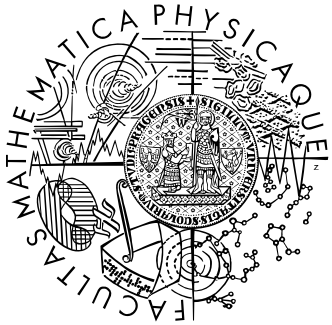
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Svoboda

Uživatelská rozhraní pro editaci přechodových funkcí

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Lukáš Maršálek
Studijní program: Informatika, Programování

2009

Na tomto místě bych rád poděkoval vedoucímu práce Mgr. Lukáši Maršálkovi za odborné vedení, vstřícnost a trpělivost.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 8.12.2009

Obsah

Seznam obrázků

Název práce: Uživatelská rozhraní pro editaci přechodových funkcí
Autor: Michal Svoboda
Katedra: Kabinet software a výuky informatiky
Vedoucí bakalářské práce: Mgr. Lukáš Maršálek
E-mail vedoucího: marsalek@cgg.ms.mff.cuni.cz
Abstrakt: Zobrazování objemových dat je v dnešní době rychle se rozvíjejícím odvětvím počítačové grafiky. Důvodem je významná role objemových dat například v lékařství, biologii či při simulaci fyzikálních experimentů.

Aby bylo možné objemová data „dobře“ vizualizovat, je nutné vytvářet tzv. „přechodové funkce“, které mapují vstupní hodnoty na zobrazitelné veličiny. Tvorba přechodových funkcí je často náročným a zdlouhavým procesem. Často je též vyžadována značná odborná znalost uživatele.

Cílem této práce je navrhnout uživatelské rozhraní pro tvorbu a editaci přechodových funkcí, se kterým by byla práce intuitivní a co možná nejjednodušší. Základním požadavkem je interaktivní odpověď zobrazovaných dat na změny jednotlivých přechodových funkcí. Výsledkem práce je uživatelské rozhraní, které neklade vysoké nároky na odborné znalosti uživatele, ale přesto mu umožňuje vytvářet informativní vizualizace.

Title: User Interfaces for Editing Transfer Functions
Author: Michal Svoboda
Department: Department of Software and Computer Science Education
Supervisor: Mgr. Lukáš Maršálek
Supervisor's e-mail: marsalek@cgg.ms.mff.cuni.cz
Abstract: Visualization of volume data is nowadays still actual theme in computer graphics. The reason is a significant role of volume data in fields such as medicine, biology or simulation of physics experiments.

In order to create informative visualizations of volume data, it is necessary to create so-called "transfer functions" that maps input values to a displayable quantity. Creating transfer functions is often difficult and lengthy process. Often considerable expertise is also required from the user.

The aim of this work is to design user interface for creating and editing transfer functions, which would work as intuitively as possible. The basic requirement is a direct interaction between transfer function and displayed data. The result of this work is user interface, which does not put high demands on specialized knowledge of the user.

Kapitola 1
Úvod

V současné době se většina programů zobrazujících tří-dimenzionální objekty zaměřuje pouze na viditelnou část, respektive jejich povrch. Pomocí, někdy i velkého množství, mnohoúhelníků se snaží co nejpřesněji aproximovat požadovaný objekt. V některých případech nám však takové zobrazení povrchu objektu nestačí.

K takové situaci dochází, když potřebujeme zobrazit nějaké vlastnosti vnitřku objektu, například chceme-li vizualizovat vnitřní hustotu, teplotu či jakékoliv jiné vlastnosti. Typickým příkladem je využití v souvislosti s lékařstvím, především CT (Computer Tomography[1]) a MRI (Magnetic Resonance imaging[2]), které v konečném výstupu produkují trojrozměrná data.

Dalším příkladem jsou takové objekty, které nelze jednoduše zobrazit pomocí povrchové reprezentace, neboť povrch jako takový nemají. Výbornými příklady jsou například oheň, mraky nebo mlha. Na základě těchto důvodů přicházejí na řadu tzv. objemová data, jejich zpracování a zobrazování.

Úkolem této kapitoly je seznámit se se základními pojmy z oblasti objemových dat, zdůvodnit důležitost přechodových funkcí a nastínit cíle této práce.

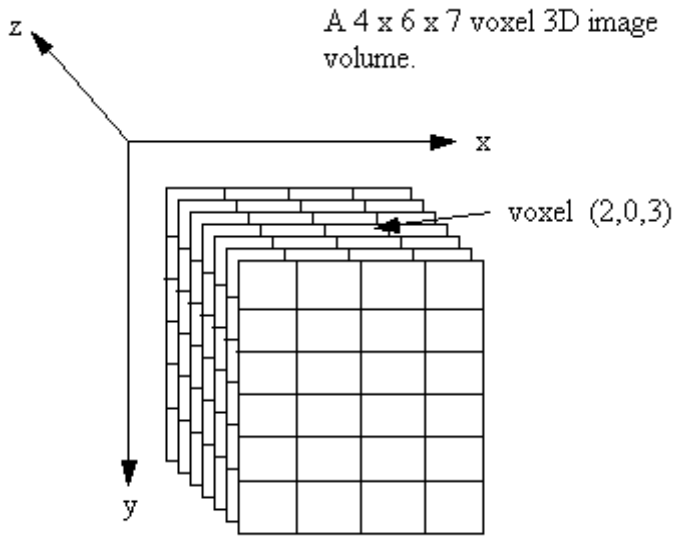
1.1 Objemová data

Rozbor možných způsobů získávání a uchovávání objemových dat není hlavním tématem této práce, proto pouze nastíníme nejčastější způsob vyčíslení objemu obsazeným tělesem. Tím je rozdělení části tří-dimenzionálního prostoru na velké množství pravidelných elementárních objemových jednotek, které se nazývají

1 Počítačová tomografie (Computer Tomography – CT) byla poprvé představena Godfreyem Hounsfieldem v sedmdesátých letech minulého století. CT pro získání objemových dat používá rentgenové paprsky, snímače rentgenových paprsků a počítač pro vytvoření obrazu příčného řezu tělem pacienta. Oproti standardní radiografii dosahuje rozlišení CT hodnot okolo 1/3 mm, což je sice poměrně hrubá jednotka, avšak pro mnoho diagnostických účelů je dostačující. Rozměry voxelové mřížky získané pomocí CT mohou nabývat hodnot až 512×512×200 voxelů s bitovou hloubkou typicky 12 bitů na voxel (při zobrazování se mnohdy provádí decimace na osm bitů).

2 Magnetická rezonance (**Magnetic Resonance** imaging – MRI) byla poprvé experimentálně ověřena v roce 1946 americkými vědci F. Blochem a E. Purcellem. V roce 1971 (P. Lauterbur) bylo využito změn gradientu magnetického pole k získání dvourozměrného tomografického obrazu rezonujících jader. V roce 1974 (P. Mansfield) byl publikován první medicínský obraz získaný pomocí MRI.

voxely (z anglického Volume Element). Při použití voxelů jako základních nosičů „objemové“ informace se původně spojitý trojrozměrný prostor diskretizuje na jednotlivé mikroobjemy. V základní podobě, kdy potřebujeme pouze vyjádřit, zda daná část v prostoru náleží či nenáleží do tělesa, mohou voxely nabývat pouze dvou stavů (obsazeno/neobsazeno). V praxi však takto jednoduchá reprezentace nenachází velké množství využití. Mnohem častěji se voxelům přiřazují nějaké další informace, ať už spojité nebo diskrétní. Například při zpracování medicínských dat získaných výše zmíněnými metodami CT a MRI, kdy je každý voxel reprezentovaný hodnotou, která se v případě metody CT vztahuje k útlumu rentgenových paprsků a v případě MRI k počtu protonových jader v nasnímaném objemu. Část prostoru, ve kterém se nachází těleso či celá scéna, je popsaná množinou voxelů, které jsou pravidelně rozmístěny v osově orientované krychli či kvádru. Ukázku rozdělení objemových dat můžete vidět na následujícím obrázku.



Obrázek 1.1: Rozdělení objemových dat pomocí pravidelné mřížky. Obrázek je převzat z [0]

1.2 Způsoby vizualizace objemových dat

Objemová data, která běžně vznikají v průběhu různých měření nebo jsou získána jako výsledek simulací, bývají velice rozsáhlá. Proto musí být zpracovávána pomocí efektivních algoritmů a výkonných výpočetních systémů. Vývoj vizualizačních nástrojů poměrně rychle pokračuje, a to jak v oblasti technických prostředků, tak i algoritmů pro efektivní práci s objemovými daty. Algoritmy pro vizualizaci vícerozměrných informací lze podle typu zpracovávaných dat rozdělit do dvou skupin:

- Algoritmy, které zobrazují skalární prostorové mřížky. Více o těchto algoritmech je uvedeno níže.
- Algoritmy, které zobrazují vektorová nebo tenzorová pole. Vzhledem k vysoké informační hodnotě a vnitřní provázanosti těchto dat je nutné tyto informace zobrazovat s použitím speciálních algoritmů. Touto částí zobrazování dat se v této práci zabývat nebudeme.

Existují různé algoritmy, které zobrazují skalární prostorové mřížky, ale každý může být popsán podobným sledem základních kroků, jak bylo shrnuto například v [1].

Základní kroky jsou následující:

1. Získávání dat
2. Předzpracování dat
3. Klasifikace dat
4. Projekce dat do rovinného obrazu pixelů
5. Zobrazení

Získání dat

V tomto kroku se měří, počítají, nebo generují objemová data. Předpokladem pro tento krok je jejich korektní zpracování tak, aby mohla být správně rekonstruována (například dle Nyquist-Shannonova teorému).

Předzpracování dat

Zejména naměřená skalární data používaná pro vizualizaci je velmi často nutné předzpracovat. To může zahrnovat nastavení jasu, kontrastu, odstranění šumu, ekvalizaci histogramu a další.

Klasifikace dat

Jedná se o proces mapování hodnot objemových dat na hodnoty použité pro vizualizaci. V případě přímého zobrazování objemových dat tento krok většinou zahrnuje přidělení barvy a neprůhlednosti objemové hodnotě. V případě metod zobrazujících povrchy je nejdříve nutné vytvořit pomocnou geometrickou strukturu.

Projekce dat do rovinného obrazu pixelů

Obecně se v tomto kroku rozhoduje, jaký efekt budou mít různé voxely na výsledný rovinný obraz (projekční rovinu). Algoritmy použité v tomto kroku se obecně dělí do dvou skupin

- algoritmy zobrazující povrchy
- přímé objemové algoritmy

Obě skupiny jsou dále diskutovány zvlášť.

Zobrazení

Posledním krokem je zobrazení výsledků z předchozích kroků.

1.2.1 Algoritmy zobrazující povrchy

Algoritmy zobrazující povrchy většinou musí nejdříve ze vstupních dat (tj. prostorové mřížky skalárních hodnot – voxelů) vytvořit pomocnou geometrickou strukturu, pomocí které je vyjádřen povrch. Tyto algoritmy tedy pracují se vstupními daty nepřímo, protože nejprve vytvoří povrch, který je reprezentován jednoduššími geometrickými primitivy, nejčastěji trojúhelníky, které tvoří nepravidelnou síť trojúhelníků (TIN – Triangular Irregular Network). Tyto trojúhelníky se posléze zobrazují s využitím standardních grafických knihoven (například OpenGL, Direct 3D) s případnou technickou podporou (tj. urychlení zobrazování) ve formě grafických akcelerátorů. Při zobrazování tímto způsobem lze využít prakticky všechny zobrazovací postupy používané v 3D počítačové grafice, například texturování, osvětlení, antialiasing, poloprůhlednost povrchu atd.

Pro převod dat z formy skalární prostorové mřížky do povrchové (hraniční) reprezentace se nejčastěji používají klasické algoritmy určené k převodu dat z objemové reprezentace. Mezi tyto algoritmy patří například algoritmus Marching cubes z [2], popř. algoritmus Marching Tetrahedra uvedený v [3].

1.2.2 Přímé objemové algoritmy

Objemové algoritmy zobrazují skalární data přímo. Neprovádí tedy prvotní převod do povrchové reprezentace. Tyto algoritmy umožňují zobrazovat i poloprůhledné objemy, čehož lze využít například při vizualizaci kapalin nebo struktur obsažených uvnitř jiných struktur. Přímé zobrazování objemu (dále jen DVR z anglického Direct Volume Rendering) využívá mapování z hodnot skalární mřížky do vizuálních vlastností, jako je barva či neprůhlednost. Toto mapování obstarávají přechodové funkce. Příkladem nejjednodušší přechodové funkce může být přímé přiřazení barvy konkrétní hodnotě objemové veličiny. Bez možnosti takového, nebo i jiného přiřazení, je velmi obtížné DVR používat. Navíc je toto mapování objemových dat díky úžasným možnostem variací přechodových funkcí velice flexibilní. Z těchto důvodů je v dnešní době na přechodové funkce kladen velký důraz.

Zároveň však díky této flexibilitě bývá velice obtížné najít smysluplnou přechodovou funkci a právě z těchto důvodů je velice důležité uživatelské rozhraní, které by umožnilo vizualizaci přechodové funkce a hlavně rychlé nalezení požadovaného výsledku.

1.3 Cíl práce

Jak bylo zmíněno výše, nalezení vhodné přechodové funkce je obtížný úkol. Stále se často používají těžkopádná uživatelská rozhraní, v kterých může uživatel vytvořit jednoduchou jedno-dimenzionální přechodovou funkci tak, že nakreslí grafy všech jednotlivých složek barvy i neprůhlednosti. Toto řešení má však dvě zásadní chyby. V první řadě je extrémně obtížné dosáhnout požadované barvy manipulací červené, zelené a modré složky barvy. Za druhé tento mechanismus nedovoluje jednoduchou manipulaci s barvou v konkrétní oblasti zájmu uživatele. V případě že chce změnit barvu pro konkrétní hodnotu, je prakticky nucen překreslit celou přechodovou funkci.

Často se u přechodových funkcí také využívá metoda „pokus-omyl“, kdy se jednoduše změní přechodová funkce a sleduje se efekt této změny. Proto je důležitá přímá a rychlá interakce mezi změnou přechodové funkce a vykreslovaným objektem.

Cílem této práce je právě naprogramování příjemného uživatelského rozhraní, které by nebylo postiženo zmiňovanými nedostatky, tedy rozhraní pro rychlé nalezení přechodové funkce zobrazovaných objemových dat a jejich interaktivní zobrazení, které umožňuje zejména:

- vizuální editaci přechodových funkcí přímo při zobrazování objemových dat
- interaktivně propojovat změny přechodových funkcí do objemové vizualizace
- lokální editaci přechodových funkcí
- více-dimenzionální přechodové funkce

1.4 Organizace práce

V této kapitole jsme se seznámili se základními pojmy z oblasti objemových dat a lehce jsme nastínili hlavní principy jejich vizualizace. Dále jsme zdůraznili složitost hledání přechodových funkcí a jejich důležitost. Na závěr jsme jako cíl této práce stanovili naprogramování přívětivého uživatelského rozhraní pro zadávání přechodových funkcí.

Kapitola 2 se zabývá teoretickou stránkou přechodových funkcí. Popisuje jejich využití a opět poukazuje na jejich důležitost. V této kapitole se také zabýváme

rozdělením přechodových funkcí a popisujeme různé způsoby jejich hledání. Součástí kapitoly je též stručný popis aktuálního vývoje v této oblasti. Na závěr kapitoly je též uveden jeden méně tradiční pohled na přechodové funkce.

V kapitole 3 se již zabýváme uživatelskými rozhraními pro zadávání přechodových funkcí. Zaměřujeme se na způsoby, jakými mohou uživatelé vytvářet přechodové funkce. Důraz je pak kladen na ta rozhraní, která byla v této práci následně implementována. Na závěr se pak opět věnujeme nejnovějšímu vývoji v tomto oboru.

Kapitola 4 se zaměřuje na implementaci této práce. Obsahuje představení použitého vývojového prostředí, popis modulu využitého pro zobrazování objemových dat a jeho úpravy. Následně je popsána struktura kódu implementovaných přechodových funkcí. Popis kódu je koncipován tak, aby odrážel principy používané při programování v knihovnách „Qt“, které jsou pro tuto práci využity.

Předposlední 5 kapitola ukazuje příklady vykreslení různých objektových dat pomocí přechodových funkcí implementovaných v této práci. Součástí je také krátké srovnání s méně sofistikovaným uživatelským rozhraním.

Závěrečná 6 kapitola podává shrnutí celé práce. Každé kapitole zde věnujeme krátké shrnutí.

Příloha A obsahuje uživatelský návod implementovaného uživatelského rozhraní.

Příloha B sepisuje obsah přiloženého DVD.

Kapitola 2
Přechodové funkce

V této kapitole se budeme zabývat pouze objemovými daty, jejichž reprezentace je založena na diskreditaci původně spojitého trojrozměrného prostoru. Konkrétně se může jednat o skalární 8, 12 či 16 bitové hodnoty. Tuto reprezentaci objemových dat jsme popsali v odstavci 1.1. Na těchto datech si popíšeme nepostradatelnost přechodových funkcí a následně pak důležitost dobrého uživatelského rozhraní. Hlavním zdrojem pro tuto kapitolu je práce [4], neboť poskytuje výborný přehled základních teorií přechodových funkcí.

2.1 Proč přechodové funkce?

Jsou-li objemová data reprezentována abstraktními skalárními hodnotami reprezentujícími nějaké prostorové vlastnosti objektu, neexistuje v obecném případě přirozená cesta, jak odvodit správné vizuální vlastnosti této hodnoty, jako jsou vyřazování, pohlcování světla a další. Na místo toho musí uživatel rozhodnout, jaké optické vlastnosti by měly mít různé struktury v objemových datech. Uživatel tedy potřebuje vytvořit mapování mezi skalárními hodnotami objemových dat a optickými vlastnostmi zobrazitelnými na obrazovce. Toto mapování je úkolem přechodových funkcí.

Dříve, než popíšeme různé přístupy hledání specifických přechodových funkcí, představíme základní přechodové funkce.

2.2 Rozdělení přechodových funkcí

Přechodové funkce jsou vlastně obyčejné funkce, jak je známe z matematiky. Jak víme, v matematice je pojem „funkce“ používán pro zobrazení z jedné množiny prvků do druhé množiny. V našem případě se jedná o zobrazení množiny hodnot objemových dat do množiny hodnot zobrazitelných na obrazovce. Abychom mohli nadefinovat přechodovou funkci, musíme v první řadě znát definiční obor a obor hodnot funkce.

2.2.1 Dělení dle oboru hodnot

V nejjednodušších přechodových funkcích jsou definičním oborem jednoduché skalární hodnoty a oborem hodnot je neprůhlednost. Tato jednoduchá přechodová funkce může být zobecněna do různých typů přechodových funkcí jednoduchým rozšířením oboru hodnot. Obor hodnot pak většinou zahrnuje i barvu,

neboť barva přirozeně vizualizuje odlišnosti mezi strukturami vstupních dat [5]. Obecně každá optická vlastnost, která je zobrazitelná v počítačové grafice, může patřit do oboru hodnot přechodové funkce, od neprůhlednosti a barvy, přes vyzařování světla, rozptyl světla dle fázové funkce [6], stínování, texturování [7], až po index lomu světla [8]. Každý z těchto elementů může být reprezentován v různé složitosti, například neprůhlednost se může lišit dle barvy, namísto rozlišení podle jednotlivých skalárních dat [9].

2.2.2 Dělení dle dimenze definičního oboru

Dalším způsobem, jak rozšířit přechodové funkce, může být zvětšení dimenze definičního oboru. Takové přechodové funkce se nazývají více-dimenzionální. V námi diskutované reprezentaci objemových dat, založené na skalárních datech, je například užitečné vzít do úvahy velikost gradientu, jako druhou dimenzi přechodové funkce. Velikost gradientu nám ukazuje, jak rychle se vstupní data mění. Využitím velikosti gradientu jako druhé dimenze přechodové funkce můžeme nalézt hranice mezi homogenními oblastmi v objemových datech, které by jinak byly neodlišitelné od homogenních oblastí se stejnou skalární hodnotou. Tyto více-dimenzionální přechodové funkce jsou speciálně užitečné například v lékařství, kde často potřebujeme rozlišit mezi různými druhy tkání.

Další možností rozšíření o druhou dimenzi může být využití druhé derivace, která je využitelná například při detekci hran v obrazu [10]. Více možností využití dvou, či dokonce tří-dimenzionálních přechodových funkcí je diskutováno v práci [11]. V konečném důsledku se ukazuje, že pro zvýraznění různých vnitřních struktur při vykreslování postačuje v přechodových funkcích velikost gradientu. To vyplývá například z použité techniky objemového vykreslování diskutované v článku [12]. Přechodové funkce založené výhradně na dvou-dimenzionálním prostoru jsou také diskutovány v práci [13].

Jiným důvodem pro více-dimenzionální přechodové funkce je zobrazování objemových dat, které nejsou složeny pouze z jednoduchých skalárních hodnot. Příkladem je magnetická rezonance, které může měřit více různých fyzikálních vlastností živé tkáně [14]. Každá z těchto měřených hodnot může být složkou definičního oboru více-dimenzionální přechodové funkce [15]. Další využití více-dimenzionálních přechodových funkcí je možné nalézt v práci [4].

2.3 Hledání přechodové funkce

Jak již bylo řečeno v úvodu, přechodové funkce jsou velice flexibilním nástrojem, což je zároveň výhodou i nevýhodou. Základní výhodou je možnost zobrazovat velké množství různých aspektů objemových dat. Avšak nalezení vhodné přechodové funkce může být extrémně složitým problémem. Běžnou metodou stále

bývá zdlouhavý způsob založený na opakovaném zkoušení různých variant, dokud není dosaženo požadovaného výsledku (tzv. metoda „pokus-omyl“). Jak je uvedeno v článku [4], jsou zde proto tři hlavní důvody:

- Prvním důvodem je zmiňovaná flexibilita přechodových funkcí. Ve velkém množství možností, které přechodové funkce nabízejí není snadné najít tu nejvhodnější variantu.
- Druhým problémem jsou samotná uživatelská rozhraní, která snadno nedokáží omezit, nebo nějakým způsobem uživatele navést k přechodové funkci, která by zobrazovala požadovanou oblast zájmu v objemových datech. To vede uživatele k opakovanému zkoušení a pozorování efektu zvolené přechodové funkce (metoda „pokus-omyl“).
- Posledním důvodem je, že v obecné podobně nejsou přechodové funkce prostorově závislé. Tím máme na mysli, že neumějí pracovat s pozicí v tří-dimenzionálních datech. Pozice přitom může být v některých případech důležitá. Například chceme-li vizuálně rozlišit konkrétní oblast mezi velmi podobnými hodnotami ve zkoumaných datech.

Právě z těchto důvodů se vývoj v oblasti přechodových funkcí soustředí na zjednodušení procesu hledání vhodné přechodové funkce. Existující metody se, stejně jako například v pracích [16], [17] či [18], dají zhruba rozdělit do dvou kategorií:

- „datově řízené“ z anglického data-driven:

  Tyto metody získávají informace ze samotných objemových dat. Získané informace pak využívají k omezení prostoru přechodových funkcí, nebo pomáhají řídit uživatele k požadovanému výsledku.

- „obrazově řízené“ z anglického image-driven:

  Tyto metody jsou založené na zkoumání vykreslených objemových dat. Touto cestou, která může být také částečně automatizována, se snaží ovlivnit rozsah prostoru přechodových funkcí na základě analýzy výsledného vykreslení.

Existují také algoritmy pro automatické generování přechodových funkcí, ale stále nedosahují dostačujících výsledků. Proto je vytvoření přechodové funkce stále ruční a často zdlouhavou záležitostí, jež vyžaduje detailní znalosti o struktuře prezentovaných dat. Proto je důležité, aby byla uživateli poskytnuta okamžitá vizuální odezva na jeho manipulaci s přechodovou funkcí.

2.4 Obrazově řízené přechodové funkce

Jak již bylo řečeno, jedním ze základních problémů při zobrazování objemových dat je nalezení té nejvhodnější přechodové funkce z mnoha variací, které přechodové funkce umožňují. Hlavní přínos metod založených na technice obrazově řízených přechodových funkcí pro uživatele spočívá v možnosti nalézt požadované zobrazení bez rozsáhlých znalostí či přímé manipulace s přechodovými funkcemi.

Jedna z prací zaměřující se na toto řešení je prezentována v [19]. Metoda popsaná v této práci v prvním kroku náhodně vygeneruje sadu přechodových funkcí. Tyto přechodové funkce pak aplikuje na objemová data, čímž vznikne odpovídající skupina náhledů, které jsou prezentovány uživateli. Uživatel pak vybere ty náhledy, které nejlépe vykreslují zobrazovaná data. Na základě vybraných náhledů, respektive přechodových funkcí, algoritmus generuje další sadu upřesňujících náhledů, z kterých může uživatel vybírat. Tyto kroky se opakují, dokud není dosaženo požadovaného výsledku. Práce též popisuje možnost zahrnout algoritmy pro zpracování obrazu, jako jsou algoritmy využívající například entropie[3], rozptylu[4] či další, které umožňují analyzovat výsledné vykreslení obrazu. Problém hledání vhodné přechodové funkce je tímto převeden na optimalizační úlohu, v které není potřeba manuálního zásahu uživatele, ale uživatel „pouze“ hraje roli optimalizační funkce.

Jiná publikace [20] převádí problém hledání vhodné přechodové funkce na, v počítačové grafice již známý, proces zvaný „zlepšování parametrů“ (z anglického „parameter tweaking“). Algoritmus popisovaný v této práci se ovšem nesnaží nalézt nejlépe vyhovující parametrické nastavení, avšak využívá uživatelem specifikovanou metriku, na kterou aplikuje co nejširší počet různých parametrických nastavení. Obdobně jako v předchozím případě pak systém vygeneruje velkou sadu přechodových funkcí, pomocí kterých opět vytváří náhledy. Tyto náhledy jsou taktéž publikovány uživateli, z kterých může vybrat nejlepší vykreslení.

V některých dalších pracích je pozornost věnována vhodné organizaci vygenerovaných náhledů, která usnadňuje uživateli nalézt požadovanou přechodovou funkci.

3 Entropie je matematický výraz udávající míru neuspořádanosti systému. V počítačové grafice se entropie využívá k segmentaci obrázků.

4 Rozptylem se v tomto případě myslí pojem z teorie pravděpodobnosti a statistiky, který vyjadřuje variabilitu rozdělení náhodných hodnot kolem její střední hodnoty. V počítačové grafice je možné použít znalostí o tomto rozdělení ke zpracování obrazu. Například k jeho vyhlazení.

Poslední prací, o které se zmíníme v této kapitole, je [21]. Práce popisuje uživatelské rozhraní organizující vygenerované náhledy do struktury grafu. Graf pak znázorňuje historii změň parametrů. Jednotlivé uzle grafu zobrazují náhledy vykreslení a hrany znázorňují změny parametrů provedené mezi odpovídajícími uzly.

2.5 Datově řízené přechodové funkce

Stejně jako u metod založených na obrazově řízených přechodových funkcích se i v tomto případě jedná o snahu nějakým způsobem vést uživatele co nejrychlejším a nejpřímější způsobem k vhodné přechodové funkci. Tato skupina metod k tomu však využívá informací, které odvozuje přímo z objemových dat, nikoli z vykresleného výsledku, jak tomu bylo u obrazově řízených metod. Metody založené na této technice často využívají histogramu a jeho analýzy. Hlavním důvodem hojného využití histogramů je jejich nezávislost na pozici v prostorových datech, která reflektuje poziční nezávislost přechodových funkcí.

Základní metodou skupiny datově řízených přechodových funkcí je vyšetřování tzv. „izoploch“ (z anglického isosurface). Izoplocha je definována skupinou hodnot shodných přes celá objemová data se zadanou hodnotou (tzv. „isovalue“ z anglického jazyka). Hledání izoplochy je možné si také představit jako hledání vhodného parametru přechodové funkce, který je pak následně aplikován na celá objemová data. Přestože se zjevně jedná o jedno-dimenzionální prostor, nalezení té správné důležité izoplochy může být stále časově náročný a zdlouhavý proces. Z tohoto důvodů se vývoj v oblasti izoploch zaměřil na pomoc uživateli s hledáním vhodné izoplochy. Významného urychlení hledání izoploch dosahují uživatelská rozhraní, která umožňují přímou interakci mezi přechodovou funkcí a vykreslením objemových dat. Při změně izoplochy tak uživatel přímo vidí změnu v objemových datech, což mu umožňuje rychle rozhodnout o důležitosti zobrazované izoplochy.

Kromě vizuální kontroly však existuje více různých způsobů, jak pomoci uživateli rozhodnout, která izoplocha pro něj může být důležitá. Tyto způsoby mohou být založeny na různých principech. Uvedeme si je v několika kategoriích. Každou vysvětlíme na příkladné metodě s odkazem na studie jejich autorů.

Jako první uvedeme metodu založenou na analýze matematických vlastností objemových dat. Například povrch, objem, průměrná velikost gradientu a další topologické vlastnosti objemových dat, jako jsou spojitost, incidence či jiné. Pomocí těchto vlastností se pak ohodnotí rozsah možných izoploch vybranou metrikou. „Váhy důležitosti“ jednotlivých izoploch se pak zaintegrují do uživatelského rozhraní, což může uživatele vést k rozhodnutí, které izoplochy jsou pro něj nejdůležitější. Příkladem této kategorie je metoda zvaná „The contour spectrum“ uvedená v [22] používána u nestrukturovaných sítí.

Další princip využívá sledování změn topologie dle izohodnot. Základní myšlenku tohoto přístupu si uvedeme na metodě publikované v článku [23]. Fujishiro používá grafu k popisu topologie izoploch v každé izohodnotě (tzv. „Hyper Reeb graph“). Tento graf může být opět ohodnocen na základně zvolených předpokladů. Informace získané z takového ohodnoceného grafu mohou být taktéž využity k vybrání vhodné izoplochy. Výhodou této metody je možnost zautomatizování pomocí klasických algoritmů pro vyhodnocování ohodnocených grafů.

Náhled na datově řízené přechodové funkce ještě doplníme o metodu z článku [24], která je tentokrát založena na statistických metodách. Pro vyšetření významných izoploch využívá faktu, že histogram může být popsán pomocí „momentů“. Momenty se v teorii pravděpodobnosti a statistiky používají k popisu charakteristik náhodné veličiny. Umožňují tak popsat chování v určitém okolí. Druhý centrální moment vyjadřuje rozptyl, třetí šikmost a čtvrtý špičatost. Tyto lokální statistické charakteristiky mohou být spočítány pro každou hodnotu objemových dat. Jak uvádějí Tenginakai a kolektiv ve své práci, mohou být tyto spočítané charakteristiky využity k vytvoření tzv. „statistického podpisu“, který může sloužit právě pro určení významných izoploch.

Jako poslední si uvedeme práci [12]. Autoři se v této práci vyhýbají přímé závislosti na datových hodnotách tím, že vytvářejí přechodové funkce založené na hodnotách gradientu. Využívají faktu, že omezením neprůhlednosti hodnot, jejichž směr gradientu odpovídá směru pohledu, dochází ke zvýraznění siluety objektu. Přechodové funkce založené na gradientu jsou dnes často využívané, neboť gradient nám dává užitečnou informaci o hranici mezi materiály. V této práci implementujeme jednu dvou-dimenzionální přechodovou funkci, která též využívá gradientu.

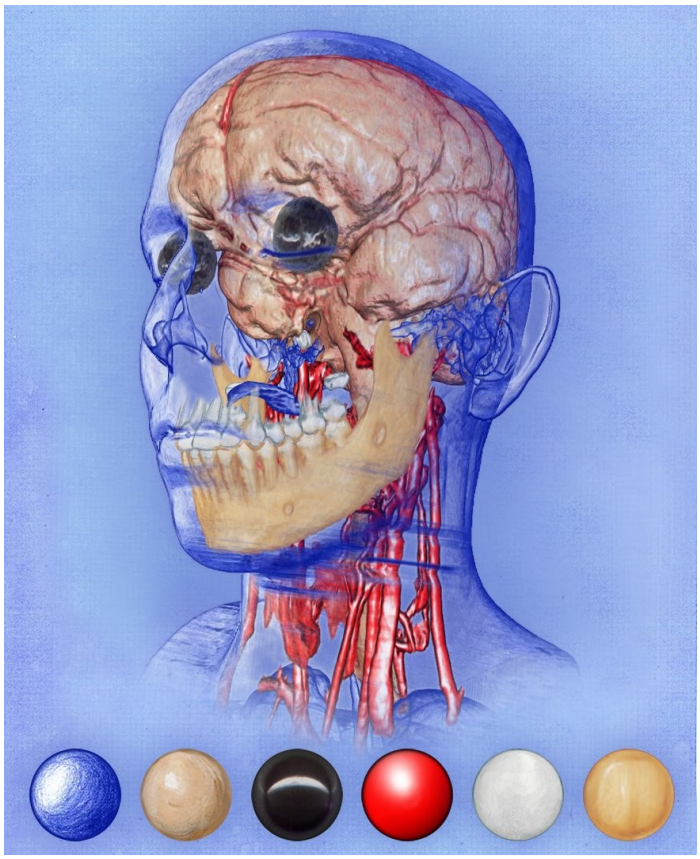
Existují další způsoby, jak pomoci uživateli nalézt významnou izoplochu. Většinou však kladou vyšší nároky na znalosti uživatele. Proto jsme se v této práci zaměřili na uživatelské rozhraní, které přímo reaguje na změny v přechodové funkci okamžitým překreslením objemových dat.

2.6 Aktuální vývoj přechodových funkcí

Kromě již popsaných způsobů vytváření přechodových funkcí se objevují nové přístupy. Jedním z příkladů mohou být přechodové funkce založené na stylech, tzv. „style transfer function“, dále jen STF. STF využívají informací získaných z projekce osvětlené koule. Tato technika umožňuje snadno dosáhnout světelně sladěného zobrazení. STF dovolují kombinovat velké množství různých stylů stínování v jednom vykreslení. Díky této technice můžeme simulovat nerealistické

ilustrace, které mohou uživateli lépe zvýraznit důležité rysy v objemových datech [25], [26]. Ukázka takové ilustrace je uvedena na obrázku 2.1.

Technika STF je podrobněji rozebrána například v článku [27].



Obrázek 2.1: Style transfer function

Příklad nerealistické ilustrace vykreslené pomocí přechodové funkce založené na stylech.

2.7 Netradiční přechodové funkce

Aby bylo představení přechodových funkcí kompletní, ukážeme si ke konci této kapitoly méně tradiční pohled na přechodové funkce. Z předešlých kapitol již víme, že přechodové funkce jsou v obecném případě pozičně nezávislé. Tím máme na mysli, že barva, průhlednost či další vizuální vlastnosti jsou přiřazovány na základě lokálně naměřené hodnoty, například velikost gradientu. Tyto hodnoty se mění s pozicí v objemových datech, ale pozice jako taková se v doméně přechodové funkce neobjevuje. Některé výzkumy v oblasti objemového vykreslování se snaží nepoužívat tradiční přechodové funkce, ale získávají informace ze struktury dat.

Díky tomu se problém nalezení vhodné přechodové funkce mění na vyhledávání určitých rysů struktury dat. Tato práce se však nezabývá těmito technikami zpracování objemových dat, proto uvedeme pouze několik příkladů s odkazy na články, v nichž byly diskutovány.

Autoři práce [28] používají tří-dimenzionální morfologické operace, jako jsou eroze nebo dilatace, aby určili charakteristické rysy, výsledek pak používají ke generování barevné vizualizace objemových dat.

Fang a kolektiv [29] vizualizují data použitím sekvence operací na zpracování obrazu, jako jsou ekvalizace histogramu, vyhlazování, ostření a detekce hran. Poté používají jednoduché lineární funkce pro přiřazení neprůhlednosti.

Jiří Hladůvka s kolektivem využívá v práci [30] analýzy vlastních čísel Hessianovy matice[5] a velikostí gradientu, pomocí které identifikuje význačné rysy objemových dat. Tyto údaje pak využívá k usnadnění vizualizace.

5 Matice druhých parciálních derivací

Kapitola 3

Uživatelské rozhraní

V předchozích částech práce jsme již několikrát zmiňovali důležitost uživatelských rozhraní v oblasti zobrazování objemových dat pomocí přechodových funkcí. Vzhledem ke stále se rozvíjejícím nárokům na zobrazování objemových dat je vývoj v oblasti uživatelských rozhraní stále aktivním tématem. Jedním z posledních trendů je snaha vytvořit uživatelská rozhraní umožňující automatické generování přechodových funkcí. Některé techniky v této oblasti již dosahují poměrně dobrých výsledků, stále je však nutné, aby uživatel rozhodl, jakou oblast dat chce zobrazit a jakým způsobem má být tato oblast vykreslena. Z těchto důvodů se v této kapitole zaměříme na způsoby, jakými mohou uživatelé vytvářet přechodové funkce. Zdůrazníme ty metody, které byly v této práci implementovány, a zakončíme odstavcem věnujícím se technologii nejnovějšího vývoje v oblasti uživatelských rozhraní přechodových funkcí.

3.1 Rozdělení uživatelských rozhraní

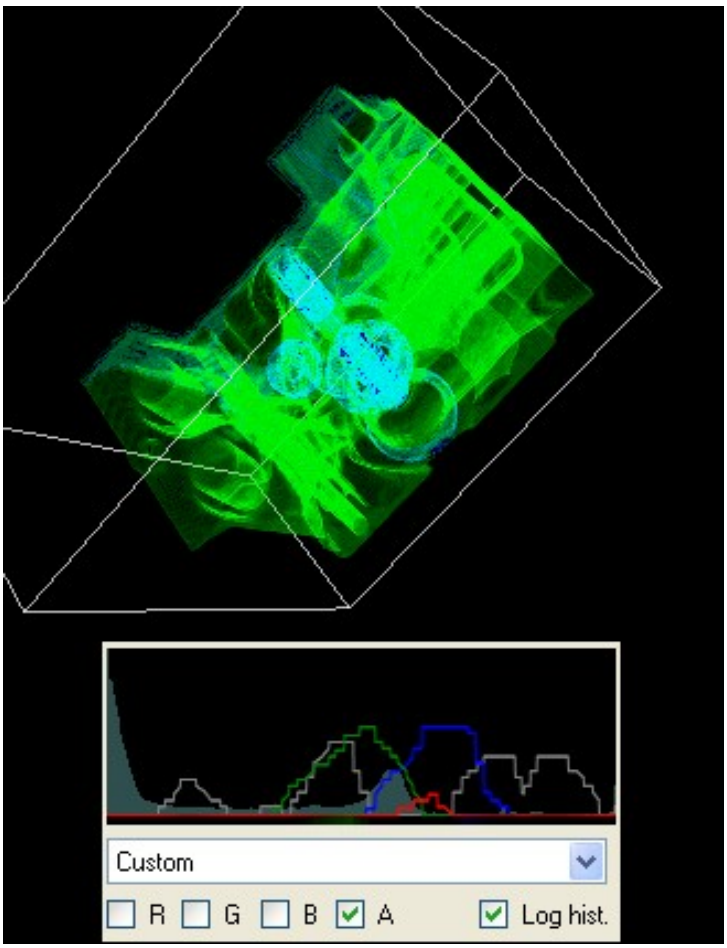
Stejně jako jsme členili přechodové funkce na jedno-dimenzionální, dvou-dimenzionální a více-dimenzionální, je možné pohlížet podobným způsobem i na rozdělení uživatelských rozhraní pro zadávání přechodových funkcí odpovídající dimenze. Snadno nahlédneme, že uživatelská rozhraní pro zadávání jedno-dimenzionálních přechodových funkcí budou představovat jednodušší problematiku, zatímco s přibývajícími dimenzemi bude složitější nejen zadávání přechodových funkcí, ale i jejich vykreslení. Naštěstí jedno-dimenzionální přechodové funkce v dnešní době stále převládají. Dvou-dimenzionální přechodové funkce mají menšinové zastoupení a více-dimenzionální, zejména díky nedostatku vhodných paradigmat pro jejich specifikaci, stále nenalezly většího užití. Začneme tedy s popisem různých uživatelských rozhraní pro zadávání jedno-dimenzionálních přechodových funkcí, a to od nejjednodušších po složitější.

3.1.1 Uživatelské rozhraní jedno-dimenzionální přechodové funkce

Nejjednodušší představou přechodové funkce může být jednoduchá tabulka přiřazující objemovým datům hodnoty zobrazitelné na obrazovce; například neprůhlednost, či jednu z barevných složek. Naproti tomu představa uživatelského rozhraní spočívajícího ve vyplňování takovýchto tabulek může být poněkud děsivá. Většina uživatelských rozhraní má proto grafickou podobu. Například v případě

„datově řízených“ přechodových funkcí je často základem histogram vyobrazený na pozadí UI.

Prvním příkladem takového uživatelského rozhraní je ručně kreslený graf jednotlivých barevných složek a neprůhlednosti. Ukázka je zobrazena na obrázku 3.1.



Obrázek 3.1: Ručně nakreslený graf jednotlivých barevných složek

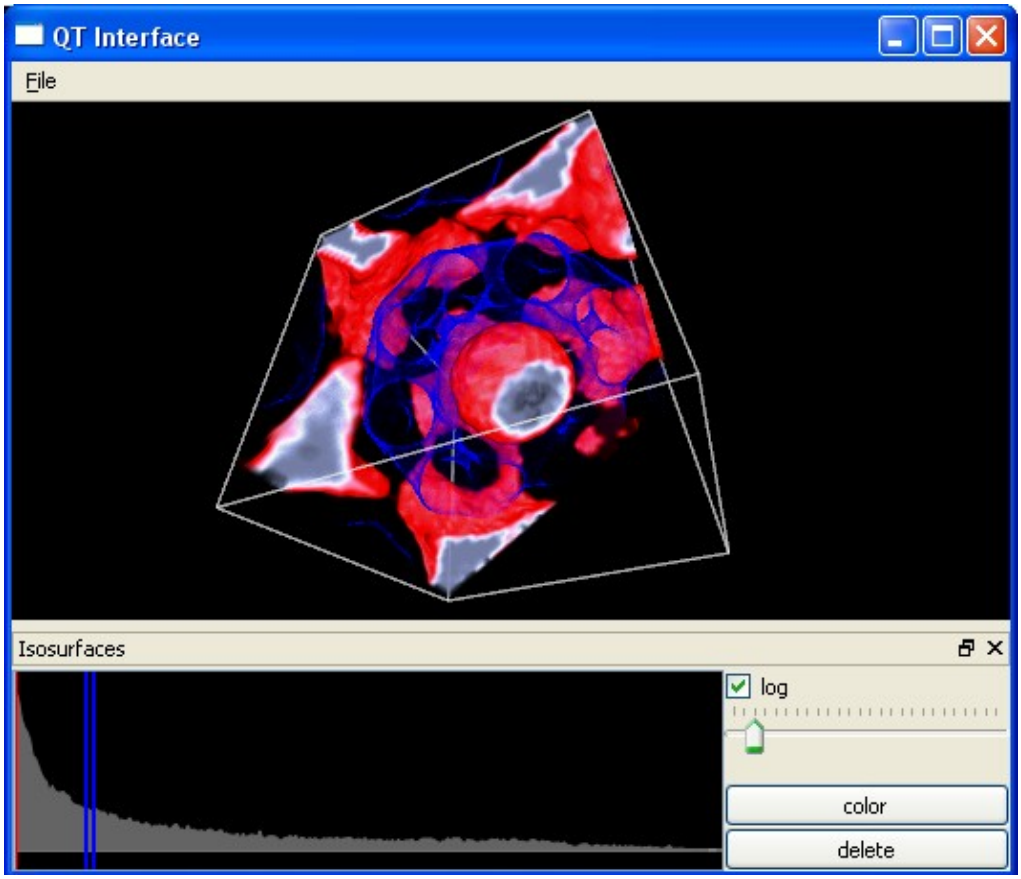
V levé části obrázek ukazuje přechodovou funkci s ručně nakresleným grafem jednotlivých barevných složek a neprůhlednosti. Na pozadí přechodové funkce je vykreslen histogram. Vpravo je pak ukázka vykreslení bloku motoru pomocí této přechodové funkce.

Jak můžeme vidět z histogramu na pozadí, vodorovná osa definuje hodnoty zastoupené v objemových datech, zatímco na vertikální osu se vynášejí jednotlivé barevné složky. Histogram na pozadí v tomto případě zobrazuje poměrné zastoupení

jednotlivých hodnot objemu, to znamená, že nejvíce zastoupená hodnota odpovídá „1“ v histogramu. Zobrazený histogram slouží v zásadě pouze pro lepší orientaci uživatele a usnadňuje výběr důležitých oblastí zájmu. Uvědomme si také fakt, že nakreslený graf barev v konečném důsledku nedefinuje nic jiného než sadu, v našem případě čtyř, tabulek, které přiřazují hodnotám objemu jednotlivé barevné složky, respektive neprůhlednost.

Toto popsané řešení má však dva zásadní nedostatky. Prvním problémem je, že je velice složité dosáhnout požadovaných vizuálních vlastností přímou manipulací s jednotlivými složkami barvy a s neprůhledností. Za druhý nedostatek tohoto uživatelského rozhraní považujeme absenci jednoduché lokální úpravy přechodové funkce, obzvláště v blíže definovaných oblastech zájmu. V situaci, kdy uživatel chce z libovolných důvodů změnit vizuální vlastnosti v konkrétním místě, je nucen překreslit buďto celou přechodovou funkci, nebo významnější část. Tato vada je podtrhnutá poziční nezávislostí, která je v obecném případě základní vlastností přechodových funkcí. V případě ručního kreslení barevných složek tato poziční nezávislost způsobuje o to větší sklíčenost při snaze zvýraznit, či vymezit konkrétní oblast zájmu. Z těchto uvedených důvodů nebyl tento způsob zadávání přechodových funkcí v této práci implementován.

Další způsob, jak zadávat přechodové funkce je definováním izoploch, které patří mezi nejzákladnější metody uživatelských rozhraní. Nejprve si opět ukážeme toto uživatelské rozhraní na příkladu, viz obrázek 3.2.

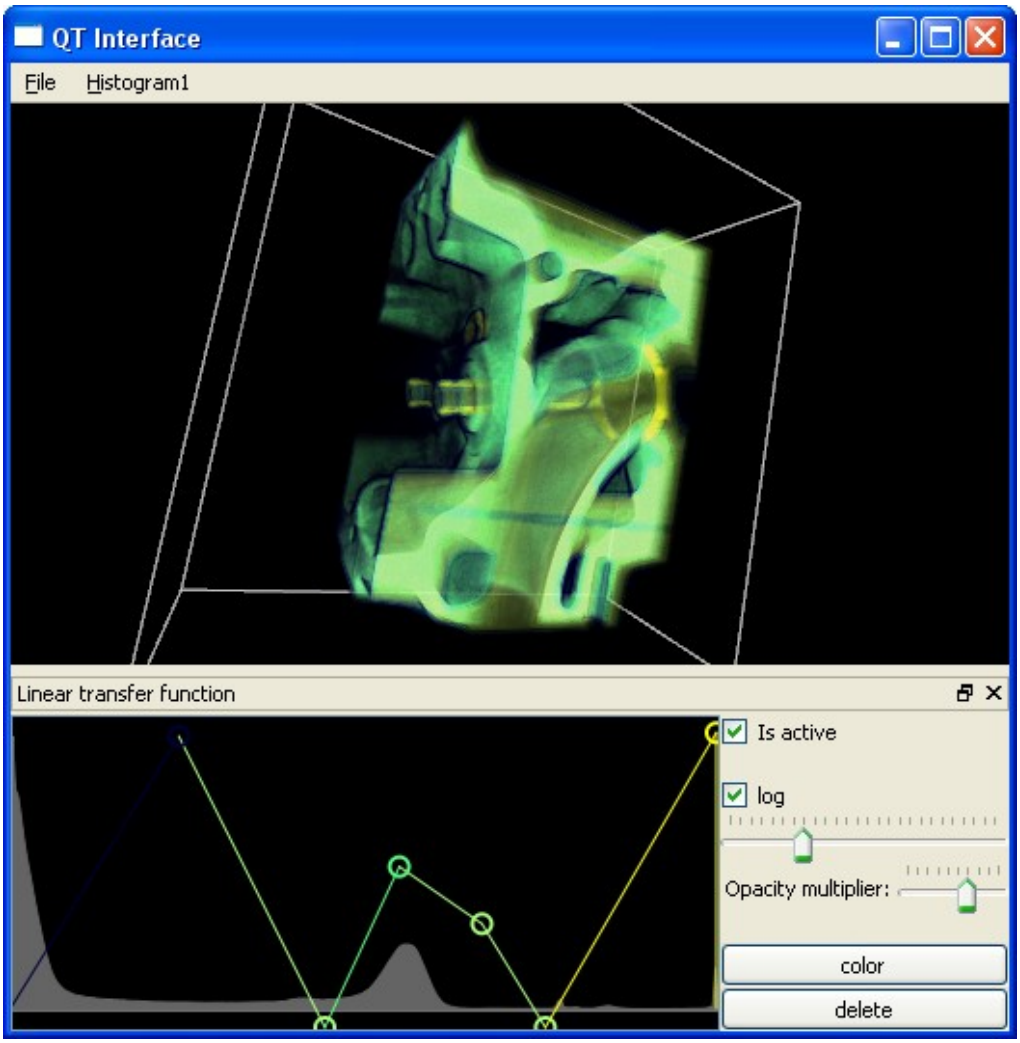


Obrázek 3.2: Ukázka vykreslení histogramu na pozadí s několika izoplochami. Izoplochy jsou primárně určeny pro zvýraznění jedné konkrétní hodnoty. Histogram je logaritmován.

Jak je vidět na obrázku, obvykle i v tomto případě bývá na pozadí zobrazen histogram sloužící uživateli pro lepší orientaci v doméně objemových dat. Problematice vyšetřování izoploch je věnována velká část odstavce 2.5 o datově řízených přechodových funkcí. Nebudeme tedy zabíhat hlouběji, ale využijeme faktu, že zadávání izoploch je jednou z oblastí, kterou tato práce implementuje, a v krátkosti si popíšeme specifika našeho řešení. Prvním a zároveň nejzásadnějším faktorem je okamžité překreslení objemových dat při jakékoli změně v oblasti izoploch. Tato interakce umožňuje uživateli rychle najít požadovanou izpolochu, aniž by musel lépe rozumět objektovým datům, respektive jejich matematickým vlastnostem, jako je například velikost gradientu či jiné. Okamžité překreslování dat je v této práci implementováno pro každou přechodovou funkci. Druhou vlastností našeho rozhraní pro zadávání izoploch je provázání s ostatními jedno-dimenzionálními přechodovými funkcemi, to znamená, že je-li použita některá z níže popisovaných přechodových funkcí a zároveň jsou definovány izoplochy, mají na

zobrazení objemových dat vliv obě nastavení. Logika je taková, že ostatní „více průhledné“ funkce dávají kontext a izoplochy zobrazují detaily objemových dat, neboť jsou dobře lokalizované a lépe vidět.

Další metoda specifikování přechodových funkcí, která je též implementována v této práci, bývá označována pojmem „lineární přechodová funkce“. V tomto případě je uživatel zbaven frustrujícího kreslení jednotlivých složek barvy celé přechodové funkce, ale pouze v oblastech svého zájmu zadává jednotlivé body, kterým přiřazuje konkrétní barevnou hodnotu. Jako v předchozích příkladech se opět používá histogramu vykresleného na pozadí, aby bylo možné identifikovat důležité oblasti. Vše vysvětluje následující ukázka.



Obrázek 3.3: Uživatelské rozhraní pro lineární přechodovou funkci

Pro nastavení této přechodové funkce jsme pro lepší orientaci využili možnosti logaritmování histogramu. Vykreslená objemová data jsme pak zvýraznili pomocí násobitele neprůhlednosti („opacity multiplier“).

Jak je z ukázky zřejmé, jednotlivé uzly lineární přechodové funkce přiřazují barvy (v obecném případě i jiné vizuální vlastnosti) konkrétním hodnotám v objemu. Neprůhlednost je pak určena výškou umístění bodu na ose y. Barevné složky, respektive hodnoty neprůhlednosti v ostatních nespecifikovaných bodech, jsou určeny lineární interpolací mezi zadanými uzly. Odtud též pochází označení této metody.

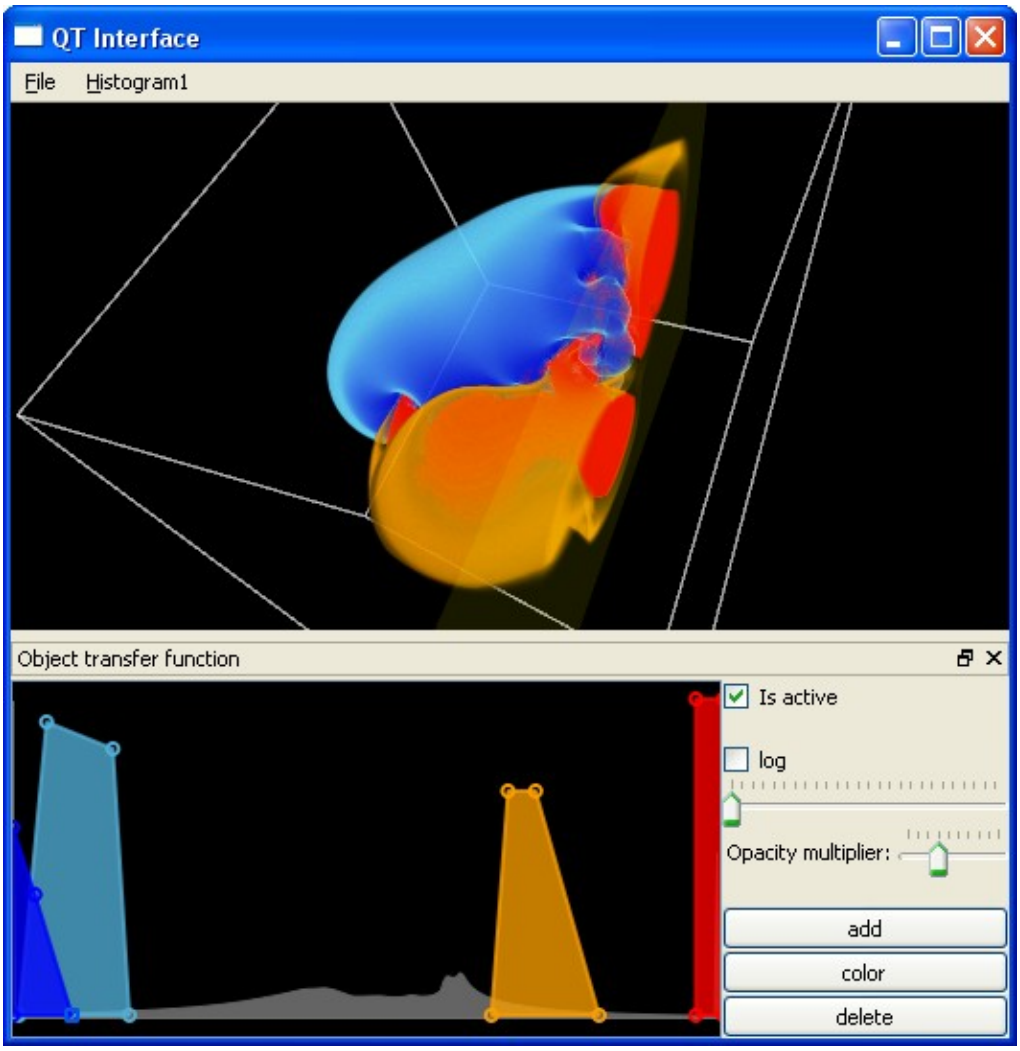
Poslední metodou, kterou si uvedeme v oblasti uživatelských rozhraní pro jednodimenzionální přechodové funkce, je rozhraní pro zadávání tzv. „objektů“. Toto rozhraní implementované v této práci má spíše experimentální charakter. Inspirace pochází z lékařství, konkrétně z používání CT. Přechodové funkce přiřazující barvu a neprůhlednost v oblasti CT využívají absorpce rentgenových paprsků, která může být využita k detekci různých typů materiálů. Tyto různé látky jsou charakterizovány rozdílem hodnot absorpce. Absorpce měřená pomocí CT se obvykle měří v jednotkách Hounsfield (HU)[6]. Různé naměřené hodnoty těchto jednotek lze připsat různým druhům látek nebo tkání. Vybrané hodnoty zobrazuje níže uvedená tabulka, která je výtažkem z tabulky uvedené v [31].

Objekt	denzita HU
vzduch	-1000
tuk	-40 až -120
voda	0
cysta	0 až 15
žluč	5 až 10
slinivka	34 až 45
slezina	45 až 55
kalcifikace	85 až 1000
kompaktní kost	1000 a více

Uživatelské rozhraní založené na zadávání objektů se snaží vyhovět myšlence zachycení složení tkáně v konkrétní lokalitě a umožňuje nastavení barvy a neprůhlednosti pro celé tyto lokality. Například nastavení barvy v okolí 40 HU odpovídá zvýraznění slinivky.

Nyní si ukážeme, jak toto rozhraní může vypadat. Příkladem nám opět bude implementace v této práci.

6 Hounsfieldova jednotka (HU) vyjadřuje stupeň absorpce záření vzhledem k absorpci záření vodou.



Obrázek 3.4: Uživatelské rozhraní pro objektovou přechodovou funkci

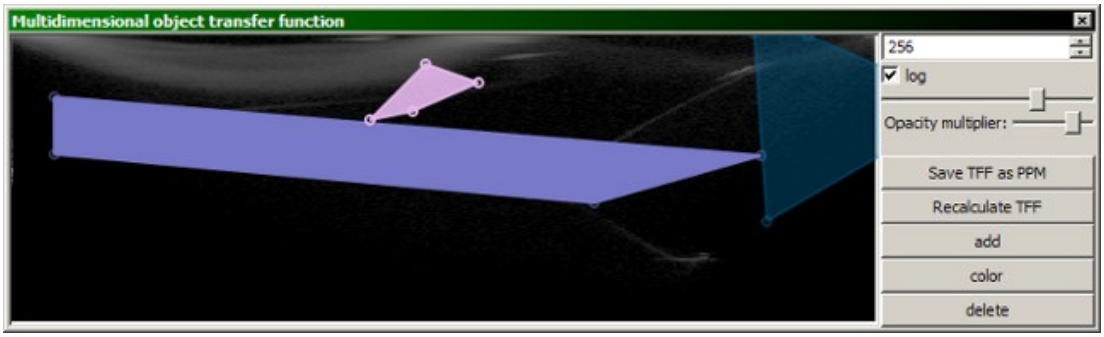
Na tomto obrázku se snažíme ukázat, jak jednotlivé objekty přidané do uživatelského rozhraní mohou reprezentovat určité jednotlivé tkáně. Také jsme při tomto náhledu využili ořezové roviny, která je v objektových datech reprezentována průsvitnou rovinou.

Jak je zobrazeno na obrázku, uživatelské rozhraní opět na pozadí vykresluje histogram s poměrným zastoupením hodnot. Na ose x jsou tedy hodnoty objemových dat a na ose y se vynáší hodnota neprůhlednosti. Přidávané objekty v rozhraní pak mají tvar obecných čtyřúhelníků. Dolní hrana čtyřúhelníku je vždy přichycena k ose x. Všem hodnotám zachycených touto hranou je pak přiřazena barva čtyřúhelníku. Neprůhlednost se přiřazuje interpolací „po“ hranách čtyřúhelníku. Zajímavá situace nastává v případě překrytí více objektů. V takovém případě je pak samozřejmě barva přiřazená objemové hodnotě kombinována ze všech zastoupených barev. Konkrétní

výpočet barvy při překrytí je podrobněji popsaný ve 4 kapitole věnované implementaci.

3.1.2 Uživatelské rozhraní dvou-dimenzionální přechodové funkce

V oblasti dvou-dimenzionálních přechodových funkcí implementuje tato práce pouze jedno uživatelské rozhraní. Toto rozhraní vychází z posledně jmenovaného jedno-dimenzionálního rozhraní pro zadávání „objektů“. Tentokrát si uvedeme pouze obrázek uživatelského rozhraní. Adekvátní obrázek je umístěn v kapitole výsledků.



Obrázek 3.5: Uživatelské rozhraní dvou-dimenzionální přechodové funkce

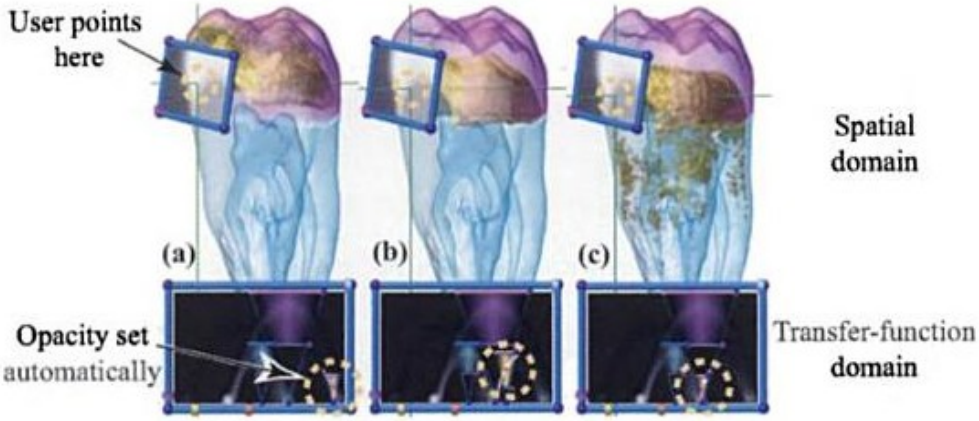
V případě tohoto uživatelského rozhraní je na pozadí zobrazen dvou-dimenzionální histogram. Jedna dimenze pro tento histogram je tvořena naměřenými hodnotami, druhou pak tvoří velikost gradientů. I v tomto případě se jedná o histogram s poměrným zastoupením hodnot. Ve vykresleném histogramu je pak zastoupení vyjádřeno odstínem šedi. Nejčastěji zastoupená kombinace hodnoty a gradientu je zobrazena bílou barvou, zatímco méně časté kombinace se zobrazují různými odstíny šedi až po černou barvu.

Přidávané objekty mohou být v tomto případě obecné polygony. Každému objektu (polygonu) je možné přiřadit hodnotu neprůhlednosti. Všem hodnotám, které jsou pak pokryty přidaným objektem, je přiřazena barva a neprůhlednost polygonu. V případě překrytí více objektů se samozřejmě barva i neprůhlednost kombinuje ze všech zastoupených objektů. Konkrétní výpočet barvy při překrytí je podrobněji popsaný v kapitole 4 věnované implementaci uživatelského rozhraní.

3.2 Dual-domain interaction

Další oblast vývoje uživatelských rozhraní je označována pod anglickým názvem. „dual-domain interaction“. Tento druh rozhraní předpokládá, že uživatel má

intuitivní představu o zobrazovaném objektu. Uživatel pak pracuje přímo se zobrazenými daty i s doménou přechodové funkce. To znamená, že uživatel může obecně určit oblast svého zájmu v zobrazených objemových datech jednoduchým kliknutím.



Obrázek 3.6: Uživatelské rozhraní typu "dual-domain interaction"

Jak ukazuje obrázek, uživatel může vybrat oblast přímo v zobrazených objemových datech, což má okamžitý efekt na doménu přechodových funkcí. Obrázek je převzat z knihy [0].

Kapitola 4

Implementace

Nyní se zaměříme na implementaci této práce. Nejprve si popíšeme vývojové prostředí, ve kterém je aplikace vyvinuta, popíšeme nástroj, na zobrazovaní objemových dat, který byl v této práci využit a v neposlední řadě si ukážeme, jaké požadavky jsou kladeny na bezproblémový chod. Dále popíšeme strukturu aplikace a nakonec se zaměříme na některé detaily implementace.

4.1 Vývojové prostředí

Uživatelské rozhraní vyvinuté v této práci má klasickou grafickou podobu, jak ji známe ze všech moderních aplikací. Největší důraz je kladen na jednoduchost ovládání. Využité jsou vývojové nástroje knihovny „Qt“, kterou vyvíjí norská společnost Trolltech (v době psaní této práce vlastněna společností Nokia).

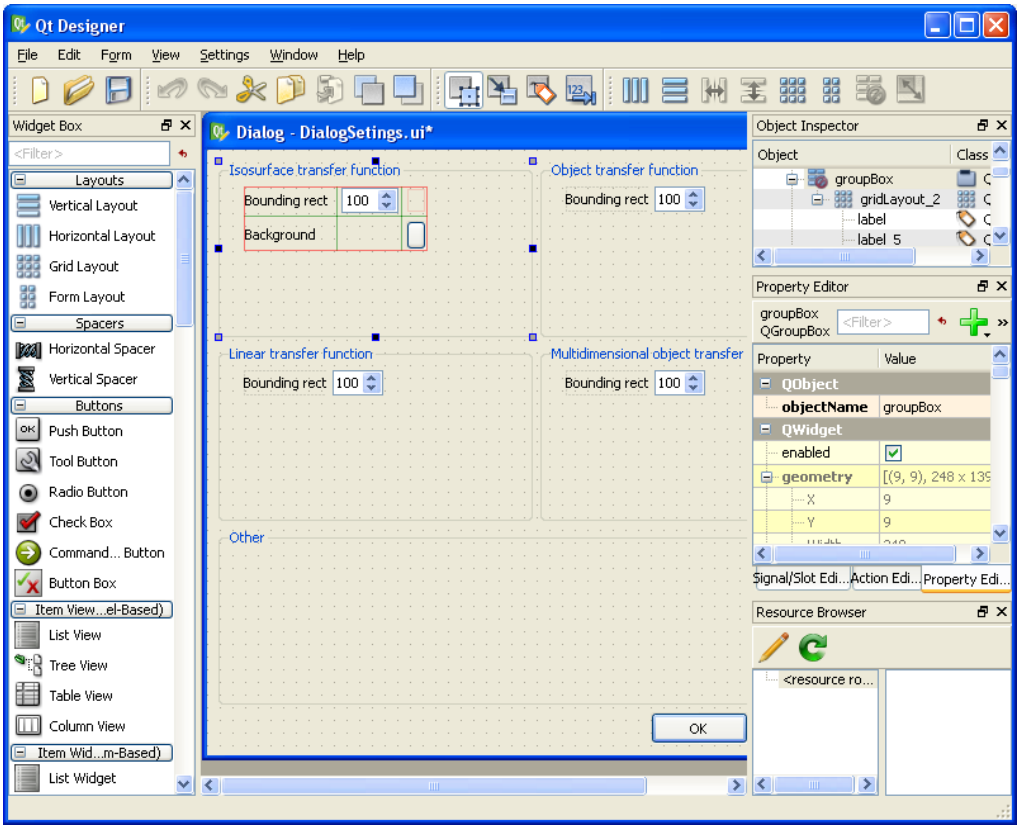
Qt je jedna z nejpopulárnějších multiplatformních knihoven pro vytváření programů s grafickým uživatelským rozhraním. Jedná se o knihovnu programovacího jazyka C++, i když existuje i pro Python (PyQt) a další verze programovacích jazyků. Podporuje lokalizaci aplikací, také SQL, zpracování XML, správu vláken a přístup k souborům. Nejznámější software využívající Qt jsou například: prostředí KDE, webový prohlížeč Opera, Google Earth, Skype a další. V době psaní této práce byla nejnovější verze 4.5 z března 2009. Na této verzi Qt je aplikace též vyvinuta. Jak již bylo řečeno, jedná se o multiplatformní knihovnu dostupnou pro Embedded Linux, Max OS X, Windows, Linux/X11, Windows CE/Mobile. Verze pro platformy Symbian, Maemo se v době psaní této práce teprve připravovaly. Veškeré další informace o knihovně Qt jsou uvedeny na domovských stránkách společnosti Trolltech http://qt.nokia.com/products.

Qt ale není pouhou sadou knihoven funkcí, ale může být plnohodnotným vývojovým nástrojem s vlastním prostředím. Tzv. „Qt Creator“ obsažený v kompletním balíčku, či dostupný zvlášť na stránkách společnosti Trolltech, je multi-platformní integrované vývojové prostředí pro potřeby vývojáře Qt. To zahrnuje C++ editor, nástroje pro správu projektů, vizuální debugger a další.

Přestože je Qt Creator dostatečným nástrojem pro tvorbu Qt projektů, nezdály se nám jeho funkce dostatečné. Proto jsme se v této práci rozhodli využít možnosti integrovat nástroje pro vytváření Qt projektů do prostředí MS Visual Studia 2005. Pro integraci jsme využili nástroj „Visual Studio Add-in“, který je volně dostupný

přímo na stránkách společnosti Trolltech. Po jednoduché instalaci se pak tedy stalo vývojovým prostředím MS Visual Studio 2005.

Dalším vývojovým nástrojem poskytovaným společností Trolltech je tzv. „Qt Designer“. Zmenšený náhled na „Qt Designer“ poskytuje následující obrázek.



Obrázek 4.1: Náhled na "Qt Designer"

Qt Designer slouží pro jednoduché navržení grafické podoby uživatelského rozhraní. Používají se k tomu formuláře s předdefinovanými grafickými komponentami a dialogy, které je možné přesouvat pomocí myši. Každému prvku je pak možné nastavit celou škálu vlastností. Okamžitý náhled pak umožňuje návrháři rychle navrhnout funkční formulář přesně tak, jak zamýšlel. Po nadefinování požadovaného vzhledu se generuje zdrojový kód, který je možné integrovat do vytvořeného projektu, kde je možné jej buď přímo využít, či rozšířit o požadovanou funkcionalitu. Bohužel vzhledem ke specifickým požadavkům na vzhled grafického rozhraní pro jednotlivé přechodové funkce jsme Designer využili pouze pro navržení podoby dialogu nastavení. Kromě této části byla celá aplikace psána ručně ve zdrojovém kódu.

4.2 Požadavky na aplikaci

Uživatelské rozhraní v této práci bylo vyvíjeno a testováno pouze pro Microsoft Windows Vista. Nicméně vzhledem k tomu, že Qt knihovny jsou dostupné pro různé platformy, je aplikace psána tak, aby ji bylo možné snadno přizpůsobit pro různé platformy. Je toho docíleno tak, že každá část aplikace je obalena tzv. „widgety“ z knihovny Qt. Pojem widget pochází z anglického jazyka a v tomto kontextu označuje grafickou komponentu.

Aplikace je též navržena tak, aby ji bylo možné použít jako součást jiných projektů. Konkrétně byla práce vyvíjena s myšlenkou integrovat ji do projektu „BALL“ na kterém spolupracuje Mgr. L. Maršálek, vedoucí této práce. Více o projektu v článcích [32] a [33]. Rovněž napojení na renderer je realizováno přes aplikační rozhraní, tudíž je možné snadno upravit uživatelské rozhraní pro připojení jiných rendererů.

Vzhledem k přístupnosti Qt knihoven pro různé platformy a jejich celkové nenáročnosti na konkrétní softwarové, či hardwarové požadavky vyplývají hlavní omezení na bezproblémový chod aplikace právě z použitého rendereru. Renderer použitý v této práci vyžaduje dynamicky linkované knihovny CUDA (Compute United Device Architecture) pro práci s procesorem grafické karty (GPU).

CUDA je patentová technologie společnosti NVIDIA uvedená na řadě grafických karet GeForce8 a vyšších. Beta verze vývojového prostředí (SDK) a kompilátoru pro CUDA byla zveřejněna v únoru 2007. Kompilátor je založen na mírně modifikované verzi jazyka C. CUDA umožňuje používat výpočetní výkon GPU bez použití grafického API. Umožňuje tedy programátorovi napsat kód, který využije masivní výpočetní výkon grafického procesoru pro jiné než grafické operace. Existují i podobné produkty jako Stream SDK pro AMD FireStream procesor, ale nikdy se dostatečně neuchytily a jsou nyní zastaralé. Na konci roku 2008 se též objevil systém zvaný OpenCL pro programování na heterogenních platformách, včetně paralelní architektury GPU. Použitý renderer je podrobněji popsán v následujícím odstavci.

4.3 Software pro zobrazování objemových dat

Aby bylo možné prezentovat uživatelské rozhraní vyvinuté v této práci, bylo nutné napojit jej na software pro zobrazování objemových dat (dále jen renderer odvozený z anglického výrazu rendering[7]). Na výběr rendereru byly kladeny dva základní požadavky:

- zobrazování objemových dat v reálném čase

7 Rendering je tvorba reálného obrazu na základě počítačového modelu, nejčastěji 3D.

- snadná napojitelnost na uživatelské rozhraní

Z těchto důvodů jsme vybrali renderer naprogramovaný Ivo Pavlíkem [34].

Práce Ivo Pavlíka zkoumá možnosti urychlení vizualizace izoploch založených na metodě vrhání paprsku pomocí oktalového stromu. Pro implementaci na GPU používá hybridní průchod objemovými daty, který kombinuje bezzásobníkový průchod stromem s přímým průchodem mřížkou objemových dat, založeným na algoritmu DDA[8]. Implementace této kombinace přinesla podle autora až 3,5-násobné zrychlení oproti původnímu DDA algoritmu. Zobrazování v reálném čase je tedy jednou z předností implementace tohoto rendereru a splňuje tedy první základní požadavek použití v této práci.

Implementace rendereru Ivo Pavlíka byla také již v základu vyvíjena pro aplikaci WisS od Vladimíra Hrinčára [35]. Proto se vyznačuje dobrým aplikačním rozhraním. Bylo tedy snadné napojit uživatelské rozhraní na požadované zobrazovací funkce. Vybraný renderer dále umožňuje kreslení tří základních typů objemových dat:

- vykreslování jednoduchých průhledných izoploch
- vykreslování neprůhledných stínovaných izoploch
- přímé vykreslování objemu (direct volume rendering)

V této práci je využita pouze poslední zmiňovaná možnost a to přímé vykreslování objemu.

Použitý renderer též poskytuje aplikační rozhraní pro návrat jedno-dimenzionálního histogramu, uživatelské rozhraní z této práce tedy pouze žádá o histogram použitý renderer a nepotřebuje tak mít žádné informace o objektových datech.

Nevýhodou použitého rendereru byla absence podpory dvou-dimenzionálních přechodových funkcí. Respektive konkrétně v našem případě chyběly prostředky pro implementaci dvou-dimenzionální objektové přechodové funkce z odstavce 3.1.2. Bylo tedy nutné renderer rozšířit. Implementace rozšíření je detailněji popsána v odstavci 4.4 Rozšíření rendereru.

4.4 Rozšíření rendereru

Renderer bylo potřeba rozšířit o podporu dvou-dimenzionálních přechodových funkcí. V prvním kroku bylo třeba vytvořit dvou-dimenzionální

8 DDA algoritmus: Jedná se o přírůstkový algoritmus pro výpočet bodů úsečky. Ve směru jedné osy s větším přírůstkem inkrementuje o krok 1 a ve směru druhé osy o krok menší než 1 podle poměru délky úsečky ve směru osy x a ve směru osy y.

histogram. Vzhledem k tomu, že vývoj rendereru nebyl primárním cílem této práce, implementovali jsme rozšíření co nejjednodušším a nejrychlejším způsobem, bez ohledu na časovou složitost. Proto se histogram vytváří ve dvou krocích:

- V prvním kroku jsou procházena celá objemová data a jsou počítány jednotlivé gradienty. Cílem tohoto průchodu je určit nejmenší a největší gradient, který je v dalším kroku využit k tomu, aby bylo možné vytvořit dvou-dimenzionální histogram s poměrným zastoupením hodnot. Bez nejmenšího a největšího gradientu by nebylo možné určit, do které části dvou-dimenzionálního histogramu spočítaná hodnota gradientu patří.

- V druhém kroku jsou gradienty počítány znovu, neboť časové nároky na výpočet gradientu nejsou příliš náročné, zatímco paměťové nároky na jejich uložení by nebyly zanedbatelné. V průběhu tohoto kroku se tedy příspěvky jednotlivých gradientů kategorizují do sekcí mezi minimálním a maximálním gradientem dle zadané výšky dvou-dimenzionálního histogramu. Velikost jedné sekce je určena pomocí vzorce ((max. gradient – min. gradient) / výška histogramu). Dle velikosti gradientu se zjistí, do které sekce tento gradient přispívá, a zvýší se počet zastoupených gradientů v této sekci o jedna.

Pro správně vytvoření histogramu bylo nutné upravit načítání dat, neboť námi použitý způsob vyžaduje mít v jednom okamžiku uložena celá objemová data. To v případě stavění oktalového stromu nebylo potřeba.

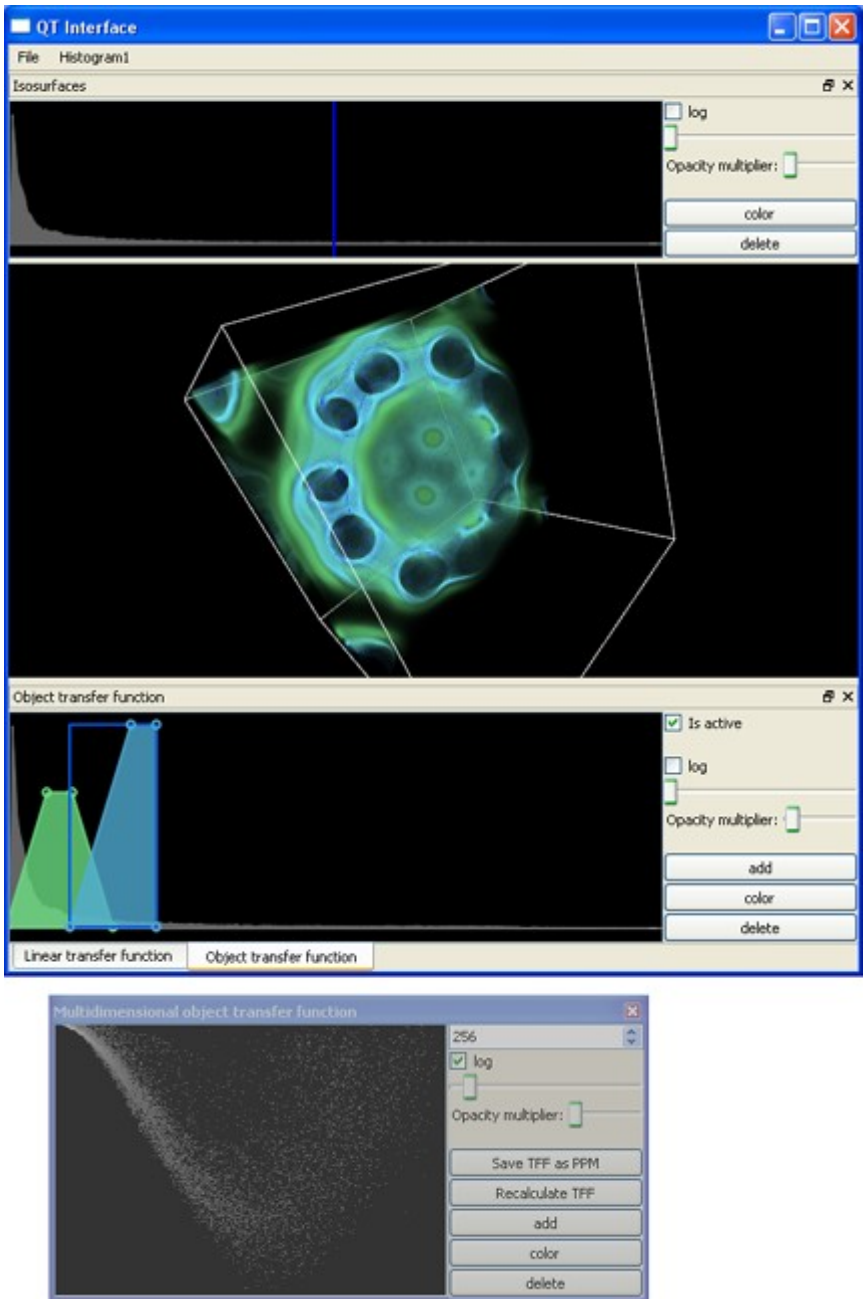
Krom tvorby dvou-dimenzionálního histogramu bylo ještě nutné upravit funkci pro nastavení přechodové funkce v rendereru. Ta byla rozšířena o parametry určující šířku a výšku histogramu. Výška histogramu je pak volitelným parametrem a není-li vyplněna, je výchozí hodnota nastavena na „1“, která určuje, že se jedná o jedno-dimenzionální přechodovou funkci.

4.5 Struktura aplikace

Uživatelské rozhraní je rozděleno dle jednotlivých implementovaných přechodových funkcí. Jednotlivé přechodové funkce jsou popsány v kapitole 3 o uživatelských rozhraních. Pro přehlednost si zde implementované přechodové funkce vypíšeme. Jedná se o:

- izoplochy
- lineární přechodová funkce
- objektová přechodová funkce
- dvou-dimenzionální objektová přechodová funkce

K jednotlivým oknům s přechodovými funkcemi navíc přibývá centrální okno aplikace pro zobrazovaná data. Příklad rozdělení ukazuje obrázek 4.2.



Obrázek 4.2: Příklad rozdělení oken aplikace

V horní části je dokové okno uvnitř hlavního okna aplikace, následuje centrální okno s vykreslenými daty, poté dvě přechodové funkce v jedné dokové oblasti a nakonec lehce průhledné samostatné okno s multi-dimenzionální přechodovou funkcí.

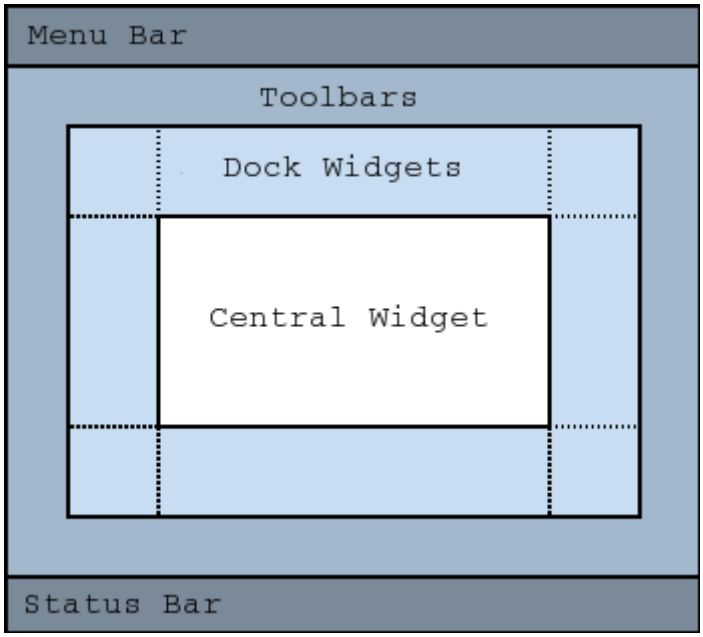
Pro uživatelský komfort byla aplikace navržena tak, aby s jednotlivými přechodovými funkcemi šlo manipulovat. Je tedy možné je zavírat, přesouvat do jiných částí aplikace, případně je ponechat jako „plovoucí“ objekty. Konkrétní

vzhled uživatelského rozhraní není tedy pevně dán a může se od obrázku lišit. Jediné, co vždy zůstává, je centrální okno se zobrazovanými daty.

4.6 Struktura kódu

Již dříve jsme zmínili, že aplikace je psána tak, aby ji bylo možné snadno použít na různých platformách. Tato vlastnost je zajištěna tak, že každá třída aplikace je potomkem adekvátní třídy předdefinované v některé z knihoven Qt. Proto dříve, než si popíšeme strukturu kódu jednotlivých přechodových funkcí, stručně si představíme základní třídy Qt, z kterých námi implementované třídy dědí. Kompletní dokumentaci je pak možné nalézt na stránkách společnosti Trolltech http://doc.trolltech.com/.

QMainWindow
Třída QMainWindow se používá jako hlavní okno aplikace. Toto okno má své vlastní uspořádání, které umožňuje přidávat klasické prvky pro dnešní uživatelská rozhraní, jako je rozbalovací nabídka („menu bar“), panel nástrojů („toolbars“), či stavový řádek („status bar“). Konkrétní uspořádání ukazuje obrázek 4.3.



Obrázek 4.3: Rozvržení oblastí hlavního okna aplikace

Obrázek též zobrazuje tzv. „dokové“ oblasti, které slouží pro umístění grafických komponent implementovaných třídou QDockWidget (viz níže).

QDockWidget
Tato třída implementuje grafickou komponentu (dále jen „widget“), která může být součástí nadřazeného okna, nebo se může chovat jako samostatné dialogové okno aplikace. Nadřazené okno implementuje třída QMainWindow popsaná v předchozím odstavci.
V případě vykreslení uvnitř nadřazeného widgetu se jedná o tzv. „dokové“ objekty, které se vykreslují v již popisovaných „dokových“ oblastech. Dokové oblasti jsou umístěné kolem centrálního widgetu, viz obrázek 4.3 výše.

Doková okna je možné v rámci aplikace přesouvat mezi různými dokovými oblastmi, je možné zobrazit více dokových oken v rámci jedné dokové oblasti pomocí záložek, minimalizovat je, či je dokonce úplně zavřít. Ukázka jedné z variant rozvržení je na výše obrázku 4.2 z našeho uživatelského rozhraní.

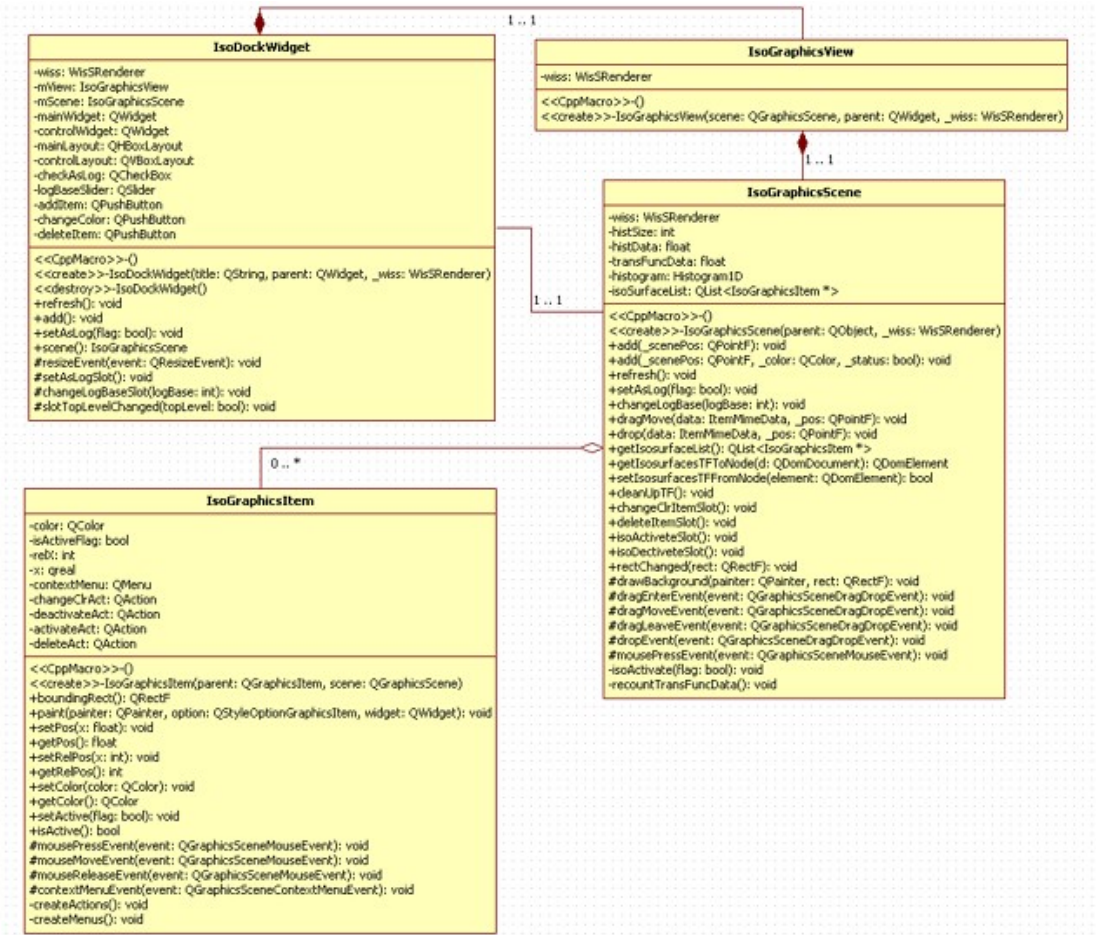
QGraphicsView
Tato třída implementuje widget, který se používá pro zobrazování obsahu třídy QGraphicsScene (viz níže). Widget může být použit pro zobrazení celé scény, nebo pouze její části. Též umožňuje skrolování scény, vycentrování scény, vycentrování k určitému objektu uvnitř scény apod. QGraphicsView nabízí více zajímavých funkcí, jako je nastavení transformační matice, která se často používá pro přiblížení, oddálení či rotaci scény. Pro naše potřeby však nebyly pokročilejší funkce třídy QGraphicsView potřebné, vystačili jsme s možnostmi třídy QGraphicsScene, které jsou popsány samostatně. Podrobný popis QGraphicsView je opět možné nalézt v dokumentaci Qt.

QGraphicsScene
Třída QGraphicsScene slouží pro správu velkého množství grafických prvků (QGraphicsItem). Poskytuje plochu pro vizualizaci dvou-dimenzionálních grafických předmětů, jako jsou čáry, obdélníky, různé polygony a další. Též umožňuje zobrazovat vlastní prvky vytvořené pomocí třídy QGraphicsItem (viz níže). Třída QGraphicsScene nemá vlastní vzhled, pouze spravuje vložené prvky. Proto se třída používá v kombinaci s třídou QGraphicsView. Správa prvků uvnitř scény umožňuje efektivní zjišťování, zda jsou položky viditelné, kde jsou umístěné a další. Samozřejmostí jsou také metody pro dynamické přidávání, odebírání nebo pohyb jednotlivých prvků.

QGraphicsItem
QGraphicsItem je základní třídou pro všechny grafické elementy vkládané do widgetu QGraphicsScene. Pro jednoduché použití existuje v knihovnách Qt několik již předdefinovaných tříd, dědících z QGraphicsItem, které implementují konkrétní grafické elementy. Například se může jednat o elipsu, přímku, polygon, ale i několik další elementů.
Hlavní výhodou QGraphicsItem ovšem zůstává možnost vytváření vlastních grafických elementů. Nejedná se ovšem o pouhý geometrický tvar. V rámci QGraphicsItem je možné implementovat způsob vykreslení, detekci kolize či události elementu a další. Dalším důležitým faktem je, že element QGraphicsItem

může obsahovat další elementy a sám může být opět obsažen v jiném elementu. Každý element může tedy mít svého rodiče a své potomky.
Třída QGraphicsItem je pravděpodobně tou nejdůležitější pro implementaci této práce.

Již z popisu použitých tříd z knihoven Qt může být struktura kódu jednotlivých přechodových funkcí zřejmá. Pro lepší představu ukazuje konkrétní uspořádání UML diagram uživatelského rozhraní pro definování izoploch.



Obrázek 4.4: UML diagram okna jedné přechodové funkce

U ostatních přechodových funkcích jsme dodrželi stejnou strukturu zanoření tříd. Rozdílná je pouze různá složitost prvků přidávaných do grafické scény. Různá složitost grafických prvků však přesně odpovídá tomu, že jednotlivé prvky mohou obsahovat další prvky, jak je uvedeno v popisu třídy QGraphicsItem. Přesto si pro jednotlivé přechodové funkce popíšeme vytvářené objekty v grafické scéně.

Izoplochy
V případě izoploch se do grafické scény přidává jednoduchá svislá úsečka. Pro potřeby přechodové funkce se uchovává pouze x-ová souřadnice v rámci scény,

relativní pozice vůči histogramu a barva. Neprůhlednost je v případě izoploch statická. Jednotlivé izoplochy je též možné aktivovat či deaktivovat.

Lineární přechodová funkce
V případě lineární přechodové funkce odpovídají přidávané elementy bodům, respektive kružnicím. Uchovává se barva a pozice [x, y] v rámci scény. X-ová pozice pak odpovídá relativní pozici v histogramu, zatímco y-ová odpovídá neprůhlednosti. Pro vykreslení lomené čáry mezi vloženými body je jednoduše přetížena metoda vykreslování scény. Výpočet barev a neprůhlednosti je uveden v odstavci 4.7 o implementačních detailech.

Objektová přechodová funkce
Přidávané elementy odpovídají obecným čtyřúhelníkům. Ke každému čtyřúhelníku jsou přiřazeny jednotlivé vrcholy, které odpovídají samostatným elementům v grafické scéně. Díky tomu je možné pracovat jak s celým čtyřúhelníkem, tak s jednotlivými body. V rámci jednoho elementu jsou tedy uchovány čtyři vrcholy a barva. Jednotlivé body pak nesou informaci o pozici v rámci scény a též barvu. Neprůhlednost opět odpovídá výšce bodu ve scéně. Přepočet barvy a neprůhlednosti pro jednotlivé hodnoty histogramu je opět popsán zvlášť v odstavci 4.7 o implementačních detailech.

Dvou-dimenzionální objektová přechodová funkce
Prvky přidávané do grafické scény jsou v tomto případě zobecněním prvků z objektové přechodové funkce. Jeden prvek tedy v tomto případě odpovídá polygonu. Vrcholy polygonu taktéž odpovídají prvkům přidaným do grafické scény. Každý element nese informaci o barvě a neprůhlednosti. Vrcholy pak obsahují též informaci o barvě a pozici v rámci scény. Výpočet údajů pro jednotlivé hodnoty histogramu též popisují implementační detaily.

4.7 Implementační detaily

V předchozím odstavci jsme popsali strukturu kódu aplikace. Často jsme poukazovali na podrobnější popis implementace výpočtu přechodových funkcí. Nyní si tedy tyto výpočty popíšeme pro všechny jednotlivé přechodové funkce implementované v této práci.

Izoplochy
V případě izoploch je generování výsledné přechodové funkce velice jednoduché, neboť se pouze přiřazuje barva izoplochy té hodnotě objemových dat z histogramu, které je izoplocha přidělena. Neprůhlednost je nastavena konstantně.

Lineární přechodová funkce
Tato přechodová funkce přiřazuje barevné hodnoty celému rozsahu objemových dat. Barva přiřazovaná hodnotě v histogramu je určena lineární interpolací mezi barvami vložených bodů. Neprůhlednost je pak zajištěna y-ovou hodnotou úsečky v daném bodě.

Objektová přechodová funkce

Pro objektovou přechodovou funkci existují dva oddělené případy, kdy buď nedochází, nebo dochází k překrytí vložených objektů.

První jednodušší případ si lze představit podobně jako u lineární přechodové funkce, a to že barva bude určena barvou objektu a neprůhlednost se určí lineární interpolací y-ových hodnot po hranách objektu.

V případě překrytí objektů je nutné přepočítat barvu přes všechny zastoupené objekty, a to v zachovaném poměru dle jednotlivých neprůhledností v počítaném bodě.

Implementace se opírá o předpoklad, že histogram má přijatelnou velikost a že přidaných objektů bude vždy relativně malé množství. Předpoklad vychází ze způsobu používání objektových přechodových funkcí. Díky této podmínce je možné procházet histogram po jednotlivých bodech, a pro každý kontrolovaný bod histogramu procházet seznam přidaných objektů. Z objektů, které pak pokrývají kontrolovaný bod, se vytváří pomocný seznam, který následně slouží pro výpočet barevné hodnoty a neprůhlednosti přechodové funkce v kontrolovaném bodě.

Pomocný seznam obsahuje barvu objektu a odpovídající neprůhlednost, která je pro daný bod spočítána pomocí lineární interpolace po hranách objektu. Dalším krokem výpočtu je průchod pomocným seznamem, při kterém se dle odpovídající neprůhlednosti vypočítává poměrné zastoupení barvy vůči neprůhlednosti maximální. Tyto přírůstky společně s přírůstkem objektu s maximální neprůhledností vytvářejí výslednou barvu. Každý objekt tak přispívá do výsledné barvy takovou částí své barvy, která odpovídá poměru jeho neprůhlednost vůči maximální neprůhlednosti ze zastoupených objektů. Pro lepší pochopení může pomoci následně uvedený výňatek ze zdrojového kódu s komentáři.

```
void ObjGraphicsScene::interpolationByMaxOpacity(QList<QColor> colorList, int
maxOpacityIndex, int histPos)
{
  float maxR = colorList[maxOpacityIndex].redF();
  float maxG = colorList[maxOpacityIndex].greenF();
  float maxB = colorList[maxOpacityIndex].blueF();
  float maxO = colorList[maxOpacityIndex].alphaF();

  //Pro jednoduchý průchod seznamem barev odstraníme maximální hodnotu
  colorList.removeAt(maxOpacityIndex);

  float rateOpacity = 0.0f;
  float totalRate = 0.0f;
  float totalR = 0.0f;
  float totalG = 0.0f;
  float totalB = 0.0f;

  //Sečteme zastoupení všech barev s menší opacitou.
  for (int i=0; i<colorList.size(); ++i)
  {
            rateOpacity = maxO * colorList[i].alphaF();
            totalR = totalR + (rateOpacity * colorList[i].redF());
            totalG = totalG + (rateOpacity * colorList[i].greenF());
            totalB = totalB + (rateOpacity * colorList[i].blueF());
```

```
            //Nasčítává se poměrné zastoupení všech barev s menší opacitou
            totalRate = totalRate + rateOpacity;
        }

        //Nakonec se připočítá zastoupení barvy s největší opacitou
        //A to její poměrné zastoupení vůči ostatním barvám
        rateOpacity = (maxO - totalRate);

        totalR = totalR + (rateOpacity * maxR);
        totalG = totalG + (rateOpacity * maxG);
        totalB = totalB + (rateOpacity * maxB);

        //Vypočítané hodnoty se uloží do přechodové funkce
        this->transFuncData[histPos * 4 + 0] = totalR;
        this->transFuncData[histPos * 4 + 1] = totalG;
        this->transFuncData[histPos * 4 + 2] = totalB;
        this->transFuncData[histPos * 4 + 3] = this->opacityMultiplier * maxO;
    }
```

Dvou-dimenzionální objektová přechodová funkce

Vzhledem k tomu, že dvou-dimenzionální objektová přechodová funkce je odvozena ze své jedno-dimenzionální verze, je výpočet přechodové funkce takřka totožný. Rozdílné je v první řadě zjišťování neprůhlednosti, která je v případě dvou-dimenzionálních objektů stejná pro celý objekt, a není tedy nutné ji lineárně interpolovat po hranách objektu.

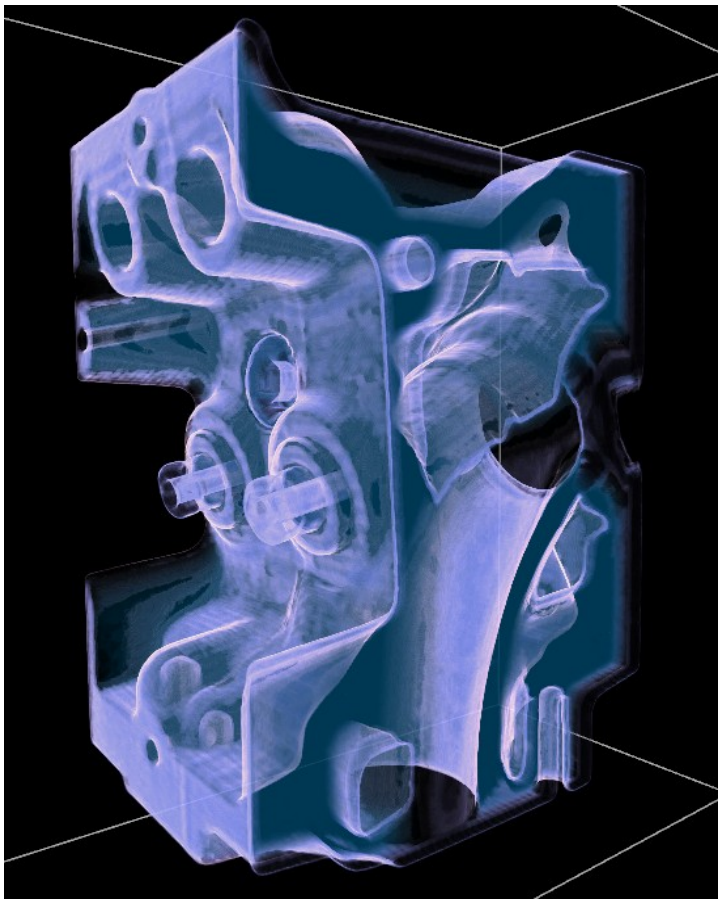
Další rozdíl vznikl z důvodu výkonosti. Procházení dvou-dimenzionálního diagramu by bylo příliš časově náročné, proto se u této přechodové funkce prochází vždy ta část histogramu, ve které došlo ke změně přidaných objektů. Například při změně barvy polygonu se prochází nejmenší obdélník opisující tento polygon.

Kapitola 5

Výsledky

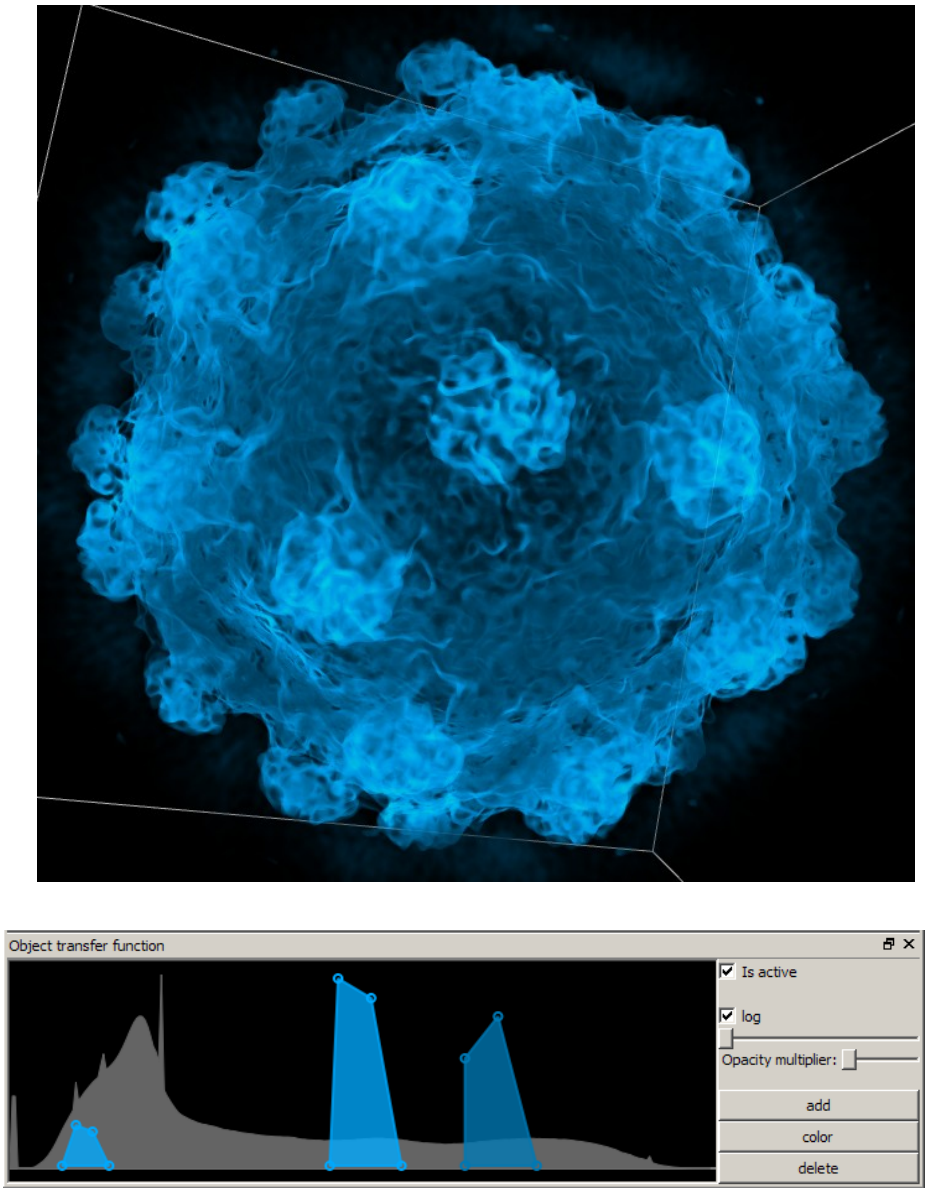
V této předposlední kapitole si ukážeme příklady vykreslení, kterých je možné dosáhnout pomocí uživatelského rozhraní vyvinutého v této práci. Základem tedy budou ilustrace různých objemových dat, na které aplikujeme námi vytvořené přechodové funkce. Abychom také poukázali na přínos našeho uživatelského rozhraní, provedeme srovnání s obyčejným uživatelským rozhraním, které umožňuje pouhé ruční kreslení přechodových funkcí.

První ilustrace, zobrazující blok motoru, je vytvořena pomocí dvou-dimenzionální přechodové funkce, kterou jsme použili jako ukázku pro popis uživatelského rozhraní v kapitole 3.1.2. Výsledkem je velice realistické znázornění bloku motoru:

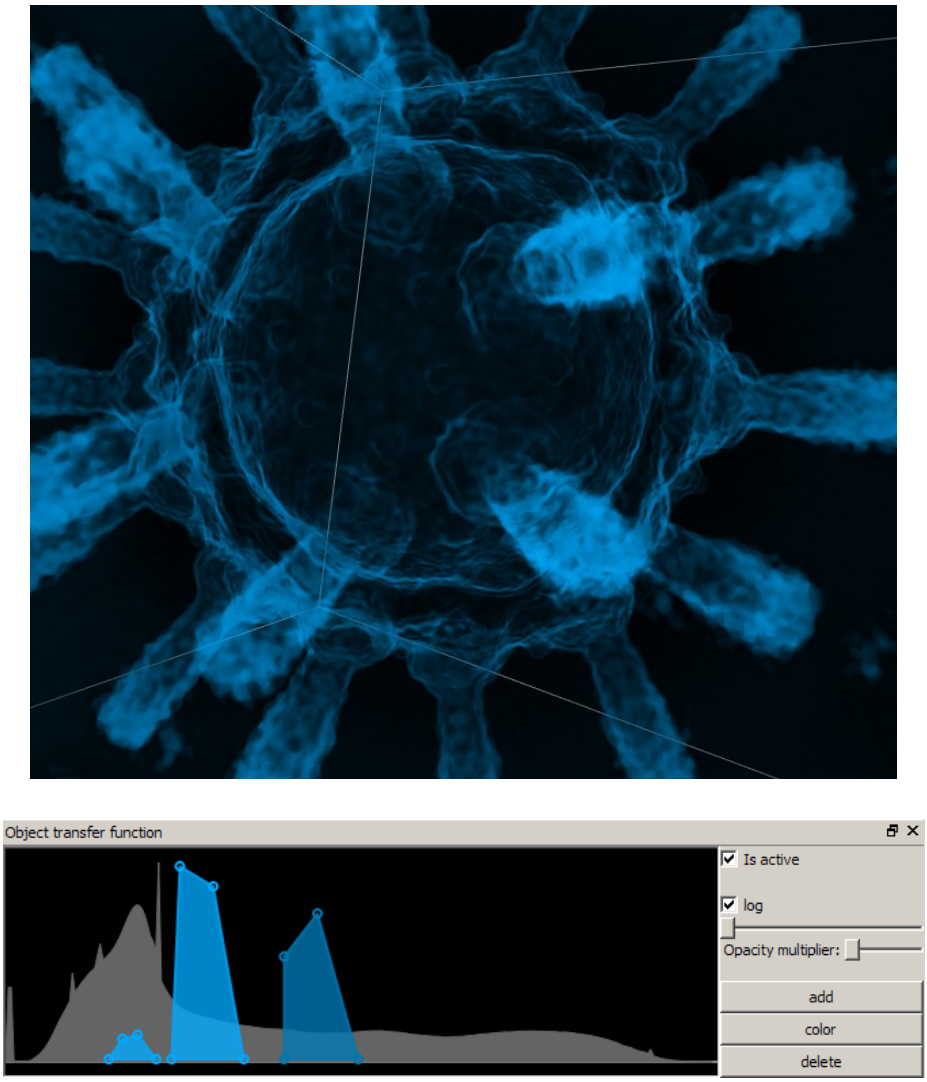


Obrázek 5.1: Blok motoru

Na následujících ilustracích viru chceme ukázat výhody, které poskytuje objektová přechodová funkce. Z těchto důvodů jsme na objemová data viru aplikovali dvě objemové přechodové funkce. Každá z nich poskytuje zcela odlišný náhled na zobrazovaný vir, přitom po vytvoření přechodové funkce bylo zapotřebí pouze minimální změny. Stačilo pouze posunout celé objekty v rámci histogramu. V případě ručně kreslené přechodové funkce by pro takovouto úpravu bylo nutné překreslit celou přechodovou funkci.



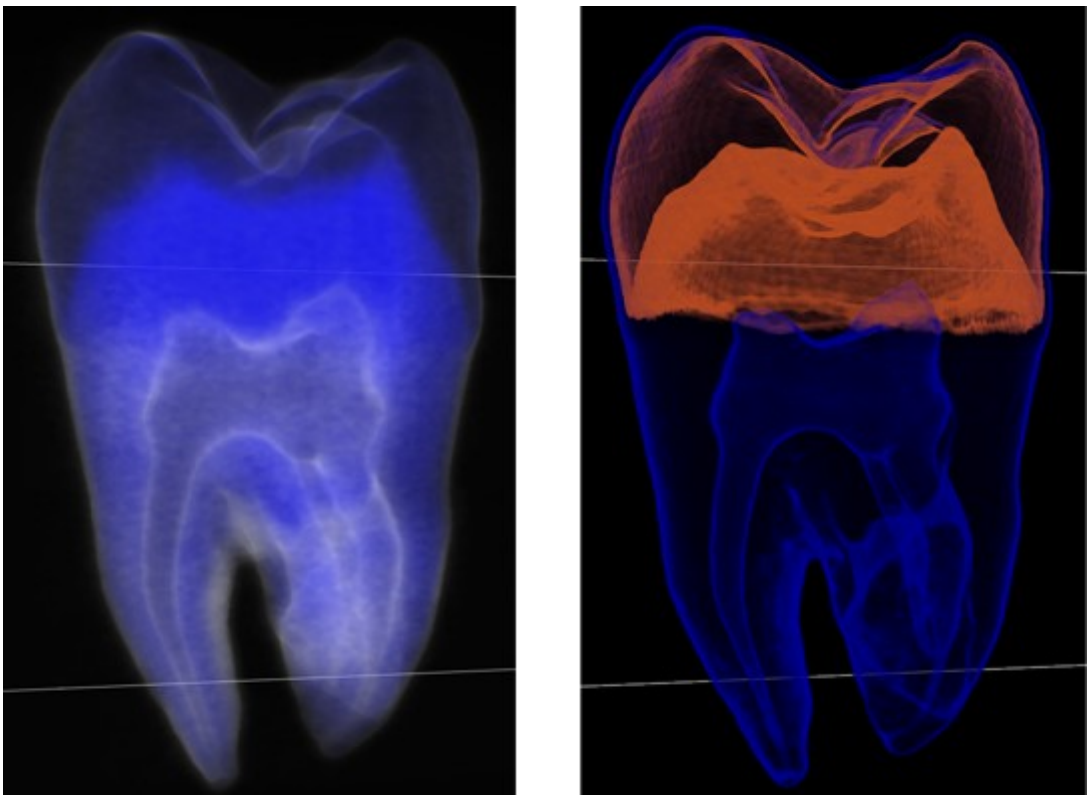
Obrázek 5.2: Zobrazení viru pomocí objektové přechodové funkce



Obrázek 5.3: Zobrazení viru po přesunutí objektů přechodové funkce

Kapitola 5: Výsledky

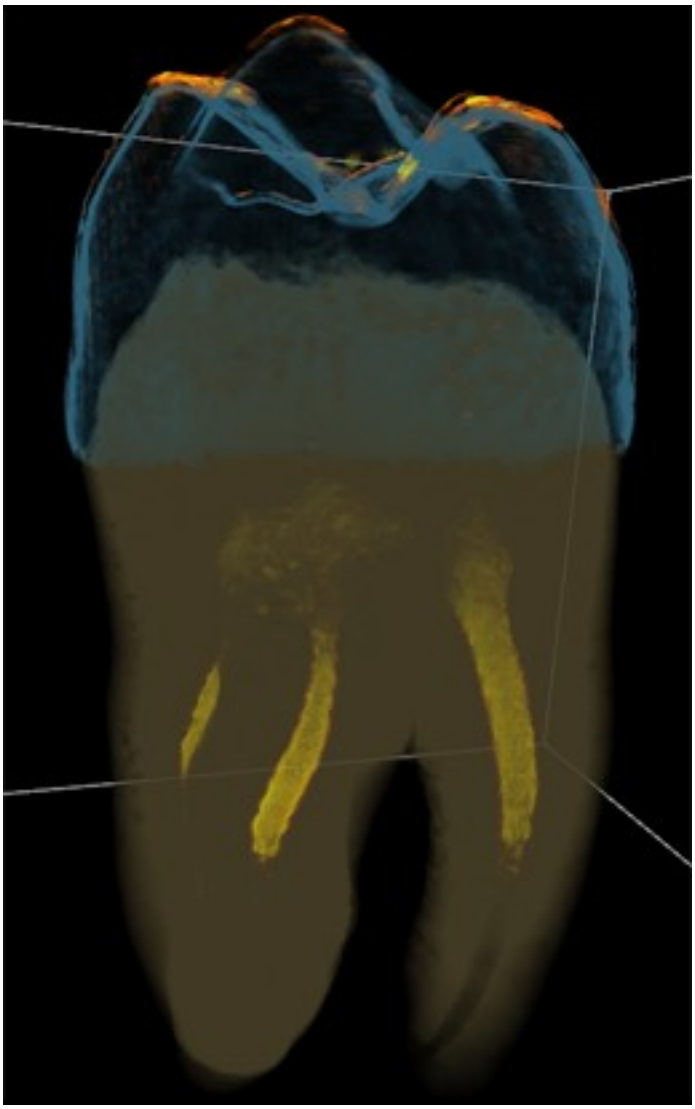
Na závěr si pomocí CT zubu vysvětlíme důvod použití dvou-dimenzionální přechodové funkce. Pokusili jsme se vykreslit obrázky zubu jak pomocí lineární, tak i objektové přechodové funkce, viz níže. V obou případech ale není možné vykreslit tělo zubu bez zobrazené skloviny, která se vždy alespoň obrysově zobrazí. Pomocí objektové přechodové funkce se sice podařilo vytvořit obrázek, který dobře zvýrazňuje hranici těla zubu, není však možné odříznout vrchní část úplně!



Obrázek 5.4: Vykreslení zubu pomocí jedno-dimenzionální přechodové funkce

Vlevo vidíme vykreslení pomocí lineární přechodové funkce, zatímco vpravo je zobrazena ilustrace pomocí objektové přechodové funkce.

Aby bylo možné vykreslit tělo zubu a sklovinu zvlášť, je nutné použít více-dimenzionální přechodovou funkci. V našem případě se jedná o implementovanou dvou-dimenzionální objektovou přechodovou funkci. Obrázek níže zobrazuje tentýž zub taktéž s vykreslenou sklovinou. Je ale důležité si uvědomit, že sklovina je v tomto případě vykreslena odlišnou barvou, neboť odpovídá v přechodové funkci samostatnému objektu! Tento objekt je samozřejmě možné odebrat a zobrazit tak tělo zubu samostatně.



Obrázek 5.5: Vykreslení zubu dvou-dimenzionální objektovou přechodovou funkcí

Kapitola 6

Závěr

Motivací pro tuto práci byla potřeba vytvoření jednoduchého a uživatelsky příjemného uživatelského rozhraní, které by usnadnilo tvorbu přechodových funkcí, která je v běžných případech složitým a zdlouhavým procesem. Cílem však nebylo pouhé naprogramování uživatelského rozhraní a popsání jeho funkcionalit, ale snahou bylo celkové uvedení čtenáře do problematiky zobrazování objemových dat a hlavně do celkové teorie přechodových funkcí, které jsou nutné pro mapování vstupních hodnot do zobrazitelných veličin.

Celou práci shrneme v těchto bodech:

- Úvod je věnován základním pojmům z oblasti objemových dat, jejichž vizualizací je tato práce inspirována. Snahou v této části je zdůraznit důležitost přechodových funkcí.

- Další část je věnována samotné teorii přechodových funkcí. Je uvedeno rozdělení přechodových funkcí dle různých kritérií a způsoby hledání vhodné přechodové funkce. Znovu je zde vysvětlena jejich nepostradatelnost při zobrazování objemových dat. V téže kapitole je také vyzdvihnuta potřeba dobrého uživatelského rozhraní!

- Třetí kapitola se již plně věnuje samotným uživatelským rozhraním pro zadávání přechodových funkcí. Snahou je zaměřit se na ty metody vytváření přechodových funkcí, které jsou v této práci implementovány, a zdůraznit jejich výhody. Závěr kapitoly je věnován nejnovějšímu vývoji v této oblasti.

- Po popsání jednotlivých uživatelských rozhraní následuje část věnovaná implementaci této práce. Kapitola obsahuje popis vývojového prostředí, renderer, který je v této práci použit, i napojení na něj pomocí aplikačního rozhraní. Větší část kapitoly je však věnována struktuře aplikace a kódu. Také je nastíněna možnost použití uživatelského rozhraní na různých platformách a pro různé aplikace. Konkrétně je práce navržena pro zaintegrování do projektu „BALL“.

- Poslední kapitola je věnována výsledkům této práce. V našem případě se převážně jedná o různé ukázky vykreslení objemových dat, kterých je možné pomocí vyvinutého uživatelského rozhraní dosáhnout. Tato kapitola rovněž

poukazuje na výhody, které přináší námi implementované uživatelské rozhraní, oproti jednodušším a méně sofistikovaným řešením.

Při tvorbě ilustrací zobrazených v kapitole výsledků jsme uživatelské rozhraní důkladně odzkoušeli. Závěrem tedy můžeme říct, že se podařilo vytvořit uživatelské rozhraní, které umožňuje nalézt vhodnou přechodovou funkci bez hlubších znalostí o zobrazovaném objektu. Proces hledání je oproti méně sofistikovaným řešením rychlou záležitostí! Díky grafickému znázornění přechodových funkcí a klasickému ovládání pomocí myši je práce s uživatelským rozhraním jednoduchou a pohodlnou záležitostí.

6.1 Budoucí rozšíření

Automatické, respektive polo-automatické metody vytváření přechodových funkcí v oblasti vizualizace objemových dat jsou v dnešní době stále velkou výzvou. Dalšího rozvoj uživatelského rozhraní by se mohl ubírat právě směrem polo-automatických metod, které by vytvářeli počáteční přechodovou funkci. Uživatel by tak hned v prvním okamžiku znal vzhled zobrazovaného objektu a mohl by se rychleji zaměřit na konkrétní oblast svého zájmu. V metodách, které umožňují rychle a jednoduše stanovit vhodnou přechodovou funkci leží budoucnost zobrazování objemových dat.

Seznam použité literatury

[0]: Derek Hill, Introduction to Medical Imaging Science, Radiological Sciences, 2008

[1]: Lukáš Maršálek, Osvětlení na GPU, Univerzita Karlova v Praze, 2005

[2]: William E. Lorensen, Harvey E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, Computer Graphics Volume 21, 1987

[3]: P. Shirley, A. Tuchman, A polygonal Approximation to Direct Scalar Volume Rendering, Computer Graphics Volume 24, 1990

[4]: Gordon Kindlmann, Transfer Functions in Direct Volume Rendering: Design, Interface, Interaction, SIGGRAPH 2002 Course Notes, 2002

[5]: Barthold Lichtenbelt, Randy Crane a Shaz Naqvi, Introduction to Volume Rendering, Prentice Hall, 1998

[6]: Wolgang Krueger, The application of transport theory to visualization of 3-D scalar data fields, Computers in Physics, 1991

[7]: Gordon Kindlmann a David Weinstein, Hue-Balls and Lit-Tensors for Direct Volume Rendering of Diffusion Tensor Fields, Proceedings Visualization, 1999

[8]: David Rodgman a Min Chen, Refraction in Discrete Ray Tracing, Proceedings Volume Graphics, 2001

[9]: Herke Jan Noordmans, Hans T M van der Voort a Arnold W M Smeulders, Spectral Volume Rendering, IEEE Transactions on Visualization and Computer Graphics, 2000

[10]: D. Marr a E. C. Hildreth, Theory of edge detection, Proceedings of the Royal Society of London, 1980

[11]: Joe Kniss, Gordon Kindlmann a Charles Hansen, Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widget, Proceedings Visualization, 2001

[12]: Balázs Csébfalvi a Eduard Gröller, Interactive Volume Rendering based on a "Bubble Model", Proceedings Graphics Interface, 2001

[13]: Jiří Hladůvka, Andreas König a Eduard Gröller, Curvature-Based Transfer Functions for Direct Volume Rendering, Spring Conference on Computer Graphics, 2000

[14]: Zhi-Pei Liang a Paul C Lauterbur, Principles of Magnetic Resonance Imaging, IEEE Press, 2000

[15]: David H. Laidlaw, Geometric Model Extraction from Magnetic Resonance Volume Data, California Institute of Technology, 1995

[16]: Gordon Kindlmann, Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering, Cornell University, Ithaca, NY, 1999

[17]: Hanspeter Pfister, Bill Lorensen, Chandrajit Bajaj, Gordon Kindlmann, Will Schroeder a Raghu Machiraju, The Transfer Function Bake-Off, Proceedings Visualization 2000, 2000

[18]: Hanspeter Pfister, Bill Lorensen, Chandrajit Bajaj, Gordon Kindlmann, Will Schroeder, Lisa Sobeierajski Avila, Ken Martin, Raghu Machiraju a Jinho Lee, The Transfer Function Bake-Off, IEEE Computer Graphics and Applications, 2001

[19]: Taosong He, Lichan Hong, Arie Kaufman a Hanspeter Pfister, Generation of Transfer Functions with Stochastic Search Techniques, Proceedings Visualization '96, 1996

[20]: J. Marks, B. Andalman, P.A. Beardsley, H. Pfister a kolektiv, Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation, ACM Computer Graphics, 1997

[21]: Kwan-Liu Ma, Image Graphs - A Novel Approach to Visual Data Exploration, Proceedings IEEE Visualization, 1999

[22]: Chandrajit L. Bajaj, Valerio Pascucci a Daniel R. Schikore, The Contour Spectrum, Proceedings Visualization '97, 1997

[23]: Issei Fujishiro, Taeko Azuma a Yuriko Takeshima, Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis, Proceedings IEEE Visualization, 1999

[24]: Shivaraj Tenginakai, Jinho Lee a Raghu Machiraju, Salient Iso-Surface Detection with Model-Independent Statistical Signatures, , 2001

[25]: D. S. Ebert, P. Rheingans, Volume illustration: non photorealistic rendering of volume models, Proceedings of IEEE Visualization 2000, 2000

[26]: P. Rheingans, D. S. Ebert, Volume illustration: Non-photorealistic rendering of volume models, IEEE Transactions on Visualization and Computer Graphics 7, 2001

[27]: S. Bruckner a M. E. Gröller, Style Transfer Functions for Illustrative Volume Rendering, Computer Graphics Forum, 2007

[28]: Christoph Lürig a Thomas Ertl, Hierarchical Volume Analysis and Visualization Based on Morphological Operators, Proceedings IEEE Visualization 1998, 1998

[29]: Shiaofen Fang, Tom Biddlecome a Mihran Tuceryan, Image-Based Transfer Function Design for Data Exploration in Volume Visualization, Proceedings IEEE Visualization 1998, 1998

[30]: Jiří Hladůvka, Andreas König a Eduard Gröller, Salient Representation of Volume Data, Proceedings of the Joint Eurographics - IEEE TVCG Symposium on Visualization, 2001

[31]: Lukáš Kohoutek, Vybrané kapitoly CT, niverzita Karlova v Praze, 2005

[0]: Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama a Daniel Weiskopf, Real-Time Volume Graphics, A K Peters, Ltd., 2006

[32]: A. Moll, A. Hildebrandt, H.-P. Lenhof, and O. Kohlbacher, BALLView: an object oriented molecular visualization and modeling framework, J Comput Aided Mol Des, 2005

[33]: A. Moll, A. Hildebrandt, H.-P. Lenhof, and O. Kohlbacher, BALLView: a tool for research and education in molecular modeling, Bioinformatics, 2006

[34]: Ivo Pavlík, Pokročilé zobrazování objemových dat na GPU, Univezita Katlova v Praze, 2009

[35]: Vladimír Hrinčár, Volume Visualization of human skulls, Charles University in Prague, 2008

Příloha A

Uživatelský manuál

V této příloze uvádíme podrobný návod, jak s uživatelským rozhraním z této práce pracovat a jaké nabízí funkce. Pokusíme se uvést důležité informace v tom pořadí, v jakém jsou pro uživatele potřebné. Proto si nejprve obecně nastíníme rozvržení aplikace, poté rozvržení oken jednotlivých přechodových funkcí a jak se samotnými přechodovými funkcemi manipulovat. A nakonec zakončíme popisem manipulace se samotným vykreslením objemových dat.

A.1 Rozdělení uživatelského rozhraní

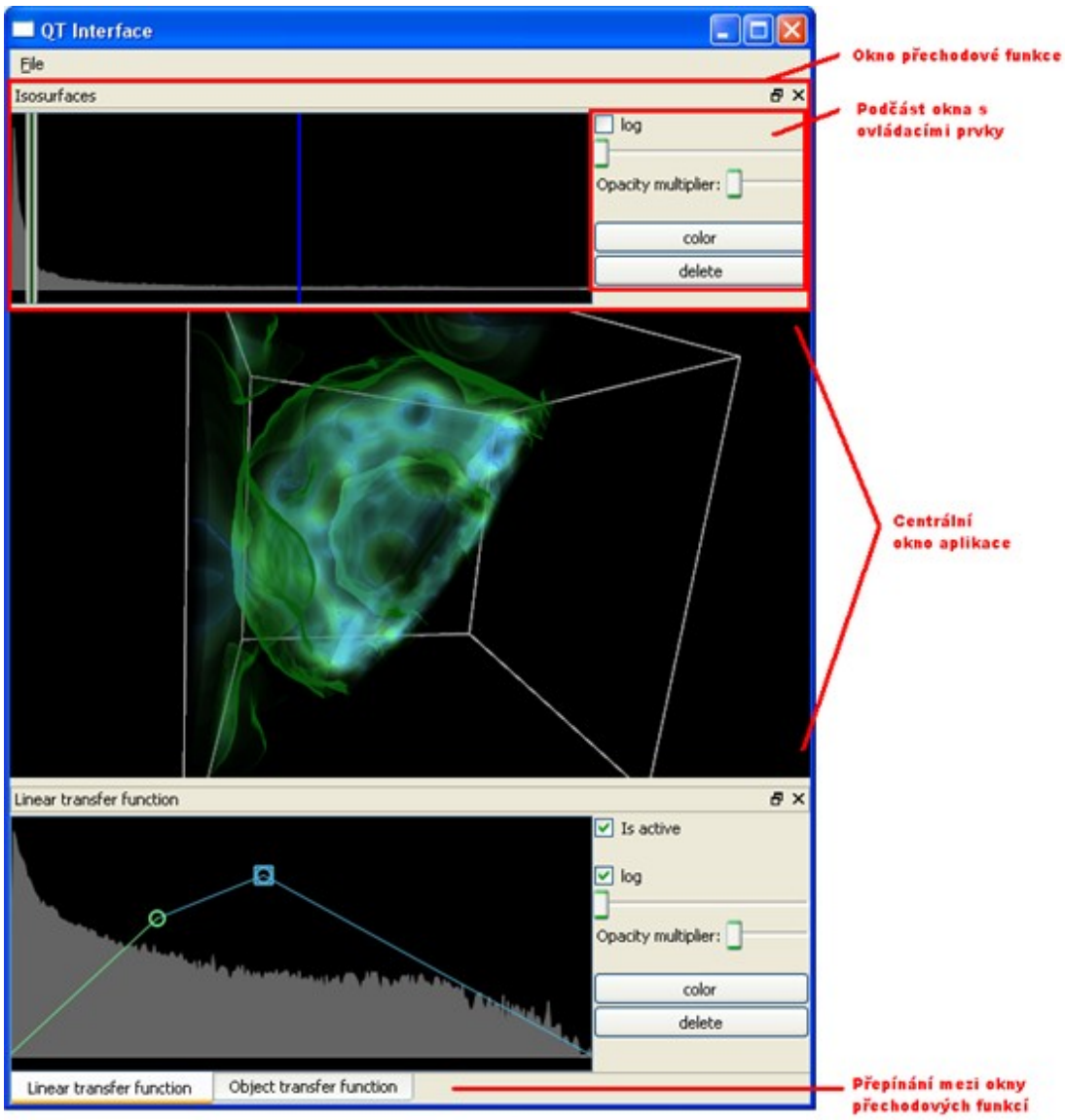
Uživatelské rozhraní je rozděleno do několika oken. Jedná se o centrální okno se zobrazovanými daty a okna pro jednotlivé přechodové funkce. Každé okno přechodové funkce může být zobrazeno jako samostatný dialog, zaintegrováno do hlavního okna aplikace, či může být úplně zavřené. Ukázka rozvržení je na obrázku 6.A níže s popisem jednotlivých částí.

A.2 Rozvržení a práce s přechodovými funkcemi

Okna přechodových funkcí jsou rozdělena do dvou částí. První slouží pro grafické znázornění samotné přechodové funkce, zatímco druhá obsahuje prvky pro jejich nastavení. Grafické rozdělení oken přechodových funkcí ukazuje obrázek A. Ovládací prvky jsou dvojího druhu:

- Horní část ovládacích elementů slouží pro manipulaci s histogramem vykresleným na pozadí přechodové funkce. Jedná se o možnost logaritmování histogramu. Slouží k tomu zaškrtávací políčko „log“. Též se dá měnit základ logaritmu, a to posuvníkem vykresleným pod zaškrtávacím políčkem. Další posuvník označený jako „opacity multiplier“ slouží jako násobitel neprůhlednosti zvolené přechodové funkce.

- Dolní část ovládání slouží pro změny vlastností označených elementů. Jak označit element přechodové funkce je uvedeno v následujícím odstavci, který je celý věnovaný manipulaci s přechodovými funkcemi.



Obrázek 6.A: Popis jednotlivých částí uživatelského rozhraní

Způsob ovládání je pro všechny přechodové funkce podobný, proto jej popíšeme obecně. Ovládání je zajištěno převážně pomocí myši. Kliknutím pravého tlačítka myši do prostoru přechodové funkce se přidávají jednotlivé elementy, například jednotlivé izoplochy v přechodové funkci izoploch, či body lineární přechodové funkce. Stisknutím stejného tlačítka nad přidaným elementem se vyvolává kontextové menu, které slouží pro nastavování vlastností jednotlivých elementů přechodových funkcí. U všech přechodových funkcí je možné změnit barvu elementu. Pouze v případě izoploch je možné jednotlivé izoplochy aktivovat nebo deaktivovat. Pro elementy objektové přechodové funkce přibývá možnost nastavení názvu. U elementů dvourozměrné přechodová funkce se též nastavuje neprůhlednost objektu. Rovněž je možné elementy pomocí kontextového menu mazat. Kliknutím pomocí levého tlačítka myši lze označit element. Vlastnosti označených elementů se

dají měnit pomocí tlačítek v kontrolní oblasti přechodových funkcí. Elementy se pohybuje pomocí stisknutého levého tlačítka myši.

Aplikace také umožňuje ukládat respektive načítat zvolená nastavení přechodových funkcí ve formě strukturovaného XML pomocí hlavního menu aplikace!

A.3 Načtení a manipulace s objemovými daty

Objemová data se načítají pomocí menu aplikace („File->Open“). Formát načítaných dat je, stejně jako možnosti manipulace s vykresleným výsledkem, závislý na použitém rendereru. V našem případě jsou požadována data ve formátu „raw“, avšak pro lepší komfort uživatele se předpokládá vstupní soubor pojmenovaný stejně, jako samotná dat s přidanou koncovkou „.vh“. Tento soubor musí obsahovat hlavičkové údaje o načítaných datech. Jedná se o následující strukturu:

- velikost dat v ose x - šířka
- velikost dat v ose y – výška
- velikost dat v ose z – hloubka
- bitová hloubka dat
- velikost pixelu v ose x
- velikost pixelu v ose y
- velikost pixelu v ose z
- nepoužitá hodnota
- hodnota neprůhlednosti předdefinované izoplochy – historický pozůstatek vyplývající z použitého rendereru, v naší aplikaci nepoužito
- izohodnota pro předdefinovanou izoplochu – historický pozůstatek vyplývající z použitého rendereru, v naší aplikaci nepoužito

Samotná manipulace s již načteným a vykresleným výsledkem je umožněna převážně pomocí myši. Jedná se o následující akce:

Rotace objektu:

Vykresleným objektem lze rotovat kolem jeho osy pomocí stisknutého levého tlačítka myši. Směr rotace je závislý na směru pohybu kurzoru. V případě uvolnění tlačítka za pohybu kurzoru zůstává objekt v rotaci. Rychlost je závislá na rychlosti pohybu kurzoru. Zastavení se provádí opětovným kliknutím myši do prostoru s vykresleným objektem.

Posun objektu:

Pomocí stisknutého pravého tlačítka myši je možné s objektem pohybovat.

Přiblížení a oddálení:

Zvětšování respektive zmenšování je umožněno pomocí kolečka myši.

Ořezová rovina:

Na vykreslený objekt je možné aplikovat ořezovou rovinu. Ve výchozím nastavení jsou data oříznuta uprostřed osy z paralelně s rovinou xy. Manipulace je zajištěna opět pomocí myši, avšak pouze při stisknuté klávese „control“ (Ctrl). Jde o posun po ose z pomocí kolečka myši a rotaci kolem středu ořezové roviny pomocí stisknutého pravého tlačítka myši. Stisknutím pravého tlačítka se pak ořezová rovina opět nastaví do poloviny osy z.

A.4 Nastavení aplikace

Dialog nastavení umožňuje nastavit pozadí jednotlivých jedno-dimenzionálních přechodových funkcí a barvu histogramu. Pro lepší ovladatelnost se také dá nastavit velikost plochy kolem jednotlivých elementů, která slouží k jejich uchopení pomocí kurzoru myši. Nejzajímavějším nastavením je však posuvník „Ray marcher precision“, pomocí kterého se nastavuje přesnost paprsku, který sníma informace o objemových datech..

Příloha B

Obsah DVD

- **bin:** Obsahuje spustitelnou verzi uživatelského rozhraní. Spouští se pomocí souboru QTInterface.exe.
- **data:** V této složce jsou umístěny příklady objemových dat.
- **doc:** V tomto adresáři je uložena elektronická podoba této práce.
- **src:** Adresář obsahuje zdrojový kód včetně souboru projektu aplikace pro MS Visual Studia 2005. Pro úspěšnou kompilaci je vyžadována instalace „Qt“..
  - **QTInterface:** Obsahuje soubory se základním zdrojovým kódem, včetně potřebných dynamicky linkovaných souborů „dll“.
    - **Histograms:** Zdrojové kódy pro implementaci jedno-dimenzionálního a dvou-dimenzionálního histogramu.
    - **IsosurfaceTransFunc:** Zdrojové kódy implementující přidávání izoploch.
    - **LinearTransFunc:** Zdrojové kódy implementující uživatelské rozhraní lineární přechodové funkce.
    - **ObjectTransFunc:** Podčást implementující objektovou přechodovou funkci.
    - **MultiDimObejctTransFunc:** Implementace multi-dimenzionální přechodové funkce.
    - **lib:** Adresář s knihovnou použitého rendereru.
  - **QTInterface.sln**