Univerzita Komenského

Fakulta matematiky, fyziky a informatiky

Práca k dizertačnej skúške

# Objemové zobrazovanie v reálnom čase s použitím grafického hardvéru

2007 RNDr. Michal Červeňanský

Univerzita Komenského
Fakulta matematiky, fyziky a informatiky
Katedra aplikovanej informatiky

Práca k dizertačnej skúške

# Objemové zobrazovanie v reálnom čase s použitím grafického hardvéru

Autor: RNDr. Michal Červeňanský
Školiteľ: Doc. Ing. Miloš Šrámek, PhD.
Bratislava 2007

## Obsah

# 1 Úvod

Počítačová grafika má veľké uplatnenie v mnohých vedných disciplínach, rovnako ako aj v zábavnom priemysle. Rozvoj počítačovej grafiky je hnaný používateľmi, ktorí žiadajú stále reálnejšie stvárnenie okolitého sveta pomocou počítača. Ako vznikajú nové a väčšinou náročnejšie algoritmy, ktoré sa snažia zobraziť reálny svet, tak spolu s týmito nárokmi na výpočtový výkon sú vyvíjané aj rýchlejšie procesory a grafické karty, ktoré sú primárne určené na zobrazovanie grafiky. Výkon týchto grafických akcelerátorov narastá a v niektorých parametroch dokonca prekonávajú parametre hlavných procesorov osobných počítačov. Tieto grafické akcelerátory sa v poslednej dobe stávajú nástrojmi na rôzne výpočty, nielen z oblasti počítačovej grafiky.

Objemové zobrazovanie (*volume rendering*) je súčasťou počítačovej grafiky, ktorá v dnešnej dobe zaberá veľké pole pôsobnosti. Je využívané hlavne na medicínsku vizualizáciu, ktorá je hnacím motorom tejto oblasti, ale aj v iných oblastiach ako sú geológia, geografia, archeológia, rôzne simulácie prúdenia tekutín a podobne. Keďže objemové zobrazovanie spadá pod počítačovú grafiku, boli navrhnuté algoritmy využívajúce grafické akcelerátory na výpočty, ktoré potrebujú vo väčšine prípadov vysoký výkon na dosiahnutie požadovaných výsledkov.

V tejto práci je uvedený stručný prehľad niektorých techník z oblasti objemového zobrazovania s využitím grafického hardvéru a prehľad súčasného hardvéru. V kapitole 4 je uvedená časť programov, ktoré slúžia na zobrazovanie objemových dát a ich spracovanie. V záverečnej kapitole je uvedený základný návrh programu pre zobrazovanie objemových dát s využitým grafických kariet, rovnako ako aj popis vlastností a požiadaviek ktoré by mal takýto program spĺňať.

# 2 Grafické akcelerátory

V súčasnosti sú grafické akcelerátory podporujúce efektívne 2D a 3D operácie dostupné na takmer každom bežnom počítači. Ich výkon enormne vzrástol a cena v pomere k výkonu výrazne poklesla. Tento hardvér je vo väčšine prípadov využívaný na multimediálne aplikácie, počítačové hry a zábavné programy.

## 2.1 Vývoj grafických kariet

V minulosti osobný počítač neobsahoval žiadnu grafickú kartu a všetky výpočty sa vykonávali na procesore počítača. Postupe s vývojom počítačového hardvéru začali firmy vyvíjať aj grafické karty. Tieto prvé grafické karty boli vyvíjané pre výkonné pracovné stanice a neboli dostupné na bežných osobných počítačoch. Postupne s ich ďalším vývojom rástol ich výkon a klesala ich cena, pričom sa stali bežnou súčasťou osobných počítačov.

Pôvodné grafické karty nepodporovali žiadne 3D operácie a slúžili hlavne na prácu s 2D grafikou. Postupné zdokonaľovanie grafických kariet ponúkalo ich využitie aj v 3D aplikáciách. Hlavne sa používajú na zobrazovanie komplexných scén, kde tieto karty zodpovedajú za celkovú transformáciu danej scény, jej premietnutie a rôzne výpočty na dosiahnutie reálnosti scény (osvetlenie, tieňovanie, …). V súčasnosti so zdokonaľovaním týchto kariet nerastie len ich hrubý výpočtový výkon, ale aj ich funkcionalita, čo je veľkou výhodou.

Dnešné osobné počítače umožňujú využívať viacero grafických kariet v jednom počítači pre dosiahnutie vyššieho výkonu. Dnešná architektúra umožňuje využiť grafické karty aj na iné výpočty, nie nutne súvisiace s počítačovou grafikou. Tieto grafické karty sa tak stávajú univerzálnym nástrojom na rôzne výpočty a dostali aj označenie *GPU – graphics processor unit* [13]. Z tohto dôvodu počítače s viacerými kartami nemusia využívať tieto karty na rovnaké výpočty, ale každá z týchto grafických výpočtových jednotiek môže vykonávať iný výpočet [15] (napr. zobrazovanie 3D scény na jednej karte a na druhej fyzikálne výpočty).

Grafické karty podporujú priame programovanie niektorých častí zobrazovacieho procesu. Týmto častiam sa hovorí aj programovateľné jednotky (*shader-s*). Bežné karty podporujú dva typy týchto jednotiek a to *vertex* a *fragment* jednotky. Každá jednotka je zodpovedná za výpočty na iných dátach, *vertex* jednotka spracováva vrcholy a vykonáva na nich transformačné operácie. *Fragment* jednotka spracováva fragmenty (pixely) a vykonáva na nich rôzne výpočty, napríklad osvetľovanie, tieňovanie a podobne. Najnovšie grafické karty zaviedli ďalšiu programovateľnú *geometry* jednotku [35], ktorá je zodpovedná za zmenu geometrie, pričom v tejto jednotke je možné generovať nové primitívy (trojuholníky), prípadne rušiť niektoré vrcholy a meniť topológiu, čo pred tým nebolo možné. Najnovšia architektúra priniesla so sebou aj pojem *unified shader* [35], čo je označenie pre univerzálnu výpočtovú jednotku, ktorá dokáže vykonávať vertex / fragment / geometry programy, podľa potreby. Pokiaľ sú na grafickej karte z väčšej časti vykonávané operácie s vrcholmi (CAD aplikácie), tak väčšina týchto jednotiek spracováva len vrcholy a ostatné jednotky sú pridelené pre fragment a geometry výpočty. Grafické karty obsahujú viacero týchto unified jednotiek a sú tak podľa potreby prideľované tým programom, ktoré ich najviac potrebujú. Predošlé architektúry mali fixný počet vertex jednotiek a fixný počet fragment jednotiek. Ak bola aplikácia nevyvážená, (výpočet sa týkal len spracovaním vrcholov alebo fragmentov), tak menej využité jednotky neprispievali k výsledným výpočtom.

## 2.2 Grafické zobrazovanie – *graphics pipeline*

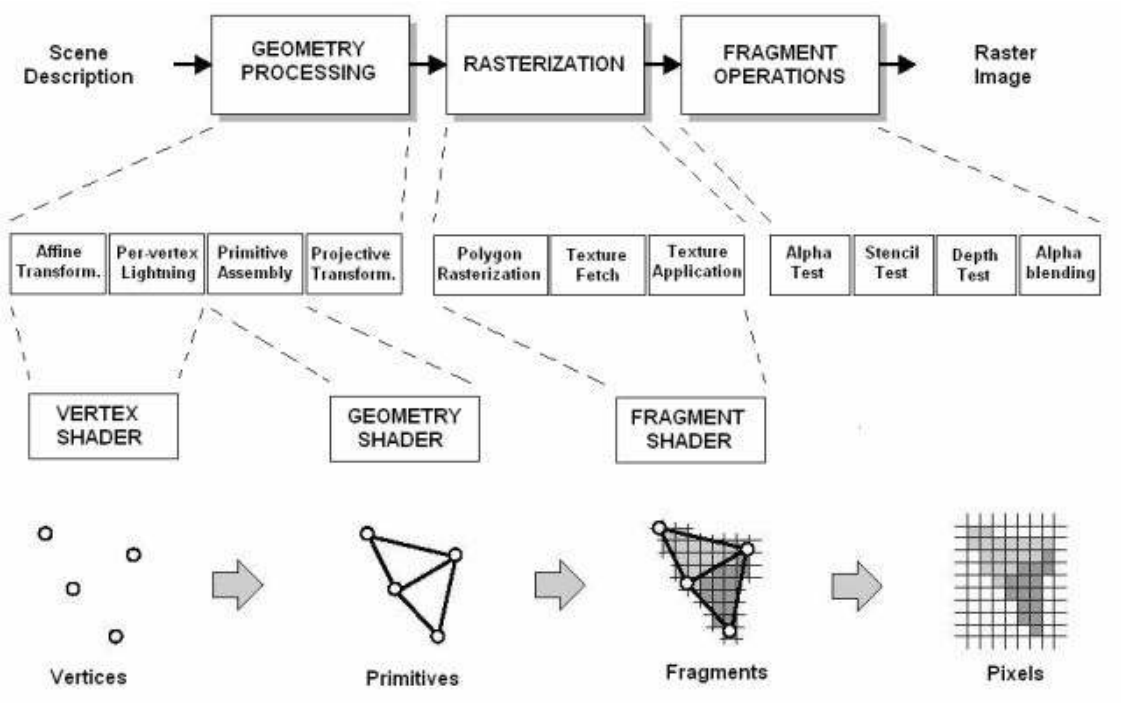
Pre grafické zariadenia musí zobrazovaná scéna pozostávať z rovinných útvarov. Proces, ktorý prevedie množinu mnohouholníkov reprezentujúcich scénu do rastrového obrazu, sa nazýva *display traversal*. Tento proces sa skladá z určitej postupnosti krokov, ktoré sú pre väčšinu grafických kariet obdobné. Poradie týchto krokov sa opisuje ako *graphics pipeline* [9] a je zobrazené na obrázku 2.1. Vstupom tohto procesu je zoznam vrcholov primitív (trojuholníkov) s dodatočnými údajmi o farbe, normále atď.. Proces dekompozície trojrozmernej scény na rovinné útvary sa nazýva *teselácia (tesellation)*. Výstupom display traversal procesu je rastrový obraz s farebnými hodnotami, ktoré sú zobrazené na obrazovke. Graphics pipeline sa delí na tri základné vrstvy.

**Spracovanie geometrie** transformuje (rotuje, posúva, škáluje atď.) vstupné údaje z modelových súradníc do svetových súradníc a potom premieta do dvojrozmernej roviny, kde sú spoločné vrcholy pospájané do geometrických primitív, útvarov (body, čiary, trojuholníky ...)

**Rasterizácia** rozkladá dané primitívy na fragmenty, ktorým priraďuje textúrové hodnoty – mapovanie textúry. Každý fragment zodpovedá jednému pixelu na monitore.

**Fragmentové (*Per-fragment*) operácie** sú aplikované následne po tom, ako boli fragmentom priradené hodnoty ako farba a priehľadnosť v rasterizačnej jednotke. Posledným krokom

pred vykreslením daného fragmentu na monitor sa uskutočňujú fragmentové testy, ktoré rozhodujú či bude daný fragment vykreslený alebo nie.




*obrázok 2.1: grafické zobrazovanie – graphics pipeline*

## 2.3 API - application programming interface

Programovacie jazyky (*C, C++, Java* atd.), ktoré chcú využívať prednosti grafických kariet, musia vedieť komunikovať s ovládačom karty. Toto umožňuje aplikačné programové rozhranie (*application programming interface - API*), ktoré efektívne využíva grafické karty od rôznych výrobcov. API odstraňuje nízkoúrovňové programovanie zobrazovacieho zariadenia, robí programovanie omnoho jednoduchším a dostupnejším a umožňuje využívanie všetkých možností tohto zariadenia. Samozrejme, že vhodne navrhnuté API uľahčuje prácu vývojárom pri vyvíjaní nového hardvéru. Najrozšírenejšie API v súčasnosti sú *Direct3D* a *OpenGL* [24,33,44,45]. Obe rozhrania (štandardy) majú rovnakú koncepciu grafického procesu ako je znázornený na obrázku 2.1.

Pre programovanie grafických jednotiek (*shader*) boli vyvinuté samostatné programovacie jazyky, ktoré kooperujú s jednotlivými API. Pre Direct3D to je HLSL (*high level shader langiuage*)[33] a pre OpenGL to je asembler podporovaný priamo v OpenGL rozšíreniach. Pre tento asembler bol vyvinutý jazyk Cg (*C for Graphics*) [8] firmou nVidia [34], ktorý je kompatibilný so syntaxou HLSL. Pre OpenGL bol navrhnutý aj jazyk glsl (*OpenGL shading languge*)[47], ktorý je priamo zahrnutý do špecifikácie 2.0.

### 2.3.1 Direct3D

Direct3D [24,33] je grafické API, ktoré je súčasťou DirectX [1], čo je technológia firmy Microsoft pre programovanie počítačových hier a zábavných aplikácií. DirectX obsahuje niekoľko súčastí, ktoré zodpovedajú nie len za programovanie grafiky, ale aj zvuku, multimédií a iných. DirectX je produktom firmy Microsoft a preto je podporované iba operačnými systémami Windows. Aktuálna verzia DirectX je 10.

### 2.3.2 OpenGL

OpenGL [44,45] je softvérové prepojenie na grafické zariadenie, ktoré obsahuje viac ako 200 odlišných príkazov (funkcií). Toto predstavuje jadro, ktoré je doplňované mnohými voliteľnými rozšíreniami [26,46]. Od uvedenia OpenGL v roku 1992 sa stalo široko používaným a podporovaným API pre 2D a 3D aplikácie. OpenGL je voľný, platformovo nezávislý grafický štandard. Špecifikáciu OpenGL má na starosti nezávislé konzorcium, OpenGL *Architecture Review Board (ARB) ,* ktoré tiež zabezpečuje spätnú kompatibilitu. Programovacie jazyky C, C++, Fortran, a ďalšie môžu využívať OpenGL. OpenGL funguje na rôznych operačných systémoch vrátane Mac OS, OS/2, UNIX, Windows 95, 98, NT, 2000, XP, Vista, Linux a BeOS. Aktuálna verzia tohto rozhrania je OpenGL 2.1.

# 3. Vizualizácia objemových dát

Objemové zobrazovanie reprezentuje metódy zamerané na spracovanie trojrozmerných skalárnych a vektorových dát za účelom ich zobrazenia v dvojrozmernej rovine [11]. Prvé požiadavky na zobrazovanie objemových dát prišli z medicínskych potrieb, avšak postupne sa oblasť používateľov objemového zobrazovania rozrástla aj do iných odvetví ako sú napríklad meteorológia, analýza molekulárnych štruktúr, fyzikálne simulácie a ďalšie, pričom objemové zobrazovanie zastrešuje veľkú časť počítačovej grafiky.

## 3.1 Objemové dáta

Objemové dáta, v porovnaní s povrchovými dátami, ktoré reprezentujú len daný povrch telesa, sú používané na popísanie celej vnútornej štruktúry objektu. Objemové dáta dovoľujú modelovať tekuté a plynné objekty rovnako, ako sa vyskytujú v prírode (napr. oblaky, hmla, oheň a voda).

Objemové dáta sa dajú získať z meraní alebo matematickým výpočtom. Pod meraním si predstavujeme získanie dát na určitom snímacom zariadení, napríklad počítačová tomografia (*computer tomography* - CT), magnetická rezonancia (*magnetic resonance* - MRI) a ďalšie. Pod matematickým výpočtom rozumieme popis nejakej simulácie pomocou rovníc, kde hodnoty v jednotlivých bodoch získavame výpočtom z popisujúcich rovníc. Ak tento model opisuje trojrozmerný objekt hovoríme o *voxelizácii.*

## 3.2 Zobrazovanie objemových dát

Proces zobrazovania objemových dát je často popisovaný ako *visualization pipeline*. Vstupom do procesu zobrazovania sú nespracované dáta, z ktorých je zvyčajne len istá časť zaujímavá. Filtrovaním sú dáta analyzované a prevzorkované (interpolované) podľa používateľom definovaných kritérií. Mapovanie tvorí podstatnú časť, kde sú

detekované štruktúry. Tento krok môže byť jednoduchý a to definovaním, ale aj komplexný a to prehľadávaním celého objemu. Cieľom je previesť abstraktnú informáciu do zobraziteľných tvarov, ktoré sú potom rozložené na základné primitívy (napr. trojuholníky) a určenie vizuálnych veličín (farba, priehľadnosť). V zobrazovacom kroku sa použijú základné primitívy na zobrazenie [9] výsledného objektu (objemu). Časti týchto krokov môžu byť urýchlené pomocou grafických akcelerátorov.

## 3.3 Fyzikálny model šírenia svetla

Pri objemovom zobrazovaní, tak isto ako v iných oblastiach počítačovej grafiky, potrebujeme definovať matematický model, ktorý bude simulovať fyzikálny jav popisujúci šírenie svetla. Od voľby tohto modelu závisia výsledky objemového zobrazovania. Keďže popis reálneho šírenia svetla nie je jednoduchý, využívajú sa rôzne aproximácie, ktoré produkujú vierohodné výsledky a nie sú príliš časovo náročné na výpočet. Medzi hlavné spôsoby interakcie medzi svetlom a okolitým médiom pozdĺž svetelného lúča môžeme zaradiť emisiu, absorpciu a rozptyl svetla. Absorpcia, emisia a rozptyl ovplyvňujú hodnotu vyžiarenej energie pozdĺž svetelného lúča. Absorpcia redukuje vyžiarenú energiu, pričom emisia naopak pridáva svetelnú energiu. Rozptyl môže pridávať a aj uberať vyžarovanú energiu pozdĺž svetelného lúča. Svetelná energia je popísaná jej vyžarovaním I, definovaným nasledovne:

$$I = dQ / ( dA_{\perp} \, d\Omega \, dt ) \qquad (1)$$

Toto vyžarovanie je nazývané aj intenzita a je definované vyžiarenou energiou Q na jednotkovú plochu A, cez jednotkový priestorový uhol Ω za jednotku času. Premenná $A_{\perp}$ označuje, že plocha A je meraná ako kolmý priemet pozdĺž pohľadového lúča na daný povrch. Riešenie tejto rovnice (1) je časovo náročné, a preto sa pre objemové zobrazovanie používajú upravené modely, kde sa isté časti tejto rovnice zjednoduchšia. Najčastejšie používaným optickým modelom pre objemové zobrazovanie je emisno-absorpčný model. Tento model v sebe zahŕňa kompromis medzi všeobecnosťou a efektívnym výpočtom. Popis šírenia svetla pomocou tohto modelu je vyjadrený rovnicou, označovanou ako rovnica objemového zobrazovania:

$$\omega . \nabla_{\mathbf{x}} I(\mathbf{x}, \omega) = -\kappa(\mathbf{x}, \omega) \, I(\mathbf{x}, \omega) + q(\mathbf{x}, \omega), \qquad (2)$$

kde $\omega.\nabla_{\mathbf{x}} I$ je skalárny súčin medzi smerom svetelného lúča $\omega$ a gradientom intenzity I v závislosti na pozícii **x.** Funkcia $\kappa$ reprezentuje absorpčný koeficient v závislosti na pozícií a smere svetelného lúča. Obdobne funkcia q reprezentuje emisný koeficient. Ak uvažujeme iba jeden svetelný lúč, tak rovnicu (2) môžeme prepísať na tvar

$$dI(s)/ds = -\kappa(s)I(s) + q(s) , \tag{3}$$

kde pozícia pozdĺž tohto lúča je označená parametrom s reprezentujúcim vzdialenosť od svetelného zdroja pozdĺž tohto lúča. Integrovaním rovnice (3) pozdĺž svetelného lúča zo začiatočného bodu $s = s_0$ ku koncovému bodu s = D dostaneme integrál pre objemové zobrazovanie:

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} q(s) e^{-\int_{s}^{D} \kappa(t)dt} \tag{4}$$

Označenie $I_0$ reprezentuje intenzitu vchádzajúcu do objemu v bode s0. Hlavným cieľom objemového zobrazovania je vypočítať rovnicu (4), ktorá popisuje šírenie svetla v objemových dátach. Priame vyčísľovanie tohto integrálu nemusí byť jednoduché, tak sa používajú rôzne numerické metódy na jeho výpočet. Postupnými metódami výpočtu, diskretizácie, integrálu môžeme dôjsť k iteratívnej forme, ktorá ho popisuje. Túto iteratívnu formu nazývame aj skladanie, kde jej tvar závisí od smeru skladania. Popis pre skladanie spredu do zadu je uvedený v jeho známej podobe:

$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst})C_{src} , \tag{5}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst})\, \alpha_{src} ,$$

kde premenné označené indexom src (*source* - zdroj) opisujú hodnoty ako vstupy z optických vlastností dát, pričom premenné s indexom dst (*destination* - cieľ) popisujú výstupné hodnoty už naakumulovaných hodnôt. Premenné označené znakom C reprezentujú farbu, typicky definovanú pomocou RGB hodnôt. Premenná označená $\alpha$ reprezentuje priehľadnosť. Podobne je možné popísať aj skladanie odzadu dopredu predpisom:

$$C_{dst} \leftarrow (1 - \alpha_{src})C_{dst} + C_{src} . \tag{6}$$

Pri tomto skladaní nie je potrebné vypočítavať hodnotu akumulovanej priehľadnosti, pretože neprispieva k farebným príspevkom. Viacej k tejto téme a jednotlivým odvodeniam môže čitateľ nájsť v [16,31,59].

## 3.4 Algoritmy zobrazovania objemových dát

Ako už bolo napísané, objemové dáta obsahujú informácie o vnútorných štruktúrach, preto sú na ich zobrazovanie potrebné špeciálne algoritmy. Tieto algoritmy môžeme rozdeliť do dvoch skupín. Prvou skupinou sú nepriame metódy, kde sa zobrazujú detekované povrchy z objemových dát. Druhou skupinou sú priame metódy, ktorých cieľom je zobrazenie všetkých významných štruktúr a aj ich rozloženie.

### 3.4.1 Nepriame metódy – povrchové zobrazovanie

Nepriame metódy vyberajú homogénne oblasti s rovnakými alebo podobnými vlastnosťami a zobrazujú povrch danej oblasti. Do tejto kategórie spadajú všetky prístupy, ktoré transformujú *voxel (volume element)* na povrchovú reprezentáciu v rámci predspracovania. Tento výsledný povrch je v závere zobrazený tradičnými zobrazovacími technikami. Vo vzťahu k zobrazovaciemu procesu zaberá najväčšiu časť výpočtu mapovanie.

Hlavné nevýhody týchto techník:

- model je oddelený od pôvodných dát a tým sa stráca podstatná časť informácií
- je potrebný veľký počet mnohouholníkov na určenie komplexného povrchu
- je náročné simulovať neostrý, beztvarý povrch (oblaky, oheň)

Na druhej strane geometrické mnohouholníky sú ľahko opísateľné s možnosťou vhodného uloženia, kompresie a prenositeľnosti. Interaktivita je dobre dosiahnuteľná použitím štandardných grafických akcelerátorov (pokiaľ trojuholníkov nie je neúmerne veľa). Medzi hlavné prístupy nepriameho zobrazovania patria algoritmy vyhľadávania kontúr [51] a implicitné povrchové pokrývanie [29]

### 3.4.2 Priame metódy – objemové zobrazovanie

Objemové zobrazovanie (*volume rendering*) zobrazuje dáta priamo bez vytvárania pomocnej povrchovej reprezentácie. Tento prístup má výhodu v tom, že môžeme vizualizovať polo priehľadné materiály a štruktúry vnorené do iných štruktúr. Tieto techniky umožňujú zobrazovať rozhranie medzi materiálmi a aj ich vnútro. Pre objemové zobrazovanie sa používajú dva hlavné prístupy. Objektovo a obrazovo orientované alebo hybridné techniky, ktoré kombinujú prvé dve. Objektovo orientované zobrazovanie využíva mapovanie dát na zobrazovaciu rovinu. V obrazovo [21] orientovaných algoritmoch je pre každý pixel zo zobrazovacej roviny vrhaný lúč cez celé dáta na určenie výslednej hodnoty daného pixela. Niektoré algoritmy na objemové zobrazovanie pozostávajú z niekoľkých krokov, kde sa ako prvé používajú objektovo orientované techniky nasledované obrazovo orientovanými technikami, ktoré určujú hodnotu výsledného pixlu alebo naopak. Tieto algoritmy spadajú do kategórie hybridného objemového zobrazovania.

#### 3.4.2.1 Objektovo orientované techniky

Objektovo orientované techniky [56] začínajú s jedným voxelom a počítajú jeho príspevky, ktoré sú premietnuté do výsledného obrazu. Tento výpočet sa iteratívne vykonáva pre všetky dátové voxely. Algoritmy pracujúce v poradí zozadu dopredu, prehľadávajú dáta v danom smere a premietajú dané hodnoty do výsledného obrazu. Ak je potrebné, tak ich prepíšu, alebo využijú na výpočet výslednej hodnoty. V opačnom poradí pracujú algoritmy, ktoré využívajú výhodu elementov premietnutých do už vykreslených regiónov, pretože ich nie je potreba uvažovať. Medzi objektovo orientované techniky spadajú aj algoritmy popísané nižšie, ktoré využívajú na zobrazovanie objemových dát textúry.

#### 3.4.2.2 Obrazovo orientované techniky

Tieto techniky posudzujú každý pixel vo výslednom obraze samostatne. Pre každý pixel je počítaná výsledná hodnota z jednotlivých príspevkov celého objemu, ktoré

zodpovedajú prechodu pohľadového lúča cez dáta. Typickým predstaviteľom tejto metódy ja algoritmus sledovania lúča ( *ray casting* ) [12,51].

# 4 Objemové zobrazovanie a grafické karty

Vyššie popísané triedenie vychádza z algoritmov, ktoré boli navrhnuté pre výpočet na hlavnom procesore počítača. Prenesenie týchto algoritmov na grafickú kartu väčšinou potrebuje určitú úpravu, aby vyhovovali ich špecifikám. Niektoré algoritmy stále nie je možné vykonávať na grafických kartách z dôvodu kompatibility, prípadne obmedzeniami grafických kariet.

Hlavným prínosom využitia grafického hardvéru pre objemové zobrazovanie, a nie len pre zobrazovanie, je jeho vysoký výkon a vhodný návrh pre počítačovú grafiku. Použitím grafických kariet sa dá dosiahnuť interaktívne objemové zobrazovanie, ktoré pri využití CPU rapídne klesá v porovnaní s GPU. Hlavným faktorom, ktorý ovplyvňuje interaktivitu zobrazovania je veľkosť dát, ktorý ovplyvňuje počet interpolácií potrebných na prevzorkovanie dát pozdĺž pohľadových lúčov. S veľkosťou dát súvisí aj ich presnosť, pričom v medicíne sa často používajú dáta o hĺbke 12 bitov. Ďalším faktorom, ktorý ovplyvňuje interaktivitu je aj prenosová zbernica, ak sa dáta nezmestia do video pamäti na grafickej karte. Algoritmy popísané nižšie sú založené na podpore grafických akcelerátorov, preto pre zobrazované dáta vyplývajú obmedzenia, ktoré požadujú jednotlivé zariadenia. Jedným z obmedzení je veľkosť textúr. V súčasnosti podporujú grafické karty rozmer 3D textúry 512x512x512 voxelov. Pri 2D textúrach sú dnes maximálne rozmery 4096x4096.

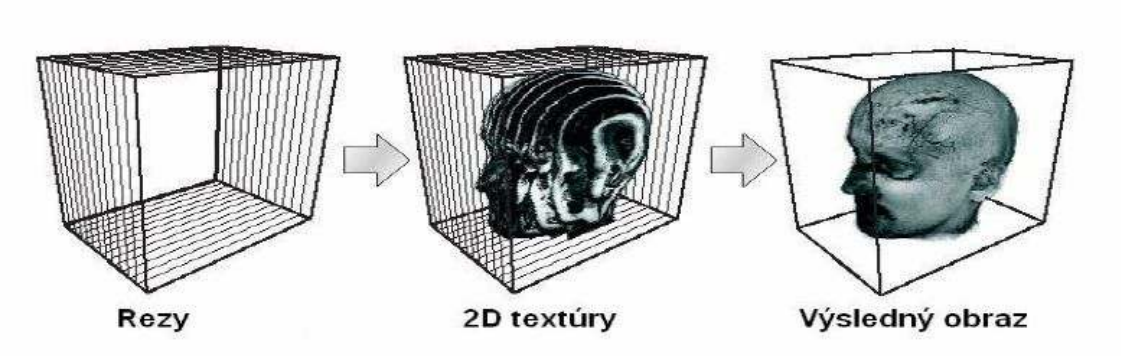
## 4.1 Objektovo orientované techniky

Medzi tieto techniky spadajú algoritmy, ktoré skladajú textúry reprezentujúce dáta do výsledného obrazu. Keďže grafická karta pracuje iba s rovinnými primitívami (trojuholníkmi) je potrebné rozložiť objem na sadu rezov, ktoré je už grafická karta schopná zobraziť a poskladať do výsledného zobrazenia. Táto sada rezov je reprezentovaná sadou textúr. Grafické karty, ktoré podporujú 3D textúry môžu využiť tieto 3D textúry, ktoré reprezentujú objemové dáta.

### 4.1.1 2D textúry

V súčasnosti každé grafické zariadenie podporuje mapovanie 2D textúr. Časť grafických kariet, ktorá je zodpovedná za mapovanie textúr, povoľuje bilineárnu interpoláciu, ktorá je vo veľkej miere zodpovedná za výpočtovú zložitosť objemového zobrazovania. Preto je dobré využiť túto možnosť grafických kariet na urýchlenie zobrazovania.

Objem je rozložený do sady *objektovo orientovaných rezov*, štvoruholníkov, v závislosti na aktuálnom smere pohľadu [3]. Grafické karty dovoľujú priamo zobrazovať jednotlivé rezy, tak ako je načrtnuté na obrázku 4.1. Orientácia rezov je závislá na minimálnom uhle medzi vektorom pohľadu a normálou rezu, čo vedie k nutnosti mať v pamäti tri sady dát (textúr), ktoré sú kolmé na súradnicové osi. Vybraná sada rezov je zobrazená transformáciou každého mnohouholníka a aktuálnou transformačnou maticou. Zobrazovanie sady rezov je vykonávané v poradí od zadných rezov ku predným. Počas rasterizácie je na každý rez mapovaná prislúchajúca textúra. Bilineárna interpolácia počas mapovania textúry je urýchľovaná grafickým zariadením.

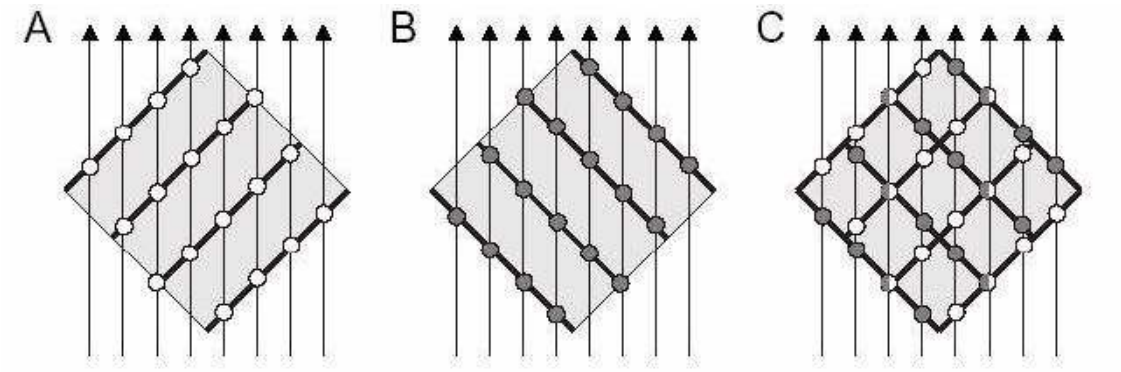



*obrázok 4.1: objekt rozložený na sadu objektovo orientovaných rezov*

Jednotlivé zobrazené rezy musia byť nejakou vhodnou formou pospájané do výsledného obrazu tak, aby vyhovovali rovniciam popísaným v kapitole 3.3. Pre skladanie rezov odzadu dopredu slúži rovnica 6, ktorá ma priamu podporu v grafických kartách ako alfa zmiešavanie (*alpha blending*).

Hlavná výhoda tohto prístupu je presunutie potrebného výpočtu interpolácií na

grafickú kartu, ktorá je na to vhodne navrhnutá, čím sa dá dosiahnuť vysoký výkon. Veľkou nevýhodou je skutočnosť, že dáta sú uložené v pamäti trikrát reprezentujúc sadu rezov. Pri zväčšovaní obrazu dochádza k typickým aliasingovým artefaktom, ktoré sú hlavne viditeľné na hranách rezov a sú spôsobené nedostatočným vzorkovaním. Počet rezov je stanovený rozmerom dát v danom smere. Pri väčšom počte rezov sa na daný rez mapuje najbližšia textúra z danej sady. Ďalší rušivý efekt je možné pozorovať pri zmene medzi rôznymi sadami rezov (obrázok 4.2). Tento problém je odstrániteľný pri použití 3D textúry. Celkovo sú v tabuľke 4.1 zhrnuté výhody a nevýhody tohto algoritmu.




*Obrázok 4.2: vzorkovacie artefakty zapríčinené zmenou sady rezov*

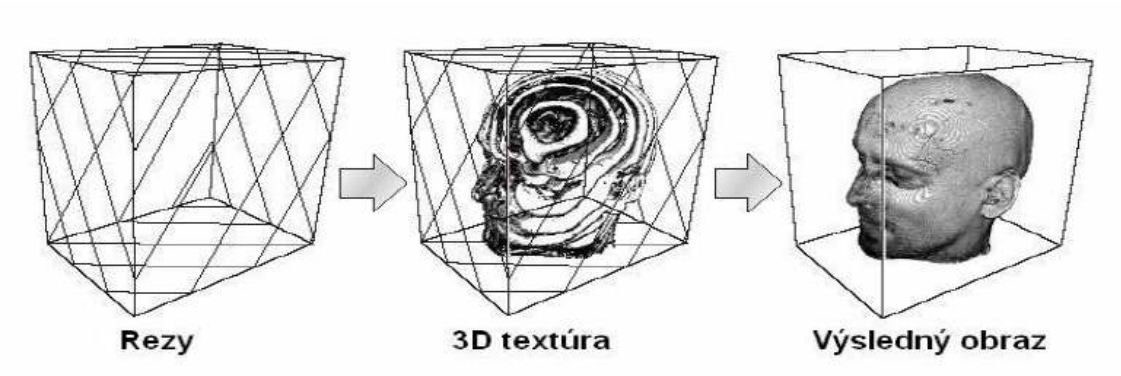
| 2D textúry | |
|---|---|
| Plusy | Mínusy |
| vysoký výkon<br>veľká dostupnosť | vysoké pamäťové nároky<br>bilineárna intrpolácia<br>vzorkovacie artefakty<br>prepínací efekt<br>nekonzistenté vzorkovanie |

*tabuľka 4.1: výhody a nevýhody zobrazovania pomocou 2D textúr*

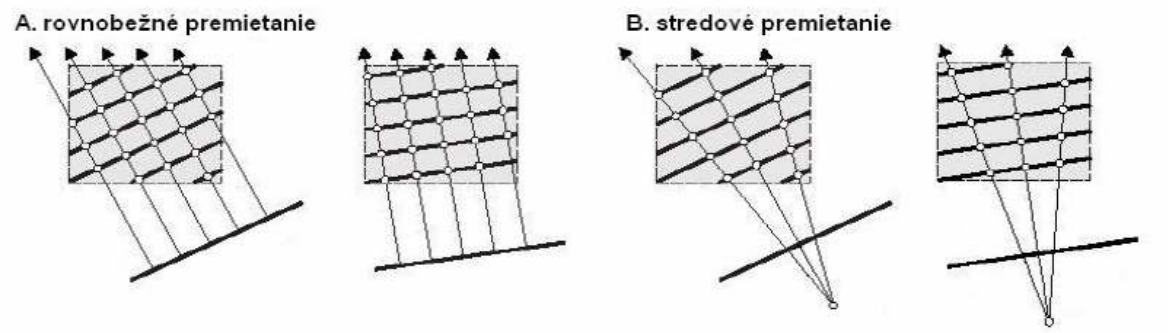
### 4.1.2 3D textúry

Metóda využívajúca 3D textúry [3,60] odstraňuje niektoré nedostatky predchádzajúcej metódy, ktoré boli zapríčinené využívaním len bilineárnej interpolácie.

Tieto nedostatky pre obe metódy sú popísané nižšie. Pri 3D textúrach sa využíva trilineárna interpolácia, čo vedie k novému prístupu objemového zobrazovania. Využívanie 3D textúr nevedie k odstráneniu potreby rozložiť objekt na rovinné mnohouholníky. Aj keď je celý objem dostupný na grafickej karte ako 3D textúra, grafické zobrazovanie nepodporuje priame zobrazovanie objemových primitív (kocka a iné), ale len rovinných. Použitie 3D textúry dovoľuje umiestniť rezové mnohouholníky ľubovoľne, tak aby bola zachovaná konštantná vzdialenosť rezov pre rôzne uhly pohľadu. Môžeme využiť rezy rovnobežné s premietacou rovinou, *pohľadovo orientované rezy* (obrázok 4.3). Avšak súradnice mnohouholníkov musia byť prepočítavané vždy keď dôjde k zmene smeru pohľadu, pričom tento výpočet je zložitejší ako pri objektovo orientovaných rezoch. Tento výpočet je možné taktiež vykonávať na grafickej karte vo vertex jednotke [7]. V prípade rovnobežného premietania vedie tento spôsob k rovnakému vzorkovaniu pre všetky lúče (obrázok 4.4a). Pre stredové premietanie nie je vzorkovanie rovnaké pre všetky lúče (obrázok 4.4b). Vzdialenosť medzi vzorkami rastie s veľkosťou uhla medzi vektorom pohľadu a normálou rezu. Pre skladanie jednotlivých rezov v tomto prípade platia rovnaké podmienky ako pre zobrazovanie pomocou 2D textúr popísané vyššie.




*obrázok 4.3: rozloženie objemu na pohľadovo orientované rezy*




*obrázok 4.4: vzorkovanie pozdĺž pohľadových lúčov pre rovnobežné (a) a stredové (b) premietanie*

Prístup využívajúci 3D textúry zvyšuje kvalitu obrazu. Hardvérová podpora pre trilineárnu interpoláciu nám umožňuje zvýšenie vzorkovania na získanie kvalitnejších výsledkov a odpadá nutnosť mať v pamäti tri sady dát. Obrazovo orientované rezy zaručujú rovnaké vzdialenosti medzi susednými vzorkami pre rovnobežné premietanie. Problém meniaceho sa vzorkovania stále nie je odstránený pre stredové premietanie. Táto nekorektnosť je však z vonkajších pohľadov málo viditeľná, pri pohľadoch z vnútra objemu sú viditeľné rušivé artefakty. Tento problém môže byť odstrániteľný použitím guľových zobrazovacích primitív namiesto rovinných [25], alebo pomocou metódy vrhania lúča. Výhody a nevýhody tohto prístupu sú zhrnuté v tabuľke 4.2. Aplikovateľnosť tohto prístupu závisí od hardvérovej podpory 3D textúry.

| 3D textúry | |
|---|---|
| Plusy | Mínusy |
| vysoký výkon<br>trilineárna interpolácia | rozmerové obmedzenia<br>nekonzistenté vzorkovanie pre stredové premietanie |

*tabuľka 4.2: výhody a nevýhody zobrazovania pomocou 3D textúr*

### 4.1.3 2D multitextúry

Multitextúring je technika využívaná na grafických kartách, kde sa na výslednú farbu pixela podieľajú viaceré textúry. Táto možnosť je využitá aj pri zobrazovaní objemových dát, kde rozširuje základný algoritmus využívajúci 2D textúry a prináša

vyššiu kvalitu do zobrazovania. V tejto metóde sa využíva fakt, že trilineárna interpolácia môže byť rozložená na 2 bilineárne a jednu lineárnu interpoláciu. Bilineárne interpolácie sú vykonávané na grafickej karte v procese filtrovania textúr. Výsledná lineárna interpolácia medzi dvoma textúrami je vykonaná vo fragment programe. Takto je možné získať rez, ktorý sa nachádza medzi dvoma origínálnymi rezmi reprezentovanými textúrami dostatočné rýchlo, počas zobrazovacieho procesu. Týmto spôsobom je možné dosiahnuť korektné vzorkovanie dát rozložených na sadu objektovo orientovaných rezov, pre kolmé premietanie. Pre skladanie rezov do výsledného obrazu platia rovnaké podmienky ako u techník spomenutých vyššie. Kedže je táto technika založená na zobrazovaní pomocou 2D textúr, tak obsahuje aj niektoré nevýhody s ňou spojené. Jedná sa hlavne o nutnosť mať 3 sady textúr v pamäti a z toho vyplývajúci prepínací efekt. Výhody a nevýhody tohoto algoritmu sú uvedené v tabuľke 4.3.

| 2D multitextúry | |
|---|---|
| Plusy | Mínusy |
| vysoký výkon<br>trilineárna interpolácia | vysoké pamäťové nároky<br>prepínací efekt<br>nekonzistentné vzorkovanie pre stredové premietanie |

*tabuľka 4.3: výhody a nevýhody zobrazovania pomocou 2Dmultitextúr*

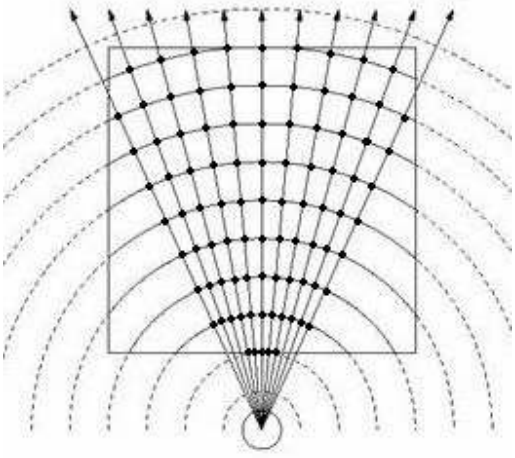
## 4.2 Obrazovo orientované techniky

Medzi tieto techniky patrí hlavný predstaviteľ a to algoritmus vrhania lúča, ktorý je tiež možné vykonávať na grafickej karte s podporou 3D textúr a s využitím pokročilejších verzií fragment jednotiek.

### 4.2.1 Algoritmus vrhania lúča ( *ray casting* )

Metóda vrhania lúča spadá do algoritmov typu obrazovo orientovaných, ako bolo uvedené vyššie, kde pre každý pixel výsledného obrazu je vrhaný lúč, ten hľadá prienik s

dátami, potom postupne prechádza dátami a akumuluje hodnoty z dát pozdĺž tohto lúča. Výsledná akumulovaná hodnota je zobrazená vo výslednom obraze. Algoritmus vrhania lúča odstraňuje problém nekonzistentného vzorkovania pre stredové premietanie (obrázok 4.5) a je to korektný algoritmus pre obe premietania. Algoritmus využíva 3D textúry a podporu trilineárnej interpolácie pre získanie jednotlivých vzoriek pozdĺž pohľadového lúča. Je ho možné priamo implementovať ako jednoprechodový algoritmus na grafických kartách, ktoré podporujú dynamické cykly (karty podporujúce *shader model 3.0c*) [49]. Implementovať tento algoritmus bolo možné aj na starších kartách ako kartách, ktoré podporujú SM 3.0c, ale iba ako viacprechodový algoritmus [23]. Pod viacprechodovým algoritmom rozumieme algoritmus, ktorý vykonáva viacero rôznych programov, ktoré môžu, ale aj nemusia závisieť na výsledkoch predchádzajúcich programov.



*obrázok 4.5: vzorkovanie pozdĺž pohľadových lúčov pre stredové premietanie*

Táto metóda využíva 3D textúru v ktorej sú uložené zobrazované dáta. Odstraňuje nevýhody vyplývajúce zo stredového premietania. Keďže táto metóda využíva 3D textúry, tak je obmedzená veľkosťou textúry. Tento algoritmus je vhodný na rôzne optimalizačné techniky, ako je preskakovanie prázdnych miest [23], adaptívne vzorkovanie, generovanie izoplôch, prípadne sa dá doplniť o lámanie a odraz lúča [49]. Možnosťou pre toto použitie sú aj rôzne osvetľovacie modely a prenosové funkcie, ktoré budú popísané nižšie. Zhodnotenie tejto metódy je možné vidieť v tabuľke 4.4.

| Metóda vrhania lúča | |
|---|---|
| Plusy | Mínusy |
| trilineárna interpolácia<br>korektný pre obe premietania<br>modulárny | rozmerové obmedzenie<br>menší výkon ako 2D/3D textúry<br>podpora shader modulu 3.0 |

*tabuľka 4.4: výhody a nevýhody metódy vrhania lúča*

## 4.3 Klasifikácia

Pri skladaní výslednej hodnoty pixela pozdĺž pohľadového lúča vo výslednom obraze je vhodné špecifikovať koeficienty vyžarovania a pohlcovania energie. Dáta získané z merania alebo simulácie vo väčšine prípadov tieto hodnoty neobsahujú a neexistuje prirodzená cesta, ktorá by ich definovala. V praxi sú tieto koeficienty pre dané vzorky dát priraďované používateľom. Tento proces priraďovania je označovaný ako klasifikácia a môže byť popísaný pomocou prenosových funkcií. Prenosové funkcie priraďujú v závislosti na vstupných hodnotách (dáta) farebné hodnoty a hodnoty priehľadnosti. Postupy ako automaticky generovať prenosové funkcie z dát alebo výsledného obrazu sú vo väčšine prípadov závislé na vstupných dátach [7]. Vo všeobecnosti navrhovanie prenosových funkcií je manuálny a zdĺhavý proces, ktorý vyžaduje dôkladné znalosti a štruktúru, ktorú dáta reprezentujú. Iné výsledky sa obdržia pri použití tých istých prenosových funkciách, ale rôznych dátach rovnakého objektu (CT, MRI). Pre vhodné definovanie daných funkcií v procese klasifikácie je veľmi dôležitá viditeľná odozva používateľových operácií. Preto je žiaduce pri modifikovaní prenosových funkcií aj súčasná interaktívna zmena počas zobrazovania objemových dát, ktoré umožňujú grafické karty. Stručne si uvedieme techniky prenosových funkcií, ktoré sú založené len na jednom parametri, jednorozmerné prenosové funkcie, a to hustote (intenzite) dát. Existujú techniky, ktoré pre základ prenosových funkcií používajú viaceré parametre, napríklad veľkosť gradientu, hlavné krivosti, druhé derivácie a iné, tieto funkcie sú označované ako viacrozmerné prenosové funkcie, alebo dátovo zamerané

(*data centric*) [17, 21]. Medzi viacrozmerné prenosové funkcie sa radí aj použitie LH Histogramu [50]. Ďalšími metódami na generovanie prenosových funkcií sú algoritmy označované ako obrazovo zamerané (*image centric*) [30] .

Prenosové funkcie sú definované ako spojité funkcie, avšak v praxi sa realizujú ako indexové tabuľky pevnej dĺžky. Koeficient vyžarovania je zvyčajne reprezentovaný ako RGB zložka, ktorá dovoľuje vyžarovanie farebného svetla, čím získavame farebné dáta. Koeficient pohlcovania je reprezentovaný ako skalárna veličina v rozmedzí 0 až 1, ktorý definuje priehľadnosť danej vzorky. Tieto koeficienty môžeme skombinovať do RGBA hodnoty.

Objemové dáta sú reprezentované ako trojrozmerné pole bodov (voxelov). Podľa teórie vzorkovania môže byť spojitý signál rekonštruovaný z týchto bodov *konvolúciou* použitím príslušného filtra (interpoláciou daného stupňa). Prenosová funkcia môže byť preto aplikovaná priamo na diskrétne body pred rekonštrukciu alebo až na spojitý signál po rekonštrukcii [40]. Obe metódy vedú k viditeľne rozdielnym výsledkom.

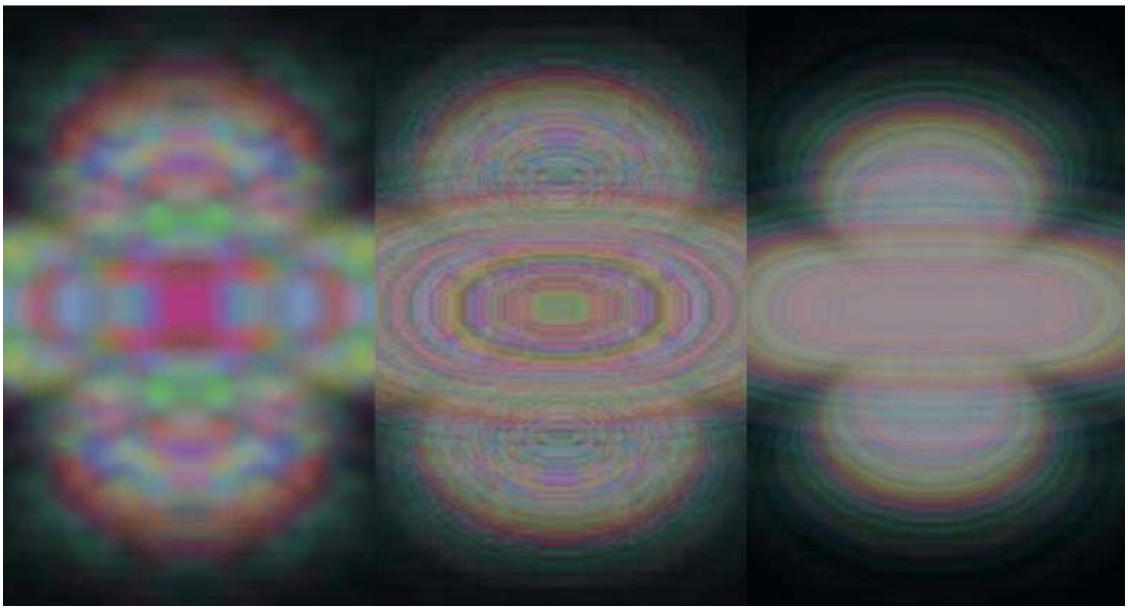
**Pred-klasifikácia** označuje aplikovanie prenosovej funkcie na diskrétne vzorky bodov pred rekonštrukciou signálu. Rekonštrukcia spojitého signálu je vykonávaná na hodnotách s už priradeným vyžarovaním a pohlcovaním, pričom sa nezachovávajú vysoké frekvencie z prenosových funkcií

**Po-klasifikácia** obracia poradie vykonávaných operácií. Aplikovanie prenosovej funkcie sa dostáva až za proces rekonštrukcie. Prenosová funkcia je tak aplikovaná na spojitý signál namiesto aplikovania na diskrétne body. Pre zachovanie vysokých frekvencií v prenosových funkciách, je treba zobrazovať veľa rezov (Nyquistovo kritérium)

Vyššie frekvencie v prenosových funkciách zvyšujú potrebu väčšieho počtu vzorkovania. Pred-klasifikácia neprenesie vysoké frekvencie z prenosovej funkcie do výsledného renderovania, po klasifikácia zachováva vysoké frekvencie, ale iba na danom reze. Na reprodukciu vysokých frekvencií, medzi rezmi, treba zobrazovať viacej rezov, čo vedie k zvýšenému času potrebnému na zobrazovanie.

**Predintegrovaná klasifikácia**, je metóda klasifikácie, ktorá čiastočne odstraňuje problém vysokých frekvencií v prenosových funkciách [7]. Táto metóda simuluje renderovanie bloku po bloku a nie rez po reze, kde pod blokom rozumieme objem medzi dvoma rezmi. Pri použití tejto techniky sa docielia kvalitatívne podobné výsledky ako pri po klasifikácii, ale s menším počtom rezov (blokov), čo vedie k rýchlejšiemu zobrazovaniu.

Všetky spomenuté metódy klasifikácie je možné implementovať na grafickej karte, pričom sa dosahujú interaktívne výsledky zobrazovania. Pre jednotlivé metódy klasifikácie existujú rôzne riešenia, či už pomocou OpenGL rozšírení, alebo priameho programovania fragment jednotiek. Pri aplikovaní prenosových funkcií sa vo fragment jednotke (OpenGL rozšírenie) dátam priraďujú hodnoty farebnosti a priehľadnosti na predvypočítanej palete. Táto paleta je generovaná z predpisu prenosových funkcií, kde pre každý kanál (RGBA) exituje samostatná funkcia. Pre interaktívne zobrazovanie a manipulovanie s prenosovými funkciami je nutné, aby toto generovanie prebiehalo tiež interaktívne. Paleta pre prvé dve metódy klasifikácie (pred- a po-klasifikácia) je totožná. Jej generovanie je priamočiare a zodpovedá zapísaním funkčných hodnôt prenosových funkcií do jednorozmernej textúry pre každú vstupnú hodnotu intenzity z dát. Generovanie palety pre pred integrovanú klasifikáciu je časovo náročnejšie. Táto paleta je reprezentovaná dvojrozmernou textúrou, ktorá je symetrická podľa diagonály. V palete sú predvypočítané numerické integrovania pozdĺž pohľadových lúčov pre všetky možné kombinácie vstupných hodnôt z dát (intenzity). Generovanie tejto palety je možné vykonávať priamo na grafickej karte [42]. Na obrázku 4.6 sú zobrazené objemové dáta s rôznym aplikovaním tej iste prenosovej funkcie



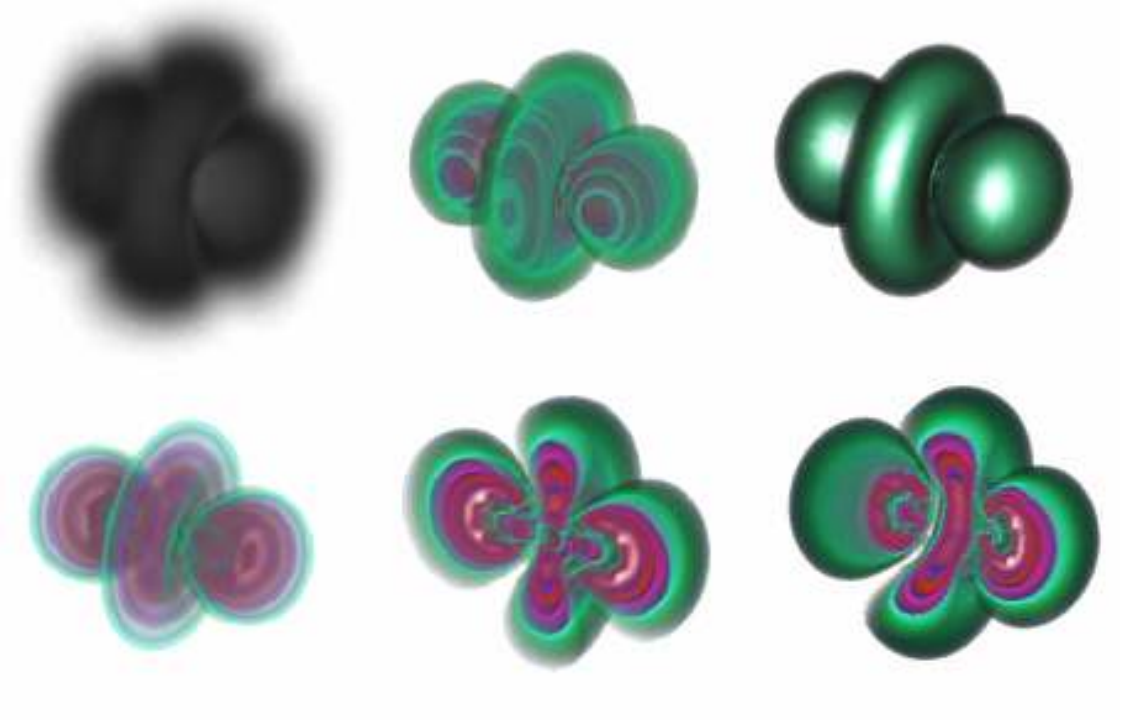
*obrázok 4.6: rôzne výsledky metód klasifikácie (zľava doprava: pred klasifikácia, po klasifikácia, pred integrovaná klasifikácia) pri aplikovaní rovnakej prenosovej funkcie*

## 4.4 Spracovanie objemových dát

Vyššie spomenuté techniky slúžia na priame zobrazovanie objemových dát. Avšak pri objemovom zobrazovaní sa nejedná len o samotné zobrazovanie týchto dát ale sú potrebné aj iné výpočty s objemovými dátami. Niektoré dodatočné výpočty sa dajú vykonávať na grafických kartách. Medzi hlavné výpočty môžeme zaradiť lokálne / globálne osvetľovanie, orezávanie [57,58], filtrovanie a iné. Veľká časť týchto výpočtov slúži na zvyšovanie kvality výsledného zobrazovania. Ďalšími algoritmami sú algoritmy zvyšujúce rýchlosť zobrazovania, pričom využívajú možnosti grafického hardvéru. Reálne medicínske dáta zaberajú veľkú časť pamäte a často sa stáva, že veľkosť týchto dát prevyšuje kapacitu video pamäti. Aj takéto algoritmy spracovávajúce veľké dáta je možné vykonávať na grafických kartách [40]. K týmto algoritmom môžeme zaradiť aj algoritmy, ktoré spracovávajú časovo závislé dáta a generujú rôzne animácie (rôzne biologické procesy, simulovanie počasia). S veľkým počtom dát sa zaoberajú aj algoritmy zobrazujúce dáta získané z rôznych modalít rovnakého objektu. Vhodným napasovaním týchto dát sa venujú techniky registrácie, ktoré taktiež spadajú do oblasti

objemového zobrazovania. Medzi špeciálne techniky spadá zobrazovanie segmentovaných dát v rámci pôvodných dát a kombinovanie rôznych techník na ich zvýraznenie, pripadne zvýraznenie istých čŕt v dátach [2,27,57]. Tieto techniky môžeme zaradiť pod oblasť nazývanú nefotorealistické zobrazovanie [4,14].

V tomto krátkom popise sa vyskytli rôzne techniky ktoré súvisia či už priamo, alebo nepriamo s vizualizáciou objemových dát. Tieto techniky sú neustále vyvíjané aj s príchodom nových grafických kariet a ich novou funkcionalitou. Objemové dáta zobrazené s použitím niektorých vyššie popísaných techník je možné vidieť na obrázku 4.7.



*Obrázok 4.7: Rôzne vizualizačné techniky (zľava, zhora): mapovanie rovnobežných rezov s priemetňou, aplikácia prenosovej funkcie (pred integrovaná klasifikácia), lokálne osvetlenie, gradientová modulácia, orezanie rovinou, orezanie zložitejšou geometriou.*

# 5 Prehľad dostupných riešení

V tejto kapitole sa budeme venovať softvérom pre zobrazovanie objemových dát. Ich štúdiom môžeme zistiť určité vlastnosti a zákonitosti, ktoré môžeme aplikovať do nášho softvéru. Tieto programy môžeme rozdeliť do dvoch hlavných tried, v podstate ako každý iný softvér a to na komerčné a nekomerčné aplikácie.

## 5.1 Komerčné aplikácie

Komerčné aplikácie slúžia hlavne pre zobrazovanie medicínskych dát a z tohto dôvodu sú kladené vysoké nároky na presnosť a kvalitu výsledných riešení. Ku komerčným aplikáciám vo všeobecnosti nie sú dostupné zdrojové kódy a ani dodatočné informácie o dizajne. Väčšinu týchto komerčných aplikácií nie je ani možné nainštalovať a používať, či už v časovo obmedzenej forme, alebo s obmedzenými vlastnosťami. K týmto aplikáciám sú dostupné predajné informácie internetové stránky a rôzne prezentácie. Cena týchto produktov je určite vysoká a tiež nie je dostupná, pretože väčšinou závisí od počtu licencií a rôznych iných dohôd. Komerčné aplikácie pre ich vysokú náročnosť čo sa týka spoľahlivosti sú vyvíjané početnými tímami ľudí a sú používané v nemocniciach. Tento softvér je väčšinou veľmi členitý čo sa týka jeho využitia. Používatelia môžu používať len niektoré vysoko špecializované časti na zobrazovanie konkrétnych dát, alebo všetky balíky, ktoré nemusia súvisieť výlučne s vizualizáciou, ale pokrývajú celkovú starostlivosť o pacienta. Tieto komerčné aplikácie často využívajú špeciálny hardvér VolumePro [38], ktorý je navrhnutý na zobrazovanie objemových dát. Hlavnými dodávateľmi takýchto aplikácií sú samotní výrobcovia zariadení na získavanie objemových dát. Medzi niektoré komerčné aplikácie môžeme zaradiť produkty týchto firiem: Siemens (Syngo) [48], Philips [39], FujiFilm [10], MeVis Research [32], TatraMed [54] a mnoho ďalších.

## 5.2 Nekomerčné aplikácie

Väčšina nekomerčných aplikácií vzniká ako súčasť grantových a univerzitných činností na overovanie výskumu a teoretických výpočtov. Tieto aplikácie sú tvorené ako jednoduché aplikácie s jednotným používateľským rozhraním a slúžia iba na úzky okruh zobrazovacích techník. Takéto aplikácie sú najrozšírenejšie, vo väčšine prípadov sú k nim dostupné aj zdrojové kódy a ich vývoj je väčšinou ukončený. Medzi takéto aplikácie môžeme zaradiť OpenQvis [41], VolView[20], Teem [19], Simian [22], f3dvr [5,53].

Väčšie aplikácie, do ktorého vývoja je zahrnutých aj viacero ľudí bývajú dobre dokumentované, s dostupnými zdrojovými kódmi a sú aj v istých časových intervaloch dopĺňané a pracuje sa na ich vývoji. Keďže tieto systémy sú už rozsiahlejšieho charakteru, tak neslúžia len na objemové zobrazovanie, ale dajú sa s nimi vykonávať aj rôzne iné výpočty. V týchto rozsiahlejších aplikáciách môže dôjsť k problému, že neskúsený používateľ môže mať problém s ich nainštalovaním, prípadne správnym nakonfigurovaným. Obdobný problém môže nastať pre vývojárov, ktorý ich chcú dopĺňať a rozširovať. Medzi takéto aplikácie môžeme zaradiť SciRun[43], Drishti [28].

Špeciálnu časť aplikácií zobrazujúcich objemové dáta tvoria aplikácie, ktoré využívajú na toto zobrazovanie už hotové knižnice na popis 3D scén, napr. OpenScenGraph [36], OpenSG [37]. Nad týmito knižnicami sú navrhnuté nové používateľské rozhrania, prípadne nové zobrazovacie algoritmy.

**Reprezentácia dát a výpočtov** – zaujímavým rozdelením môže byť aj reprezentácia dát a vykonávanie procesov. Pri jednoduchých aplikáciách sú väčšinou dáta jednoducho reprezentované a spracovávané jednoduchou a priamou cestou. Pri zložitejších aplikáciách môžeme rozdeliť návrh týchto aplikácii na dve skupiny. Prvá skupina je založená na grafe scény, kde aplikácia pracuje s vytvorenou scénou. Túto scénu tvorí koreň, na ktorý sú pripájané dáta tvoriace túto scénu. Dáta nemusia byť priamo pripojené na koreň scény, ale môžu byť pripojené aj na iné dáta a tým sa určuje platnosť týchto dát. Takto pripravená scéna je ďalej spracovávaná a zobrazovaná. Hlavnými predstaviteľmi sú OpenSceneGraph a OpenSG. Druhým prístupom je “tok dát” kde nie je zostavená

komplexná scéna, ale na jednotlivé dáta sú aplikované čiastkové výpočty a výstupy z týchto čiastkových výpočtov sú ďalej spracovávané, pričom znovu tento proces tvorí istý graf. V tomto grafe sú postupne spájané dáta s výpočtovými jednotkami, ktoré môžu generovať nové dáta. V takomto systéme sú presne špecifikované možnosti konexií dát s výpočtovými jednotkami a naopak. Hlavným predstaviteľom takejto aplikácie je program SCIRun.

## 5.3 Zhrnutie

S príchodom nového grafického hardvéru vznikajú nové algoritmy, ktoré by mali byť implementované do už existujúcich riešení. Preto je žiaduce aby tieto riešenia boli vhodne navrhnuté a umožnili jednoduchú implementáciu nových techník. Nejedná sa len o nové možnosti grafických kariet, ale aj o hlavné procesory osobných počítačov, pretože tie sa tiež neustále vyvíjajú a prinášajú nové možnosti. Preto je vhodný návrh takéhoto softvéru veľmi podstatný, aby v budúcnosti umožnil jednoduchú správu softvéru, prípadne oddialil doň veľké zásahy.

# 6 Projekt dizertačnej práce

Pre skúmanie a pozorovanie objemových dát je nevyhnutné mať vhodný nastroj – program, ktorý umožňuje zobrazenie týchto dát a následne ich štúdium. Je veľmi ťažké a časovo náročné skúmať jednotlivé rezy, z ktorých sa dané objemové dáta skladajú, pretože týchto rezov môže byť neúmerne veľa – dnešné tomografy produkujú až 2000 rezov v jednom vyšetrení. Pri pozorovaní objemových dát ako celku a ich vnútorných štruktúr môžu byť niektoré hľadané časti výraznejšie a lepšie viditeľné ako keby ich mal používateľ hľadať v jednotlivých rezoch a potom z nich skladať nejaký 3D útvar. Preto je veľmi potrebné takýto nástroj mať pre potreby objemovej vizualizácie. Je aj veľmi vhodné aby bolo zobrazovanie dát interaktívne, pričom interaktivitu pre objemové zobrazovanie môžeme považovať už od 10 snímkov za sekundu. Toto interaktívne zobrazovanie umožňuje tiež skrátiť čas na skúmanie potrebných vlastností a štruktúr dát.

Ako bolo spomenuté v predchádzajúcich kapitolách, tak existujú rôzne programy, ktoré umožňujú objemové zobrazovanie a využívajú výkon grafických kariet na urýchlenie tohto zobrazovania.

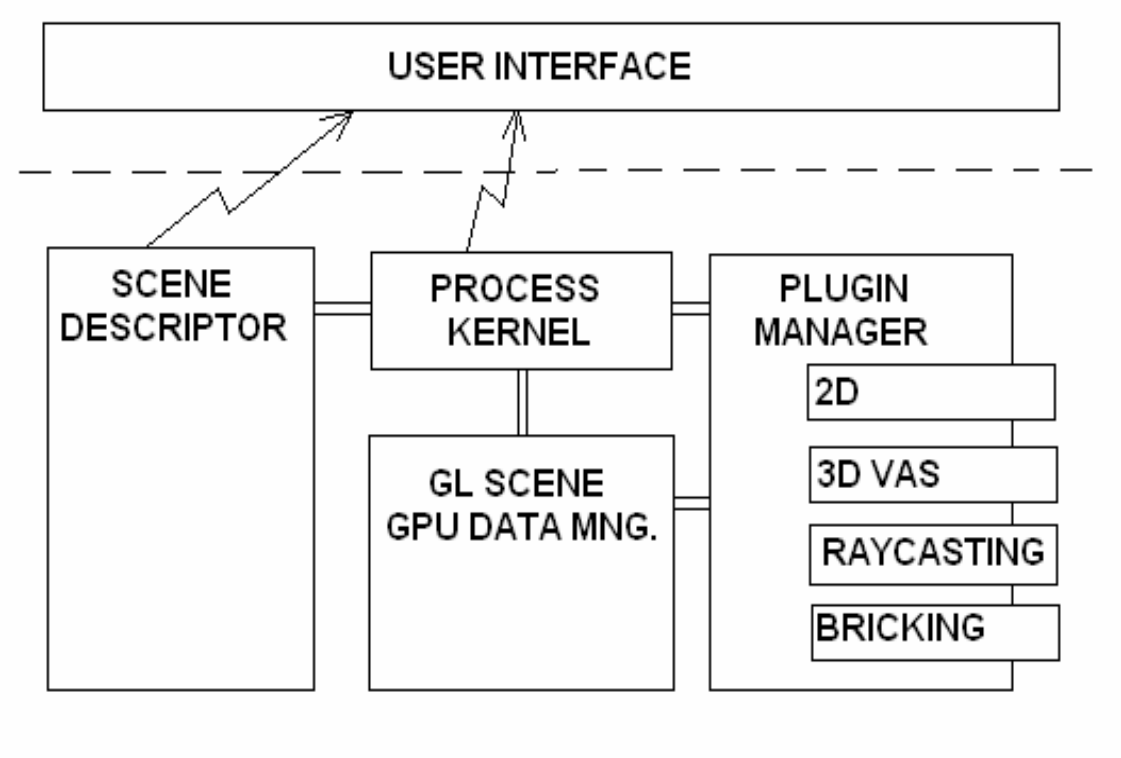
## 6.1 Cieľ práce

Cieľom dizertačnej práce je vytvoriť takýto program na zobrazovanie objemových dát. Mal by umožňovať zobrazovanie objemových dát rôznymi spôsobmi potrebnými pre používateľov a hlavne jednoducho implementovať nové algoritmy, ktoré vznikajú pri práci s objemovými dátami a ich overenie v praxi.

Tento program by mal byť jednoducho rozšíriteľný o nové zobrazovacie módy a algoritmy, prípadne ich kombinácie. Jeho samozrejmosťou by mala byť ľahká prenositeľnosť medzi rôznymi používateľmi a teda aj funkčnosť pod rôznymi operačnými systémami. Mal by taktiež vo veľkej miere využívať grafické karty na zobrazovanie dát ale aj na ich spracovanie, prípadne dodatočné výpočty, aby bola dosiahnutá interaktivita, ak to súčasné riešenia umožňujú.

## 6.2 Základný návrh

Základný návrh softvéru pre zobrazovanie objemových dát môžeme rozdeliť na 5 časti. Prvá časť zodpovedá za vytvorenie zobrazovanej scény a tvorí ju používateľ s ohľadom na jeho požiadavky. Ďalšou častou je manažér modulov, ktorý je zodpovedný za jednotlivé moduly. Moduly tvoria samostatnú časť, kde pre každý zobrazovací mód (algoritmus), prípadne významný výpočet existuje samostatný modul. Významnou časťou je aj GPU dáta manažér, ktorý zodpovedá za vytvorenie GPU scény z dátovej scény, kde GPU scéna udržuje dáta transformované na grafickú kartu pre potreby modulov. Poslednou časťou je jadro tohto softvéru, ktoré je zodpovedné za koordináciu výpočtov a slúži aj na komunikáciu s používateľským rozhraním. Na obrázku 5.1 je zobrazený náčrt základného delenia programu.




*Obrázok 5.1: základný návrh programu*

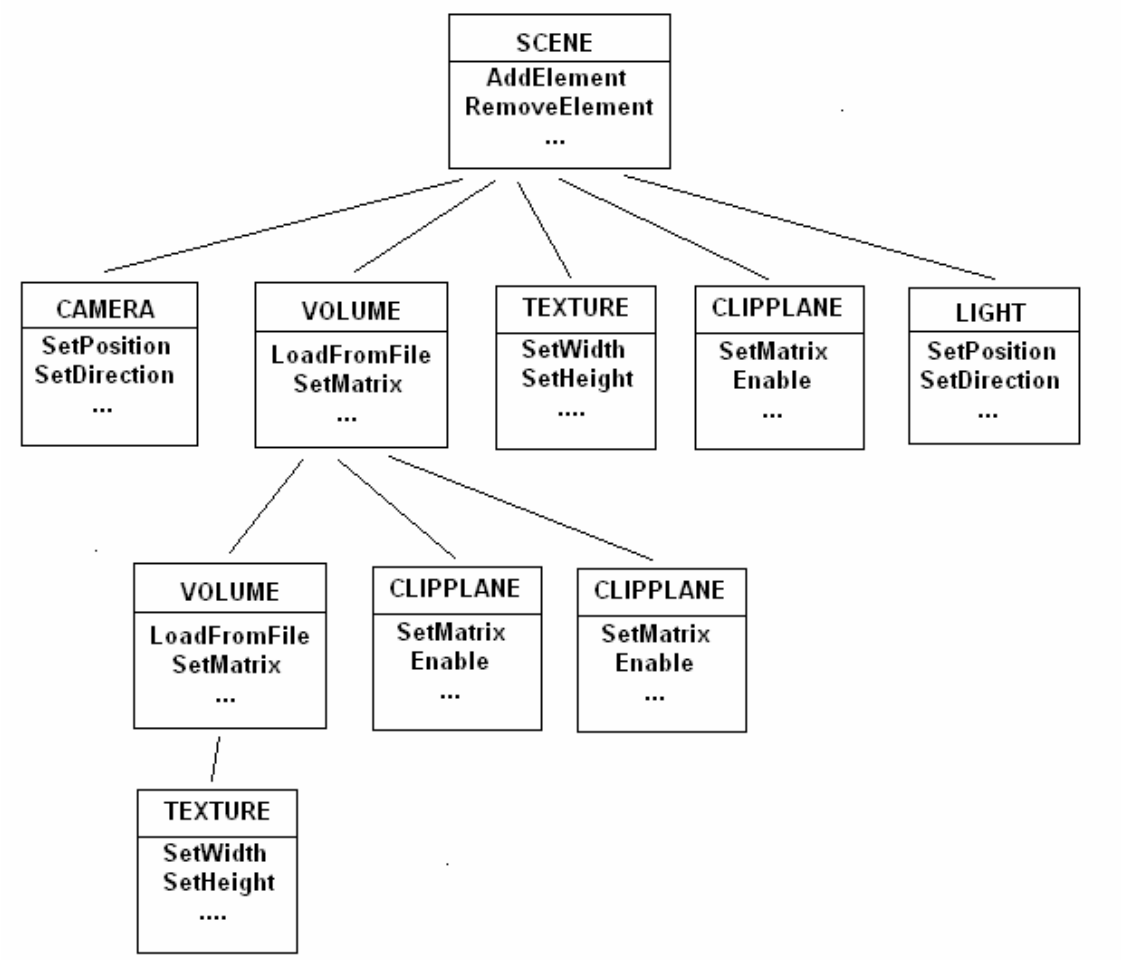
### 6.2.1. Reprezentácia dát

Pre objemové zobrazovanie v najjednoduchšej miere stačia samotné dáta a ich rozmery. Avšak pre pokročilejšie techniky je potreba k týmto dátam pridať aj dáta priamo či nepriamo súvisiacimi s týmito objemovými dátami. Naša reprezentácia týchto dát bude vychádzať z grafu scény, kde v koreni tohto grafu je scéna a na ňu sú pripojené dáta , ktoré tvoria danú scénu. Týmto dátam budeme hovoriť aj scénové elementy. Medzi hlavné scénové elementy, ktoré musí obsahovať každá scéna sú kamera s jej nastaveniami (smer pohľadu, typ premietania a iné) a objem, ktorý reprezentuje samotné objemové dát a poskytuje aj informáciu o rozmeroch dát, presnosti a ďalších vlastnostiach objemových dát. Medzi dodatočné scénové elementy môžeme zaradiť svetlá (bodové, plošné, ...), textúry, ktoré reprezentujú prídavné dáta (napr. prenosové funkcie), orezávacie roviny, prípadne iná reprezentácia jednoduchej geometrie, ktorá je využitá podľa potreby. Aplikovateľnosť niektorých scénových elementov závisí od ich pripojenia na rodiča. Napríklad už spomínané orezávacie roviny môžu byť pripojené priamo na scénu (koreň), v tomto prípade majú globálny charakter a sú aplikované na všetky objemy v scéne. Alebo môžu byť pripojené iba na konkrétny objem a v tom prípade sú aplikované iba na tento objem. Za vytvorenie scény je zodpovedný používateľ, ktorý cez príslušné možnosti vytvorí scénu s objemovými dátami a táto scéna je následne spracovávaná a zobrazovaná. Niektoré scénové elementy sa môžu vyskytovať v scéne viackrát na rovnakej úrovni. Na obrázku 5.2 je načrtnutá jednoduchá scéna.

### 6.2.2 Manažér modulov

Plugin manažér je časť programu, ktorá je zodpovedná za manažovanie modulov. Každý nový renderovací algoritmus je konštruovaný ako samostatný modul, ktorému zodpovedá príslušná knižnica. Plugin manažér je zodpovedný za nahratie a údržbu dostupných modulov. Je zodpovedný za zistenie ich funkčnosti (použiteľnosti) na danej hardvérovej platforme. Vykonáva hlavný funkčný kanál medzi scénou (používateľom) a

modulom. Zodpovedá aj za poskytnutie informácii o module a jeho potrebných nastaveniach. Drží si zoznam aktuálnych modulov a k nim potrebných informácií.




*Obrázok 5.2: náčrt jednoduchej scény*

### 6.2.3 GPU dáta manažér

GPU dáta manažér je zodpovedný za prípravu dát pre grafickú kartu a ich údržbu. Medzi dáta, ktoré sa spracovávajú, patria dáta popisujúce scénu, ktoré majú právo využívať všetky moduly na svoje výpočty. Tieto dáta môžu byt reprezentované pomocou textúr, zoznamom zobraziteľnej geometrie (*displaylistov*), zoskupeniami vrcholov (*vertex arrays*), časťami videopamäti (*fragment buffer object - FBO*), a inými dostupnými a potrebnými reprezentáciami, ktoré poskytujú grafické karty. GPU dáta manažér spracuje vstupnú dátovú scénu a vytvorí potrebnú reprezentáciu podľa potreby

aktuálneho modulu. Keď používateľ zmení renderovací modul, ktorý potrebuje inú reprezentáciu scény alebo jej časti, tak GPU dáta manažér zodpovedá za korektné vytvorenie potrebnej reprezentácie. Pôvodná reprezentácia sa bude uchovávať pre prípad opätovného použitia predošlého modulu alebo iného modulu, ktorý potrebuje danú reprezentáciu. Ak pri potrebe nového modulu na novú reprezentáciu nastane nedostatok pamäte, tak bude niektorá predošlá reprezentácia, prípadne jej časť, vymazaná na základe zvoleného kritéria (frekvencia použitia, veľkosť pamäte, prípadne iné faktory).

Jednoduché príklady iných reprezentácií pre rozdielne moduly sú reprezentácie samotných objemových dát ako 3D textúry pre algoritmus vrhania lúča a ako sady 2D textúr pre algoritmus mapujúci tieto 2D textúry na projekčnú rovinu (*shear warp* algoritmus).

Program na zobrazovanie objemových dát je primárne určený na využívanie grafických kariet, avšak to nebráni tomu aby bol implementovaný aj modul, ktorý bude vykonávať všetky výpočty na procesore počítača. V takom prípade modul nezískava údaje z GPU scény, ale z pôvodnej dátovej scény.

### 6.2.4 Jadro

Jadro je významnou časťou programu pre zobrazovanie objemových dát. Jeho hlavnou úlohou je rozdeľovanie príkazov od používateľa jednotlivým častiam. Je tiež zodpovedné za synchronizáciu dátových častí. Vykonáva volania a komunikuje s aktívnymi modulmi. Táto komunikácia je presne špecifikovaná a totožná pre všetky moduly. Tak isto je presne špecifikovaná komunikácia medzi jadrom a používateľom. Táto komunikácia je univerzálna a je možné komunikovať z rôznymi používateľskými rozhraniami. Naskytuje sa možnosť úplného oddelenia používateľského rozhrania od jadra v zmysle kompilácie programu, pričom komunikácia by prebiehala pomocou komunikačných protokolov, prípadne iných medziprocesových komunikačných techník.

## 6.3 Zhrnutie

V tejto časti si zosumarizujeme a v príslušných bodoch zhrnieme, čo požadujeme od takéhoto softvéru zobrazujúceho objemové dáta na základe vyššie popísaných vlastností.

1, Systémovo nezávislý (Windows, Linux – prípadne ľahko rozšíriteľný aj na iné operačné systémy),
2, Sada knižníc tvoriaca jadro softvéru pre objemové zobrazovanie a knižnice tvoriace jednotlivé zobrazovacie módy (oddelenosť od používateľského rozhrania) ,
3, Jednoduché používateľské rozhranie, umožňujúce demonštrovať možnosti jednotlivých modulov (systémovo nezávislé),
4, Dátová časť softvéru založená na grafe scény popisujúca potrebné elementy scény pre potreby zloženia jednoduchej scény. Jednoduchá rozšíriteľnosť (možnosť pridania) o nové scénové elementy,
5, Manažér modulov, ktorý je zodpovedný za načítanie a manažovanie modulov pri zobrazovaní. Umožnenie jednoduchého dodávania nových modulov, prípadne využitia a modifikácia už hotových na vytvorenie nových modulov,
6, Manažér dát pre grafické karty umožňujúci zdieľanie rovnakých textúr a iných dát pre potreby modulov,
7, Podpora pre základné formáty vstupných dát – f3d [52], raw formát [41], nrrd formát [22,55], prípadne iné.
8, Výstup zobrazovaných dát na obrazovku, prípadne ich uloženie na disk počítača,
9, Jednoduchý editor jednorozmerných prenosových funkcií a využitie týchto funkcií pri zobrazovacích módoch (po-klasifikácia, pred-integrovaná klasifikácia [6])
10, Podpora pre multispektrálne dáta získavané z konfokálnej mikroskopie, prípadne iných modalít generujúcich takéto dáta.
11, Implemenácia základných modulov na zobrazovanie objemových dát pomocou 2D, 3D textúr, algoritmu vrhania lúča a *bricking* algoritmu.
12, Implementovanie základných osvetľovacích modelov (Phongov osvetľovací model) pre príslušné moduly.

# Referencie


[1] B. Bargen, P. Donelly. *Inside DirectX.* Microsoft Press, 1998.

[2] S. Bruckner, S. Grimm, A. Kanistar, M. E. Gröller. *Illustrative Context-Preserving Volume Rendering.* In Proc. Eurographics, 2005

[3] B.Cabral, N. Cam, J. Fortan. *Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware*. ACM symp. On Vol. Vis., 1994

[4] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. *Fast visualization of object contours by non-photorealistic volume rendering.* In Proc. of EUROGRAPHICS, 2001.

[5] M. Červeňanský. *f3dvr – f3d volume renderer.* 2007. Dostupné na internete: www.sccg.sk/~cervenansky/f3dvr.

[6] K. Engel, T. Ertl. *Interactive High-Quality Volume Rendering with Flexible Consumer Graphics Hardware*. In Proc. Eurographics, 2002

[7] K. Engel, M. Hadwiger, J. Kniss, Ch. Rezk-Salama, D. Weiskopf. *Real-Time Volume Graphics.* A K Peters, 2006

[8] R. Fernando, M.J. Kilgard. *The Cg Tutorial*. Addison-Wesley, 2003

[9] J.Foley, A van Dam, S. Feiner, J. Hughes. *Computer Graphics, Principles And Practice*. Addison-Weseley, 1993

[10] FujiFilm Medical Systems. 2007. Dostupné na internete: http://www.fujimed.com/.

[11] R.S. Galagher. *Comupter Visualization: graphics techniques for scientific and engineering analysis.* CRC Pres, 1995

[12] A. Glassner. *Principles of DigitalImage Synthesis.* San Francisco: Morgan Kaufmann, 1995

[13] GPGPU. *General-Purpose Computation on GPUs.* 2007. Dostupné na internete: http://www.gpgpu.org/.

[14] H. Hauser, L. Mroz, G. Bischi, and M. E. Gröller. *Two-level volumerendering fusing MIP and DVR.* In Proceedings of IEEE Visualization, 2000.

[15] Havok. 2007. Dostupné na internet: http://www.havok.com/.

[16] H.C. Hege, T. Höllerer, D. Stalling. *Volume Rendering – Mathematical Models and Algorithmic Aspects.* Report TR 93-7, ZIB, Berlin, 1993

[17] J. Hladůvka, A. König, E. Gröller. *Curvature-Based Transfer Function for Direct Volume Rendering.* In Proc. SCCG, 2000

[18] J. Kajiya, B. Von Herzen. *Ray Tracing Volume Densities*. In Proc. Sigraph, 1984

[19] G. Kindlmann, M. Itkis, D. Weinstein, et all. *Teem.* 2005. Dostupné na internete: http://teem.sourceforge.net/.

[20] Kitware. *VolView.* 2005. Dostupné na internete: http://www.kitware.com/products/volview.html.

[21] J. Kniss, G. Kindlmann, Ch. Hansen. *Multidimensional Transfer Functions for Interactive Volume Rendering.* In Proc. IEEE Transactions on Visualization and Computer Graphics, 2002

[22] J. Kniss, G. Kindlmann, and Ch. Hansen. *Simian.* 2002. Dostupné na internete: http://www.cs.utah.edu/~jmk/simian/index.htm.

[23] J. Krüger, R. Westermann. *Acceleration Techniques for GPU-based Volume Rendering* In Proc. IEEE Visualization, 2003

[24] P. Kovach. *Inside Direct 3D*. Microsoft Press, 2000

[25] E. LaMar, B. Hamman, K. Joy. *Multiresolution Techniques for Interactive Texturebased Volume Visulation*. In Proc. IEEE Visualization, 1999

[26] E. Lengyel. *The OpenGL Extensions Guide.* Charles Rivier Media, Inc., 2003

[27] M. Levoy. *Display of surfaces from volume data*. IEEE Computer Graphics and Applications, 1987

[28] A. Limaye, *Drishti - Volume Exploration and Presentation Tool,* Poster presentation,Vis 2006, Baltimore. Dostupné na internete: http://sf.anu.edu.au/Vizlab/drishti/index.html.

[29] W. Lorensen, H. Cline. *Marching Cubes: A High Resolution 3D Surface ConstructionAlgorithm.* Comp. Graphics, 21(4):163-169, 1987

[30] J. Marks, W. Ruml, K. Ryall, J. Seims, et al. *Design galleries: a general approach to setting parameters for computer graphics and animation*. Siggraph, 1997

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

[32] MeVis Group. 2007. Dostupné na internete: http://www.mevis.de/mevis/english.php.

[33] Microsoft. *DirectX.* 2007. Dostupné na internete: http://msdn.microsoft.com/directx/.

[34] Nvidia. *Cg.* 2007. Dostupné na internete: http://developer.nvidia.com/Cg/.

[35] Nvidia. *Nvidia Developer Website.* 2007. Dostupné na internete: http://developer.nvidia.com/.

[36] OpenSceneGraph. 2007. Dostupné na internete: http://www.openscenegraph.com/.

[37] OpenSG. 2007. Dostupné na internete: http://opensg.vrsource.org/.

[38] H. Pfister, J. Hardenberg, J. Knittel, H. Lauer, L. Seiler. *The VolumePro Real-Time Ray-casting System*. Sigraph, 1999

[39] Philips. *Philips Medical Systems.* 2007. Dostupné na internete: http://www.medical.philips.com/main/index.asp.

[40] Ch. Rezk-Salama. *Volume Rendering Techniques for General Purpose Graphics Hardware*. Der Technischen Fakultät der Universität Erlangen-Nürnberg, 2001

[41] Ch. Rezk-Salama, K. Engel, and F. Higuera. *The OpenQVis project.* 2002. Dostupné na internete: http://openqvis.sourceforge.net/.

[42] S. Rttger, S. Guthe, D. Weiskopf, and T. Ertl. *Smart Hardware Accelerated Volume Rendering.* In Procceedings of EG/IEEE TCVG Symposium on Visualization VisSym, 2003

[43] Scientific Computing and Imaging Institute, University of Utah. *SCIRun.* 2007. Dostupné na nternete: http://software.sci.utah.edu/scirun.html.

[44] M. Segal, K.Akley. *The OpenGL Graphics System: A Specification*. 2007. Dostupné na internete: http://www.opengl.org/documentation/specs/.

[45] SGI. *OpenGL.* 2007. Dostupné na internete: http://www.opengl.org.

[46] SGI. OpenGL extensions registry. 2007. Dostupné na internete: http://oss.sgi.com/projects/ogl-sample/registry/.

[47] SGI. *OpenGL Shading Language.* 2006. Dostupné na internete: http://www.opengl.org/documentation/glsl/.

[48] Siemens. *Siemens Medical Solutions - Syngo.* 2007. Dostupné na internete: www.siemens.com/syngo-webspace.

[49] S. Stegmaier, M. Strengert, T. Klein, T. Ertl. *A Simple and Flexible Volume rendering Framework for Graphics-Hardware-based Raycasting.* In Proc. Volume Graphics, 2005

[50] P. Šereda, A. Vilanova Bartrolí, I.W.O. Serlie, and F.A. Gerritsen. *Visualization of Boundaries in Volumetric Data Sets Using LH Histograms.* In Proc. IEEE Transactions on Visualization and Computer Graphics, 2006

[51] M.Šrámek. *Visualization of Volumetric Data by Ray Tracing*. Österreichische Computer Ges., 1998

[52] M. Šrámek, and L. I. Dimitrov. *f3d project home page.* 2007. Dostupné na internete: http://gerlach.viskom.oeaw.ac.at/f3dtrac.

[53] Milos Šrámek, Leonid I. Dimitrov, Matúš Straka, and Michal Červeňanský. *The f3d tools for processing and visualization of volumetric data.* Journal of Medical Informatics and Technologies, pages MIP-71-MIP-79, 2004.

[54] TatraMed. 2007. Dostupné na internete: http://www.tatramed.com.

[55] Teem. *Nrrd file format.* 2005. Dostupné na internete: http://teem.sourceforge.net/nrrd/format.html.

[56] C. Upson, M. Keeler. *V-BUFFER: Visible Volume Rendering*. In Proc. Sigraph, 1988

[57] I. Viola, A. Kanitsar, M. E. Gröller. *Importance-Driven Volume Rendering.* In Proc. IEEE Visualization, 2004

[58] D. Weiskopf, K. Engel, T. Ertl. *Volume clipping via per-fragment operations in texture-based volume visualization.* In Proc. IEEE Visualization, 2002

[59] P.L. Wiliams, N. Max. *A volume density optical model*. 1992 Workshop on Volume Vizualization, pages 61-68, 1992

[60] O.Willson, A. Van Gelder, J. Wilhelms. *Direct Volume Renderng via 3D-textures*. Technical Report UCSC-CRL-94-19, Univ. of Carolina, Santa Cruz, 1994