Ročníkový projekt

Programátorská dokumentace

Jan Bím

Název projektu: HapticInterface

Vedoucí projektu: Mgr. Václav Krajíček

2010

Obsah

1 Použité knihovny

Knihovny, které byly použity v tomto projektu jsou popsány v následující kapitole. Kromě verze a stručného popisu funkcí knihovny je zde stručně uvedeno využití dané knihovny v aplikaci HapticInterface.

1.1 MedV4D framework

Tento framework poskytuje nástroje pro práci s lékařskými daty. Jedná se hlavně o sadu různých algoritmů pro zpracování lékařských dat a následné numerické a statistické analýzy, zobrazovací metody jak pro 2D tak 3D zobrazení lékařských dat implementované pro výpočet na GPU, interaktivní a přívětivé uživatelské rozhraní a nástroje pro práci se standardy, které jsou běžné v medicínské praxi jako je například DICOM nebo PACS.

Celá aplikace byla navržena a optimalizována právě pro budoucí integraci do projektu MedV4D framework. Jedna z hlavních tříd tohoto frameworku se stala kostrou aplikace a byla pouze dále upravována, přičemž zachovala rozhraní které jí MedV4D framework poskytuje. Pomocí tohoto rozhraní se například načítají vstupní data pro aplikaci a je do budoucna zajištěna komunikace s uživatelským rozhraním, které framework nabízí.

1.2 VTK

Verze: 5.0.3

VTK neboli „Visualisation Toolkit“ je knihovna určená pro práci s 3D počítačovou grafikou, zpracování obrázků a zobrazovaní. Knihovna VTK nabízí širokou škálu algoritmů pro libovolnou práci s daty, jako jsou metody pro skalární, či vektorová data, textury, nebo objemová data. VTK též nabízí mnoho modelovacích technik, jako jsou například implicitní modelování, ořezávání, vyhlazování, nebo Delaunayova triangulace.

VTK je v aplikaci použito pro zobrazení dat a práci s nimi. Jak 3D zobrazení, tak zobrazení jednotlivých řezů dat je implementováno přímo nástroji knihovny VTK. V aplikaci jsou hojně využity metody pro modelování objektů, díky kterým je reprezentován avatar kurzoru a hranice oblasti ve které se kurzor může pohybovat. Dále jsou zde použity filtry pro předzpracování dat a algoritmus „Marching cubes“ který v objemových datech nalezne isoplochy vhodné pro třídimenzionální zobrazení. A v posledním případě je VTK použito na zobrazení řezu přímo z objemových dat.

1.3 Chai3D

Verze: 2.0.0

Nejdůležitějším co zprostředkovává tato knihovna je práci s haptickými zařízeními. Je zde implementována podpora pro nejčastěji se objevující zařízení s třemi, šesti a sedmi stupni volnosti. Dále jsou v této knihovně obsaženy nástroje pro zobrazování a interaktivní simulaci.

Tato knihovna je použita pouze pro komunikaci s haptickým zařízením. Využity jsou metody pro identifikaci haptického zařízení, získání různých informací o daném zařízení, získání pozice, na které se zrovna zařízení nachází a pro nastavení haptické odezvy.

1.4 Qt

Verze: 4.3.3

Knihovna Qt je univerzální sadou nástrojů pro tvorbu libovolných aplikací a uživatelského rozhraní nezávisle na cílové platformě. Obsahuje velmi velké množství různých metod pro tvorbu uživatelského rozhraní, práci s 3D i 2D grafikou, práci s vlákny, práci s multimédii, metody pro využívání XML a různých databází a též metody pro práci se sítí.

Tato knihovna je využita pro tvorbu uživatelského rozhraní, které je realizováno pomocí formuláře a oblastí pro vykreslování přechodové funkce. Do této oblasti je kresleno právě pomocí metod knihovny Qt na práci s 2D grafikou a též je zde použito několik metod pro interakci s uživatelem pomocí myši. Dále jsou využity základní metody knihovny Qt pro spuštění aplikace, zobrazení okna a komunikace mezi jednotlivými částmi aplikace. Tato komunikace je realizována pomocí systému signálů a slotů, který tato knihovna obsahuje.

1.5 Boost

Verze: 1.38

Knihovna Boost je rozšířením knihovny STL (Standard Template Library) pro programovací jazyk C++. Obsahuje velké množství funkcí pro práci se systémovými nástroji. Například to jsou nástroje pro práci se souborovým systémem, časem, vlákny, sítí a správou paměti.

Knihovna Boost je využita pro tvorbu vlákna, ve kterém probíhá komunikace s haptickým zařízením. Dále jsou použity nástroje pro řízení více vláknové aplikace.

2 Struktura aplikace

Aplikace se skládá z těchto logických celků: Zobrazení objemových dat, Haptický kurzor a Rozhraní pro nastavení aplikace. Tyto logické celky jsou sestaveny do jednoho spustitelného souboru. Pro implementaci byl použit programovací jazyk C++. Všechny uváděné soubory zdrojového kódu jsou uvedeny bez přípon a všechny jsou složeny ze dvou souborů, jeden z příponou .cpp a druhý hlavičkový s příponou .h.

2.1 Zobrazení objemových dat

Třídy ve kterých je tato část implementována:

- aggregationFilterForVTK
- m4dGUIOGLHapticViewerWidget
- ViewerWindow

Tato část je hlavní kostrou celé aplikace. Je vlastníkem ostatních částí a zprostředkovává jejich komunikaci. Její samotná úloha spočívá v třech základních úkolech. Tyto úkoly jsou: načítání a předzpracování dat, zobrazení 3D modelu objemových dat a zobrazení objemových dat ve formě řezů. Využívá funkcí z knihovny Qt (10), MedV4D framework a knihovny VTK (11).

2.2 Haptický kurzor

Třídy ve kterých je tato část implementována:

- cursorInterface
- hapticCursor
- transitionFunction

Jak napovídá název této části aplikace, tak jejím hlavním úkolem je uchovávat informace o kurzoru a zprostředkovávat haptickou odezvu uživateli. Implementuje funkce pro zjištění aktuální pozice kurzoru a zjištění pozice a rozměrů oblasti, ve které se může kurzor pohybovat. Implementace je provedena pro různá haptická zařízení, se kterými umí komunikovat knihovna CHAI 3D. Testována však je jen pro zeřízení Novint Falcon, protože jiné nebylo v době vývoje k dispozici. Pro některé části správy paměti a pro práci s vlákny je použita knihovna Boost (12).

Tato část aplikace by měla být použitelná i v jiných projektech než je tento. Třída hapticCursor vyžaduje pro své vytvoření pouze ukazatel na

data, která jsou reprezentována třídou vtkImageData, ukazatel na vtkRenderWindow a ukazatel na přechodovou funkci, kterou zastupuju třída transitionFunction.

2.3 Rozhraní pro nastavení aplikace

Třídy ve kterých je tato část implementována:

- settingsBoxWidget
- transitionFunctionRenderArea

Poslední částí aplikace je grafické rozhraní pro nastavování všech nastavitelných parametrů aplikace. Majoritní částí tohoto rozhraní je editor přechodové funkce. Dalšími prvky lze nastavovat zvětšení a zmenšení oblasti, ve které se pohybuje kurzor, navrácení kurzoru do původní pozice a zapnutí či vypnutí záznamu pozic kurzoru. Celé rozhraní je implementováno pomocí knihovny Qt.

3 Předzpracování dat

Z důvodu výskytu šumu v datech, pro která je aplikace implementována, se předpokládá předzpracování dat. Data by měla být přefiltrována pomocí mediánového filtru. Předzpracování dat není realizováno aplikací HapticInterface. Tato aplikace předpokládá, že toto předzpracování proběhne a na vstupu pro tuto aplikaci již budou předzpracovaná data.

Mediánový filtr je velmi častá technika používaná při digitálním zpracování obrazu. Tento filtr je používán převážně k redukci šumu, neboť právě redukce šumu je častým případem předzpracování obrazu pro následné použití jiných technik, jako například pro hledání hran. Právě pro tento účel se mediánový filtr hodí nejvíce, protože zachovává hrany. Principem mediánového filtru je projít všechna data a do každého pole dat dosadit medián spočítaný v daném poli a jeho okolí. Velikost tohoto okolí je předem stanovena a závisí na ní výsledek tohoto filtru. Jeho funkci lze předvést na jednoduchém příkladě. Nechť pole x = [2, 80, 4, 3, 60, 1]. Aplikací mediánového filtru (velikost okolí nechť je rovna 3) na toto získáme pole y, které má obsahovat hodnoty po aplikaci filtru, následovně:

```
y[1] = medián(2, 2, 80) = 2
y[2] = medián(2, 80, 4) = medián(2, 4, 80) = 4
y[3] = medián(80, 4, 3) = medián(3, 4, 80) = 4
y[4] = medián(4, 3, 60) = medián(3, 4, 60) = 4
y[5] = medián(3, 60, 1) = medián(1, 3, 60) = 3
y[6] = medián(60, 1, 1) = medián(1, 1, 60) = 1
y = [ 2, 4, 4, 4, 3, 1 ]
```

Při použití tohoto filtru ve vícerozměrných polích se používá okolí tvaru kruhu, čtverce, nebo například kříže. Provedení různých testů ukázalo, že pro nejlepší výsledek při generování haptické odezvy je vhodné zvolit poloměr tohoto filtru na hodnotu 3. Je třeba dbát na vhodné zvolení poloměru filtru pro úkol, který je s daty vykonáván, protože mediánový filtr může odstranit i struktury, které byly žádoucí, například tenké oblasti dat, které se nacházejí mezi dvěma oblastmi se stejnou hodnotou.

Data je též vhodné ořezat, aby zde zbyla pouze ta část, která má být předmětem zkoumání. Toto je hlavně z důvodu rychlosti a též kvůli odstranění nesmyslných dat. Například na snímcích CT z oblasti abdomenu se velmi často vyskytuje lehátko na kterém pacient ležel v průběhu snímání. Velikost dat ovlivňuje rychlost startu aplikace, protože se zde data zpracovávají několika průchody přes celá data. Pro příklad, nechť jsou dána data o rozměrech 512x512x100 voxelů. Po ořezání částí, ve kterých jsou data, která nensou žádnou hodnotu o zkoumaném objektu, se zmenší rozměr dat na 380x300x100 voxelů. Velikost tohoto ořezání je běžná i v praktickém použití a ubýtek datových polí je větší než polovina, protože původní data obsahovala zhruba 26 miliónů voxelů, ale ořezaná data obsahují jen zhruba 11 miliónů voxelů. Rychlost startu aplikace může v takovémto případě být až dvakrát větší. Nástroje pro provedení ořezání a aplikování mediánového filtru obsahuje MedV4D framework v projektu „tools“. Popis použití těchto nástrojů je popsán v uveden B.

4 Implementační detaily

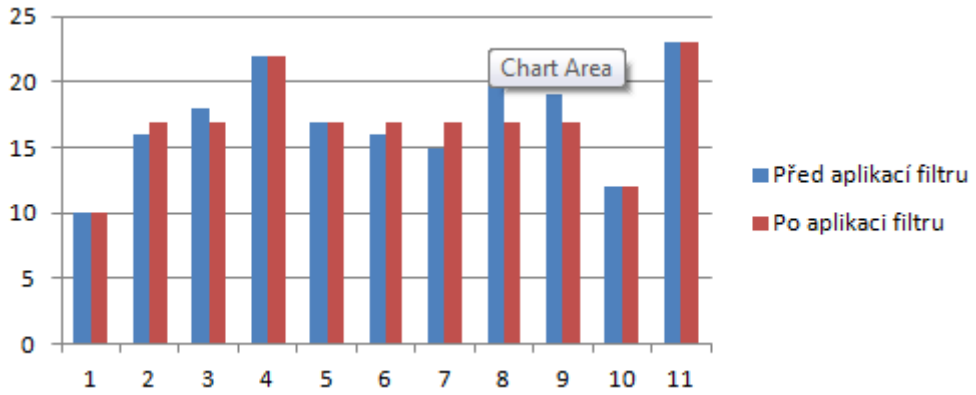
4.1 Data

Načítání dat je realizováno z již předem předpřipraveného souboru, který může obsahovat jakákoliv objemová data a nutné informace o nich. Do tohoto souboru jsou data převedena aplikací DICOM2ImageDump, které je součástí frameworku MedV4D.

Načítaný soubor se načítá do datových struktur tohoto frameworku a poté se přidělí aplikaci. Při spouštění aplikace v hlavní metodě main je pomocí třídy ImageFactory načten do třídy AImage::Ptr a z této je pak vytvořena třída ConnectionTyped, která se předává hlavní třídě aplikace m4dGUIOGLHapticViewerWidget v jejím konstruktoru. Po načtení do paměti v podobě datových struktur MedV4D frameworku jsou tyto pomocí nástrojů, které framework obsahuje převedeny do datových struktur se

kterými pracuje knihovna VTK. Jedná se o nástroje části frameworku MedV4D, která se nazývá vtkIntegration. Pro tento učel je vytvořena třída m4dImageDataSource, která již umí vracet vtkAlgorithmOutput a to je třída, se kterou umí pracovat všechny třídy, které se používají v konstrukci VTK pipeline. Vizualizace této pipeline je v následující kapitole na obrázku 3-2.

V jedné větvi jsou ještě data předzpracována pomocí agregačního filtru. Agregační filtr má přednastavený interval a všem hodnotám z tohoto intervalu přiřazuje jednu předem danou hodnotu. Příklad použití agregačního filtru je na obrázku 3-1.



Obrázek 3-1: Příklad použití agregačního filtru. Hodnoty z intervalu <15, 20> byly všechny převedeny na hodnotu 17.

Filtrování je z důvodu následného použití algoritmu „Marching cubes“. Tento algoritmus se používá k hledání isoploch v objemových datech a pro dostatečně kvalitní zobrazení 3D modelu je třeba zmenšit přesnost dat. Tímto se odstraní výkyvu hustot v jednotlivých tkáních a je zaručeno, že následný 3D model, který se z těchto dat tvoří, je celistvý.

4.2 Zobrazení

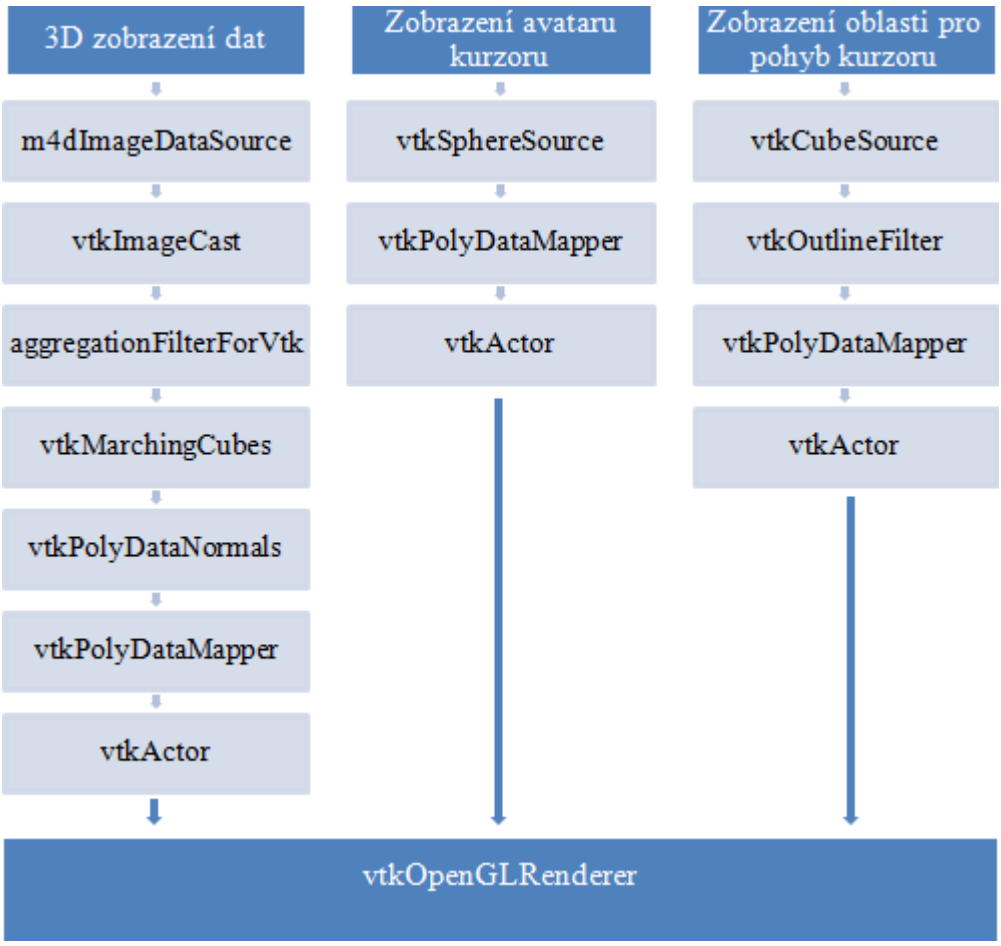
Vykreslování třídimenzionálního modelu dat je zprostředkováváno knihovnou VTK. Vykreslování je standardně nastaveno tak, aby bylo prováděno na grafické kartě. Nástroje VTK k tomu používají knihovnu OpenGL. V případě, že v počítači nebude přítomna grafická karta podporující správnou verzi knihovny OpenGL, tak bude použit k vykreslení procesor.

Vykreslují se předem vybrané isoplochy. Právě kvůli isoplochám, vykreslovaní 3D modelu používá filtrovaná data. V aplikaci je předem nastaveno a lze snadno změnit, kolik isoploch a pro jaké hodnoty hustoty se má v datech hledat. Nastavení je možné směnit pouze úpravou přímo v kódu. Ve stávající implementaci jsou hledány isoplochy pro hodnotu 600, která je však předtím pomocí agregačního filtru dosazena na místo všech hodnot v intervalu <550, 700>. Toto nastavení je třeba volit velmi uvážlivě, protože průběh algoritmu „Marching cubes“ je velmi náročný a jeho zpracování například pro datové pole o rozměru 512x512x300 voxelů trvá v řádech desítek sekund pro jednu hodnotu isoploch.

Jednotlivé isoplochy jsou vykresleny každá jinou barvou a jinou průhledností. Tyto vlastnosti lze v kódu snadno ovlivnit. Při ladění aplikace se ukázalo jako nejlepší řešení to, co je aktuálně v aplikaci implementováno, což znamená vzkreslování právě pouze jedné isoplochy. Při použití příliš mnoha isoploch se ztrácí přehlednost 3D zobrazení. Též přílišné používání průhlednosti činí zobrazení velmi nepřehledným. Průhlednost je však třeba používat z důvodu dobré viditelnosti avataru kurzoru.

Vykreslení je prováděno pomocí VTK pipeline. Tato pipeline začíná třídou vtkImageCast, které předala data výše zmiňovaná třída m4dImageDataSource. Další částí pipeline je agregační filtr zastoupen třídou aggregationFilterForVtk. Tato pak předává data třídě vtkMarchingCubes, která zprostředkovává průběh stejnojmenného algoritmu. Následujícími třídami jsou vtkPolyDataNormals, která přidá k získaným polygonům normály a třída vtkPolyDataMapper, která mapuje získaná data na grafická primitiva a nakonec je zde zařazena třída vtkActor, která zastupuje danou entitu ve scéně, která se poté vykresluje. Samotné vykreslení zprostředkovává třída vtkOpenGLRenderer. Vizualizace této pipeline je na obrázku 3-2. Při použití s více isoplochami se tato pipeline za agregačním filtrem větví a je třeba vytvořit novou třídu vtkMarchingCubes a všechny třídy, které v pipeline následují znovu.

Krom isoploch jsou v třídimenzionálním zobrazení ještě následující objekty: avatar kurzoru a oblast, v níž se může kurzor pohybovat. Avatar kurzoru je zobrazen jako zelená neprůhledná koule. Tyto parametry spolu s velikostí avataru jsou výsledkem ladění a testování. Tuto kouli reprezentuje třída vtkSphereSource, která nese údaje o pozici a velikosti. Oblast, v níž se kurzor smí pohybovat, je znázorněna drátovým modelem modré krychle. Model krychle je vytvořen pomocí třídy vtkCubeSource, která opět uchovává informace o pozici a své velikosti, ze které je vytvořen drátový model třídou vtkOutlineFilter. Barvy jsou upraveny až ve třídě vtkActor.



Obrázek 3-2: Vizualizace VTK pipeline pro vykreslování třídimenzionálního modelu dat, kurzoru a oblasti pro pohyb kurzoru

Implementace zobrazovací oblasti 3D modelu obsahuje též metody pro interakci s uživatelem. Toto je realizováno pomocí třídy vtkRenderWindowInteractor, která zpracovává události vytvořené pohyby myší, mačkáním tlačítek myši a používáním kolečka na myši. Reakce jsou přibližování a oddalování modelu z isoploch, jeho otáčení a posun.

Vykreslení objemových dat v podobě řezu, ve kterém se nachází kurzor, je též implementováno pomocí knihovny VTK a to konkrétně pomocí třídy vtkImageMapper, které se pak tvoří vtkActor2D. Kvůli větší přesnosti toto zobrazení používá nefiltrovaná data. Též toto zobrazení neobsahuje žádnou jinou interakci s uživatelem, než zobrazení aktuálního řezu podle pozice kurzoru, případně řezu žádného, pokud kurzor není v oblasti s daty.

Zobrazení řezu je realizováno jako plocha, jejíž každý bod se obarví patřičnou barvou podle hodnoty bodu v datech. Hodnoty dat se automaticky přiřadí na černobílou stupnici, černé body mají nejnižší hodnotu, kterou data obsahují a bíle naopak nejvyšší.

V zobrazení jednoho řezu se též vykresluje avatar kurzoru a oblast ve které se daný kurzor smí pohybovat. Avatar je zastoupen zeleným křížem a oblast pro pohyb je vyznačena modrým čtvercem. Tyto dvě věci jsou obě vykresleny pomocí čar. Čáry jsou zařazeny do pipeline tak jak popisuje následující text. Zdrojem dat je zde vtkLineSource, který uchovává pozici a délku čáry, a ten je poté mapován pomocí vtkPolyDataMapper2D a zastupován na scéně pomocí vtkActor2D. Všechny třídy vtkActor2D jsou poté předány třídě vtkRenderer, která ma na starosti samotné vykreslování.

4.3 Přechodová funkce

K výpočtu haptické odezvy je využívána přechodová funkce, která určuje hodnoty odporu prostředí pro jednotlivé hodnoty v objemových datech. Všechny dále popisované vlastnosti implementuje třída transitionFunction.

Funkce je uložena pomocí bodů, kterými prochází. Pro získání hodnoty funkce v daném bodě se získává dvěma způsoby. Pokud daný bod je jeden z těch, které reprezentují funkci, tak se vrací přímo jeho hodnota, pokud je to bod, který ve funkci není, poté se jeho hodnota určí jako hodnota v daném bodě funkce přímky, spojující dva nejbližší body, pomocí kterých je přechodová funkce definována. Přechodová funkce je vždy určena alespoň dvěma body a to jsou okrajové hodnoty intervalu všech hodnot, které se nacházejí v objemových datech.

Další implementovanou funkcí přechodové funkce je uchovávání dvou hranic. Tyto jsou reprezentovány nějakými hodnotami z objemových dat. Hranice určují, do které a od které hodnoty se má prostředí s danou hodnotou hustoty chovat jako neprostupné. Tyto nemusí být vždy nastaveny.

V Přechodové funkci jsou též implementovány metody pro uložení, nebo načtení ze souboru. Soubor, do kterého se funkce ukládá, je textový a obsahuje data oddělená koncem řádky. První co se v něm vyskytuje je počet bodů funkce, poté následují jednotlivé body, vždy obě souřadnice jednoho bodu na jedné řádce oddělené mezerou a na konci jsou x-ové souřadnice hranic pro neprostupné prostředí.

4.4 Práce s haptickým zařízením

Implementaci, o které hovoří tato kapitola zapouzřuje třída hapticCursor. Pro umožnění práce s haptickým zařízením je nejprve provedena jeho detekce. Aplikace se při svém spuštění pokouší nalézt libovolné haptické zařízení, se kterým umí spolupracovat knihovna Chai3D. Toto se děje prostřednictvím třídy cHapticDeviceHandler. Pokud takové zařízení neexistuje, potom se kurzor nastaví do středu dat podle všech souřadných os a již s ním není dále možno hýbat. Pokud takové zařízení existuje, načtou se do datových struktur, které uchovává třída cGenericHapticDevice, všechny informace o něm a provede se kalibrace zařízení, je-li to třeba. Kalibraci též zprostředkovává tato třída. Získané informace, které tato třída uchovává obsahují například meze pro pohyb haptického zařízení, jméno a výrobce zařízení, maximální sílu haptické odezvy, kterou je zařízení schopno působit a podobně.

Kvůli citlivosti haptických zařízení je třeba dosáhnout dostatečně velké obnovovací frekvence směru a síly haptické odezvy aby zařízení nevykonávalo nějaké neplynulé reakce. Obnovovací frekvence pozice zařízení je 1kHz. Pro dostatečně kvalitní haptickou odezvu je třeba, aby aplikace dokázala komunikovat se zařízení a obnovovat hodnoty jeho síly alespoň s frekvencí 500Hz. Z tohoto důvodu se veškerá komunikace s haptickým zařízením odehrává v separátním vlákně. Toto vlákno provádí pouze několik matematických výpočtů, které nezaberou téměř žádný procesorový čas, proto je vlákno v každém cyklu uspáno na dobu 1ms. Při této implementaci dosahuje průměrná frekvence komunikace se zařízením hodnoty 700Hz. Tato hodnota je pro kvalitní odezvu dostatečná.

Komunikace s haptickým zařízením je prováděna v cyklu, který vykonává následující operace: zjištění aktuální pozice zařízení, přepočet souřadného systému haptického zařízení do souřadného systému objemových dat, výpočet haptické odezvy a nastavení směru a síly, kterou má zařízení působit. Haptická odezva se zařízení předává jakožto třísložkový vektor.

Přepočet souřadného systému haptického zařízení na systém, který se nachází v datech, je takový, že krajní pozice haptického zařízení odpovídají krajním pozicím oblasti pro pohyb kurzorem. Při zmenšení oblasti se tím dosahuje mnohem větší přesnosti, protože je fyzický pohyb zařízení mapován na menší pohyb virtuální. Přepočtu odpovídá následující pseudokód:

Do proměnné X se uloží vektor aktuálních souřadnic haptického

```
X.x = SouřadnicePoziceHaptickéhoZařízení.y
X.y = SouřadnicePoziceHaptickéhoZařízení.z
X.z = SouřadnicePoziceHaptickéhoZařízení.x
s = DelkaHranyKrychleOblastiProPohyb / 2,0 /
MaximálníVýchylkaHaptZařízení
SouřadniceKurzoruVDatech = StředOblastiProPohyb + X / s
```

zařízení, ale s osami přemapovanými na osy v datech. Poté se do proměnné s uloží měřítko. Toto zaručuje, že i při změně velikosti oblasti pro pohyb kurzoru je namapován pohyb kurzoru zařízení na pohyb v celé této ovlasti. Souřadnice v datech se pak již jednoduše určí přičtením upravených souřadnic haptického kurzoru pomocí měřítka k středu oblasti pro pohyb zařízení.

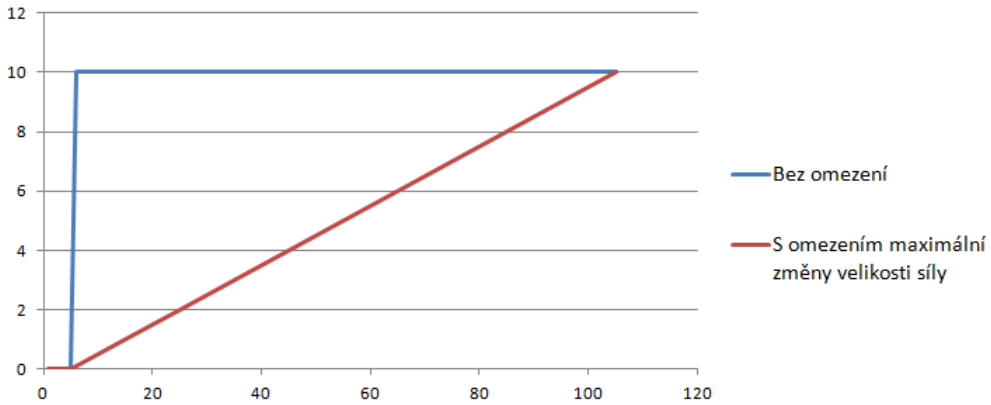
Samotný výpočet haptické odezvy je rozdělen na dva stavy. Stav kdy je kurzor v běžném prostředí a stav kdy se nachází kurzor v prostředí definovaném přechodovou funkcí jako neprostupné. Při spuštění aplikace je vždy stav zařízení nastaven na běžný režim. Jakmile se kurzor dostane do neprostupné oblasti, jsou uloženy patřičné informace a stav je přepnut pro výpočet v neprostupných oblastech. Pokud zařízení splní podmínky pro vystoupení z této oblasti, a nachází se skutečně v datech tam, kde je běžná hodnota přechodové funkce, vrací se stav do normálního režimu.

Výpočet haptické odezvy v běžném režimu má simulovat tuhost prostředí neboli velikost odporu prostředí vůči procházení nějakým tělesem. Nejdříve se určí směr síly, který odpovídá směru opačného vektoru k vektoru pohybu kurzoru bezprostředně před výpočtem haptické odezvy. Toto simuluje, že síla jde vždy přesně proti směru pohybu zařízení. Poté se směrový vektor normalizuje. Pomocí přechodové funkce se určí velikost síly jakou má haptická odezva působit a směrový vektor se jí vynásobí. Tento výsledný vektor se pošle zařízení jako nastavení haptické odezvy.

Z důvodů přílišné přesnosti zařízení, které způsobuje, že kromě krajních pozic zařízení nikdy nevrátí svou pozici shodnou s předchozí pozicí, je potřeba vektor pohybu zařízení před výpočtem též vypočítat, z pohybu zařízení na větší vzdálenosti, než je rozdíl posledních dvou pozic. Pokud by se tak nedělo, tak i v případě, že se uživatel snaží udržet zařízení v nějakém bodě v klidové poloze, se toto nemůže podařit a při příliš pomalém pohybu by docházelo k odchylkám směrového vektoru vůči správnému směru. V implementaci aplikace se průměruje posledních 30 směrových vektorů, které se získají jako rozdíl posledních dvou pozic v každém průběhu cyklu pro komunikaci s haptickým zařízením. Toto číslo je nastaveno jako konstanta přímo ve zdrojovém kódu třídy hapticCursor s názvem NUMBER_OF_VECTORS. Velikost tohoto čísla byla testována a hodnota 30 vykazovala nejlepší vlastnosti. Průměr vektorů je tedy při průměrné obnovovací frakvenci 700Hz počítán z pohybu, který proběhl v posledních 0,043 sekundy. Tento časový interval je dostatečně krátký aby zohledňoval pouze pohyb bezprostředně před výpočtem a zároveň je dostatečně dlouhý pro zjištění směru tohoto pohybu. Díky tomuto zprůměrování haptické

zařízení dává haptickou odezvu skutečně do protisměru pohybu tohoto zařízení.

Implementace obsahuje ještě jeden prvek pro korekci výpočtu haptické odezvy. Pokud se totiž velikost síly odezvy změní skokově, zařízení provede příliš nárazovou reakci a vychýlí uživatele z požadované pozice. Na takto rychlé „kopnutí“ není schopen lidský organismus včas reagovat, proto je zde implementována kontrola velikosti změny síly, kterou má haptická odezva působit. Pokud velikost změny překročí danou konstantu, je tato síla změněna právě jen o zmiňovanou konstantu. Tímto způsobem nedochází k náhlým „kopnutím“, ale k plynulé změně síly. Znázornění rozdílu nárůstu síly při použití tohoto algoritmu a bez něj znázorňuje obrázek 3-3. Konstantu pro omezení této změny je třeba volit velmi obezřetně. V případě příliš velké této konstanty bude i nadále docházet ke „kopání“ a v případě příliš malé konstanty bude mít reakce na prostředí příliš velké zpoždění, nebo se dokonce znemožní dosažení maximální síly vůbec. Ve stávající implementaci programu je tato konstanta nastavena na 0,1, což je číslo vyjadřující změnu síly v Newtonech. Lze ji změnit ve zdrojovém kódu třídy hapticCursor a její název je EPSILON.



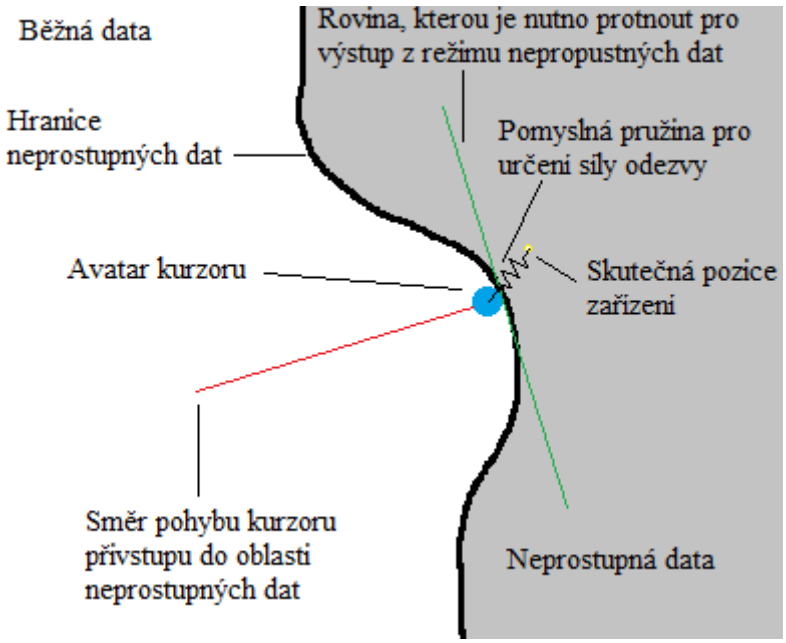
Obrázek 3-3: Znázornění rozdílu v nárůstu síly haptické odezvy při omezení maximálního nárůstu této síly a bez něj. Na ose x je uveden čas v počtu průběhů cyklu pro komunikaci s haptickým zařízením a na ose y je velikost síly v Newtonech.

Pokud je výpočet haptické odezvy nastaven na stav pro neprostupné oblasti, vypočítává se haptická odezva algoritmem, který simuluje natažení virtuální pružiny mezi kurzorem fyzického zařízení a bodem, kde došlo k průniku do oblasti neproniknutelných dat. V tomto bodě též zůstává avatar kurzoru, aby navodil iluzi, že kurzor skutečně nemůže pokračovat

skrz tuto oblast. Algoritmu s natažením pružiny se využívá hlavně kvůli plynulému nárůstu síly, aby ani zde nedocházelo ke „kopání“ ze strany haptického zařízení. Tento algoritmus se běžně používá pro výpočet haptické odezvy nad geometrickými daty. Zařízení, pro které byla tato aplikace implementována má maximální sílu haptické odezvy 10N. Tato velikost síly průnik běžnému člověku neznemožní. Proto síla musí narůstat dostatečně rychle, aby tato změna budila dojem neprostupnosti. Situaci při používání této metody ilustruje obrázek 3-4.

Algoritmus se spustí v okamžiku, kdy se kurzor dostane do místa, které přechodová funkce označí jako neprostupné. Kurzor přestane vykazovat změny pozice a tím pádem dojde k zastavení pohybu avataru kurzoru na obrazovce. Z místa kde zůstal avatar se k fyzickému zařízení natahuje pomyslná pružina o předem dané tuhosti. Tuhost pružiny musí být volena se stejným zřetelem jako omezení pro maximální změnu síly v předchozí metodě, jinak bude haptická odezva vykazovat totožné chyby. Haptická odezva má pak směr ve kterém je fyzický kurzor přitahován touto pružinou k místu kde zůstal avatar a velikost určenou podle velikosti síly, kterou je fyzický kurzor přitahován k avataru.

Pro změnu metody výpočtu na původní je potřeba aby se fyzické zařízení vrátilo do oblasti dat, kde je běžný odpor prostředí a aby při tomto návratu protnulo tzv. vstupní rovinu kurzoru do neprostupné oblasti. Tato rovina má následující parametry: prochází bodem, kde zůstal avatar kurzoru a její normálový vektor je roven vektoru pohybu kurzoru bezprostředně předtím, než se dostal do oblasti neprostupného prostředí. Opět je pro určení normálového vektoru roviny použito vektoru, který vznikl průměrem několika posledních směrových vektorů. Průchod rovinou je nutný pro správné detekování opuštění neprostupné oblasti, protože zařízení se svou odezvou sice snaží vracet kurzor přesně do místa, kudy se do neprostupné oblasti dostal, ale při pohybu zpět nemusí kurzor protnout přesně stejné souřadnice.

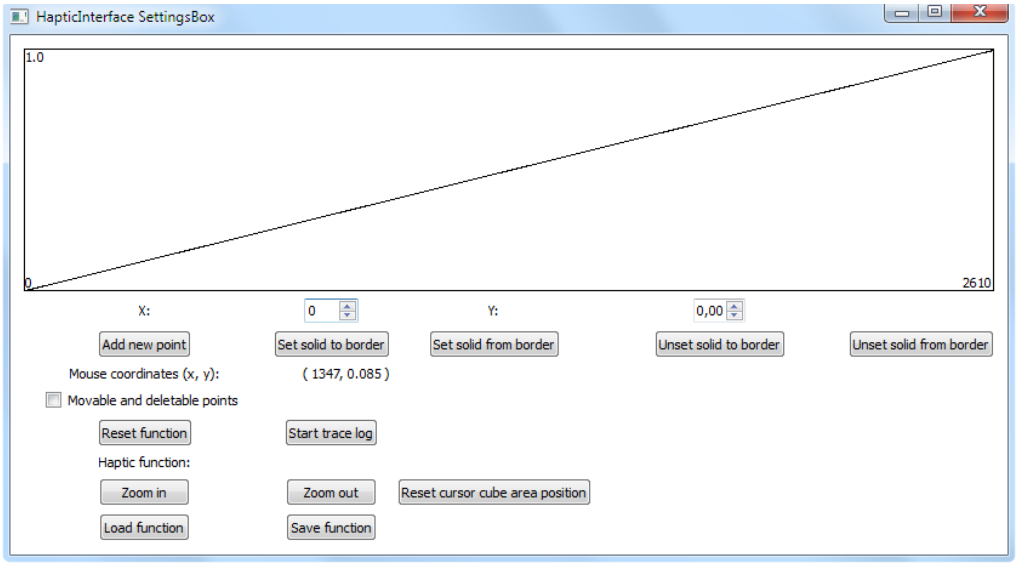


Obrázek 3-4: Znázornění stavu, kdy je výpočet haptické odezvy přepnut do režimu naprostupných dat, ve chvíli, kdy kurzor již vstoupil do oblasti s neprosupnými daty.

Zařízení rovněž implementuje funkce, pomocí kterých je možno spustit, či ukončit ukládání souřadnic haptického kurzoru. Souřadnice jsou ukládány do textového souboru oddělené koncem řádky. Ukládání není závislé na čase, nebo rychlosti pohybu kurzoru, ale uloží se vždy ty souřadnice, které jsou rozdílné od souřadnic, na nichž se kurzor nacházel v minulém cyklu komunikace s haptickým zařízením. Jedná se o souřadnice v souřadnicovém systému objemových dat, tudíž zde není problém s přílišnou přesností haptického zařízení.

4.5 Rozhraní pro nastavení aplikace

Rozhraní pro nastavení aplikace je složeno z dvou hlavních částí. Část, kde se vykresluje přechodová funkce a je ji možno editovat za pomoci myši a přilehlých nástrojů a část, která obsahuje právě zmíněná nástroje a další ovládací prvky, kterými je možno ovlivňovat chování aplikace. Rozhraní je zobrazeno na obrázku 3-x.



Obrázek 3-x: Okno s rozhraním pro nastavení aplikace

Část pro vykreslení má implementovány dva stavy. V prvním, implicitním stavu reaguje na kliknutí do oblasti pro kreslení pouze přidáním bodu do přechodové funkce. Při přepnutí do druhého stavu se kolem každého bodu funkce vykreslí zelený kruh, který symbolizuje oblast, ve které se označí daný bod a je možno s ním provádět úpravy.

V módu pro úpravy přechodové funkce se při každém kliknutí levého tlačítka myši spočítá, zda se kurzor v momentě kliknutí nacházel v blízkém okolí nějakého bodu. Pokud ano, tak se tento považuje za označený a při pohybu myši se tento bod přesunuje. Prostředí kontroluje, zda není přesunut mimo interval sousedních dvou bodů. V takovém případě je implementována grafická odezva, funkce v okolí tohoto bodu zčervená. Toto chování je identické pro pokus o jakoukoliv neplatnou operaci. Kontrolují se souřadnice kurzoru vůči této ploše, zda se neocitl mimo ni. Aplikace též implementuje reakci na událost stisknutí pravého tlačítka myši. Pokud bylo stisknuto tlačítko v blízkosti některého z bodů, pokusí se jej aplikace odstranit.

Zobrazovací oblast má též implementováno vykreslení hranic pro neprostupné prostředí. Pokud přechodová funkce tyto hranice obsahuje tak jsou znázorněny svislými barevnými čarami. Celá tato část je implementována pomocí nástrojů pro kreslení ve 2D prostředí knihovny Qt. Veškeré vykreslování se odehrává při vyvolání události žádosti o aktualizaci. Tato událost se vyvolává při libovolné změně přechodové

funkce, nebo při jakémkoliv pohybu myši, když je označen nějaký bod, aby se zaručilo uživateli vždy aktuální zobrazení přechodové funkce.

Zbytek rozhraní pro nastavení aplikace je implementován pomocí běžných formulářových prvků. Jsou to hlavně tlačítka zprostředkovávající zvětšení či zmenšení oblasti pro pohyb kurzoru, přidání hranic hodnot funkce pro neprostupné prostředí, vyvolání dialogů pro uložení a načtení funkce ze souboru a dalších funkcí. Všechny akce, které jsou vyvolány v tomto rozhraní, jsou předány pomocí mechanismu signálů a slotů knihovny Qt hlavní části aplikace.

5. Překlad

Aplikace funguje pouze s operačním systémem Microsoft Windows XP, Vista a Seven. Byla vyvíjena a překládána pomocí programu Microsoft Visual Studio 2008 a překlad pomocí jiných překladačů nemusí být funkční. Pro překlad aplikace je nutné provést následující kroky:

1. Stáhnout aktuální verzi frameworku MedV4D z repozitáře, který se nachází na adrese

   `svn://cgg.mff.cuni.cz/MedV4D/trunk`.

2. Nainstalovat program Cmake. Tento program lze získat na adrese

   `http://www.cmake.org`.

3. Stáhnout ze stránky

   `http://cgg.mff.cuni.cz/trac/medv4d/wiki/PrecompiledLibPrecom`

   ze sekce pro Microsoft Visual Studio 2008 všechny soubory. A ze stránky

   `http://cgg.mff.cuni.cz/trac/medv4d/attachment/wiki/HapticInterfaHa/`

   stáhnout soubor chai3d-2.0.0-precompiled.zip.
4. Rozbalit a zkopírovat obsah souborů dcmtk_msvc2008.zip, vtk-with-qt_msvc2008.zip, qt-reduced_msvc2008.zip a OtherLibs.zip do adresáře trunk, který byl stažen z repozitáře. A soubory

boost_root.zip a chai3d-2.0.0-precompiled.zip rozbalit každý do svého adresáře.

5. Nastavit systémovou proměnnou CHAI3D_ROOT na cestu k adresáři, do kterého byl vykopírován obsah souboru chai3d-2.0.0-precompiled.zip a proměnnou BOOST_ROOT na cestu k adresáři, do kterého byl vykopírován obsah souboru boost_root.zip.
6. Spustit program cmake-gui.exe a do pole, které se jmenuje „Where is the source code:“ vložit adresu

   ```
   trunk/src/cmake_project/Applications/HapticInterface
   ```

   kde trunk je adresář stažený z repozitáře.
   Do pole které se jmenuje „Where to build the binaries:“ napsat adresu adresáře do kterého chcete vygenerovat soubory projektu.
7. Stisknout tlačítko „Configure“. Při výběru překladače zvolte Microsoft Visual Studio 2008. Pokud po zpracování tohoto požadavku bude v proměnných, které se zobrazí vprostřed okna zaškrtnuto políčko CG_ENABLED, tak toto políčko vyškrtnout. Znovu stisknout „Configure“ a poté stisknout tlačítko „Generate“
8. V adresáři, který byl zadán pro vygenerování souborů projektu se vytvoří soubor HapticInterface.sln, který je možno otevřít v Microsoft Visual Studiu 2008.