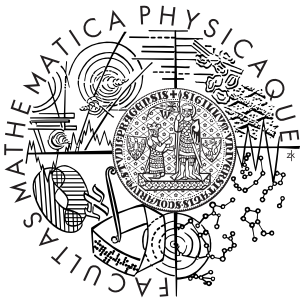
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Kavalík

Editor objemových dat

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. Václav Krajíček
Studijní program: Informatika, Programování

2010

Děkuji vedoucímu práce RNDr. Václavu Krajíčkovi za rady, připomínky a čas, který mi věnoval.

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

V Praze dne 27. 5. 2010 Jiří Kavalík

Obsah

Název práce: Editor objemových dat
Autor: Jiří Kavalík
Katedra (ústav): Kabinet software a výuky informatiky
Vedoucí bakalářské práce: RNDr. Václav Krajíček
e-mail vedoucího: Vaclav.Krajicek@mff.cuni.cz

Abstrakt: Cílem této práce je vytvoření jednoduchého nástroje k základní práci s objemovými daty, jako je například změna velikosti, otočení, prahování a retušovaní, jež by bylo možné dále podle potřeb rozšiřovat. V dnešní době existuje mnoho nástrojů pro vizualizaci objemových dat, lékařských i jiných a několik jednoduchých editorů, například pro úpravy počítačových her s voxelovou grafikou. Běžnému uživateli však není dostupný editor se základní sadou nástrojů pro úpravy objemu v rozlišení na úrovni lékařských skenerů. Námi navržený a implementovaný editor funguje na běžně dostupném hardware, uživateli nabízí sadu jednoduchých štětců a filtrů a programátorům možnost implementace vlastních nástrojů v prostředí C# formou pluginů.

Klíčová slova: editace objemu, .NET, plugin, štětec, paralelní výpočty

Title: Volume editor
Author: Jiří Kavalík
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Václav Krajíček
Supervisor's e-mail address: Vaclav.Krajicek@mff.cuni.cz

Abstract: In the present work we want to design and implement simple tool for basic editing of volume data like resizing, rotating, tresholding retouching, which could be extended as needed. Currently, there are tools for volume data visualisation and simple editors for modding computer games with voxel-based graphics. There is not an editor with basic set of tools for editing volumes in resolution of medical scanners available to common user. The designed and implemented editor can run on common hardware and offers set of simple brushes and filters to the user and means to implement different tools as plugins for C# programmer.

Keywords: volume editing, .NET, plugin, brush, parallel computation

Kapitola 1

Úvod

Ve 2D grafice má uživatel na výběr z mnoha nástrojů rasterových i vektorových. Od nejjednodušších, jež bývají součástí grafických operačních systémů, přes mnoho děl open-source scény až po profesionální balíky, které tvoří hlavní produkty některých firem. Používaných 3D editorů je také mnoho, ale zaměřují se téměř výhradně na povrchovou reprezentaci objektů. Totéž platí i o hardwarově urychlovaném zobrazování 3D scén. Důvodem je menší paměťová i výpočetní náročnost této reprezentace a jejího zobrazení.

Oproti tomu trojrozměrná obdoba rasterové grafiky, objemová reprezentace, je náročná jak prostorově, tak výpočetně. Proto se dosud používala hlavně v lékařství a dalších vědních disciplínách k ukládání výsledků měření a simulací v případech, kde mřížková struktura odráží realitu věrněji. Současný hardware však umožňuje zpracování stále většího množství dat rychlostí dostačující pro real-time úpravy a vizualizaci, čehož se nyní využívá například v medicíně nebo antropologii. Dříve po pořízení snímků, například z výpočetního tomografu, bylo tyto možno buď prohlížet po řezech (elektronicky nebo vyvolané), nebo na specializovaném zařízení. Dnes díky vývoji grafických akcelerátorů poslouží k pokročilému zobrazování získaných dat každý osobní počítač. Medicína si většinou vystačí s vizualizací, v některých případech doplněnou o segmentaci, ale třeba v archeologii jsou dostupná data často neúplná, vyžadují vyčištění, případně by bylo vhodné spojit do jedné scény objekty získané z různých zdrojů.

1.1 Cíle

Cílem této práce je navrhnout a implementovat software k přímé editaci objemu pomocí štětců, filtrů a dalších nástrojů, který by zároveň sloužil jako framework pro tvorbu nástrojů implementujících různé segmentační nebo editační algoritmy. Navržený editor umožní zpracování dat získaných z lékařských a jiných skenerů stejně jako modelování a tvorbu objemu úplně od základů. Naším záměrem je vytvořit editor, který bude snadno ovladatelný, způsobem práce připomínající běžné rasterové editory. Zároveň by však měl pracovat s daty skutečně jako s objemem se všemi jeho charakteristikami. Jak vyplívá z provedené rešerše, takových nástrojů je v současnosti nedostatek.

1.2 Struktura textu

Tento text je rozdělen následujícím způ sobem:

V kapitole 2 se zabýváme možnostmi využití objemových dat, shrnujeme metody jejich editace a vlastnosti existujícího software.

Kapitola 3 popisuje implementaci editoru, včetně úprav, které jsme provedli na frameworku pro reprezentaci a vizualizaci dat. Také zde dáváme přehled o možnostech knihovny nástrojů a o standardních nástrojích.

V kapitole 4 prezentujeme dosažené výsledky, ukázky možných úprav a tvorby. Testujeme možnosti paralelizace a popisujeme způsob tvorby vlastního nástroje.

Kapitola 5 shrnuje celou práci a dává několik námětů na vylepšení.

V dodatku lze nalézt uživatelskou dokumentaci k editoru a vybrané detaily implementace.

Kapitola 2

Objemová data

Z pohledu počítačové grafiky lze objemová data chápat jako zobecnění rasterového obrazu, základní jednotkou je zde *voxel* (volume pixel) - kvádr, který má určenou pozici v prostoru, rozměry a hodnotu (barvu, hustotu nebo třeba vektor rychlosti). Pozice ani rozměr typicky není implicitní součástí hodnoty voxelu. Pozice je určena reálnou polohou voxelu v datové struktuře a rozměr, společný pro všechny voxely, je vlastností konkrétního objemu. Samotný objem je trojrozměrná pravidelná pravoúhlá mřížka voxelů (lze si jej představit jako sérii bitmap, které mají ve třetím rozměru určitou tloušťku a jsou naskládané na sebe). Objem je diskretizovanou reprezentací původních objemových dat, každý voxel reprezentuje oblast prostoru, která ve skutečnosti nemusí být homogenní, a jeho hodnota tedy odpovídá buď hodnotě naměřené na odpovídajícím bodu mřížky, nebo integrálu přes danou oblast.

Nejčastějšími zdroji objemových dat jsou v lékařství i jiných oborech výpočetní tomograf (CT), magnetická rezonance nebo ultrazvuk. Tyto poskytují skalární data: hodnota voxelu udává velikost jedné veličiny. U CT je to hustota materiálu (resp. jeho schopnost pohlcovat rentgenové záření). Jindy je objem výstupem fyzikální simulace, např. proudění turbínou, a v každém voxelu je definováno několik hodnot (*vektorové pole*).

Binární objemová data jsou data, kde voxely nabývají pouze hodnot 0 a 1. Takový objem definuje množinu pixelů "uvnitř"a její doplněk. Tato reprezentace se dá využít například tam, kde stačí definovat pouze "masku"nad daty - označení vybraných voxelů, výstup prahování a podobně. Wang a Kaufman [17] navrhli systém, kde vnitřek objektu má hodnotu 1, okolní prostor

0, a navíc definovali přechodovou oblast (povrch), kde voxely nabývají hodnot z intervalu (0,1) podle vzdálenosti od hranice objektu.

2.1 Běžné aplikace objemových dat

Objemová data se dnes využívají v mnoha oborech, avšak jejich zastoupení v porovnání s povrchovou reprezentací je mnohem menší. Většinou se jedná o speciální aplikace.

2.1.1 Věda

Lékařství

Jak bylo výše zmíněno, objemová data se často využívají v lékařství, a to jako prostředek diagnózy. CT nebo MRI různých částí těla je neinvazivní metodou pohledu na zlomeninu, tumor nebo pro virtuální kolonoskopii. Relativně nově se lze, hlavně v ordinacích gynekologů, setkat s tzv. 3D ultrazvukem, který nabízí i necvičenému oku budoucí maminky dostatečně srozumitelný pohled na vyvíjející se plod.

Další vědní obory

Antropologie pomocí CT zkoumá nálezy, které by bylo problematické dokumentovat jinak, například mumie zakonzervované v sarkofágu, a může k nim tak umožnit alespoň virtuální přístup mnohem širšímu publiku [4]. Materiálový průmysl využívá CT o vysokém výkonu například ke kontrole kovových odlitků (hledání prasklin na bloku motoru).

Designéři využívají vizualizace výstupu simulací například k ověření aerodinamiky návrhu nebo zatížení struktur.

2.1.2 Počítačové hry

Různé počítačové hry využívají objemovou reprezentaci a rendering jako součást herního engine.

Objemový terén je alternativou k terénu generovanému z výškové mapy. Narozdíl od výškové mapy, voxelový terén umožňuje vytvářet jeskyně a jiné

překryvy. Navíc použití zničitelných voxelů dává hráči šanci terén upravovat, tvořit zákopy nebo krátery. Tato technika se poprvé objevila ve hře Comanche již v roce 1992 a umožnila zobrazit na svou dobu nevídané detaily.

Definice objektů například vozidel, se objevila ve dvou dílech série *Command & Conquer* z přelomu tisíciletí. Takové modely jsou vytvořeny ve velice nízkém rozlišení, nejvýše několik stovek voxelů na jednotku. Pro zobrazení poškození jednotky stačí odstranit nebo začernit voxely nejblíže k místu výbuchu.

Speciální efekty jako kouř, mlha nebo mraky působí mnohem realističtěji, pokud jsou vytvořeny jako objem a správně renderovány.

2.1.3 Metody zpracování objemových dat

Filtrování a segmentace

Před samotným zobrazením je často vhodné data například prahovat nebo segmentovat, zvýraznit důležité části nebo provést obarvení podle přechodové funkce.

Segmentace je velmi často používaná úprava. Původní data obsahují spoustu informací a pokud například lékař hledá ledvinový kámen, CT zároveň nasnímá velkou část břišní dutiny. Když nejdříve vysegmentujeme samotnou ledvinu, lékař má možnost si ji "prohlédnout", aniž by mu ve výhledu překážely další orgány. Případná automatická segmentace navíc může nalézt samotný kámen a lékař už výsledek pouze zkontroluje.

Jako další příklad vezměme archeologický nález, zkamenělinu, kterou není z různých důvodů možné očistit od usazenin. Archeolog může vysegmentovat samotný objekt zájmu a studovat jeho tvar bez rušení nánosy.

Fúze dat z několika zdrojů (CT a MRI, nebo například sken s kontrastní látkou a bez ní) umožňuje společně zobrazovat kvality, které nelze zachytit jedním snímáním. K tomu je nutné dva různé snímky namapovat na sebe tak, aby se odpovídající objekty co nejpřesněji překrývaly. Tato úloha se nazývá *Registrace*.

Hledání objektů je úloha, která se snaží v objemu nalézt souvislé oblasti, které reprezentují nějaký reálný objekt, například orgán na lékařském snímku. Určení a pojmenování objektů může sloužit například k otagování dat pro vyhledávání, ale také pomáhá při používání předchozích úloh.

Zobrazování

Metoda zobrazení závisí na požadovaném použití výstupu.

Zobrazení po řezech je výpočetně nenáročná metoda, analogie prohlížení vyvolaných snímků. Nejčastější je směr řezů v axiální rovině, neboť takto data vystupují například z CT, ale není problém vytvořit z objemu sadu řezů v jiné hlavní rovině nebo dokonce libovolné uživatelem definované rovině.

Izoplocha je množina voxelů o stejné hodnotě. Pro všechny různé hodnoty voxelů objem určuje různé izoplochy, proto je jedním z parametrů takového zobrazování i hodnota požadované izoplochy. Izoplochu je možné renderovat přímo, nebo vygenerovat odpovídající trojúhelníkovou síť a zobrazovat tu.

Přímé renderování objemu raycastingem nebo raytracingem je výpočetně dost náročné, ale při použití současných programovatelných grafických karet už lze dosáhnout dostatečné rychlosti pro realtime zobrazování i s možností otáčení a přibližování.

2.1.4 Ukázková data

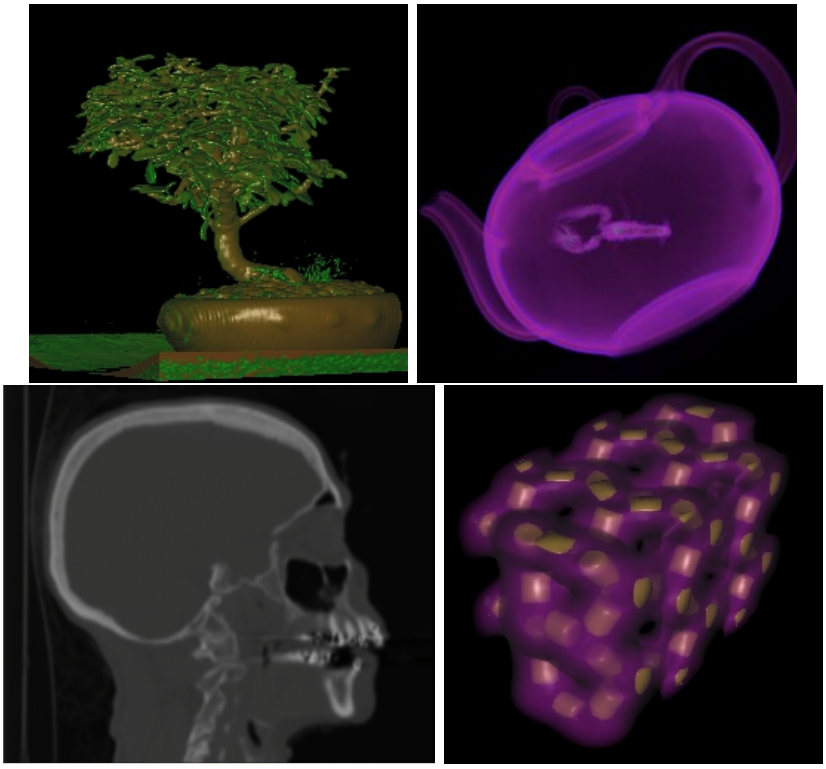
Na Internetu je možné stáhnout množství různých datasetů. Patří mezi ně objemy na obrázku 2.1 ([16]), anonymizovaná DICOM data i uživatelská tvorba (ta však bývá v nepříliš rozšířených formátech).

Objemová data lze získat například při lékařském vyšetření, kdy lékař pacientovi po vyšetření na požádání zkopíruje získané snímky na CD.

My budeme, kromě dat vytvořených přímo v editoru, při testování využívat například snímek hlavy - obrázek 2.1 (c)

2.2 Editace objemových dat

Metod editace bylo v literatuře popsáno několik. Některé zobecňují techniky používané ve 2D, jiné byly navrženy specificky pro objemová data.



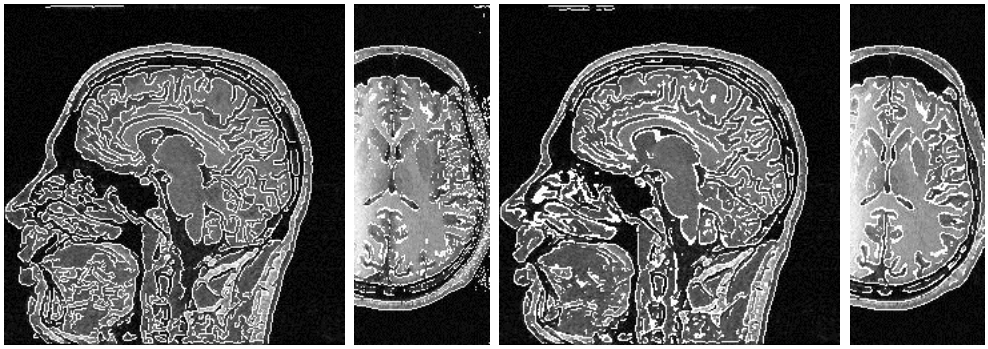
Obrázek 2.1: Ukázky objemobých dat. První snímek zobrazuje tomogram bonsaie pomocí několika barvených izoploch. Na druhém snímku vidíme přímé zobrazení skenu čajové konvice s překvapením. Třetí snímek představuje jeden řez objemen získaným při CT hlavy pacienta. Poslední snímek proti tomu ukazuje simulaci krystalové mřížky křemíku.

2.2.1 Transformace

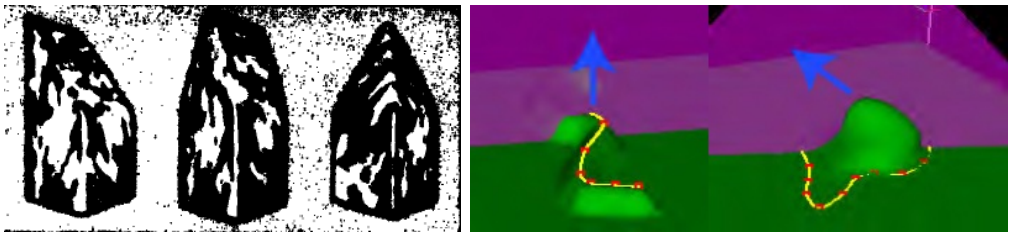
Běžné 2D editory umožňují měnit rozměry obrázku, otáčet jej nebo jeho části, provádět zrcadlení a podobně. Všechny tyto transformace mají význam i ve 3D, jen přibyde další stupeň volnosti. Implementace takových transformací je přímočarým rozšířením dvourozměrných verzí.

2.2.2 Filtry a konvoluce

Stejně jako transformace lze do třetího rozměru rozšířit i například Gaussovo rozostření a konvoluce obecně [10]. Je možné provádět takové úpravy



Obrázek 2.2: Detekované hrany zakreslené do původních dat. Vlevo detekce po řezech. Vpravo algoritmus upravený pro 3D [9].



Obrázek 2.3: Ukázkové nástroje pro rytí [17]a editační operátory [3].

samostatně na jednotlivých řezech, ale ve třetím rozměru mohou vznikat artefakty, jak je vidět na obrázku 2.2.

2.2.3 Volume sculpting

Český překlad "vyřezávání"nebo "sochařství"velmi přesně vystihuje hlavní myšlenku tohoto postupu. Na začátku uživatel vybere základní objekt, kvádr, kouli, případně strukturu poskládanou z geometrických primitiv, a tento následně opracovává virtuálním dlátem.

Galyean a Hughes [5] představují binární objem jako jílovou sochu, kde 1 značí přítomnost materiálu a 0 prázdný prostor, a kterou je možné upravovat několika nástroji:

- *Tuba* slouží k přidávání materiálu do scény, jako vymačkáváním zubní pasty.
- *Horkovzdušná pistole* postupně odstraňuje (vypařuje) materiál

- *Brusný papír* vyhlazuje hrany a hrboly a vyplňuje rýhy
- Geometrická primitiva kreslená podobně jako ve 2D, místo úsečky válec a torus jako alternativa kruhu
- Barvení přidává původně binárnímu objemu další rozměr.

Ve své práci také předpokládají vznik deformačních nástrojů, jako je kroucení nebo ohýbání.

Wang a Kaufman [17] navrhují sadu nástrojů různých tvarů pro rytí (obrázek 2.3 vlevo) a operaci *řezání*. Při řezání uživatel definuje uzavřenou křivku na povrchu objektu a tak "vyřízne"část objemu až do určité hloubky. Tato operace umožňuje například odhalit vnitřní strukturu objektu v případech, kdy není možné zobrazovat průhledně.

2.2.4 Level-Set modely

Jiný přístup uvažuje objem (resp. jeho povrch) jako implicitní matematickou funkci. Matematicky definovaný povrch je jednoduchý, spojitý a uzavřený. Navíc jej lze navzorkovat v rozlišení podle potřeby bez problémů s aliasingem. To nám dává možnost zobrazovat během editace menší data a dosáhnout tak interaktivity. Přitom je možné provedené operace aplikovat na objekty ve vysokém rozlišení před finálním renderingem. Eyiyurekli a Breen [3] implementují nad touto reprezentací sadu editačních operátorů:

- Vytažení bodu nebo křivky vytváří detaily na povrchu objektu (obrázek 2.3 vpravo).
- Rytí vytváří v povrchu vzory nebo nápisy
- Načrtnutí nového tvaru povrchu umožňuje rychlé přidání hrubších struktur
- Vyhlazování odstraňuje detaily ve formě výstupků i zářezů

Alternativou je generování konkrétního objemu vzorkováním matematických funkcí, například definovat vzorec pro kouli nebo sinovou vlnu a každému voxelu spočítat, zda je "uvnitř", resp. "pod"definovanou plochou.

2.2.5 CSG a skládání primitiv

Tvar složitějšího objektu můžeme vytvořit skládáním z geometrických primitiv, na které předtím aplikujeme potřebné transformace (posunutí, změna velikosti a další). Dalším stupněm je pak hierarchická struktura takových primitiv a množinové operace, tedy objemová verze CSG. Tuto metodu lze s výhodami využít například k procedurálnímu generování scény.

2.2.6 Štětce

Tato práce navrhuje editor s nástroji, které rozšiřují způsoby běžného dvojrozměrného kreslení, štětce a lokální i globální filtry. Objemový štětec je objekt, který má svou velikost, tvar a hodnotu ("barvu"). Aplikace takového štětce předpokládá navíc zadání polohy a způsobu aplikace.

Příklad: Kulový štětec o poloměru 15 voxelů a hustotě 100 jednotek aplikujeme na souřadnicích počátku odečtěním. Průnikem obou objemů je osmina koule. Na tomto průniku se od hodnoty každého voxelu editovaného objemu odečte 100 jednotek a nová hodnota se uloží zpět do voxelu.

Tvar štětce lze definovat libovolným objemem (včetně možnosti načíst dříve vymodelovaný objekt a používat jej jako "razítko"), nejčastěji však půjde o procedurálně definované množiny (koule a jiná primitiva).

Filtr může být lokální s oblastí působnosti definovanou jako u štětce, nebo globální pracující vždy s každým voxelem upravovaného objemu. Použití filtru znamená, že se pro každý voxel objemu spočítá nová hodnota jako kombinace staré hodnoty a hodnot jeho sousedů.

2.3 Existující software

2.3.1 Vizualizace a zpracování

Pro potřeby zpracování výstupu je k dispozici několik profesionálních nástrojů (např. Amira [14], Avizo [15]), které poskytují mnoho funkcí například pro analýzu dat (měření objemů), pokročilou segmentaci, filtrování, registraci, i rozsáhlou paletu možností zobrazování.Součástí jsou také nástroje pro export objektu do modelu povrchové reprezentace (extrakce izoplochy).

Na poli free a open-source existují frameworky pro vizualizace. VTK je rozsáhlý multiplatformní projekt, jehož součástí jsou i moduly pro zobrazování objemu. Knihovna Volume Library [7], kterou jsme v teto práci

použili, umožňuje zobrazování a poskytuje prostředky pro tvorbu paralelních algoritnů filtrace či segmentace.

2.3.2 Editory

Díky využití voxelové grafiky v některých počítačových hrách jsou dostupné specializované editory pro účely tvorby modifikací. Nejblíže možnostem obecného editoru jsou z nich programy pro tvorbu vojenských jednotek a podobných objektů, které umožňují v nízkém rozlišení úpravy jednotlivých voxelů.

Následující přehled popisuje základní vlastnosti dostupných programů:

Komerční programy

Voxel3D - jednoduchý editor pro objemy do několika tisíc voxelů. Kreslení probíhá vkládáním voxelů do jednotlivých řezů, nebo přímo ve 3D přilepováním nových voxelů k již existujícím. Program umí importovat trojúhelníkovou síť a přibližně ji vyplnit objemem. Pracuje s vlastním formátem souborů a neumí import jiných objemových dat. V dostupné trial verzi [2] neumí ukládat.

Paint3D - sofistikovanější editor, který podle autora zvládne i objemy o rozměrech 128^3 [1]. Ve 3D pohledu zobrazuje aktivní řez jako čtvercovou síť, úpravy pak probíhají jako 2D kreslení, na tomto řezu (s nástroji jako *tužka, přímka, plechovka*). Poté je možné nakreslený tvar rozkopírovat do dalších řezů nástrojem *extrude*. Editor podporuje skriptování v jazyce IronPython [8]. Neumí importovat RAW data.

3D Coat - nástroj zaměřený na umělce. Pracuje s povrchovou reprezentací a texturami, ale jako speciální funkci umožňuje vytvořit základní tvar pomocí objemových primitiv a volume sculptingu (nástroje pro rytí nebo rozšiřování objemu). Výsledný objekt lze převést na trojúhelníkovou síť s možností ručně ovlivnit triangulaci. Reprezentovaný objem je však pouze binární a i když program zvládne import RAW, provede při něm automaticky prahování.

VG Studio MAX - průmyslový systém pro analýzu, segmentaci a další postprocessing objemových dat, včetně funkcí pro filtrování a zpracování

obrazu. Umožňuje import mnoha datových formátů, včetně dat tomografů různých výrobců, zvládá také import a export RAW nebo DICOM. Mezi editory se VG Studio řadí od verze 1.2, kde přibyl nástroj *štětec*, který funguje jako čtvercový nebo krychlový štětec (přepínatelný mezi 2D a 3D módem) v řezu. Mezi editační funkce se dá považovat i možnost kopírovat do objemu vysegmentované objekty.

Sandbox 2 - editor podporující tvorbu voxelového terénu pro hry založené na CryEngine 2, například Crysis(2007), která někdy slouží jako benchmark při porovnávání výkonnosti grafických karet i celých počítačových sestav.

Acropora - nástroj pro procedurální modelování objemu. Umí voxelizovat povrchové modely a objemovou tvorbu převádět na trojúhelníkovou síť.

Volně dostupné programy

Voxel Section Editor II/III je open-source editor pro modifikaci her *Tiberian Sun* a *Red Alert 2*. Editace probíhá v řezech přímým vkládáním barevných voxelů pomocí nástrojů *tužka, obdélník* a *plechovka*.

ImageJ - public domain 2D editor napsaný v Javě. Kromě standardní palety editačních funkcí nabízí možnost načíst sadu obrázků jako řezy ze souboru RAW nebo i z více samostatných souborů. Tak je možné objem editovat po řezech. Kromě běžných grafických formátů umí načítat i snímky DICOM. V offline verzi [11] je dostupný plugin pro objemovou vizualizaci sady řezů, ale kvalitnější metoda renderingu je pomalá (několik sekund na snímek).

Q-BLOCK, Voxelart - webové aplikace pro jednoduché skládání barevných kostiček. Místo práce se souborem obsahují online galerie výtvorů.

Jak lze vidět v přehledu, přímo na editaci voxelů jsou zaměřené hlavně editory ke hrám. Ty však poskytují jen základní funkce a podporují pouze proprietární formáty. Pojem objemového štětce se z představených programů vyskytuje pouze ve VG Studiu, a i tam jen ve velmi omezené verzi. Editory jsou navíc většinou spíše voxelové než objemové, tedy pracují s jednotlivými voxely, ne s objemem jako celkem.

Kapitola 3

Implementace

Program je napsán v jazyce C# v prostředí MS Visual Studio 2008. Pro běh vyžaduje .NET 2.0, MS Windows XP 32/64bit a novější, 512MB RAM a grafickou kartu se SM3 a minimálně 256MB paměti. GUI je vytvořeno ve WinForms, rendering využívá OpenGL. Program umožňuje přímou vizualizaci dat v několika pohledech a editaci pomocí různých nástrojů. Samotná data ve formě sady 2D řezů je možné načítat z binárního souboru bez metadat (RAW) nebo z formátu DICOM [6] a exportovat do RAW. Editor pracuje s "projekty", tj. uspořádanou sadou 3D vrstev a jejich vazeb, a vlastním datovým formátem pro jejich ukládání.

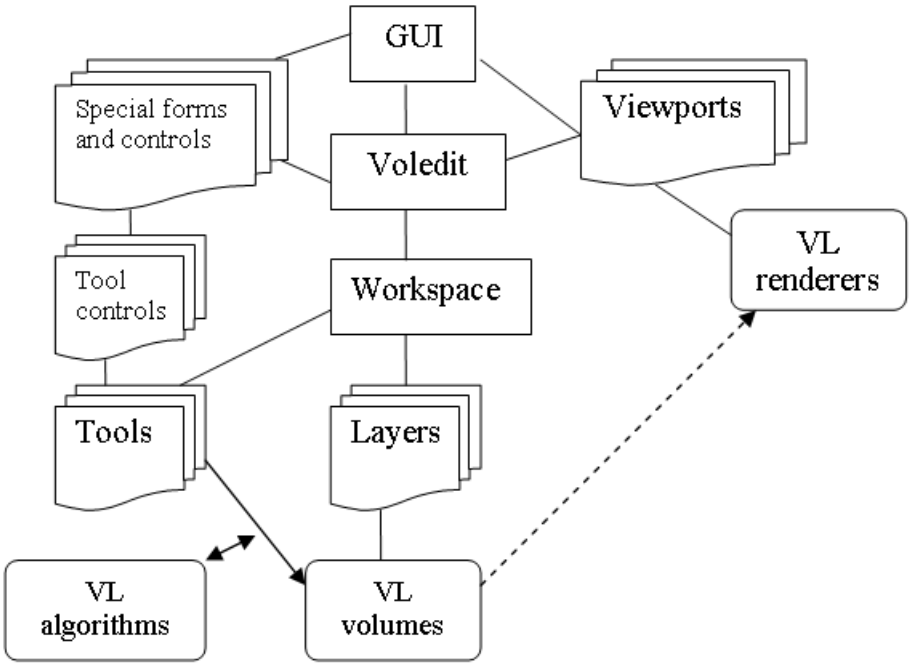
Program byl vyvíjen a odladěn na notebooku HP Compaq s dvoujádrovým procesorem s taktem 2,16GHz, 3GB RAM a GPU Ati Mobility Radeon 3430HD na systému MS Winndows 7 32bit.

Použité nástroje a knihovny MSVS08, C# a .NET, VL framework [7], The Tao Framework [12], nástroje pro import DICOM [6]

3.1 Struktura aplikace

Projekt je rozdělen do dvou hlavních částí - editor a knihovna nástrojů, dále pak obsahuje knihovnu VL [7] a knihovnu publikující rozhraní pro tvorbu pluginů.

Editor je samostatný program obsahující jádro (implementované třídou Voledit), GUI, integrované základní nástroje, grafické komponenty a další třídy.



Obrázek 3.1: Struktura a závislosti hlavních částí editoru.

Knihovna nástrojů je základní plugin - jedna dynamická knihovna obsahující třídy všech standardních nástrojů.

Volume Library je framework pro zpracování objemových dat. Využívá se pro reprezentaci, úpravy a zobrazování objemů.

3.2 VL framework

Autor Mgr. Jakub Hlaváček ve své diplomové práci navrhl a experimentálně implementoval VL jako nástroj k vývoji systémů pro zpracování a zobrazování medicínských dat v prostředí .NET.

Jím definované rozhraní i část implementace jsou platformově nezávislé, ale komponenty jsou implementovány pomocí WinForms a zobrazování využívá Tao framework[12] včetně Tao.Platform.Windows, kterou nelze zprovoznit na patformě Mono/X11 ani nahradit knihovnou Tao.Platform.X11. Vzhledem k tomu jsme se rozhodli projekt zpracovat pro Microsoft Windows,

což nekoliduje s požadavkem na dostupnost editoru běžným uživatelům.

Původní VL framework počítá hlavně s použitím segmentačních algoritmů, jejichž výstupem je pokaždé kompletní nový objem. Tento přístup se pro editor příliš nehodí, rozhodli jsme se tedy zároveň s tvorbou editoru framework mírně přizpůsobovat pro naše potřeby.

3.2.1 Struktura VL frameworku

Framework obsahuje kompletní sadu tříd pro základní práci s objemem:

- načítání dat z disku a jejich reprezentace v paměti
- obecné algoritmy pro zpracování objemu s různou granularitou (jednotlivé voxely, řezy nebo celý objem) a s podporou paralelizace na CPU
- 2D zobrazování řezů
- 3D rendering izoploch a přímý rendering objemu
- přechodové funkce pro vizualizaci, včetně editoru

Namespace VL.Api definuje obecná rozhraní

Namespace VL.Utils je nástrojem pro matematiku v prostoru homogenních souřadnic. Obsahuje třídy vektorů a matic, včetně implementace běžných operátorů a dalších metod (normalizace vektoru, translace matice, skalární i vektorový součin).

3.2.2 Provedené úpravy

Renderer2DControl

Původní 2D renderer zobrazoval v jednom OpenGL okně zároveň řezy ve třech základních směrech. To nevyhovovalo našim požadavkům na konfigurovatelné pohledy do objemu, a tak byla vytvořena verze, která podle nastavení zobrazuje v celém OpenGL okně jen jeden z těchto řezů.

Rendering 2D se v původním projektu Example používal pouze pro 16bit objemy z DICOM a tak bylo třeba doimplementovat zobrazování 8bit řezů.

In-place editace

Třída Algorithm neumožňuje zpracování takové, kde zdrojový a cílový objem je stejný objekt (VolumePtr). Definovali jsme tedy alternativní třídu AlgorithmEd, která toto zvládne.

Takový přístup dává zásadní omezení - při výpočtu nové hodnoty jednoho voxelu nelze přistupovat na jeho sousedy neboť některé mohou být už zpracované, zatímco jiné obsahují ještě starou hodnotu. Vzhledem k možnému paralelnímu běhu navíc nelze předpokládat že některé konkrétní sousední voxely budou v konkrétním stavu. Stejný princip znemožňuje interakci se sousedními řezy při zpracování po celých řezech. Navíc algoritmus tohoto typu nelze efektivně použít v pipeline (její výpočet se musí pozastavit v době běhu editačního algoritmu). Nicméně má toto řešení i pozitivní efekt, je možné definovat podobjem - například obalovým kvádrem, kde se má editace provést, a voxely mimo tuto oblast si zachovají původní hodnotu. Tímto lze dosáhnout vysoké rychlosti, pokud prováděné úpravy jsou lokálního charakteru. Také narozdíl od původní verze je potřebná paměť značně menší, neboť není nutné mít v jednom okamžiku alokované zároveň zdrojový i cílový objem.

Tato verze algoritmu se v editoru používá pro objemové štětce a filtry, které pracují s jednotlivými voxely (například inverze).

Kurzor jako vlastnost rendereru

Původní 2D renderer obsahuje koncept 3D kurzoru - při zobrazení všech řezů je jejich vzájemná poloha vyznačena čarami. Pozice třech řezů tak dávají kompletní prostorovou pozici. Tento koncept jsme rozšířili o vyvolávání událostí a doplnili i do 3D rendereru.

Ve 3D jsme se rozhodli také umožnit "ukazování"myší. Poloha myši v okně je zpětně transformována do souřadnic objemu. Vzdálenost kolmo na rovinu obrazu se nastavuje kolečkem myši.

Kamera - otáčení

Ve 3D rendereru byla implementována kamera ovladatelná myší, ale úprava probíhala na úrovni Eulerových úhlů a rotační matice se generovala z nich, to vedlo k prohazování orientace os během rotace. Implementovali jsme proto metodu, kdy dochází k postupnému pootáčení samotné matice [Příloha B.2].

Navíc jsme přidali tlačítka pro nastavení kamery na základní směry pohledu (odpovídají směrům 2D řezů).

Hlášení o průběhu operace

V původním nasazení se zpracováním objemu pomocí pipeline se průběh výpočtu zobrazoval tak, že renderer postupně vykresloval již zcela zpracované řezy. S přidáním editačních algoritmů toto nestačí. Navrhli jsme proto jednoduchý systém hlášení průběhu. Do třídy VLGlobal byly přidány delegáty pro ohlášení začátku nebo konce akce a pro update. Každá akce určí počet kroků, které provádí, a volitelně textový popisek. Při ukončení zadá jen počet provedených kroků. Jedno volání update přidá jeden krok jako zpracovaný. V GUI je pak impelemntována obsluha tak, že všechny běžící akce se sčítají (potřebné i již provedené kroky) a zobrazují jedním indikátorem průběhu. Všechny třídy, které provádějí déletrvající výpočty (algoritmy, načítání dat), byly upraveny, aby takto hlásily stav. Systém je používán i dalšími součástmi editoru.

Načítání a ukládání dat

Bylo třeba doimplementovat načítání 16bit Raw dat ve VolumePtr.VolumeFactory a podporu Little/Big Endian.

Součástí původního API není žádný způsob ukládání výsledků do souboru a načítání Raw je omezeno na jeden objem na soubor a vyžaduje druhý soubor s odpovídající hlavičkou. Proto jsme přidali metody pro načítání z libovolného datového streamu a zápis do streamu i souboru a navíc jako test ukládání komprimovaného streamu GZip. Tento postup zmenší výsledný soubor až o 50%, pokud se jedná například o data z lékařského vyšetření, a o 70-90% v případě, kdy obsah byl vymodelován uživatelem.

3.3 Knihovna PlugInBase

IVoledit Rozhraní, které poskytuje jádro editoru nástrojům.

ITool Základní rozhraní každého nástroje. Zajišťuje jeho registraci a interakci s uživatelem.

IPaintTool Editační štětce a filtry tímto rozhraním definují metodu pro provedení kreslící akce.

ISelectionTool Selekční nástroje, například segmentační upravují vždy přimo masku výběru, nezávisle na aktuálně vybrané vrstvě.

ITransferTool Nástroje pro přesuny mezi vrstvami

Load/SaveInfo Třídy používané k synchronizaci vnitřních stavů nástrojů s workspace.

UndoInfo Třída obalující informace, které nástroj potřebuje pro odvolání provedené akce.

eMenuCategory Výčtový typ definující kategorie nástrojů. Používá se

eTransferBlendingMode Metody kombinace voxelů při slučování vrstev transportními nástroji

3.4 Komponenty editoru

Třída Voledit.Layer

Layer - vrstva je základní datovou položkou editoru, obaluje objem uložený jako VL.VolumePtr a přidává mu jméno, pořadí v projektu a blending ("krytí"). Umí se uložit do streamu včetně objemových dat.

Třída Voledit.Project

Project je reprezentací aktuálního workspace. Uchovává uspořádaný seznam vrstev, informace o aktivní vrstvě, výběr a jeho obalový kvádr, instanci VL.VolumeParams s parametry objemu, nad kterým je workspace vytvořen. Také obsahuje položku pro vrstvu "náhledu", kam se ukládá sjednocení všech vrstev. Implementuje metody pro práci s vrstvami a pro ukládání/načítání souborů v datovém formátu "VEW - Volume Editor Workspace"[Příloha B.1]

Voledit.Viewport, Voledit.ViewportSettings

Viewport je vizuální komponenta, která implementuje VL.Api.IRenderer a obaluje 2D a 3D renderery.

ViewportSettings je komponenta obsahující prvky pro konfiguraci Viewportu.

Každý Viewport je pevnou součástí okna GUI, ale renderer, který je v něm umístěn, je možné měnit za běhu programu. Defaultně všechny Viewporty sdílejí jednu pozici kurzoru a jednu přechodovou funkci, ale je možné toto provázání přenastavit.

Voledit.LayoutGenerator, Voledit.Layout2x2

Abstraktní třída LayoutGenerator definuje metodu GenerateLayout(Panel, Voledit), která vytvoří, zaregistruje a na zadaný WinForms panel umístí Viewporty.

Layout2x2 implementuje vygenerování 4 stejně velkých Viewportů (pomocí SplitPanelů). Tři z nich zobrazují v základní 2D pohledy a čtvrtý je perspektivní 3D "Direct Volume Rendering".

Generátor má kompletní kontrolu nad počtem i rozvržením viewportů, lze tedy implementovat různá rozvržení, nebo dokonce konfigurovatelný layout, který by se řídil například skriptem nebo XML souborem.

Další vizuální komponenty

Dialogová okna:

- ImportForm - nastavení preprocessingu při importu dat z cizích formátů
- NewForm - definice parametrů nového workspace
- SettingsForm - konfigurace globálních vlastností editoru
- DicomImportForm - zobrazení hlavičky DICOM souboru a výběr objemu k importu

Konfigurační panely integrovaných nástrojů:

- Move/Rotate/ResizeControl umožňují numericky zadávat parametry odpovídajícím nástrojům a poté potvrdit nebo zrušit nastavenou transformaci. V RotateControl se provádí přepočet rotační matice na Eulerovy úhly [13]

- ReferenceControl - dává možnost během náhledu transformace měnit zobrazenou referenční vrstvu - při změně se vytvoří náhledový objem nově vybrané vrstvy
- Layers - panel zobrazující vrstvy v aktuálním workspace

Filtry: Rozhraní ResizeFilter definuje konvoluční filtr - getSamplingRadius() určuje velikost okénka a getSampleWeight(float) vrací váhu bodu v zadané vzdálenosti od středu okénka. Získanou sadu vah je třeba normalizovat. Implementované filtry jsou:

- BoxFilter - počítá průměr přes celé okénko
- TriangleFilter - váha vzorku je přímo úměrná vzdálenosti
- BellFilter - vrací gaussovské váhy
- CubicFilter - kvalitní, ale výpočetně náročný filtr

3.5 Jádro editoru - třída Voledit

Základní funkčnost editoru obsahuje třída Voledit. Spravuje workspace, registruje a spravuje pluginy a nástroje. Také obsluhuje renderery, zpracovává pozici kurzoru, spouští editační akce nebo undo.

Jádro implementuje rozhrani IVoledit, konkrétně:

- Init() - inicializace editoru z GUI
- konfigurace jádra před spuštěním:
  - setToolMenu()
  - setLayoutPanel()
  - setViewportSettingsPanel()
  - setLayersPanel()
  - setToolPanel()
  - setToolStrip()
  - setTransferFunctionControl()
  - transferFunctionControl_TFChanged()

- getToolPanel() - vrací prostor, kam nástroje mohou umisťovat konfigurační komponenty
- CreateButton() - vytvoří na panelu nástrojů tlačítko s popiskem nebo ikonou
- CreateMenuItem() - přidá odkaz do podmenu vybrané kategorie
- getCursorPos() - aktuální souřadnice kurzoru
- getCursorVol(), setCursorVol() - slouží kreslícím nástrojům k nastavení vlastního objemového kurzoru
- obsluha událostí GUI:
  - NewProject()
  - LoadProject()
  - SaveProject()
  - ImportRaw()
  - ImportDicom()
  - ExportRawLayer() - exportuje aktuální vrstvu
  - ExportRawWorkspace() - exportuje sjednocení všech vrstev
  - ShowSettings() - zobrazí SettingsForm
- getLayersNames() - vrátí jména, podle kterých se pak dá vyhledávat v objemech
- getActiveLayerName()
- getLayerVolume() - vrátí objem podle zadaného jména
- createLayer()
- getActiveLayerVolume(), setActiveLayerVolume(), getSelectionVolume();

3.6 Použití třídy Voledit

Program po spuštění vytvoří novou instanci třídy Voledit, nastaví odkazy na panely a menu GUI, které jádro používá ke zobrazování viewportů, nastavení a informací, vsechno postupným voláním setToolMenu(), setLayoutPanel(), setViewportSettingsPanel, setLayersPanel. V případě, že je v programu použit objekt TransferFunctionControl, tak se nastaví i obsluha události TFChanged, která zajistí aktualizaci zobrazení po uživatelské úpravě přechodové funkce. Tím je jádro připraveno k inicializaci.

Inicializace jádra

Spustí se voláním Voledit.Init(). Provádí vytvoření vnitřních struktur i komponent GUI.

1. vytvoření tlačítka Undo
2. vygenerování menu kategorií nástrojů
3. prohledání adresáře PlugIns - knihovny s nástroji mají jméno 'Tool*.dll' a uložení všech tříd nalezených pomocí Reflection, které implementují PlugInBase.ITool do seznamu tools
4. pokus o registraci všech detekovaných nástrojů - každý z nich si může zažádat o vytvoření tlačítka a položky v menu
5. inicializace defaultního objemového kurzoru - používá se malá krychle
6. vytvoření viewportů a jejich rozložení - každý viewport se přihlásí do jádra (kolekce viewportRegister) a zaregistruje obsluhu událostí MouseUp/Down/Move/Click
7. svázání kurzorů a přechodových funkcí všech viewportů (svázané viewporty tvoří komponenty grafu), zaregistrování obsluhy PosChanged - událost změny polohy kurzoru
8. zobrazení panelu Layers
9. inicializace prázdného workspace - 8bit 256^3
10. alokace objemu pro uložení selekce

11. nastavení a spuštění rendererů
12. umístění kurzoru do středu objemu
13. aktivace "prázdného"nástroje Cursor

Načítání dat ze souboru

Načtení projektu Soubor VEW [Příloha B.1] obsahuje hlavičku projektu, data vrstev i přechodovou funkci. Vše se přímo načte ze GZipStreamu.

Import Raw dat Soubor .raw nebo .vrt obsahuje jen posloupnost voxelů, parametry je možné načíst ze souboru .*.vh, pokud existuje. Takto získané hodnoty se pak přednastaví v dialogu ImportForm, kde je možné je upravit, pokud nesouhlasí. Hlavička může také obsahovat údaje o rozměrech voxelů, tyto se promítnou do přednastavené úpravy velikosti objemu tak, aby výsledné rozměry odpovídaly voxelům o hraně 1. U 16bit dat lze zvolit Endian. Dále jsou zde nastavení prepocessingu, zmenšení a přeškálování 12bit dat. Tlačítko "Import as Project"před importem vytvoří nový workspace s parametry podle importovaných dat (uvažováno po preprocessingu), jinak se objem načte do aktivní vrstvy.

Preprocessing Sestává z několika fází

- načtení objemu do paměti - v případě že je příliš velký, načítá se postupně po několika vrstvách podle potřeby a dříve načtené vrstvy se zahazují
- změna velikosti interpolací
- přeškálování - pokud se liší datový typ voxelů workspace a importovaných dat, provede se přeškálování automaticky, jinak jej lze vynutit v případě, že importovaný dataset je 12bit
- transformovaný objem se zkopíruje do aktuální vrstvy

Import DICOM dat Vybráním jednoho souboru .dcm se načte celý adresář (pokud jich obsahuje více) a zobrazí se DicomImportForm, kde lze zkontrolovat hlavičku souboru a vybrat volume k importu. Následuje kontrola souborů a načtení všech dostupných řezů do jednoho objemu. Tento

objem je poté prohlášen za Raw data a zobrazí se ImportForm s možnostmi preprocessingu.

Workspace a vrstvy

Workspace je základním datovým objektem editoru. V každém okamžiku je aktivní právě jeden workspace a na něm se provádějí všechny operace. Workspace může obsahovat několik objemů - vrstev (vždy minimálně jednu), které mají stejné parametry - 3 rozměry a datový typ voxelu (bitovou hloubku). Implementovány jsou typy VL.Api.eType.vl_byte a vl_ushort, neboť dostupná data jsou 8- a 16-bitová, ať už RAW objemy nebo DICOM sety. Pro podporu vl_uint a vl_float by bylo třeba doimplementovat všechny editační a transformační funkce, které se spouštějí pomocí VL.Algorithm.

Vzhledem k tomu, že .NET generiky neumožňují nad parametrizovanými typy provádět aritmetiku, nebylo možné tyto funkce implementovat obecně pro všechny dostupné typy.

Vrstvy tvoří lineárně uspořádaný seznam. Workspace udržuje seznam vrstev, provádí změny v uspořádání, přidává a odebírá vrstvy. Dále udržuje speciální vrstvu selekce a odkaz na náhledovou vrstvu, do které se zapisuje výsledek sjednocení všech vrstev z workspace.

Sjednocení probíhá podle uspořádání slučováním každé vrstvy s mezivýsledkem předchozích. Výsledek sjednocení je ovlivňován nastavením 'krytí' (blending) jednotlivých vrstev - další vrstvu je možné k předchozím přičíst, odečíst nebo spodní vrstvy překrýt (pak se hustota 0 uvažuje jako transparentní).

Vrstvy jsou pojmenované. První vrstva ve workspace se jmenuje "Base", přidávané jsou automaticky číslovány, ale jméno je možné měnit. Pojmenování slouží k lepší orientaci například při nastavování referenčního objemu u náhledu transformací.

Selekce je objem stejných rozměrů jako má workspace, ale vždy 8bit, neboť slouží jen jako maska, kde nenulová hodnota znamená, že voxel je ve výběru přítomen.

Selekce je společná pro všechny vrstvy, lze tedy v jedné vrstvě vybrat část objemu, například pomocí segmentačního nástroje, vytvořit vrstvu novou a v té danou oblast vyplnit na libovolnou hustotu. Odstranění selekce

jako vrstvy znamená vyprázdnění výběru. Pokud je výběr prázdný, editace probíhají na celém objemu, jinak vždy jen v průniku s výběrem.

3.7 GUI

Uživatelské rozhraní je vytvořeno ve WinForms. Jeho úkolem je inicializovat jádro, zprostředkovat mu přístup k menu, panelu nástrojů a definovat několik kontejnerů pro umístění komponent, jmenovitě:

- panel do kterého se vygeneruje rozložení viewportů
- prostor pro komponentu ViewportSettings
- panel, kde se zobrazí ovládání vrstev
- místo pro konfiguraci nástrojů
- podmenu a panel nástrojů, kam budou pluginy přidávat tlačítka
- ovládací prvek přechodové funkce

a předávat jádru uživatelem generované události:

- operace s workspace - New/Load/Save
- import Raw/DICOM a export vrstvy nebo sjednoceného workspace
- výběr transformační/selekční operace
- zobrazení dialogu nastavení

Také je vhodné nastavit delegáty pro výstup textových informací z knihovny VL a pro hlášení průběhu operací.

Pokud je toto dodrženo, na samotné struktuře GUI nezáleží. Současná verze je navržena v jednom okně s použitím záložek pro úsporu místa, ale nic nebrání implementaci, kde každý konfigurační panel bude plovoucí nebo v samostatném okně.

3.8 Knihovna nástrojů

Knihovna demonstruje základní sbírku nástrojů a princip fungování pluginů. Celá knihovna je implementována jako jeden plugin, soubor 'ToolLibrary.dll'. Všechny v ní definované nástroje publikují rozhraní PlugInBase.ITool.

Další nástroje lze do aplikace dodávat jako pluginy 'Tool*.dll', ve kterých budou definovány a implementovány odpovídající třídy. Editor při spuštění automaticky prohledá všechny dostupné pluginy a zaregistruje nalezené nástroje. Podrobněji se tvorbě nástrojů věnuje kapitola 4.4

3.8.1 Kategorie

V PlugInBase je definováno několik kategorií, podle nichž se při spuštění editoru vygenerují příslušná podmenu. Každý nástroj se může při registraci rozhodnout, do které kategorie se hlásí.

3.8.2 Nástroje

Modelový nástroj sestává z třídy s rozhraním ITool, vizuální komponenty s ovládacími prvky pro konfiguraci parametrů nástroje (pokud je třeba) a libovolného množství pomocných tříd a typů.

Kreslící nástroje

Všechny vytvořené kreslící nástroje umožňují odvolat úpravy pomocí Undo a pokud pracují s netriviálním výběrem, nekreslí mimo vybranou oblast.

Štětce - Sphere a Cube jsou ukázkou objemových štětců. Tvar aktuálního štětce se v editoru přímo zobrazuje jako kurzor a dává tak kontrolu nad výsledkem operace. Oba nástroje poskytují obdobnou konfiguraci - velikost štětce, "barvu-hustotu a způsob nanesení (přičtení, odečtení nebo překrytí dosavadní hodnoty).

Filtry - Fill (Plechovka), Invert, Treshold (Prahování) a Rescale (Škálování) - Velmi podobné svou strukturou - přečti voxel a spočítej novou hodnotu. Provedeno s pomocí AlgorithmEd, tedy bez možnosti pohledu k sousedům. Nejzajímavější je škálování, které požádá VolumePtr o nalezení maxima

a minima z voxelů aktivní vrstvy a tyto hodnoty nabídne uživateli jako optimální pro roztažení na celý obor hodnot podle parametrů workspace.

Globální operace - Flip (Zrcadlení) a Rotate (Překlopení) provádějí složitější transformaci a nemohou tedy využít AlgorithmEd. Proto při spuštění vytvoří dočasnou kopii objemu, kterou poté transformují zpět

3.8.3 Nástroj pro přesun

Přesune výběr nebo celý objem z aktivný vrstvy do jiné, kde provede sjednocení podobně, jak jsou definované interakce mezi vrstvami projektu

3.8.4 Segmentace

Wand (Kouzelná hůlka) je nástroj provádějící výběr pomocí seed growing segmentace. Implementován je jako 3D záplavové vyplňování a neběží paralelně.

3.8.5 Poznámky

Nástroj Memo prezentuje mechanismus ukládání dat do workspace. Ke každé vrstvě lze připsat textovou poznámku, která se při ukládání zapíše do workspace, aby byla po nahrání opět dostupná tak dlouho, dokud bude v editoru nahraný nástroj Memo. V případě, že by nástroj Memo nebyl mezi registrovanými při načítání projektu, přidaná data se zahodí a při příštím uložení už ve workspace nebudou.

Kapitola 4

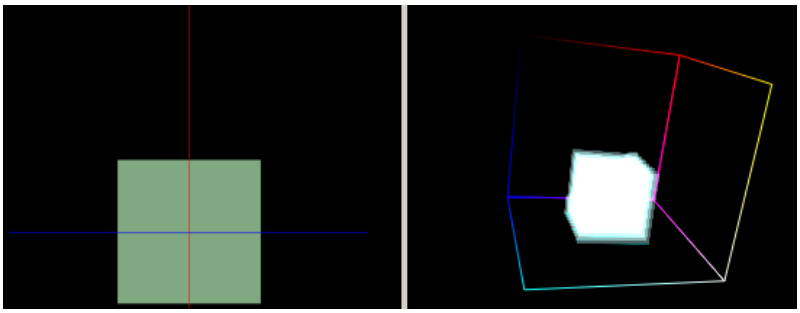
Výsledky

Testování grafického editoru má dvě základní roviny: test schopností editoru vzhledem k jeho účelu, tedy zda program poskytuje použitelné editační nástroje a je ovladatelný, a test rychlosti zpracování editačních operací, jež jsou na objemových datech výpočetně náročné.

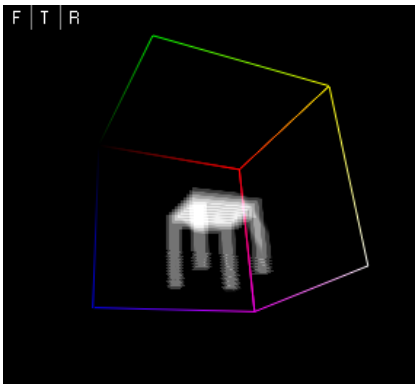
4.1 Modelování

Jako ukázku možností nástrojů jsme se rozhodli vymodelovat jednoduchý objekt a zdokumentovat postup jeho tvorby. Wang a Kaufman ve své práci jako jednu z demonstrací vyřezali z kvádru židli [17]. My jsme stejného cíle dosáhli pomocí objemových štětců a dalších nástrojů.

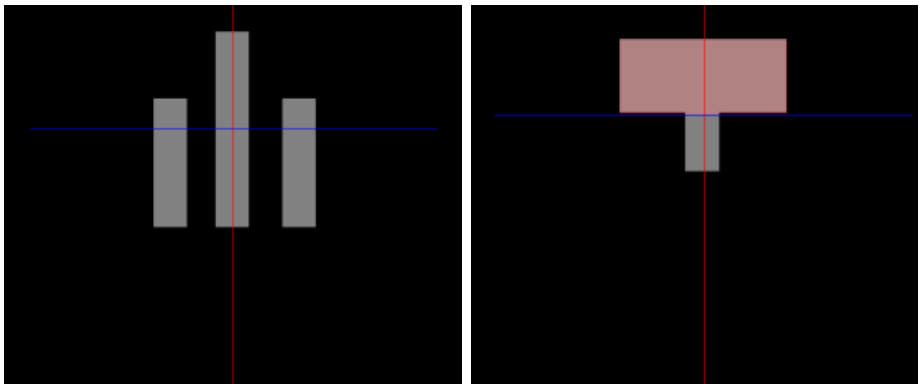
Za základní blok jsme si zvolili krychli, konkrétně krychlový štětec o hraně 100 a hustotě 200, která je vidět na obrázku 4.1. Od této krychle jsme několikrát odečetli menší štětec tak, aby vznikl sedák a 4 nohy (obrázek 4.2). Extrakcí nohy do jiné vrstvy a jejím rozkopírováním vznikl základ opěradla, jehož vrchní část jsme vytvořili použitím plechovky na kvádrový výběr (obrázek 4.3). Hranatý vršek nepůsobí dobře, využili jsme proto možnost přímo upravovat objem definující výběr a určili voxely k ořezání, jak je vidět na obrázku 4.4. Samotné ořezání provedeme nástrojem *plechovka* nastaveným na hodnotu 0. Nyní stačí opěradlo a sedák sloučit do jedné vrstvy. Obrázek 4.5 ukazuje výsledek po exportu do programu ImageJ [11] a softwarovém renderingu.



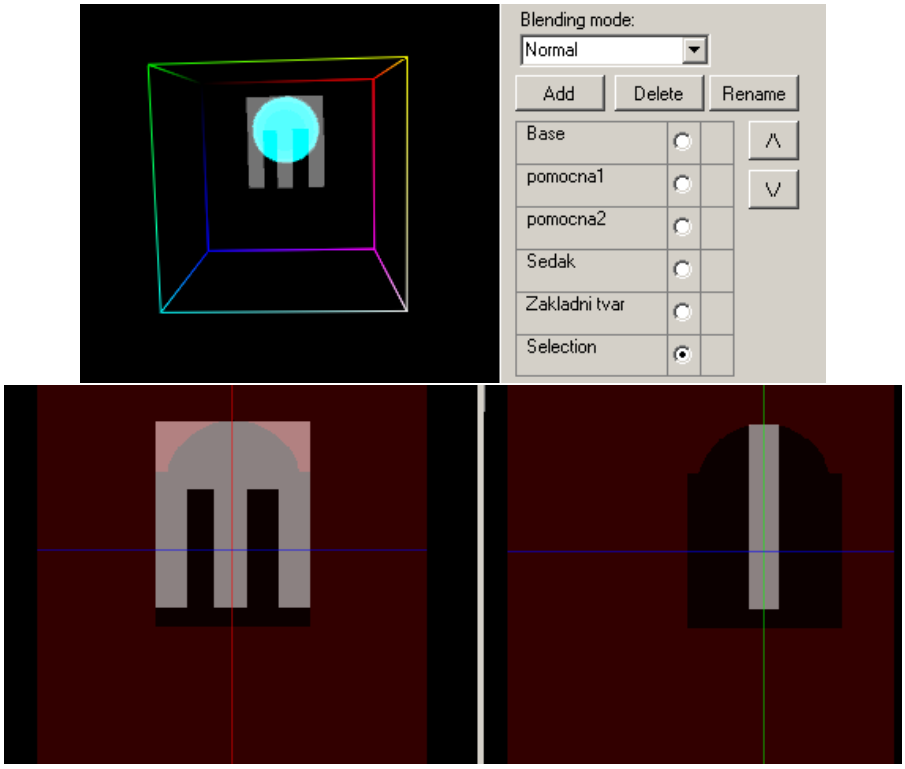
Obrázek 4.1: Krychlový štětec a nakreslený objem ve 2D i 3D pohledu.



Obrázek 4.2: Štětec s nulovou hustotou maže.



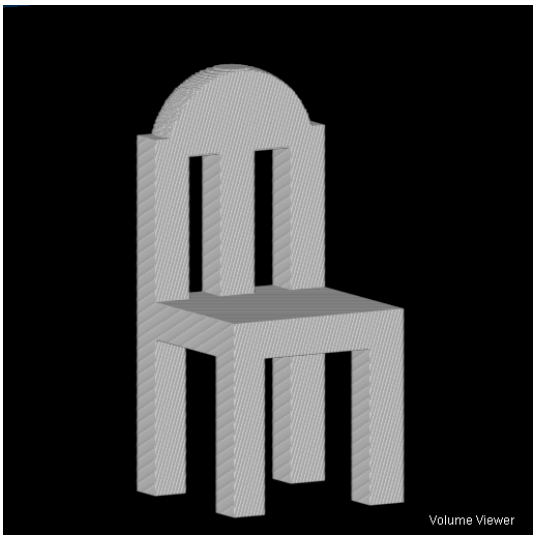
Obrázek 4.3: Kostra opěradla a výběr určený k vyplnění.



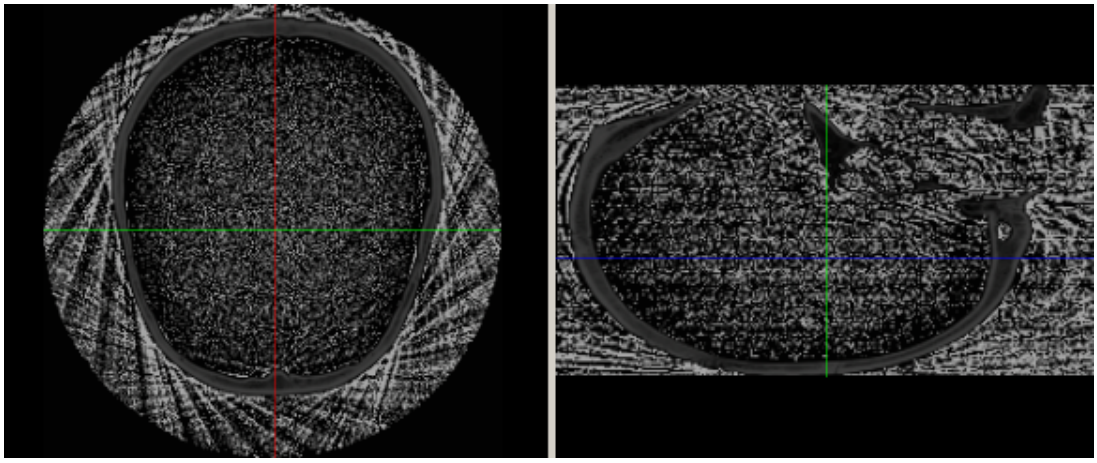
Obrázek 4.4: Vlevo kulový štětec použitý na selekci. Vpravo invertovaná selekce určuje, které voxely se smažou (nastaví na nulovou hustotu). Dole pak výsledný tvar.

4.2 Čištění získaných dat

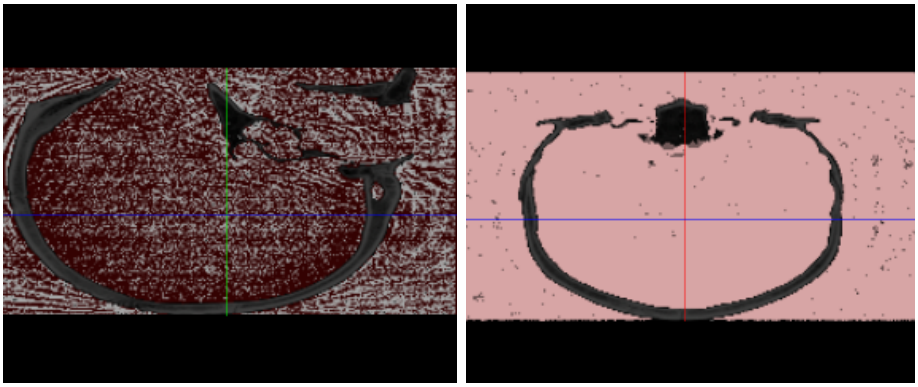
Z průmyslového výpočetního tomografu jsme získali sken lebky - milimetrové řezy v rozlišení 1024x1024. Protože editor pracuje s homogenními voxely, převedli jsme objem do rozlišení 256x256x138, které umožňuje zpracování i na pomalejším počítači s menší grafickou pamětí. Data jsou velmi zašuměná, jak je vidět na obrázku 4.6. Nástrojem *kouzelná hůlka* jsme vysegmentovali část šumu (obrázek 4.7 vlevo) a vzniklý výběr jsme vyplnili na hustotu z opačné části spektra. To umožnilo vybrat opakováním segmentace větší část "objemu pozadí"(obrázek 4.7 vpravo) a postup opakovat. Po několika iteracích je šum pryč a lebku vidíme na obrázku 4.8.



Obrázek 4.5: Softwarový render výsledného modelu. Delší výpočet dává kvalitnější vizualizaci oproti rychlému zobrazování na starší GPU.



Obrázek 4.6: Kvůli šumu není vlastní lebka téměř vidět.

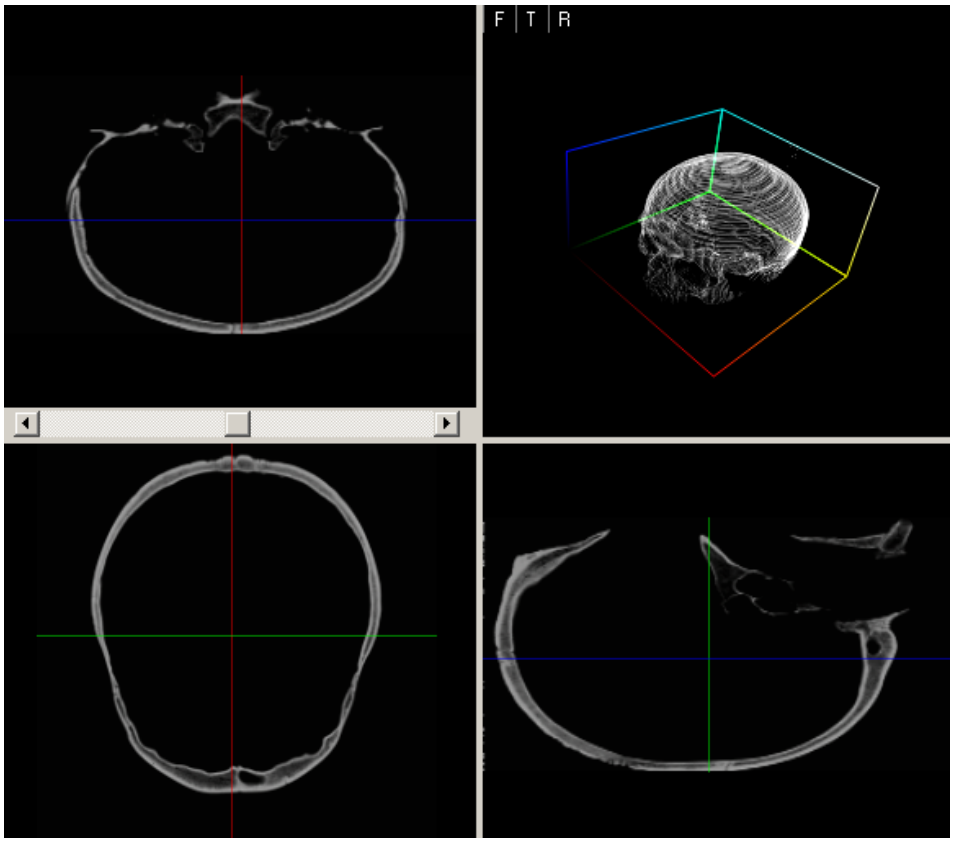


Obrázek 4.7: Vlevo: Tmavá složka šumu byla z větší části vybrána. Vpravo: Většina šumu byla již nastavena na kontrolovanou hodnotu.

4.3 Měření rychlosti

VL framework poskytuje možnost paralelizace časově náročných operací, včetně použití tříd Algorithm nebo AlgorithmEd na zpracování objemu po voxelech a řezech. V obou případech je implementována granularita dat po řezech. Ke zpracování se pak použije tolik vláken, kolik je v systému logických procesorů. Těmto vláknům jsou postupně přidělovány dosud nezpracované řezy. Tuto paralelizaci lze zakázat nastavením flagu OneThreadOnly, což využíváme k porovnání seriového a paralelního zpracování.

Tabulka 4.1 shrnuje výsledky testů na notebooku s dvoujádrovým procesorem, tabulka 4.2 pak ukazuje časy dosažené na PC s procesorem Core i7 (čtyřjádrový s hyperthreadingem). Každou testovanou akci jsme provedli na stejném objemu pětkrát s paralelizací a pětkrát bez paralelizace a výsledný čas představuje průměr z těchto pokusů. Důvodem takového postupu byla snaha omezit vliv náhodných krátkodobých interferencí ze strany operačního systému. Výsledky ukazují, že paralelní zpracování není příliš výhodné pro lokální editaci s pomocí štětců, případně při použití filtrů omezených malým výběrem. Naopak při zpracování většího objemu se zrychlení zvětšuje s tím, jak narůstá složitost výpočtu jednoho každého voxelu (složitost "kernel"funkce algoritmu). Test rychlosti segmentace proběhl pouze na seriové verzi, neboť standardní rozložení na nezávislé řezy není možné a paralelní seed growing segmentace nebyla implementována.



Obrázek 4.8: Výsledný vyčištěný objem.

Akce	Jedno vlákno	Dvě vlákna
Kulový štětec o průměru 30	1,09s	1,60s
Vyplnění celého objemu 256^3	2,47s	2,41s
Vyplnění výběru 256^3	11,18s	8,23s
Posunutí	5,74s	4,11s
Otočení	16,62s	10,20s
Segmentace měkkých tkání hlavy	4,99s	X

Tabulka 4.1: Výsledky testů na dvoujádře Intel T3400 2x 2,16GHz, 3GB RAM, GPU Ati Mobility Radeon 3430HD, Windows 7 32bit. Testován byl Release build, několik namátkových testů Debug verze dává časy přibližně dvojnásobné.

Akce	Jedno vlákno	Dvě vlákna
Kulový štětec o průměru 30	0,44s	0,56s
Vyplnění celého objemu 256^3	1,70s	1,34s
Vyplnění výběru 256^3	7,31s	3,02s
Posunutí	3,09s	1,63s
Otočení	9,61s	2,99s
Segmentace měkkých tkání hlavy	2,61s	X

Tabulka 4.2: Výsledky testů Intel Core i7 920, 8x 2,67GHz, 3GB RAM, GPU NVidia GeForce 210, Windows XP 32bit.

4.4 Tvorba nástrojů

Jedním z cílů při vývoji tohoto software byla možnost dalšího rozšiřování. Rozhodli jsme se pro vytvoření jádra editoru s možností dodávat další nástroje jako pluginy. K tomuto účelu lze s výhodou využít možnosti .NET Frameworku, konkrétně možnost spolupráce několika Assemblies (CLI verze dynamicky linkovaných knihoven) a jejich analýza pomocí Reflections.

Navrhli jsme minimální rozhraní, které musí nástroj implementovat, a sadu metod, kterými může komunikovat s editorem. Tyto definice jsou popsány v knihovně PlugInBase.dll, jež tak slouží jako spojovací článek mezi pluginem a editorem.

Jako ověření tohoto návrhu je celá knihovna standardních nástrojů implementována jako plugin, čímž jsme docílili toho, že celý systém je nejen funkčně navržený, ale také funkčně implementovaný.

Jak vypadá plugin

Plugin je samostatná Assembly představovaná jedním soborem 'Tool*.dll' a umístěná v adresáři aplikace v podadresáři 'PlugIns'. Tato Assembly by měla definovat alespoň jednu třídu implementující rozhraní PlugInBase.ITool a referencovat knihovny 'PlugInBase.dll' a 'VL.dll' (parametr Copy Local může být false, dokonce je to vhodné). Plugin může obsahovat několik tříd s rozhraním ITool a mimo to libovolné jiné třídy a typy.

Editor při startu projde všechny pluginy a pokusí se registrovat jako nástroj každou třídu, která je public a implementuje ITool. V případě problémů ohlásí chybný nástroj a pokračuje dalšími. Přehled načtených pluginů

a jejich nástrojů lze za běhu najít na druhé záložce dialogu Settings.

Požadavky na nástroj

- Nástroj musí být definovaný jako public a implementovat ITool, nesmí být abstraktní. Pokud chcete použít abstraktního předka, deklarujte ho jako private.
- Každý nástroj používá unikátní jméno jako identifikátor, případná kolize vyřadí nástroj z dalšího provozu.
- Je implementována metoda RegisterTool(), která zažádá o vytvoření tlačítka a nebo položky v menu nástrojů, bez toho bude Váš nástroj uživateli nedostupný.
- Dále jsou nutné metody Activate() a Deactivate(), které reagují na stisk příslušného tlačítka nebo položky a obsluhují zobrazení případného konfiguračního panelu.

Možnosti nástroje

Nástroj splňující základní požadavky může s uživatelem interagovat pomocí vlastních ovládacích prvků na konfiguračním panelu a pracovat se všemi metodami editoru (tyto jsou definované v rozhraní PlugInBase.IVoledit) či využívat možnosti VL. Většinou však nástroje spadají do některé z definovaných kategorií a implementují navíc odpovídající interface:

- Kreslící nástroje, ať už štětce, geometrická primitiva nebo nástroje pro voxelizaci implicitních funkcí stejně jako filtry, využijí interface IPaintTool, který umožňuje vyvolat akci na žádost uživatele a zaregistrovat Undo akci.
- ISelectionTool slouží hlavně pro segmentační nástroje, i když pracovat s ním může i třída pro zadávání jinak parametrizovaných výběrů.
- Pro případy přesunů mezi vrstvami je tu ITransferTool, který může být také základem importních a exportních pluginů.

Undo

Kreslící nástroj by měl pro pohodlí uživatele zvládat odvolání provedené akce. Součástí volání Do() je i parametr typu UndoInfo, který umožňuje zabalit a uschovat libovolná data potřebná k navrácení objemu do předchozího stavu (v nejjednodušším případě serializovanou verzi celého objemu, což ale může být paměťově náročné) a flag, který značí že Undo je platné a proveditelné. Nástroj se tedy může rozhodnout o odvolatelnosti akce až na základě konkrétních parametrů. V případě, že se uživatel pokusí Undo použít, zavolá se příslušná metoda a v parametrech dostane odpovídající UndoInfo a objem, kterého se povel týká.

Load a Save

Při ukládání workspace umožní editor každému nástroji zapsat nějaká data pomocí BinaryWriteru za účelem permanentního uložení. Když metoda Save() není implementována, nebo na konci neprovede potvrzení zápisu, nestane se nic.

Pokud během otevírání workspace narazí editor na taková data, pokusí se vyhledat odpovídající nástroj a data doručit voláním Load().

Ukázková akce kreslícího nástroje

Objem je reprezentován pomocí tříd VL a k práci s ním doporučujeme použít VL.Algorithm nebo AlgorithmEd. Definice samotné akce se provede pomocí krátké funkce, jež obdrží vstup a vrátí vypočítanou hodnotu voxelu (to v případě práce *per-voxel*), nebo dostane odkaz na výstupní objem a zpracuje jeho přidělenou část.

Následující ukázka předvádí implementaci inverze pomocí editačního algoritmu:

```
public void Do(VL.Api.IVolume vol, PlugInBase.UndoInfo undo)
{
    // počkáme na dokončení případných předchozích akcí
    vol.WaitForFinish();
    // vytvoříme objekt editačního algoritmu
    VL.AlgorithmEd a = new VL.AlgorithmEd();
    // podle datového typu objemu zavoláme odpovídající
    // implementaci
    switch (vol.Params.DataType)
```

```
    {
        case VL.Api.eType.vl_byte:
            // algoritmus poběží po voxelech
            // vstup a výstup jsou totožné, jiný vstup není
            a.ExecuteOnVoxels<byte>(paint, vol, null, vol);
            break;
        case VL.Api.eType.vl_ushort:
            a.ExecuteOnVoxels<ushort>(paintU, vol, null, vol);
            break;
        default:
            throw new NotImplementedException
                ("Current voxel type not supported");
    }
    // inverze je jednoduše reverzibilní
    // proto si jen poznamenáme, že undo je možné
    undo.Data = null;
    undo.IsValid = true;
}

// "kernel" funkce pro zpracování jednoho voxelu
// dostane souřadnice voxelu, třídu pro přístup ke vstupu
// a uživatelské parametry (v tomto případě null)
public byte paint(int x, int y, int z,
      VL.Api.DataAccess[] inputData, object userParams)
{
    // načtení původní hodnoty ze vstupního objemu
    byte t = inputData[0].Get<byte>(x, y, z);
    byte s = byte.MaxValue;
    // výpočet inverze a předání nové hodnoty
    return (byte)(s - t);
}
```

Vzhledem k nemožnosti používat standardní aritmetiku v generických metodách je třeba implementovat všechny verze funkce volané algoritmem - byte a ushort stejně jako varianty pracující s výběrem.

Kapitola 5

Závěr

V této práci jsme navrhli a implementovali jednoduchý editor pro práci s objemovými daty. Náš návrh buduje základní strukturu pro interaktivní práci s objemovými daty a dává k dispozici několik jednoduchých nástrojů, přičemž počítá s dalším rozšiřováním, pro něž poskytuje dostatečné zázemí.

Z technik představených ve druhé kapitole jsme v nástrojích implementovali transformace, filtry a objemové štětce, ale tvorba dalších nástrojů není těmito technikami omezena.

Struktura aplikace dovoluje kromě jednoduchého přidávání sady nástrojů i výměnu některých modulů samotného editoru. Taková úprava není proveditelná na úrovni pluginů, ale případný zásah do kódu editoru se bude týkat jen měněné části, nebo napojení nového modulu, potřebná rozhraní jsou definována obecně. Takto lze změnit GUI nebo přidat nový 3D renderer.

Vytvořený editor se liší od exitujících aplikací podobného zaměření tím, že je zaměřený obecně na editaci objemových dat pomocí nástrojů. Od začátku pracuje s objemem ve třech rozměrech a přitom využívá a zobecňuje principy použité v běžném kreslení. Navrhli jsme strukturu editovaného projektu podobnou kvalitním rasterovým editorům - možnost pracovat s několika vrstami, samostatnými objemy, které spolu interagují podle potřeb uživatele, nebo práce s výběrem ovlivňujícím chování nástrojů.

V implementaci chybí několik vlastností ze specifikace:

- Není nastavitelné krytí (průhlednost) při slučování vrstev.
- Při importu probíhá případné převzorkování trilineární interpolací namísto interpolace vyššího řádu nebo nastavitelného filtru.

- Bounding box výběru se neupravuje postupně při kreslení do masky, ale až před použitím nástroje, který se výběrem bude řídit. Toto způsobuje několikavteřinové zpoždění před prvním použitím výběru nástrojem. Proto je v nastaveních možnost generování obalového kvádru zakázat. Výsledkem je zhoršení efektivity všech operací, protože musí vždy projít celý objem, a větší paměťová náročnost Undo.

5.1 Možná budoucí vylepšení

Multiplatformnost

V současnosti prochází betatestováním knihovna OpenTK, která by měla pokrývat většinu funkcí frameworku Tao a to jak na Windows, tak v prostředích kompatibilních s Unix/X11. Podle propagace by navíc měla být s Tao částečně kompatibilní pro usnadnění portování už hotových projektů. Tato iniciativa by stála za prozkoumání, ale mezi nutné úpravy pro takový případ patří přepsání všech OpenGL volání.

Identifikace akcí

Všechny složitější výpočty hlásí svůj průběh do společného počítadla bez možnosti identifikace jednotlivých akci. Následkem toho není možné v editoru vytvořit složitější strukturovaný ukazatel průběhu, neboť mezi spuštěním a ukončením akce není žádná vazba kromě počtu kroků. Kdyby se tento systém doplnil o generování a udržování identifikátorů jednotlivých akcí, dal by se zobrazovat průběh různých operací nezávisle například pomocí proměnného počtu progress barů.

Náhledy úprav

Globální úpravy by mohly zobrazovat náhled v nízkém rozlišení, což by se hodilo například pro doladění parametrů prahování. Tuto funkci může implementovat samotný nástroj.

Import a export, nástroje

Pro zlepšení využitelnosti editoru veřejností by se hodila možnost zpracovávat datové formáty používané pro modifikaci počítačových her. V ta-

kovém případě lze vytvořit mapování mezi barvami v herních formátech a hustotou v reprezentaci VL frameworku pomocí přechodové funkce.

Knihovna obsahuje jen základní a spíše prototypové nástroje. Hlavním směrem další tvorby by měly být mimo jiné štětce různých tvarů nebo geometrická primitiva.

Kreslení pomocí křivek

Ovladatelnosti by prospěla možnost zadávat křivky jako cesty, podél kterých by pak kreslil štětec. To ovšem vyžaduje podporu v rendereru a použitelnou metodu zadávání křivek ve 3D.

Příloha A

Uživatelská dokumentace

Ovládání editoru je navrženo tak, aby co nejvíce dodržovalo běžné zvyklosti. Nabízí klasicky strukturované menu a běžný panel nástrojů. Mimo to však má uživatelské rozhraní některá specifika, například využití záložek z důvodů úspory místa.

A.1 Instalace

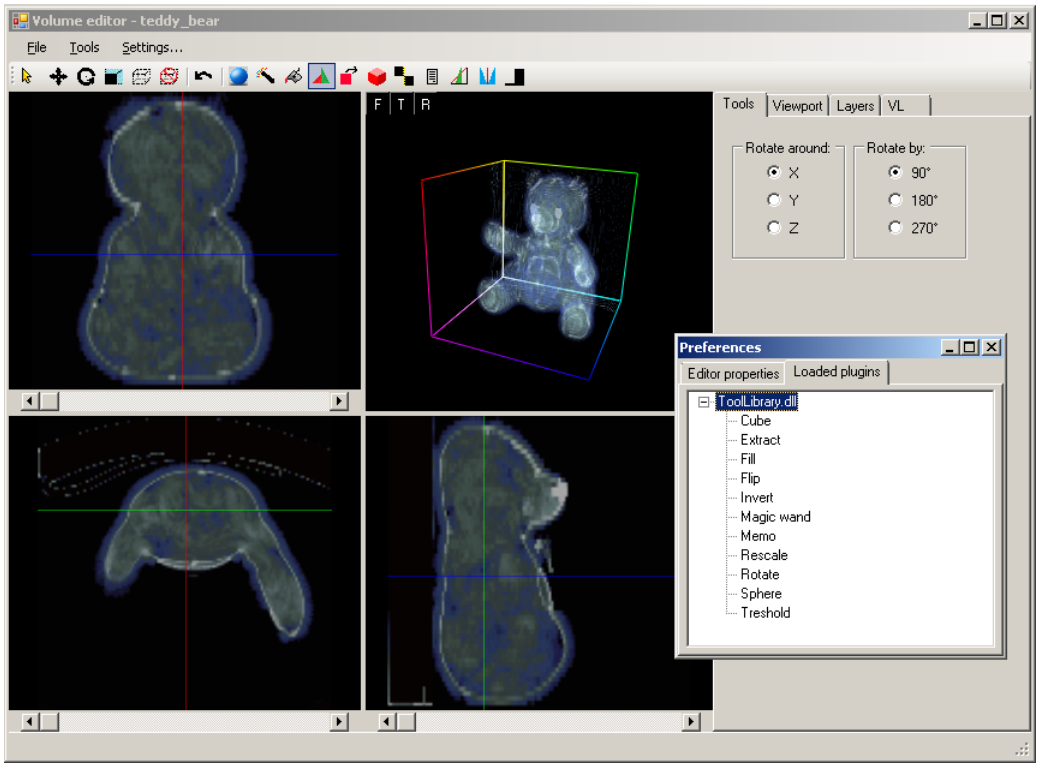
Na přiloženém CD (podrobněji příloha C), lze najít jak zdrojové kódy editoru (včetně potřebných knihoven), tak samorozbalovací archivy se zkompilovaným programem pro 32- i 64-bitové systémy. Archiv stačí rozbalit do libovolného adresáře.

A.2 Hlavní okno

A.2.1 Menu a panel nástrojů

Hlavní menu:

- File obsahuje příkazy pro práci s workspace
  - New Workspace - zobrazí dialog pro zadání rozměrů a datového typu. Po potvrzení (tlačítkem Create) se zruší aktuální workspace a vytvoří nový prázdný.
  - Load/Save Workspace - zobrazí dialog pro výběr souboru k otevření nebo uložení workspace (soubory typu *.vew)



Obrázek A.1: Hlavní formulář aplikace se zobrazením dvoj- a trojrozměrných pohledů na scénu. Na popředí je pak dialogové okno s výpisem aktivních pluginů a nástrojů. Zobrazený objem je teddy_bear.raw [16].

  - Import Raw.. - Po výběru souboru zobrazí ImportForm
  - Import DICOM.. - Po výběru souboru zobrazí DicomImportForm s výpisem jeho obsahu. Kliknutí na některý volume (modrý text) tento vybere a po chvíli načítání pokračuje zobrazením ImportForm
  - Export Raw
    * Layer.. - uloží aktuální vrstvu do souboru
    * Workspace.. - provede sjednocení všech vrstev a do zvoleného souboru uloží výsledek

- Tools zpřístupňuje nástroje nahrané z knihovny roztříděné do základních kategorií

  - Drawing - štětce a editační nástroje s lokálním rozsahem
  - Selection - segmentační a další nástroje pro úpravu výběru
  - Filters - editační nástroje globálního rozsahu
  - Transfer - nástroje pro přesun obsahu mezi vrstvami
  - Other - ostatní
- Settings.. - zobrazí dialog pro nastavení několika vlastností systému (kvalita náhledů pro rotaci apod.)

Panel nástrojů obsahuje tlačítka:

- kurzoru (neprovádí úpravy dat)
- integrovaných nástrojů (Move, Rotate, Select, Resize)
- funkce "Undo"(Zpět)
- nástrojů z knihovny

A.2.2 Viewporty

Čtyři viewporty zabírají většinu okna. Ve výchozím rozložení jsou to tři 2D ortogonální pohledy pro různé směry základních řezů a jeden 3D renderer s perspektivním zobrazením. Oddělovače mezi viewporty umožňují měnit jejich velikosti.

Data jsou vykreslena barevně podle přechodové funkce. Přes ně se poloprůhledně zobrazuje kurzor (zelenou barvou) a aktuální výběr (červeně).

2D pohled zobrazuje pozici interního kurzoru - ve svém směru jako řez a v dalších dvou rozměrech barevnými čarami. Souřadnici určující řez lze měnit kolečkem myši nebo posuvníkem pod pohledem.

Kamera 3D pohledu se ovládá pomocí myši následovně:

- Rotace - se stisknutým kolečkem
- Zoom - se stisknutými levým a pravým tlačítkem

Také lze kameru nastavit na předem zvolené směry pohledu pomocí tlačítek v levém horním rohu 3D rendereru (F, T, R)

Pozice kurzoru se mění pohybem se stisknutým pravým tlačítkem v rovině kolmé na směr pohledu. Polohu této roviny lze upravit otáčením kolečka myši (i bez tlačítka) nebo pohybem myši se současným stisknutím pravého tlačítka a klávesy Control

Stisknutím levého tlačítka uvnitř libovolného pohledu dojde ke spuštění aktuálního (editačního nebo segmentačního) nástroje z knihovny

A.2.3 ProgressBar

Ve spodní části okna se během déletrvajících akcí objevuje ukazatel zobrazující průběh jejich vykonávání a případně popisek.

A.2.4 Záložky

VL Do textboxu se vypisují textové výstupy různých akcí, jako jsou například doba načítání objemu ze souboru a jeho parametry, nebo čas provádění algoritmu.

Tools Obsah této záložky se mění v závislosti na vybraném nástroji. Když je aktivní pouze kurzor, je tato záložka prázdná.

Viewport Vždy zobrazuje editor přechodové funkce. Křivku přechodové funkce lze upravovat kreslením levým tlačítkem. Na výběr pro úpravu jsou barevné komponenty Red, Green a Blue i průhlednost Alfa. Na pozadí křivky se zobrazuje histogram objemu aktuální vrstvy.

Dále lze 'aktivovat' libovolný viewport stiskem tlačítka myši (i kolečka), čímž se zobrazí další možnosti:

- Výběr rendereru (různé 2D pohledy nebo 3D)
- Tabulka závislosti kurzoru a sdílení přechodové funkce. Při posunutí kurzoru v jednom viewportu se nová poloha převede do všech viewportů, které na něm závisí. Viewporty, které sdílejí přechodovou funkci, používají stejné mapování mezi hustotou voxelu a zobrazovanou barvou.
- Nastavení 3D vizualizace (pouze pokud daný viewport zobrazuje 3D)

Layers Seznam vrstev v otevřeném workspace. Aktuální vrstvu lze zvolit přepínačem. Tlačítka slouží k přidávání, mazání a pojmenování vrstev a k jejich přeuspořádání.
"Blending mode"umožňuje nastavit způsob sjednocení vrstvy s předchozími vrstvami:

- Normal - obsah této vrstvy přepíše předchozí vrstvy kromě voxelů s hustotou 0, ta určuje transparentní voxel
- Add - voxely se sčítají
- Subtract - hustota voxelů této vrstvy se odečte od voxelů sjednocení předchozích vrstev

Vrstva "Selection"obsahuje definici výběru. Stiskem tlačítka Delete dojde pouze ke zrušení výběru, ne k odstranění vrstvy samotné. Pokud je tato vrstva vybrána jako aktuální, je možné výběr upravovat pomocí štětců i filtrů stejně jako při editaci objemu.

Tlačitko "Merge layers and show"provede výpočet sjednocení všech vrstev a výsledek dočasně zobrazí ve všech viewportech. Zruší se vybráním některé vrstvy. Data jednotlivých vrstev tento výpočet neovlivní, jde pouze o náhled.

A.3 Dialogová okna

New workspace

Definuje rozměry a bitovou hloubku nového workspace. Při zobrazení načte hodnoty aktuálně otevřeného workspace. Tlačítko Create vytvoří workspace, Cancel zachová původní.

Import settings

Při importu RAW nebo DICOM datasetu lze nastavit parametry importovaných dat a transformace, které se při importu provedou.

Data Format Určuje rozměry a bitovou hloubku zdrojových dat. Při otevření načte hodnoty z hlavičky (DICOM resp. soubor *.vh u RAW dat). Pokud hlavičku nenajde, zobrazí varování. U vícebytových dat je možné vynutit konverzi z Big Endian

Resize Při importu je možno objem zvětšit/zmenšit. Zvětšovací faktor je nastavitelný ve třech rozměrech nezávisle, nebo výběrem "Homogenous"se řídí šířkou. Když hlavička datasetu obsahuje informace o rozměrech voxelů, nastaví se zvětšení, aby transformovaný objem měl stejné rozměry a přitom voxely o hraně 1. Pokud je RAW dataset větší než 100MB, provede se případné zmenšení postupným načítáním tak, aby se snížila paměťová náročnost.

Rescale Při importu dat rozdílné bitové hloubky se automaticky provádí přeškálování podle parametrů workspace, ale některé dvoubytové datasety jsou 12bitové namísto 16bit, a tak při importu "Unsigned Short"dat je tu možnost zvolit odpovídající korekci.

Potvrzení Tlačítko Import zahájí načítání, transformaci a načtení dat do aktuální vrstvy. Cancel zruší celý import. "Import as Project"nejdříve zruší otevřený workspace a vytvoří nový s parametry podle importovaných dat (včetně změny velikosti a přeškálování).

Settings

Nastavuje globální proměnné editoru.
Velikost náhledů v procentech - menší náhled lze rychleji aktualizovat pro získání o výsledku představy

- velikost náhledu při posouvání
- velikost náhledu při rotaci

Zda používat vícevláknové implementace algoritmů. Počet úprav, které je možné vrátit tlačítkem "Undo"- data potřebná pro zrušení efektu filtrů a jiných nástrojů pracujících nad celým objemem nebo alespoň jeho velkou částí jsou nezanedbatelná a toto nastavení by mělo odpovídat velikosti operační paměti i editovaného objemu.

Předvolba určující, zda po úpravě selekce spočítat její obalový kvádr. Tato operace zabere nějaký čas, ale následné operace probíhají už jen na takto zmenšeném objemu, což výrazně šetří čas i paměť pro Undo.

A.4 Nástroje

Nástrojů jsou dva druhy - integrované (kurzor, transformace a obdelníková selekce) a pak externí shromážděné v knihovně, která se načítá při spuštění a případně v dalších pluginech.

Každý nástroj může při aktivaci určit obsah záložky Tools - zobrazit ovládací prvky pro konfiguraci.

A.4.1 Integrované nástroje

Cursor () Kurzor neprovádí žádnou operaci nad objemem, pouze interaguje s GUI

Move, Rotate a Resize ( ,  , ) Parametry transformace je možné nastavit myší s realtime náhledem (ve sníženém rozlišení kvůli rychlosti) nebo na záložce "Tools". Nějaká vrstva (defaultně ta transformovaná) se zobrazuje jako referenční ve formě dat a transformovaná nahradí kurzor. Parametry se volí myší:

- Posunování probíhá stejně jako změna pozice kurzoru s pravým tlačítkem nebo posunem kurzoru libovolného 2D pohledu - souřadnice kurzoru určují střed posunované vrstvy
- Při rotaci se zmrazí otáčení 3D kamery a namísto ní při zmáčknutí kolečka myši rotuje náhled transformované vrstvy. Přepočítávání nové rotace je výpočetně náročnější než posunutí, a tak může docházet ke zpoždění a skokům v zobrazení podle výkonu počítače a nastavené kvality náhledu.

Výběr nástroje zapne režim náhledu transformace

- do záložky Tools se přidají ovládací prvky nástroje a ReferenceControl, který umožňuje vybrat, jaká vrstva se zobrazuje jako reference
- vygenerují se zmenšené náhledy transformované a referenční vrstvy

Po každé změně parametrů se aktualizuje náhled. Transformaci lze potvrdit stiskem tlačítka na záložce "Tools"- dojde k přepočítání původních dat podle zadaných parametrů (transformovaný objem nahradí původní netransformovaný obsah vrstvy). Tlačítko Cancel nebo aktivace jiného nástroje zruší režim náhledu beze změny dat.

Select () Kvádrová selekce se provádí změnou pozice kurzoru ve 3D rendereru Pohybem myši se stisknutým pravým tlačítkem. Vykresluje se jako kvádrový kurzor. Druhým stisknutím se kurzor zakreslí do selekce jako vybraná oblast

Undo () Každý stisk odvolá jednu akci. Pokud není žádná odvolatelná v zásobníku, tlačítko je šedé. Název akce, která je další na řadě, se zobrazuje jako "ToolTip"nad tlačítkem při najetí myší.

A.4.2 Knihovna nástrojů

Knihovna je balík základních editačních a jiných nástrojů pro práci s objemem. Každý nástroj může při načítání umístit své tlačítko včetně obrázku nebo popisku na panel nástrojů a také položku do zvolené kategorie v menu Tools. Nástroje jsou načítány v pořadí určeném .NET frameworkem a operačním systémem, takže uspořádání tlačítek na panelu nástrojů se může měnit, i když v rámci jednoho počítače zůstává většinou stabilní.

Nástroj se aktivuje stiskem příslušného tlačítka nebo položky menu a deaktivuje výběrem jiného nástroje nebo změnou ve workspace (výběr jiné vrstvy, load/import).

Funkce kreslících nástrojů se spouští kliknutím levým tlačítkem do některého z viewportů.

Sphere a Cube ( , ) Nejjednodušší objemové štětce. Umožňují přímo kreslit do dat. Kurzor se změní na odpovídající tvar a rozměr.
Nastavitelné parametry jsou:

- Diameter/Edge - velikost štětce (změna se projevuje na kurzoru)
- Density - "barva"štětce
- Blending
  - Fill - nahradí původní hodnotu zvolenou hustotou
  - Add - přičte hustotu k aktuální hodnotě voxelu
  - Subtract - odečte od aktuální hodnoty hustotu štětce

Pokud existuje neprázdný výběr, tak nekreslí mimo vybranou oblast (ořezává se výběrem).

Wand - Kouzelná hůlka () Seed growing segmentace spouštěná z pozice kurzoru. Nástroj pro úpravu selekce. Pokud vybírá větší část objemu, je časově náročná. Nelze zobrazit rozumně průběh pomocí ProgressBaru, tak se v jeho popisku zobrazuje aktuální velikost fronty zpracovávaných voxelů. Parametry:

- Treshold - tolerance pro zahrnutí voxelu do výběru - absolutní rozdíl hustot
- Method
  - Area - tolerance se počítá vzhledem ke startovnímu voxelu
  - Border - zahrnuty jsou všechny voxely kromě těch, které mají zadanou hodnotu. Jde o výběr nehomogenní oblasti s přesně definovanou hranicí.
  - Gradient - tolerance platí pro rozdíl sousedních voxelů

Fill () Slouží k vyplnění vybrané oblasti nebo celé vstvy jednou hodnotou. Hustota výplně je jediný parametr.

Invert () Provede inverzi objemu tak, že v každém bodě odečtě aktuální hodnotu od maximální. Pokud existuje selekce, zpracuje jen vybrané voxely. Nemá konfigurovatelné parametry.

Treshold () Prahování objemu podle nastavené hraniční hustoty. Navíc nabízí výběr ze tří metod:

- Theshold - klasické binární prahování
- Lesser - zachová hodnoty menší než práh a ty větší nastaví na maximální hustotu
- Greater - zachovává vyšší hodnoty a nuluje nižší

Extract () Nástroj pro extrakci části objemu do jiné vrstvy. Pokud není selekce, kopíruje celý objem, jinak jen vybranou oblast. Na záložce Tools umožňuje vybrat cílovou vrstvu pokud je nějaká dostupná a zahájit extrakci do této vybrané vrstvy (kde přepíše původní obsah) nebo do nové, kterou vytvoří. Také je možné zvolit metodu sjednocení s obsahem cílové vrstvy.

Memo () Na záložce Tools se v textovém poli zobrazuje/upravuje popisek aktuální vrstvy. Stiskem tlačítka "Save"dojde k zapamatování popisku. Při ukládání workspace na disk se tyto popisky ukládají také.

Rescale () Po aktivaci analyzuje objem aktuální vrstvy a v konfiguraci navrhne nalezené extrémy přeškálovat na celý rozsah použitého datového typu voxelu.

Flip () Podle nastavení provede zrcadlení podle roviny půlící objem v jednom ze 3 hlavních řezů.

Rotate () Otáčí objem o násobky 90° podle jedné z hlavních os procházejících středem objemového kvádru.

Příloha B

Detaily implementace

B.1 VEW - formát souboru

Soubor VEW obsahuje jeden GZip stream (prakticky je to archiv obsahující jediný nepojmenovaný soubor) a všechny další operace probíhají nad obecným streamem. Samotná data jsou rozdělena do chunků čtyř typů:

- WorkspaceHeader - obsahuje zápis VolumeParams, počet vrstev (int) a stav počítadla pro generování jmen nových vrstev
- LayerData - reprezentace třídy Layer - hlavička (jméno, BlendingMode a příznak, zda vrstva obsahuje alokovaný VolumePtr) a data případného objemu zapsaná pomocí VolumePtr.SaveToStream()
- TFData - LUT přechodové funkce zapsaná jako počet hodnot a odpovídající množství čtveřic složek RGBA
- SpecialData - schránka pro data, která žádá uložit některý nástroj, obsahuje ID nástroje (string) a data
- SpecialData s nulovou délkou - slouží jako indikátor konce souboru

Chunk tvoří typ (přetypovaný na int), velikost dat (int64) a samotná data. V současné verzi parseru je vyžadováno pevné pořadí chunků - jedna hlavička, všechny vrstvy ve správném pořadí, přechodová funkce, libovolný počet záznamů pro nástroje (teoreticky dovoluje i několik záznamů pro jeden nástroj, ale takový soubor v programu nevznikne) a nakonec prázdný SpecialData. Pokud parser narazí na chunk, který nezná, nebo který se nepatří na dané místo, korektně ho přeskočí a pokračuje dalším.

Bylo by možné implementovat parser, který nejprve každý chunk přečte, "rozbalí"a data pošle ke zpracování příslušné metodě. V současnosti však je tento formát využíván pouze prezentovaným projektem a pořadí je garantováno.

B.2 Metoda otáčení kamery

Námi implementovaná metoda spočívá v uchovávání aktuální rotační matice a jejím postupném otáčení podle pohybu myši, přičemž osa rotace je definována směrem pohybu a úhel určen vzdáleností, podrobně:

1. zjistíme relativní pohyb myši - buď přímo nebo jako rozdíl od minulé polohy
2. uložíme délku tohoto pohybu
3. v rovině okna vypočteme kolmici na pohyb myši $(-y, x)$ - technická poznámka: při převodu souřadnic z okna WinForms do okna OpenGl je $Y_{OGL} = Height - Y_{WF}$
4. vzniklý vektor doplníme třetím rozměrem $z = 0$
5. a provedeme jeho inverzní transformaci pomocí aktuální rotační matice: $v * M^{-1} = v * M^T$
6. výsledek normalizujeme a použijeme jako osu rotace
7. úhel rotace musí být kladný, neboť orientace vypočtené osy závisí na směru relativního pohybu myši. My používáme vzorec $\frac{(|x|+|y|)}{3}$

Příloha C

Obsah CD

Sources\

Obsahuje projekt MSVC, zdrojový kód a použité knihovny

src\editor\

Jádro editoru a GUI

src\PlugInBase\

Rozhraní pro tvorbu pluginů

src\ToolLibrary\

Knihovna základních nástrojů

VL\

VL framework

Voledit\

Obsahuje samorozbalovací archiv s přeloženým projektem

Voledit64\

Přeložená 64bitová verze projektu

Data\

Ukázková data ve formátech RAW, DICOM i VEW

Docs\

Text bakalářské práce a uživatelská dokumentace v samostatném souboru

README.TXT

Popis obsahu a aktuální informace

Literatura

[1] Atomic Corporation: Paint3D, `http://www.paint3d.net`, 2010.

[2] Everygraph: Voxel3D, `http://www.everygraph.com/voxel3d/`, 2008.

[3] Eyiyurekli M., Breen D.: *Interactive free-form level-set surface-editing operators*, 2009, in preparation,

[4] Forte M.: 3D facial reconstruction and visualization of ancient Egyptian mummies using spiral CT data, ACM SIGGRAPH 99, p.223, 1999.

[5] Galyean T. A., Hughes J. F.: *Sculpting: An Interactive Volumetric Modeling Technique*, SIGGRAPH, pp.267-274, Proceedings of the 18st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991.

[6] Gnandt A.: openDICOM.NET, `http://opendicom.sourceforge.net/`, 2009.

[7] Hlaváček J.: *Zpracování medicínských dat na GPU*, Master's thesis, MFF UK, 2008.

[8] Hugunin J., Microsoft: IronPython, `http://ironpython.net`, 2010.

[9] Malandain G.: *Edge Detection*, `http://www-sop.inria.fr/epidaure/personnel/malandain/segment/edges.html`, 2001.

[10] Motofumi T. S., Yoshitomo Y., Tsuneo Y., Yasutaka S.: *A Shape Feature Extraction Method Based on 3D Convolution Masks*, ISM, pp.837-844, Eighth IEEE International Symposium on Multimedia (ISM'06), 2006

[11] Rasband W.: ImageJ, `http://rsbweb.nih.gov/ij/`, 2010.

[12] Ridge R.: The Tao Framework, http://sourceforge.net/projects/taoframework/, 2008.

[13] Shoemake K.: Euler Angle Conversion.
Graphics Gems IV. Paul Heckbert (ed.)
Academic Press, 1994, ISBN: 0123361567. pp. 222–229.

[14] Visage Imaging: Amira, http://www.amira.com/, 2010.

[15] Visualization Sciences Group: Avizo, http://www.3dvisual.com.au/html/avizo.html, 2010.

[16] Volvis: Volvis homepage, http://www.volvis.org/, 2005.

[17] Wang S. W., Kaufman A. E.: *Volume Sculpting*, Symposium on Interactive 3D Graphics, pp.151-156, pp.214, 1995.