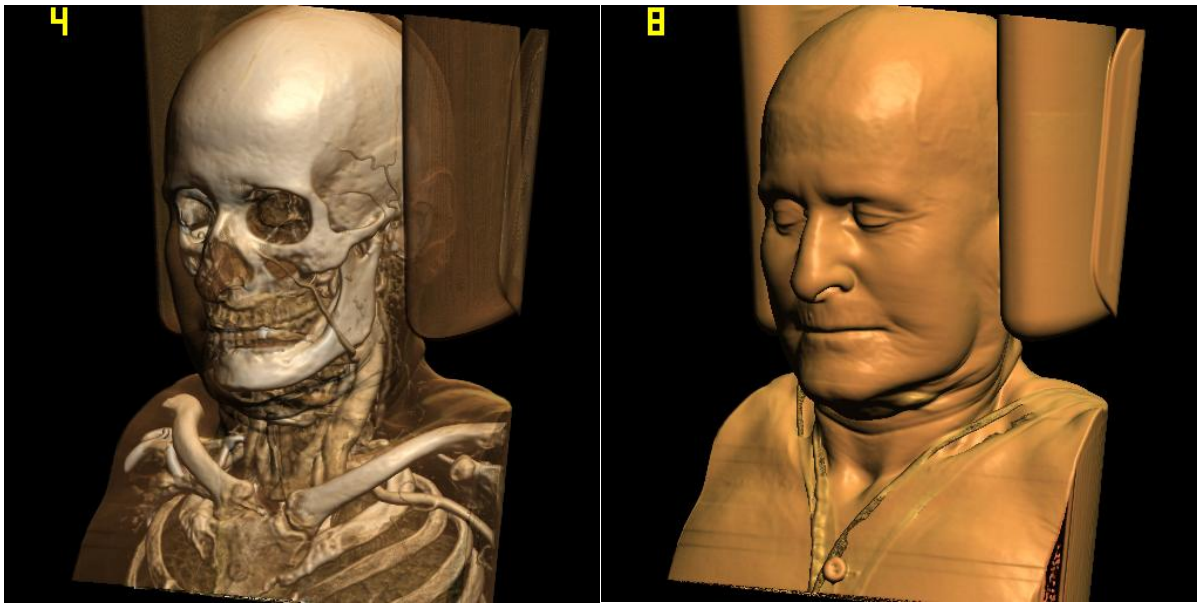
Na současných grafických kartách je počet obrázků za sekundu v rozmezí několika jednotek až desítek v závislosti na zvolené kvalitě. Následující ukázky jsou vykresleny do okna o rozměrech 512×512 obrazových bodů. Test byl proveden na kartě GeForce 8600 GTS. Počet obrázků za sekundu je v levém horním rohu.



Obrázek 4-1 Rychlost vykreslování rentgenového zobrazení a MIP.



Obrázek 4-2 Rychlost zobrazování v režimu vysoké rychlosti.



Obrázek 4-3 Rychlost zobrazování v režimu vysoké kvality.

GPU	*RTG*	*MIP*	*HQ průhledný*	*HQ povrch*	*LQ průhledný*	*LQ povrch*
GF 6600 GT	8	8	<1	~1	2	4
GF 8600 GTS	17	17	4	8	12	17
GF 8800 +	30+	30+	15+	15+	30+	30+

Tabulka 4-1 Tabulka srovnání rychlostí (fps) pro výše uvedené režimy.

4.2 Kvalita zobrazování

Následující podkapitola rozebírá různé příčiny artefaktů ray castingu a popisuje dosažené výsledky. Kvalita výstupu je velmi úzce spjatá s rychlostí, kterou je možné daný objekt zobrazit. Při zobrazování průhledných struktur je zobrazení pomalé, protože paprsky musí aktivně prozkoumat delší dráhu, ale dochází k vylepšení prostorového dojmu a vyhlazení nepřesností skrytých pod průhlednými oblastmi.

Velmi důležitá je volba správné přenosové funkce (vyhledávací tabulky). Tato tabulka určuje to, co a jak se má zobrazit. Zajímavou vlastností algoritmu je volnost světla, které lze libovolně umístit v prostoru. Pozice světla předurčuje rozložení osvětlených a tmavých oblastí, odlesky a stíny (pouze teoreticky).

Obrázek 4-4 Demonstrace kvality ray castingu.

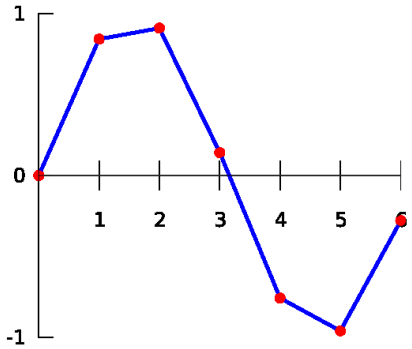
4.2.1 Možné zdroje nežádoucích artefaktů

4.2.1.1 Vzorkování

Nízká vzorkovací frekvence může výrazně zhoršit kvalitu výstupu. Optimální vzorkování je alespoň dvakrát pro každý voxel [5]. Toto je dodrženo v režimu maximální kvality. V náhledovém režimu je tato hodnota prodloužena o 50%. Chyby vzniklé vzorkováním také pomáhá vyhlazovat náhodné posunutí počátku paprsku pomocí šumové look-up textury [5]. Tato textura zajistí, že body kde dochází ke vzorkování dat, neleží v jedné rovině. Částečně je tím potlačen vznik barevných přechodů na jinak hladkém povrchu. Tyto přechody jsou způsobeny rozlišením dat a je možné je potlačit právě krátkými vzorkovacími skoky, které neleží v jedné rovně.

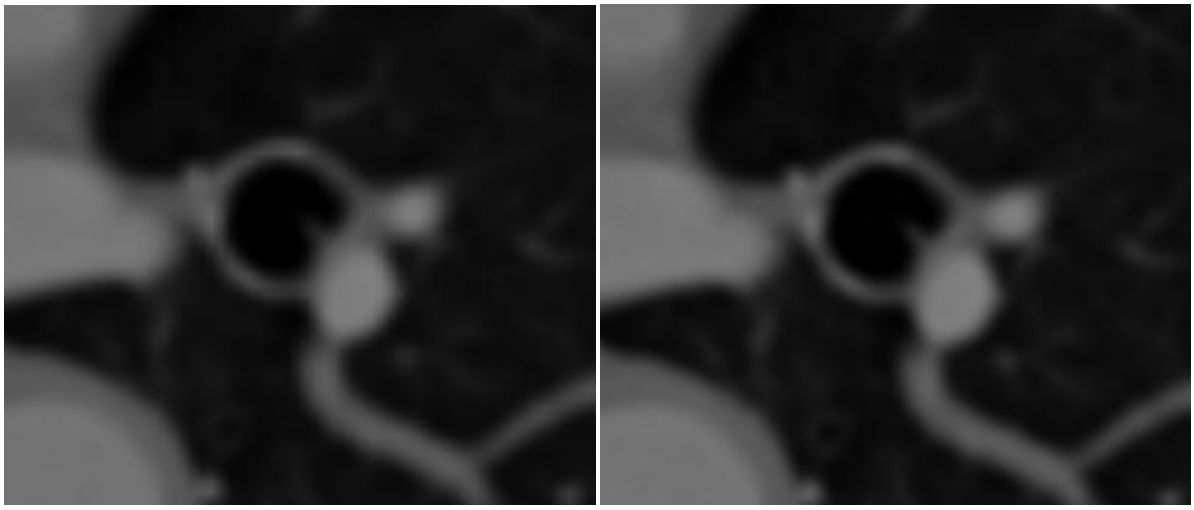
4.2.1.2 Filtrování

Grafický procesor obsahuje hardwarovou podporu pro trilineární filtrování 3D textur. Toto však není zcela ideální řešení, protože se jedná o velmi jednoduchou metodu. Implementace softwarové trikubické interpolace by přicházela v úvahu pouze na nejnovějších GPU a pravděpodobně pouze v režimu segmentace, kde by tento náročný výpočet byl omezen pouze na důležité oblasti dat.



Obrázek 4-5 Lineární interpolace.

Trilineární filtrování je sekvence lineárních filtrování v 3D prostoru. Lineární interpolace bere v úvahu pouze dva sousední body a vytváří tak ostré přechody (viz obrázek 4-5). Tento přístup vede (potenciálně) ke vzniku dalších artefaktů v obraze. Následující obrázek ilustruje nedostatky současné interpolační metody.



Obrázek 4-6 Vlevo lineární interpolace, vpravo kubická interpolace.

4.2.1.3 Klasifikace

V případě, že 1D look-up tabulka obsahuje větší množství vysokých frekvencí, dochází ke vzniku artefaktů. Tyto nežádoucí barevné skoky jsou však viditelné i u relativně plochých LUT. Řešení problému klasifikace je zavedení předintegrované vyhledávací tabulky, která pracuje s celým úsekem mezi vzorky.

Základní nevýhoda předintegrovaných tabulek je jejich velikost a čas, který je nutný k výpočtu. Přesto jsou tyto vylepšené vyhledávací tabulky zásadní pro kvalitní obrazový výstup a téměř eliminují klasifikační artefakty.

4.2.1.4 Stínování

Problém spojený s předpočítanými normálovými vektory. V tomto projektu jsou normály počítány s vysokou přesností v reálném čase a následně normalizovány. V režimu vysoké kvality jsou normálové vektory počítány Sobelovým operátorem v plné 32 bitové kvalitě, což zaručuje vysoce přesné odhady. Jejich výpočet je náročný na propustnost pamětí, proto je vhodné normály počítat pouze v předem určených oblastech (segmentace).

4.2.1.5 Integrace

Chyby integrace jsou spojeny s metodou texturování, kde je barva akumulována v 8 bitech na kanál. V tomto projektu pixel shader akumuluje barvu paprsku přímo v registrech, které mají přesnost převyšující jakékoliv nároky ray castingu.

5 Závěr

Kvalitu výstupu volume renderingu lze považovat za velmi dobrou. Program i shadery přesto zůstaly poměrně jednoduché a flexibilní. Pro reálné nasazení je důležitá kvalita, při které zůstává odezva programu (rotace, zoom, atd.) rychlá. Tento faktor velmi silně závisí na charakteru zobrazovaných dat a použité grafické kartě. Velké množství průhledných oblastí podstatně zpomaluje výpočet. Při použití téměř libovolné karty střední třídy (2009) je ovšem výkon dostatečný na kvalitní zobrazení i komplexních a průhledných dat. Jádro programu je, díky své jednoduchosti a závislosti na OpenGL 2.0, možné integrovat do aplikací, které pracují interně s medicínskými daty.

Mezi nedostatky aplikace lze zařadit neschopnost programu efektivně přeskakovat prázdný (nezajímavý) prostor v datech. Shadery vzorkují celou svoji cestu, pokud vyhledávací tabulka označí procházená data za neprázdná, dochází k pomalému výpočtu normálových vektorů a osvětlování, i když se jedná o průsvitnou a nezajímavou část dat. S tímto problémem i úzce souvisí nemožnost vytvoření „oblasti zájmu“ v datech. Tato oblast by měla být předem označena jednou ze segmentačních metod a zvýrazněna při volume renderingu.

Další rozvoj aplikace je možné vidět právě v řešení zmíněných problémů. Poslední verze shaderů (4.0, 2009) dovoluje použití pomocných trojrozměrných bitových map, které by oba problémy vyřešily. Pomocná textura by obsahovala dva bity na jeden voxel zdrojových dat (16 bitů na voxel). Tento přístup, který šetří paměť i výpočetní výkon, by umožnil přeskakovat prázdný prostor, definovat oblast zájmu a segmentovat specifické tkáně. Další vývoj knihovny je možné dosáhnout zapouzdřením OpenGL příkazů a shaderů do obecnější knihovny (například Open Scene Graph). Součástí této nové knihovny by byly další funkce usnadňující použití volume renderingu. Např. automatické generování předintegrovaných tabulek, pokročilý editor look-up tabulek, automatické nastavování parametrů, apod.

Přestože je kvalita výstupu velmi dobrá, je jí možné dále vylepšovat zapojením pokročilé metody filtrování textur, osvětlení a automatickou adaptací dat na vyhledávací tabulky.

Literatura

[1] ŽÁRA, Jiří, et al. Moderní počítačová grafika : 2. vydání. [s.l.] : [s.n.], 2005. 612 s. ISBN 80-251-0454-0.

[2] Wikipedia. [s.l.] : [s.n.], 2009. 4 s. Dostupný z WWW: <wikipedia.com>.

[3] ENGEL, Klaus, KRAUS, Martin, ERTL, Thomas. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. [s.l.] : [s.n.], 2001. Dostupný z WWW: <http://www.vis.uni-stuttgart.de/~engel/pre-integrated/>.

[4] ENGEL, Klaus, HADWIGER, Markus, REZK SALAMA, Christof. Tutorial 7: Real-Time Volume Graphics. University of Siegen, Germany, 2006.

[5] Tutorial Notes : Real-time volume graphics. Eurographics [online]. 2006 [cit. 2009-05-10]. Dostupný z WWW: <http://www.real-time-volume-graphics.org/?page_id=14>.

[6] KRŠEK, Přemysl. 3D geometrické modelování v medicíně a jeho klinické aplikace. [s.l.], 2007. 81 s. Habilitační práce.

[7] ŠPANĚL, Michal. Zobrazování a zpracování objemových dat : přednáška. Počítačová grafika [online]. 2009 [cit. 2009-05-10].

[8] KRŠEK, Přemysl. Grafická API a OpenGL : přednáška. Základy počítačové grafiky [online]. 2009 [cit. 2009-05-10].

[9] NVIDIA Developer Zone [online]. [2009] [cit. 2009-05-10]. Dostupný z WWW: <http://developer.nvidia.com/page/home.html>.

[10] BRUCKNER, SEEMANN. Shear Warp [online]. [2009] [cit. 2009-05-10]. Dostupný z WWW: <http://www.cg.tuwien.ac.at/courses/Visualisierung/2002-2003/Beispiel1/2-BrucknerSt-SeemannR/index.htm>.

Seznam příloh

Příloha 1. DVD s touto prací, zdrojovými kódy a spustitelnou aplikací.