VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ MANIPULACE S 3D OBJEKTY SE SILOVOU ZPĚTNOU VAZBOU

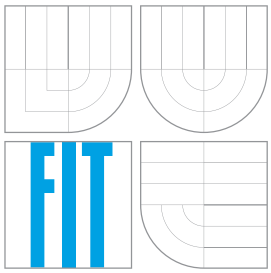
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN BĚLÍN

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ MANIPULACE S 3D OBJEKTY SE SILOVOU ZPĚTNOU VAZBOU

INTERACTIVE MANIPULATION WITH 3D OBJECTS WITH FORCE FEEDBACK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN BĚLÍN
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. PŘEMYSL KRŠEK, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

K moderní názorné manipulaci s objekty ve virtuálním prostoru se přidává i možnost fyzické haptické interakce. Haptická technologie je v náplni této diplomové práci zastoupena zařízením SensAble Phantom Omni a souvisejícím toolkitem OpenHaptics. Čtenář je v úvodu seznámen s matematickými základy manipulace a s historií i současností haptických technologií. Následuje představení toolkitu OpenHaptics a především popis haptických knihoven HDAPI a HLAPI, které jsou jeho součástí. Tento základ byl použit při tvorbě demonstračních aplikací, jenž názorně ukazují základní i rozšířené možnosti haptického zařízení Phantom Omni. Aplikace se snaží ukázat funkčnost na příkladech integrující známé elementární fyzikální zákony a jevy.

Abstract

Physical haptic interaction is added to the modern manipulation with objects in virtual space. In content of this master's thesis the haptic technology is represented by SensAble Phantom Omni device and OpenHaptics toolkit, which is related to the device. Reader is initially introduced into mathematical basics of manipulation and into haptic technology history including current state. The introduction into Openhaptics toolkit follows as well as HDAPI and HLAPI libraries description. As a result of this theoretical basics demo aplications have been created, that show basic and advanced abilities of the Phantom Omni device. Demos represent the functionality of the device as examples integrating well-known elementary physical laws and events.

Klíčová slova

Transformace v trojrozměrném prostoru, matice, manipulace, haptické zařízení, Sensable Phantom Omni, OpenHaptics toolkit, knihovny HDAPI a HLAPI.

Keywords

3D transformations, matrixes, manipulation, haptic device, Sensable Phantom Omni, OpenHaptics toolkit, libraries HDAPI and HLAPI.

Citace

Jan Bělín: Interaktivní manipulace s 3D objekty se silovou zpětnou vazbou, diplomová práce, Brno, FIT VUT v Brně, 2009

Interaktivní manipulace s 3D objekty se silovou zpětnou vazbou

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Přemysla Krška, Ph.D..

.....
Jan Bělín
26. 5. 2009

Poděkování

Chci poděkovat vedoucímu práce panu Doc. Ing. Přemyslovi Krškovi, Ph.D. za poskytnutou možnost pracovat s haptickým zařízením, za trpělivost a především za cenné rady, které mi pomohli při tvorbě této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Současné aplikace vyžadují uživatelsky přívětivé ovládání, nejlépe intuitivní s minimálním zastoupením textových popisů jednotlivých úkonů. Podle hesla „Jeden obrázek je za tisíc slov.“ se vyžaduje kvalitní grafické uživatelské rozhraní, jenž dovolí pohodlně ovládat funkce programu kurzorem počítačové myši. Kurzor myši se za léta používání stal naprosto neodmyslitelným nástrojem při interakci s aplikacemi, že si lze jen velmi těžko představit společnost, která by se odhodlala vytvářet aplikace určené pro širokou veřejnost a přesto by vyvíjela systémy ovládané příkazovým řádkem s čistě textovým uživatelským rozhraním.

Počítačová myš je více než dostačujícím ovládacím prvkem u klasických programů, které zobrazují své uživatelské rozhraní ve dvou rozměrech, příkladem je jakákoli současná kancelářská aplikace, ale pokud se přesuneme do skupiny programů pracujících s objekty v trojrozměrném prostoru, stává se ovládání myší komplikovanější. Pohyb a manipulace v prostoru se potom musí řešit skrze pomocné klávesové zkratky, které přepínají stavy manipulace, nebo se zobrazuje více pohledů na prostor tak, aby myš mohla využít svých dvourozměrových předností.

Člověk si ovšem složité činnosti, především ovládání všeho druhu, mile rád zjednodušuje a hledá technologie, jenž mu dovolí lepší a jednoduší funkčnost. Proto se pro aplikace pracující s třetím rozměrem vyvíjejí speciální ovladače. Mezi ty kompaktnější ovladače patří rozhodně všelijaké trackballs nebo 3D ovladače společnosti 3Dconnexion [13], které používají pro kontrolu 3D prostoru zařízení připomínající směrové, otočné tlačítko. A jestliže existují kompaktní řešení, je pravidlem, že si lze pořídit i rozměrnější zařízení. Do skupiny složitějších zařízení patří i skener Immersion Microscribe popsaný v mé bakalářské práci [1], která byla základním stavebním kamenem pro tuto práci. Skener Microscribe lze jednoduše popsat jako 3D myš umožňující získávání polohy snímacího hrotu ve všech třech dimenzích.

Jenomže i přes existenci 3D myši, která v podstatě řeší problém manipulace v trojrozměrném prostoru, člověk neukončí své snažení. Uživatel chce více, chce nějakým způsobem interagovat s objekty a chce je vnímat co nejvíce smysly. A v tomto okamžiku přichází ke slovu haptická technologie. Technologie, jenž dopřeje uživateli reálně cítit virtuální objekty a opět tak rozšíří možnosti ovládání aplikací. Jednou z předních společností zabývající se haptickými technologiemi je SensAble Technologies [21], která vyrábí i haptická zařízení řady Phantom.

Práce a vytváření aplikací s využitím haptického zařízení Phantom je obsahem této diplomové práce. Se zařízením Phantom Omni jsem byl schopen pracovat, neboť se nachází na Ústavu počítačové grafiky a multimédií VUT v Brně a měl jsem k tomuto zařízení přístup.

První kapitola seznámí čtenáře s nutným matematickým aparátem, který se používá

v dnešních aplikacích pro manipulaci s objekty. Jsou popsány matice, což jsou struktury pro uchovávání polohy i jako prostředek reprezentace trojrozměrných transformací, dále samotné transformace, druhy transformací a jejich účinky na objekty v prostoru. Následně je zmíněna struktura graf scény a její vnitřní vlastnosti využívající právě matic a transformací pro manipulaci a zobrazování 3D objektů. Text je převzat z mé předešlé práce [1]. V závěru kapitoly je představena haptická technologie, její stručná historie a současnost prostřednictvím společnosti SensAble Technologies a zařízení Phantom.

Druhá kapitola obsahuje představení toolkitu OpenHaptics, také vyvíjeného společností SensAble Technologies, a jeho použití při vytváření aplikací využívajících zařízení Phantom Omni. Stěžejními částmi toolkitu jsou knihovny HDAPI a HLAPI, kterým je věnována většina kapitoly. Jsou popsány základní i pokročilé vlastnosti knihoven pro implementaci haptické aplikace. Druhou část kapitoly tvoří popis návrhu demonstračních aplikací, které jsou výstupem této práce a prakticky ukazují možnosti zařízení i knihoven.

Následující kapitola se zabývá implementací demonstračních aplikací, použitými prostředky a uplatněnými technologiemi a postupy.

Předposlední kapitola shrnuje dosažené výsledky, funkčnost aplikací, jejich vývoj a vlastnosti toolkitu OpenHaptics.

Poslední částí této práce je závěrečné zhodnocení práce, výsledků, použitých nástrojů a využití haptického zařízení Phantom Omni a toolkitu Openhaptics. V této kapitole naznačuji další směry zájmu v oblasti haptického programování.

Kapitola 2

Teoretický rozbor

2.1 Matice

Teorie matic [18] tvoří úvod a základ lineární algebry[1]. V matematice se matice společně s determinanty [6] aplikují při řešení soustav lineárních rovnic. V počítačové grafice se matice hojně využívají při vyjádření transformací (dále část 2.2), protože toto uspořádaní podporují i moderní grafické procesory.

Matice je možno definovat [6] následovně:
Matice A $= (a_{ij})$ typu m/n nad množinou $X \neq \emptyset$ je schéma složené z $m \cdot n$ prvků množiny X zapsaných do m řádků a n sloupců. Přesněji matice **A** typu m/n nad X je zobrazení množiny $\{1, \ldots, m\} \times \{1, \ldots, n\}$ do množiny X.

Množina X bývá často číselná, zj. $X \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}\}$. Prvky matice mohou být ale i komplikovanější objekty, například algebraické výrazy, nebo funkce.

Jednoduše je možno definovat matice jako schématické uspořádání objektů – prvků matice (nebo také elementů matice) do m řádků a n sloupců. Takové matice potom označujeme jako matice typu $m \times n$ nebo m/n.

Matici typu $m \times n$ zapisujeme ve tvaru

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix},$$

nebo krátce ve tvaru $\mathbf{A} = (a_{ij})$, kde i je řádkový index a j je sloupcový index. Pro názornou ukázku je zde matice

$$\mathbf{A} = \begin{pmatrix} 1 & 7 & 3 \\ 2 & 6 & 4 \end{pmatrix}.$$

Matice **A** je typu 2×3 nad množinou $\mathbb{N}$. Platí například, že prvek $a_{23} = 4$, protože tento prvek leží ve druhém řádku a třetím sloupci matice **A**.

2.1.1 Názvosloví matic

Aby nedocházelo k omylům v termínech používaných při práci s maticemi.

- Je-li $m = n$, nazývá se matice **čtvercová**.

[1] Lineární algebra je odvětvím matematiky, která se zabývá vektory, vektorovými prostory, soustavami lineárních rovnic a lineárními transformacemi [17]

- V obecném případě $m \neq n$ je matice **obdélníková**.
- Množina všech prvků se stejným řádkovým a sloupcovým indexem se nazývá **hlavní diagonála** matice.
- **Nulová matice O** je matice, jejíž všechny prvky jsou nuly.
- **Jednotková matice E** je čtvercová matice, jejíž prvky mimo hlavní diagonálu jsou nuly a prvky na hlavní diagonále jsou rovny jedné.
- Matice **A** se nazývá **trojúhelníková** matice, přesněji **dolní trojúhelníková**, pokud pro libovolné dva indexy i, j platí $i > j \Rightarrow a_{ij} = 0$. Dolní trojúhelníková matice má nuly pod hlavní diagonálou.
- Analogicky je definována **horní trojúhelníková matice**, která má nuly nad hlavní diagonálou.
- Dvě matice $\mathbf{A}, \mathbf{B}$ se rovnají, když mají stejný typ a pro libovolné indexy i, j platí $a_{ij} = b_{ij}$. Pak píšeme $\mathbf{A} = \mathbf{B}$.

2.1.2 Operace s maticemi

Operace nad maticemi jsou velmi jednoduché.

Násobení matice číslem – každou matici $\mathbf{A}$ typu $m \times n$ lze vynásobit prvkem $c \in X$. Výsledkem $c\mathbf{A}$ je matice $\mathbf{C} = (c_{ij})$ typu $m \times n$, kde

$$c_{ij} = c \cdot a_{ij}. \tag{2.1}$$

Sčítaní matic – matice $\mathbf{A}, \mathbf{B}$ lze sečíst, když mají stejný typ $m \times n$. Pak výsledek $\mathbf{A} + \mathbf{B}$ je matice $\mathbf{C} = (c_{ij})$ typu $m \times n$, kde

$$c_{ij} = a_{ij} + b_{ij}. \tag{2.2}$$

Odečítání matic – odečítání matice $\mathbf{A}, \mathbf{B}$ lze pak definovat pomocí vztahů 2.2 a 2.1.

$$\mathbf{A} - \mathbf{B} = \mathbf{A} + (-1)\mathbf{B} \tag{2.3}$$

Násobení matic – pro násobení matic platí komplikovanější vztahy. Přesně dvě matice $\mathbf{A}, \mathbf{B}$ lze vynásobit v tomto pořadí, tj. vytvořit součin $\mathbf{A} \cdot \mathbf{B}$, když typy matic na sebe navazují v následujícím smyslu: pokud typ $\mathbf{A}$ je $m \times k$, typ $\mathbf{B}$ je $k \times n$, pak typ $\mathbf{A} \cdot \mathbf{B}$ je $m \times n$. Výsledkem násobení je tedy matice $\mathbf{C} = c_{ij}$ typu $m \times n$, přičemž platí

$$c_{ij} = \sum_{s=1}^{k} a_{is} b_{sj}. \tag{2.4}$$

Prvek ležící v i. řádku a j. sloupci výsledné matice tedy získáme tak, že procházíme i. řádek v matici $\mathbf{A}$ a jeho prvky postupně násobíme prvky ležícími v j. sloupci matice $\mathbf{B}$ a vytvořené součiny sečteme.

Transponování matice – libovolnou matici $\mathbf{A} = a_{ij}$ typu $m \times n$ lze transponovat. Výsledkem transpozice je matice $\mathbf{A}^T = aji$ typu $n \times m$.

Inverzní matice – nechť $\mathbf{A} = a_{ij}$ je matice typu $n \times n$. Čtvercová matice $\mathbf{B}$ typu $n \times n$ se nazývá **inverzní** k matici $\mathbf{A}$, když

$$\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A} = \mathbf{E}. \tag{2.5}$$

Jestliže je matice $\mathbf{B}$ inverzní maticí k matici $\mathbf{A}$, pak se tato inverzní matice značí jako matice $\mathbf{A}^{-1}$.

2.2 Transformace

Následující část je převzata z knihy Moderní počítačová grafika [23].

Geometrické transformace jsou jedněmi z nejčastěji používaných operací v počítačové grafice. Transformace je možno rozdělit na lineární a nelineární. Mezi lineární patří otáčení, posunutí, změna měřítka, zkosení a operace vzniklé jejich skládáním. S nelineárními transformacemi se v počítačové grafice setkáváme při složitějších změnách tvaru grafických objektů, např. deformace prostorových modelů nebo warping obrazu. Zvláštní transformací je potom projekce, která převádí objekty z vícerozměrného prostoru do prostoru o méně rozměrech. Nejčastěji se setkáváme s projekcí trojrozměrné scény do roviny obrazu.

Objekty jsou popsány svými souřadnicemi, které jsou vztaženy ke zvolenému souřadnicovému systému. Geometrické transformace mohou být aplikovány na jednotlivé souřadnice objektu, který tak mění svou polohu. Další možností je podrobit transformaci souřadnicový systém. To obyčejně činíme za účelem získání výhodnější reprezentace objektu pro jeho další zpracování.

Dále budeme pracovat s bodem P, který má kartézské souřadnice $[X,\ Y,\ Z]$ ve třech rozměrech. Transformací bodu P získáme bod P' o souřadnicích $[X',\ Y',\ Z']$. Transformací objektu budeme rozumět aplikaci transformace na všechny body, ze kterých se objekt skládá nebo, pokud to transformace a současně reprezentace objektu umožňují, aplikaci transformace na parametry, které objekt jednoznačně určují. Například posunutí koule reprezentované středem a poloměrem nebudeme řešit transformací každého povrchového bodu, stačí pouze transformovat středový bod.

Pro zjednodušení výpočtů transformací se s výhodou používá reprezentace pomocí homogenních souřadnic. Tato reprezentace se používá z několika důvodů. Homogenní souřadnice umožňují vyjádření nejčastěji používaných lineárních transformací pomocí jedné matice, což v nehomogenních kartézských souřadnicích není možné. Skládání transformací se v tomto kontextu realizuje jako násobení matic, inverzní transformace je reprezentována inverzní maticí, atd.

Uspořádaná čtveřice čísel $[x,\ y,\ z,\ w]$ představuje homogenní souřadnice bodu P s kartézskými souřadnicemi $[X,\ Y,\ Z]$ ve třech rozměrech, platí-li:

$$X = \frac{x}{w},\ Y = \frac{y}{w},\ Z = \frac{z}{w},\ w \neq 0$$

Bod P je svými homogenními souřadnicemi určen jednoznačně. Souřadnici w se také nazývá váhou bodu. Často se volí $w = 1$, potom jsou homogenní souřadnice bodu $[X,\ Y,\ Z,\ 1]$. Homogenní souřadnice transformovaného bodu P' s kartézskými souřadnicemi $[X',\ Y',\ Z']$ budeme označovat $[x',\ y',\ z',\ w']$. Rozdíl dvou bodů $A = [a_0,\ a_0,\ a_2,\ 1]$ a $B = [b_0,\ b_1,\ b_2,\ 1]$ určí vektor $\vec{p} = (a_0 - b_0,\ a_1 - b_1,\ a_2 - b_2,\ 0)$, sečtením bodu a vektoru dostaneme bod.

Obecnou matici typu 4×4 reprezentující lineární transformaci bodu $P = [x,\ y,\ z,\ w]$ na bod $P' = [x',\ y',\ z',\ w']$ budeme označovat $\mathbf{A}$, její speciální případy pak podle druhu

transformace, např. **T** (translace), **R** (rotace). Transformaci souřadnic zapíšeme

$$P' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \mathbf{A}_{4\times4} \cdot P = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

2.3 Trojrozměrné geometrické transformace

Lineární transformace v prostoru jsou zobecněním rovinných transformací. V počítačové grafice se pro transformace používají matice typu 4×4.

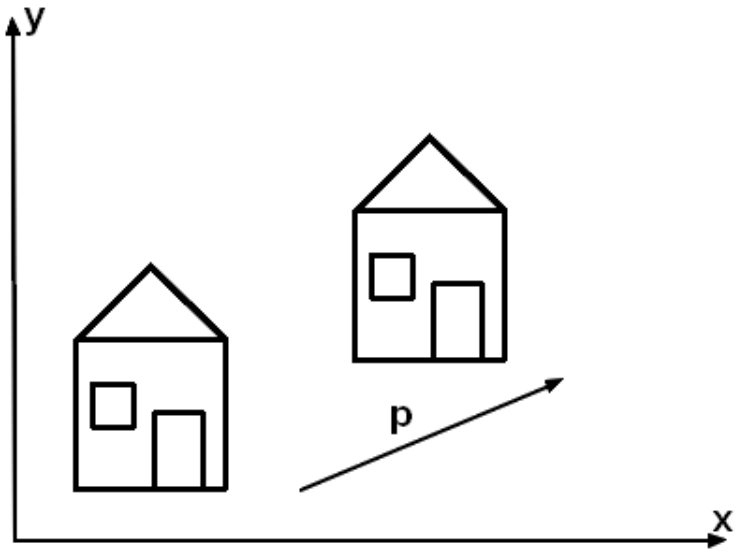
2.3.1 Posunutí

Posunutí je určeno vektorem posunutí $\vec{p} = (X_t,\ Y_t,\ Z_t)$. Posunutí je znázorněno na obrázku 2.1. Transformační matice posunutí **T** má tvar

$$\mathbf{T} = \mathbf{T}(X_t,\ Y_t,\ Z_t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_t & Y_t & Z_t & 1 \end{bmatrix}$$

a inverzní matice $\mathbf{T}^{-1}$

$$\mathbf{T}^{-1} = \mathbf{T}(-X_t,\ -Y_t,\ -Z_t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_t & -Y_t & -Z_t & 1 \end{bmatrix}$$



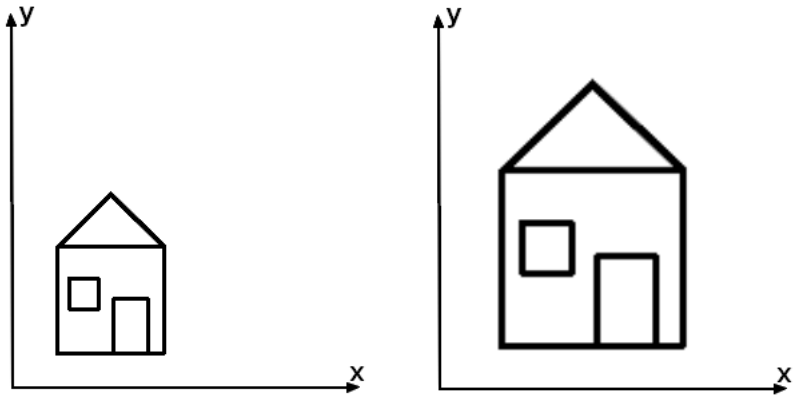
Obrázek 2.1: Posunutí podle vektoru $\vec{p}$ (Zdroj [1])

2.3.2 Změna měřítka

Změnu měřítka (scale) v prostoru popisují matice:

$$\mathbf{S}(s_x,\ s_y,\ s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{S}^{-1}(s_x,\ s_y,\ s_z) = S(\frac{1}{s_x},\ \frac{1}{s_y},\ \frac{1}{s_z}),$$

v níž koeficienty $s_x \neq 0,\ s_y \neq 0$ a $s_z \neq 0$ určují změnu ve směru příslušné souřadnicové osy. Pomocí měřítkových koeficientů můžeme realizovat některou z transformací souměrnosti v prostoru (středovou souměrnost, souměrnost podle roviny a osovou souměrnost). Např. souměrnost podle roviny xy bude realizována pomocí koeficientů $s_x = 1,\ s_y = 1,\ s_z = -1$.



Obrázek 2.2: Změna měřítka s koeficienty $s_x = 1{,}5,\ s_y = 2$ (Zdroj [1])

2.3.3 Zkosení

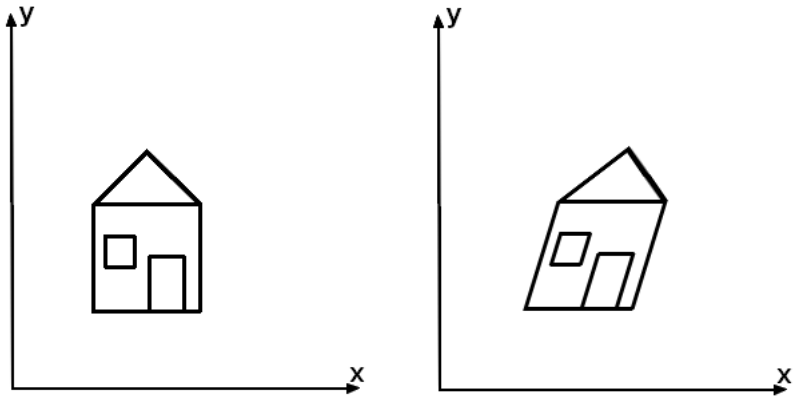
Operaci zkosení (shear) ve třech rozměrech můžeme rozdělit na tři případy zkosení ve směru jednotlivých rovin xy, xz, yz. Ve všech třech případech určují koeficienty sh_x, sh_y a sh_z míru zkosení v odpovídajícím směru. Matice jednotlivých transformací zkosení:

$$\mathbf{Sh}_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Sh_x & Sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Sh}_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_x & 1 & Sh_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Sh}_{yz} = \begin{bmatrix} 1 & Sh_y & Sh_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.4 Otáčení

Otáčení ve třech rozměrech může být jedním z podpřípadů otáčení kolem jednotlivých souřadnicových os. Matice $\mathbf{R}_x$ reprezentuje otáčení kolem osy x o úhel α a odpovídající inverzní matice:

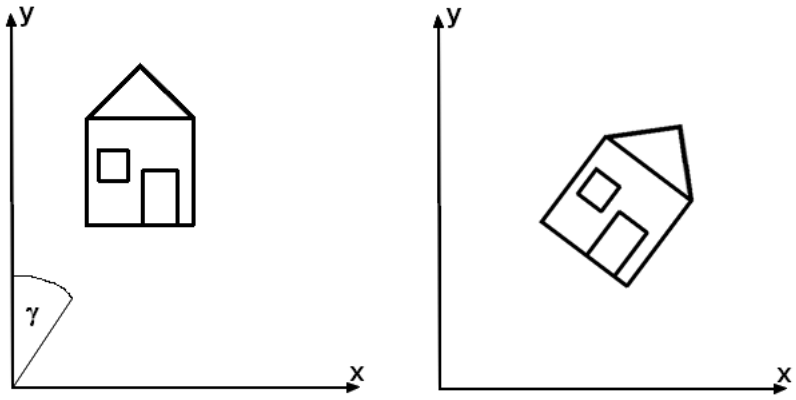
$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_x^{-1}(\alpha) = \mathbf{R}_x(-\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Obrázek 2.3: Zkosení ve směru osy x (Zdroj [1])

Odpovídajícím způsobem jsou sestaveny matice pro otáčení kolem osy y a z:

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Obrázek 2.4: Otočení objektu kolem osy z (Zdroj [1])

2.3.5 Otáčení kolem obecné osy

Otáčení kolem obecné osy v prostoru lze realizovat složením několika dílčích transformací kolem os x, y, z, nalezení příslušných transformací (úhlů otočení) však není jednoduché. Lze však využít Rodriguesovy formule, která předvádí rotační úlohu na promítaní a skládání několika vektorů. Výchozí situace je znázorněna na obrázku 2.5. Předpokládejme, že osa otáčení je určena počátkem souřadnicového O systému a jednotkovým vektorem $\vec{a}$. Polohový vektor $\vec{x}$ transformovaného bodu X je kolem osy této osy otočen o úhel α a výsledný polohový vektor je označen $\vec{x}'$. Za těchto předpokladů lze rotaci popsat vztahem

$$\vec{x}' = \cos\alpha \cdot \vec{x} + (1 - \cos\alpha)(\vec{a}.\vec{x})\vec{a} + \sin\alpha(\vec{a} \times \vec{x}).$$

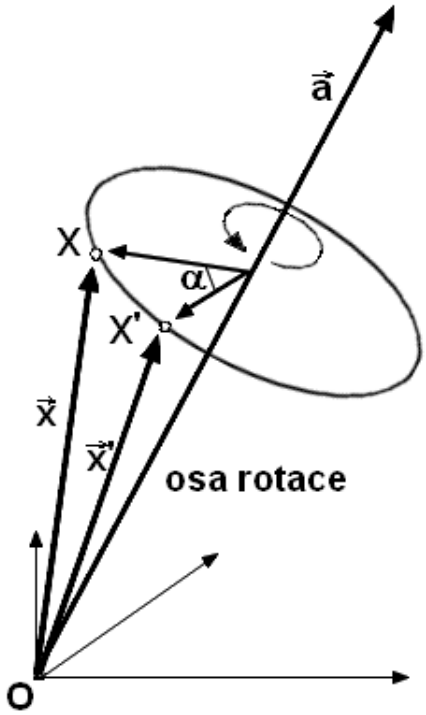
Rotaci bodu X lze přepsat do maticového tvaru s využitím skalárního a vektorového součinu, matice $\mathbf{I}$ je jednotková matice (identita).

$$X' = \mathbf{R}(\vec{a}, \alpha).X.$$

$$\mathbf{R}(\vec{a}, \alpha) = \cos\alpha.\mathbf{I} + (1 - \cos\alpha).\begin{bmatrix} a_x^2 & a_x a_y & a_x a_z & 0 \\ a_x a_y & a_y^2 & a_y a_z & 0 \\ a_x a_z & a_y a_z & a_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \sin\alpha.\begin{bmatrix} 0 & -a_z & a_y & 0 \\ a_z & 0 & -a_x & 0 \\ -a_y & a_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Zvolíme-li například osu rotace shodnou s osou x, tj. $\vec{a} = [1,\ 0,\ 0]$, po dosazení do předešlého vztahu dostaneme matici rotace kolem osy x. Obecnou rotaci řešíme obdobně, jako u obecné rotace v rovině, tj. na ose rotace zvolíme bod $P = [P_x,\ P_y,\ P_z,\ 1]$, vypočteme vektor ve směru osy $\vec{a}$, zvolený bod posuneme do počátku souřadnicového systému, otočíme transformované objekty o daný úhel a zpětným posunutím vrátíme výsledek do výchozí pozice. Transformaci matice $\mathbf{A}$ bude složena ze tří základních transformací

$$\mathbf{A} = \mathbf{T}(P_x,\ P_y,\ P_z).\mathbf{R}(\vec{a}, \alpha).\mathbf{T}(-P_x,\ -P_y,\ -P_z).$$



Obrázek 2.5: Rotace kolem obecné osy (Zdroj [1])

2.3.6 Kvaterniony

Pro reprezentaci rotací nejsou matice vždy nejvýhodnější. Jednak obsahují nadbytečné údaje (devět čísel místo tří hodnot úhlů natočení) a hlavně jsou obtížně interpolovatelné.

Přechod z jedné obecné polohy do jiné pomocí interpolovaného otáčení ve třech směrech nelze pomocí matic jednoduše vyřešit. Kvaterniony byly navrženy irským matematikem W. R. Hamiltonem v 19. století jako analogie komplexních čísel v prostoru. Praktický význam teorie kvaternionů byl rozpoznán nejen v kvantové mechanice, ale i při řešení animačních úloh v počítačové grafice. Podrobnější matematické vlastnosti kvaternionů lze nalézt v [23] nebo v práci [7].

Kvaternion $\mathbf{q}$ je reprezentován čtveřicí $\mathbf{q} = w + xi + yj + zk$, kde $w,\ x,\ y,\ z$ jsou reálná čísla a $i,\ j,\ k$ jsou kvaternionové jednotky (i odpovídá komplexní jednotce). Kvaternion *sdružený* ke kvaternionu $\mathbf{q}$ definujeme jako $\mathbf{q}^* = w - xi - yj - zk$. Velikost kvaternionu $\mathbf{q}$ je $|\mathbf{q}| = \sqrt{w^2 + x^2 + y^2 + z^2}$. Kvaternion jehož velikost je jedna nazýváme *jednotkový* kvaternion. Pro jednotkový kvaternion $\mathbf{q}$ platí $\mathbf{q}^*\mathbf{q} = \mathbf{q}\mathbf{q}^* = 1$.

Kvaternion si můžeme také představit jako dvojici složenou ze skalární (s) a vektorové (v) části. Tento pohled vede k jednoduché notaci

$$\mathbf{q} = (s, \vec{v}).$$

Libovolnou rotaci lze popsat úhlem α a jednotkovým vektorem $\vec{a} = (a_0,\ a_1,\ a_2)$, který reprezentuje osu otáčení. Taková rotace odpovídá jednotkovému kvaternionu

$$\mathbf{q} = \cos(\alpha/2) + a_0 \sin(\alpha/2)i + a_1 \sin(\alpha/2)j + a_2 \sin(\alpha/2)k.$$

Tento zápis se obvykle zkracuje na

$$\mathbf{q} = \cos(\alpha/2) + \mathbf{a} \sin(\alpha/2), \tag{2.6}$$

přičemž $\mathbf{a} = (0,\ \vec{a})$ chápeme jako jednotkový kvaternion se skalární částí $s = 0$, tj. $\mathbf{a} = a_0 i + a_1 j + a_2 k$. Přiřazení jednotkového kvaternionu k rotaci není jednoznačné, neboť $-\mathbf{q}$ odpovídá téže rotaci jako $\mathbf{q}$. Kvaternion $1{+}0i{+}0j{+}0k$ představuje identitu (rotaci s nulovým úhlem). Sdružený kvaternion $\mathbf{q}^*$ reprezentuje inverzní rotaci.

Vektor $\vec{v} = (v_0,\ v_1,\ v_2)$ můžeme chápat jako kvaternion $\mathbf{v}$ s nulovou reálnou částí, tedy $\mathbf{v} = v_0 i + v_1 j + v_2 k$. Otočení vektoru $\vec{v}$ jednotkovým kvaternion $\mathbf{q} = \cos(\alpha/2) + \mathbf{a}\sin(\alpha/2)$ kolem osy $\mathbf{a}$ o úhel α spočítáme pomocí kvaternionového násobení

$$\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^*. \tag{2.7}$$

Platí, že reálná část kvaternionu $\mathbf{v}'$ vyjde vždy nulová, tedy $\mathbf{v}' = v'_0 i + v'_1 j + v'_2 k$. To nás opravňuje tvrdit, že vektor $(v'_0,\ v'_1,\ v'_2)$ představuje rotaci vektoru $\vec{v}$ zadanou kvaternionem $\mathbf{q}$.

Vzorec (2.7) má důležitý důsledek pro skládání rotací. Mějme rotaci reprezentovanou jednotkovým kvaternionem $\mathbf{r}$ a otočme jím vektor $\vec{v'}$ reprezentovaný kvaternionem $\mathbf{v}'$. Výsledek otáčení lze napsat jako

$$\mathbf{v}'' = \mathbf{r}\mathbf{v}'\mathbf{r}^* = \mathbf{r}\mathbf{q}\mathbf{v}\mathbf{q}^*\mathbf{r}^*. \tag{2.8}$$

Vidíme že je to totéž, jako kdybychom vektor $\vec{v}$ otočili pomocí jednotkového kvaternionu $\mathbf{rq}$. Můžeme tedy shrnout: složení rotací odpovídá násobení kvaternionů.

Výpočet otočení vektoru podle rovnice (2.7) je pomalejší, než násobení vektoru rotační maticí. Potřebujeme tedy převod mezi kvaterniony a rotačními maticemi. Je důležitý i proto, že některé knihovny a grafický hardware používají rotace v maticovém tvaru. Převod jednotkového kvaternionu $\mathbf{q} = w + xi + yj + zk$ na rotační matici $\mathbf{R}$ je snadný:

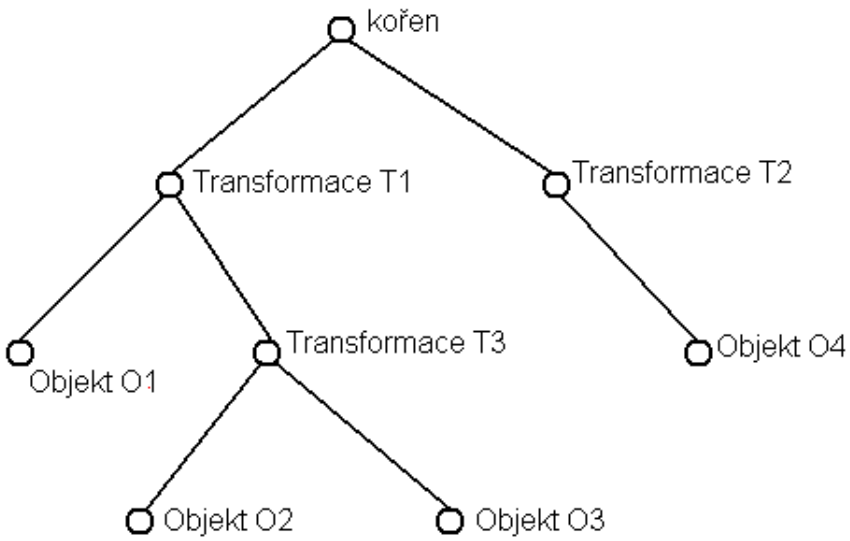
$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy & 0 \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx & 0 \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2.4 Graf scény

Scénou nazýváme množinu prostorových objektů doplněnou dalšími informacemi potřebnými pro jejich zobrazení. Přestože se zdá, že vytvoření scény je poměrně jednoduchým závěrečným krokem po předchozím vymodelování individuálních objektů, je tvorbu prostorové scény možno chápat jako samostatnou úlohu. Zatímco systémy pro modelování těles zpracovávají geometrická data definující tvar jednotlivých objektů, systémy pro tvorbu scén přidávají k objektům transformace (2.2) do jejich cílové polohy, určují informace potřebné pro zobrazování (světla, kamery) a především umožňují do scény opakovaně vkládat stejně nebo podobně vypadající objekty (instance). Scéna obvykle obsahuje:

- nezobrazované objekty – kamery, osvětlení scény
- zobrazované objekty – jejich geometrie, barevné vlastnosti, textury
- prvky definující logickou strukturu scény – definice skupin a jejich instancí
- transformace – definované hierarchicky kvůli snadnější manipulaci s objekty

Více detailů o jednotlivých položkách scény je možno nalézt v knize [23].



Obrázek 2.6: Schéma jednoduchého grafu scény (Zdroj [1])

Tělesa a další zobrazované objekty je vhodné uspořádat do datové struktury, která umožňuje seskupovat logicky k sobě patřící části, efektivně je transformovat a jejich instance vkládat úsporným způsobem do prostoru scény. Tato struktura se obecně nazývá *graf scény* (obrázek 2.6).

Graf scény je n-ární strom, tj. takový graf, v němž lze pro každý uzel nalézt právě jednoho předchůdce. Výjimku tvoří kořen stromu, který stojí na nejvyšší úrovni. Graf scény může obsahovat i několik stromů, tzv. les. Graf scény není ve všech systémech definován stejným způsobem, odlišnosti jsou v typech uzlů, v pravidlech pro stavbu stromu i pro interpretaci dat ve stromu uložených. Dále jsou uvedeny principiální vlastnosti grafů scény bez ohledu na systém.

Důležitou vlastností stromu je schopnost vyjádřit vztahy mezi uzly. Jedním z těchto vztahů je *dědičnost*. Umožňuje v jednom místě stromu definovat vlastnost, která bude platná pro řadu dalších uzlů. Rozsah platnosti je dán vzájemnou polohou uzlů v rámci stromu. Lze například stanovit, že vlastnost definovaná v uzlu je platná pro všechny následníky tohoto uzlu. Pravidla pro dědění mohou být být definovány různými způsoby.

V rámci této části se soustředíme pouze na význam transformací v grafu scény. Je zřejmé, že každé těleso může být pevně umístěno do své cílové pozice, tj. veškeré souřadnice tělesa mohou být předem transformovány pomocí některé z trojrozměrných transformací (viz část 2.3). Pro manipulaci se scénou je však mnohem výhodnější ponechat těleso v jejich základních polohách (lokálních souřadnicových systémech) a potřebné transformace zapsat do grafu scény, například v podobě transformační matice. Hierarchické uspořádaní scény umožní transformace skládat a změnou jedné transformace ovlivnit celý podstrom.

Na obrázku (2.6) grafu scény jsou některé uzly označeny transformace Tx. Názorný příklad skládání transformací v grafu scény si uvedeme právě pro toto schéma. Objekt $O1$ bude ovlivněn pouze transformací $T1$. Objekty $O2$ a $O3$ budou ovlivněny složením transformací $T1.T3$ a objekt $O4$ bude opět ovlivněn pouze transformací $T2$. Jestliže ovšem změníme transformaci $T1$, ovlivníme výsledné transformace pro objekty $O1$, $O2$ a $O3$.

2.5 Haptická technologie

Jak je možné se dočíst na Wikipedii [16] nebo na stránkách společnosti Immersion [4] slovo „haptický“ pochází z řečtiny a znamená dotyk nebo pocit dotyku. Což doslova znamená, že haptický je každý náš úkon, při kterém dochází ke kontaktu (psaní na klávesnici, vytáčení na telefonním přístroji, atd.). V počítačovém světě se haptickou technologií rozumí zařízení, které pomocí síly, vibrace nebo pohybu zprostředkovává uživateli pocit dotyku při práci s virtuálními objekty, které nemají hmotnou formu.

Haptické technologie jsou digitálním výsledkem zkoumání lidského dotyku a fyzické interakce s okolním prostředím. Díky této technologii bylo možné pozorovat a zkoumat lidský dotyk a lépe pochopit jeho podstatu, především pak mozkové pochody spojené právě s prováděním dotyků a získáváním podnětů. Důvod k takovému vývoji je zřejmý, neboť dotyk je přirozeným, velmi jednoduchým a intuitivním prostředkem lidské interakce.

Je vhodné zmínit, že haptické technologie jsou opakem měřicích senzorů, které zaznamenávají sílu vyvinutou člověkem, zatímco haptické zařízení vyvíjí silovou odezvu proti uživateli. Nejlepším příkladem jednoduchého haptického systému jsou dnešní herní ovladače, které díky zpětné odezvě dokáží uživateli simulovat odpor, například herní volanty, které hapticky simulují odpor kladený autem při jízdě nebo joysticky pro ovládání leteckých simulátorů.

Historie haptických zařízení se podle [11] datuje do roku 1948, kdy byl vyroben první dálkově ovládaný manipulátor [20] Národní laboratoří Argonne (obr. 2.7). Tyto zařízení, které spojovaly poznatky z elektroniky, hydrauliky a mechaniky, dovolily zacházet s objekty pomocí, jak bychom dnes řekli, mechanických ramen. Jednalo se o první zařízení využívající systému Master–Slave, kdy se pracovní část zařízení (slave) pohybuje a provádí stejné úkony jako ovládací část (master). Později v roce 1949 byly podobná zařízení použitá v jaderných reaktorech pro transport materiálu a jednotlivých částí reaktoru.

První haptické zařízení se objevily v letectví, kde sloužily k oznámení vyjimečné situace. Tato zařízení využívala servo systémů a jejich úkolem bylo skrze třes ovládacích prvků naznačit pilotovi nebezpečný stav letadla, například při strmém stoupání.

Obrázek 2.7: První master–slave manipulátor (Zdroj: [5])

V dnešní době se haptická zařízení používají v mnoha odvětvích lidské činnosti, počínaje hrami a uměním přes design a strojírenství až po robotiku a medicínu.

Především pak v medicíně našla haptická zařízení uplatnění při náročných operacích, kde se vyžaduje vysoká přesnost nebo se minimalizuje riziko. Haptická zařízení tohoto uplatnění byla vyrobena již v 50. letech 20. století, ale až na konci 80. let se objevila první skutečná zařízení, která simulovala odezvu odpovídající skutečnému pocitu dotyku. Dnes se tyto zařízení používají při laboratorních pracích nebo při operacích na dálku (obr. 2.8), kdy chirurg pomocí několika ovladačů a kamer provádí operaci na druhém konci světa.

2.5.1 Společnost SensAble Technologies

Jednou z předních světových společností zabývající se vývojem haptických zařízení je v současnosti společnost SensAble Technologies [21]. Společnost byla založena v roce 1993 a již od počátku je společnost hlavním výrobcem zařízení a technologií pro virtuální dotyk, které umožňují uživatelům objekty na obrazovkách nejenom vidět a slyšet, ale také je reálně cítit. Společnost se vyvinula s akademického výzkumu prováděného na MIT[2] v 90. letech 20. století průmyslovými průkopníky Thomasem Massie a Dr. Kennethem Salisbury. S 32 uznanými patenty a s více jak 6000 systémy, které jsou nainstalovány po celém světě, jsou haptická zařízení společnosti SensAble Technologies využívána v širokém spektru lidské činnosti od medicínského prostředí přes výrobu hraček a obuvi až po uplatnění zařízení při výzkumu. Společnost produkuje společně se svými zařízeními i vlastní řešení pro 3D modelování.

V roce 2006 založila společnost divizi SensAble Dental, která se zaměřuje na vývoj

[2]angl. Massachusetts Institute of Technology, Massachusettský technologický institut je soukromá výzkumná univerzita ve městě Cambrige, USA.

Obrázek 2.8: Chirurgický robot pro vzdálené ovládání

integrovaných digitálních řešení pro stomatologický průmysl.

Některé výrobky společnosti SensAble Technologies (více informací o jednotlivých položkách na [21]):

- Modelovací systém FreeForm(R)
- SensAble Dental Lab system
- Zařízení Phantom(R)
- OpenHaptics(R) software

2.5.2 Haptický skener Phantom Omni

Jak jsem již zmínil v předchozí kapitole o společnosti SensAble 2.5.1, společnost produkuje několik haptických zařízení řady Phantom. Jednotlivé modely se liší podle nároků, které jsou na ně kladeny ze strany průmyslových i vědeckých uživatelů.

Zařízení modelové řady Phantom Premium (obr. 2.9) dovoluje uživateli pracovat s největším pracovním prostorem, s nejpřesnějším snímáním a s největší silovou odezvou, kterou mohou haptická zařízení poskytnout. Tato zařízení jsou určena pro profesionální využití v průmyslu, například ve strojírenství, kde je míra přesnosti kladena nade vše. Modely Premium jsou schopny pracovat s rozlišením v řádech tisícin milimetrů a generovat sílu o velikosti až desítek Newtonů.

Střední modelovou řadou je Phantom Desktop a následuje řada Phantom Omni (obr. 2.9 a 2.10). Řada Desktop je přesnější a může generovat vyšší sílu než řada Omni. Cílovou skupinou uživatelů jsou pochopitelně na přesnost méně náročné průmyslové disciplíny a výzkumné skupiny, které vyžadují velkou výkonnost.

Obrázek 2.9: Haptické skenery Phantom Desktop (vlevo) a Phantom Premium (vpravo) (Zdroj [21])

Modely Phantom Omni (obr. 2.10) jsou, jak sám výrobce tvrdí, nejvýhodnějším haptickým zařízením, které je dnes možné koupit. Koncovými uživatelů jsou především skupiny zabývající se haptickým programováním. Přesnost snímání pracovního prostoru zařízení je 450 dpi, což je rozlišení prostoru v řádech desetin milimetrů a velikost sil, které je zařízení schopno generovat se pohybuje v řádech desetin Newtonů.

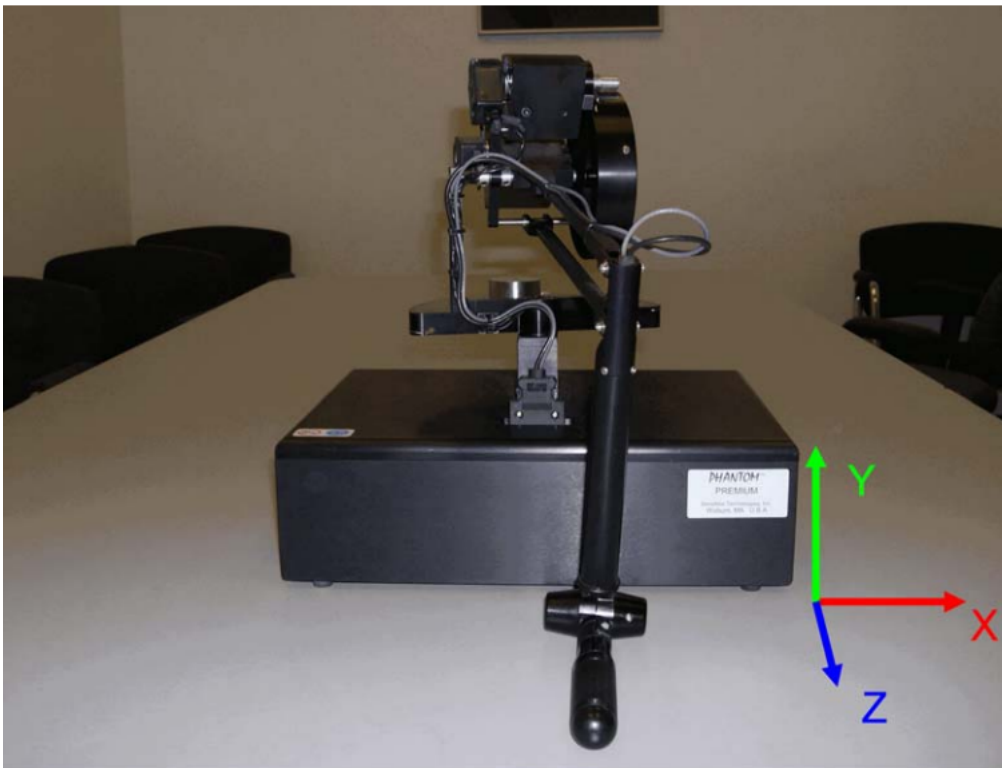
Obrázek 2.10: Haptický skener Phantom Omni (Zdroj [21])

Model Phantom Omni je velmi kompaktním zařízením. Zařízení Omni dovoluje až 6DOF[3], což znamená, že zařízení vrací, stejně jako skener Microscribe [1], aktuální souřadnice snímacího hrotu ve směrech os X, Y a Z kartézského souřadného systému společně s orientací hrotu. Je však možné použít pouze pozici, tedy pouze souřadnice ve směru os a nepracovat s orientací. Takový stupeň volnosti se poté označuje 3DOF.

A stejně jako u skeneru Microscribe i zde jsou použity pro vyjádření orientace úhly otočení kolem os X, Y a Z, které se podle nazývají Tait-Bryanovy úhly [22]. Na obr.

[3] 6DOF – z angl. Six Degrees Of Freedom, 6 stupňů volnosti.

2.11 je znázorněna orientace jednotlivých souřadných os haptických zařízení Phantom. Pro snímání polohy je použit trojrozměrný kartézský souřadný systém, jehož kladný směr osy Z směřuje k uživateli zařízení a osa Y určuje vertikální složku pozice a osa X směřuje doprava od uživatele.



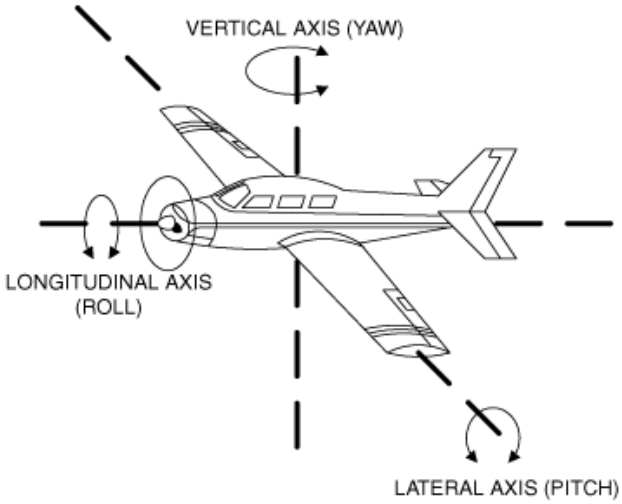
Obrázek 2.11: Orientace souřadných os haptických zařízení Phantom (Zdroj [10])

Způsob vyjádření orientace tělesa za použití Tait-Bryanových úhlů se používá v letectví (obr. 2.12), používá se i anglické názvosloví rotací kolem os:

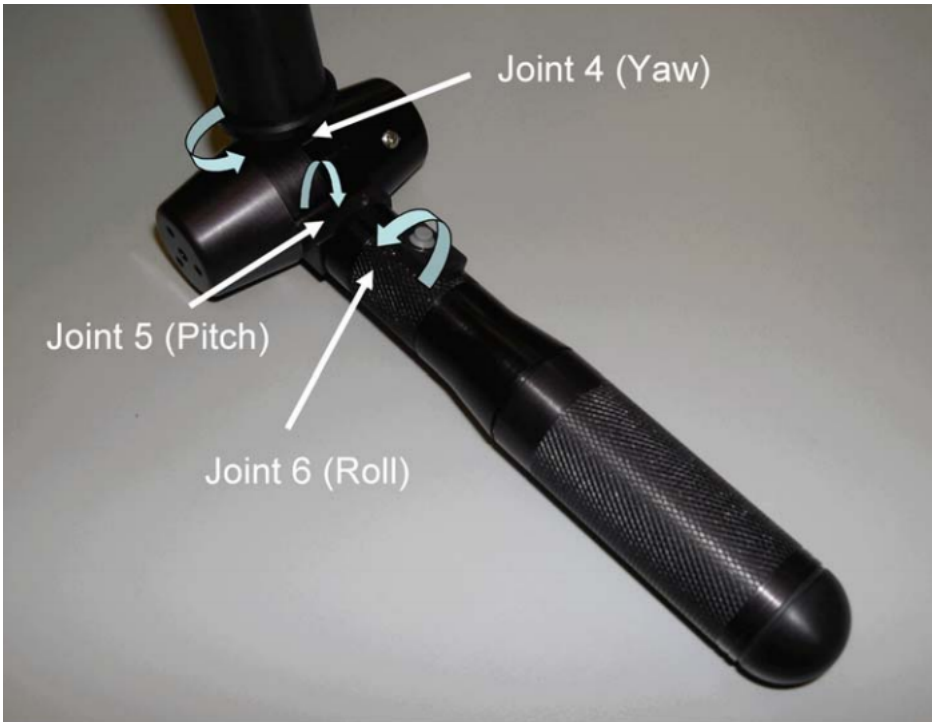
- „Roll“ – rotace snímacího hrotu zařízení kolem osy Z.
- „Pitch“ – rotace kolem osy X.
- „Yaw“ – rotace kolem osy Y.

Jednotlivá otočení v souřadném prostoru zařízení Omni jsou znázorněna na obr. 2.13.

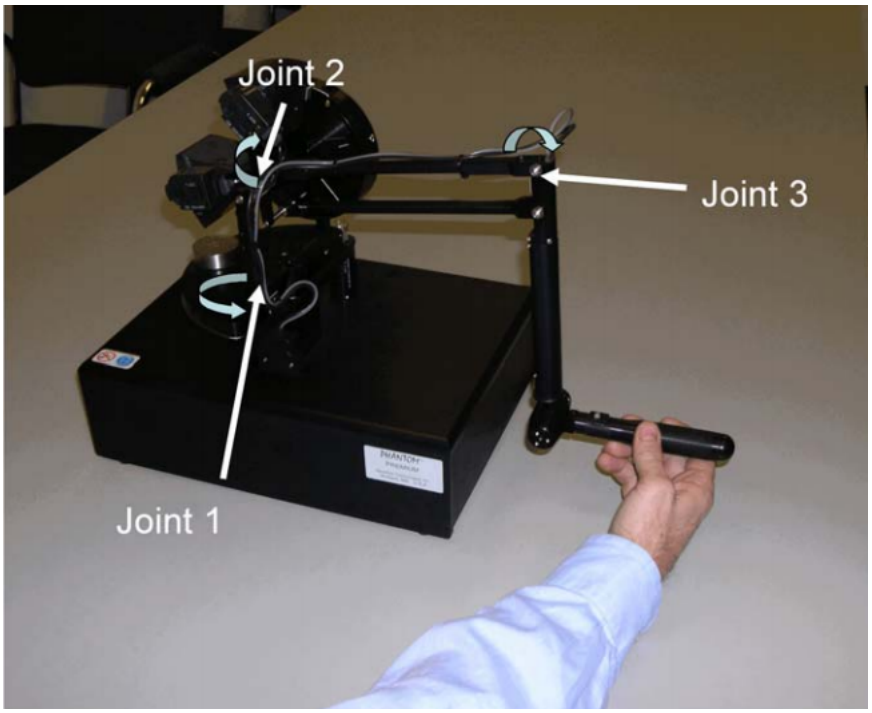
Jelikož je Phantom Omni haptickým zařízením, musí se generovat silová odezva. Silové působení se produkuje ve třech kloubech zařízení (obr. 2.14) a každý kloub tak vytváří silovou odezvu pro adekvátní směr. Jelikož se síla generuje pouze do směrů, zařízení Phantom Omni se označuje jako 3DOF haptické zařízení. Pouze některé modely řady Premium jsou schopny generovat sílu 6DOF, tzn. silově působit i při rotaci snímacího hrotu. Síla je tvořena velmi sofistikovaným systémem servo motorů, který se vyvíjel po dlouhý časový úsek 2.5 a ovlivňuje pohyb a pozici koncové části ramena – snímacího hrotu.



Obrázek 2.12: Pořadí a význam otočení Tait-Bryanových úhlů v letectví (Zdroj [1])



Obrázek 2.13: Popsání orientace rotací haptického zařízení Phantom (Zdroj [10])



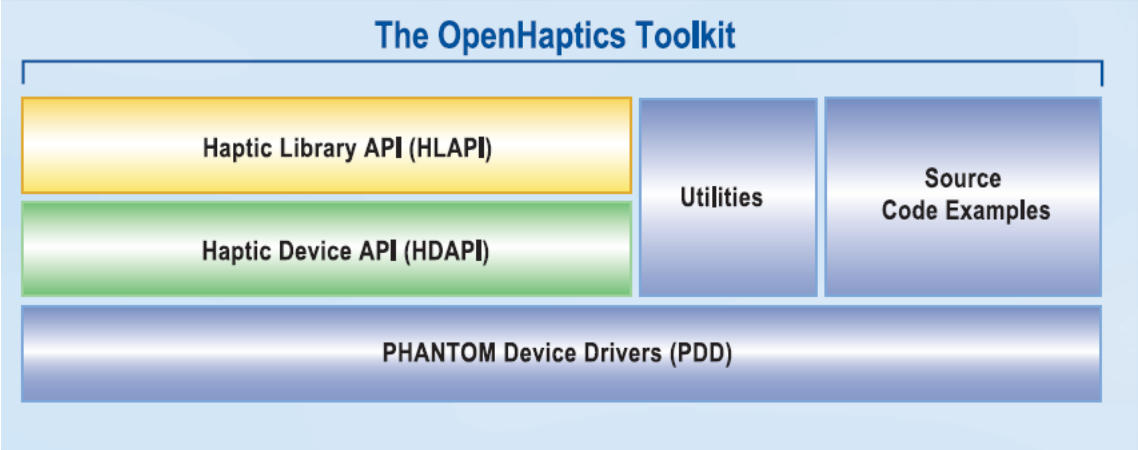
Obrázek 2.14: Pozice jednotlivých kloubů generujících sílu (Zdroj [10])

Kapitola 3

Návrh haptické aplikace

3.1 OpenHaptics Toolkit

Společnost SensAble dodává ke svým výrobkům vývojový toolkit OpenHaptics [8], který umožňuje vývojářům implementovat programy využívající haptická zařízení. Za pomoci tohoto toolkitu[1] je programátor schopen vytvářet aplikace využívající všechny modely řady Phantom 2.5.2 a i několik zařízení najednou. Toolkit je tvořen relativně velkým souborem ovladačů, knihoven a podpůrných nástrojů (obr. 3.1), jenž pokrývají v podstatě celou problematiku práce se zařízením a vývoj aplikací využívajících haptického programování a případně grafický výstup. Dále v práci popíši dvě hlavní části toolkitu OpenHaptics, knihovny HDAPI a HLAPI (kapitoly 3.1.1 a 3.1.3). O zbývajících částech toolkitu se zmíním okrajově a jejich úplnou funkčnost nebudu zpracovávat. Více o OpenHaptics toolkitu je možné se dočíst na stránkách výrobce [21] nebo v dokumentaci [10]. Přesné popisy všech zmíněných funkcí a parametrů jsou k nalezení, také v [9], ale tento dokument není volně dostupný a lze jej získat jedině s toolkitem po registraci u výrobce.



Obrázek 3.1: Struktura a součásti OpenHaptics toolkitu (Zdroj [8])

Haptické programování zprostředkovává uživateli různé silové podněty, především zamezení v žádaném pohybu nebo úpravu pohybu pomocí síly (silová odezva). Na rozdíl od

[1] Angl. toolkit je možné přeložit jako „souprava nástrojů“. V kontextu této práce budu termín toolkit používat pro soubor knihoven a pomocných nástrojů určených pro práci se zařízením.

vytváření grafických aplikací má haptické programování několik odchylek, které vycházejí z fyzikálních vlastností vnímání lidskými smysly.

Jedním z hlavních rozdílů je frekvence, se kterou se musí obnovovat haptický obraz. Aby lidské oko vidělo pohyb, je nutné produkovat obraz rychlostí alespoň 24 snímků za sekundu. Lidský hmat je ovšem podstatně citlivější a je tedy nutné produkovat haptický vjem několikanásobně rychleji. Knihovny OpenHaptics toolkitu standardně pracují na frekvenci 1000 Hz (1000 snímků za sekundu), aby byla zajištěna spojitost silového vjemu.

Dalším problémem je různorodost fyzikálních sil a jejich popisů. I přes nnožství variací je výsledná síla vždy vektor, který má počátek, udaný směr a velikost, i když se může průběh síly v čase značně měnit. Podle [10] se v haptickém programování nejčastěji využívají tři hlavní skupiny silových účinků: síly závislé na pohybu, síly závislé na času a kombinace předešlých dvou. Účinek síly se skoro vždy vztahuje ke koncovému bodu, kterým je v případě haptického zařízení snímací hrot.

Přehled polohově závislých silových účinků:

Pružina – pružinový efekt je pravděpodobně nejčastěji používaným silovým efektem vůbec. Popis průběhu je velmi jednoduchý a univerzální. Působící síla se počítá skrze Hookův zákon:

$$F = k \cdot x \tag{3.1}$$

kde k je tuhost pružiny a x je vektor posunu, který je rozdílem pozice pevného bodu uchycení pružiny a koncového bodu. Síla v každém okamžiku směřuje k bodu uchycení.

Tlumení – používá se hlavně pro redukci vibrací. S fyzikálního pohledu je tlumení koncového bodu závislé na rychlosti bodu. Síla tlumení se vypočítává:

$$F = -b \cdot v \tag{3.2}$$

kde b je konstanta tlumení a v je současná rychlost koncového bodu. Znaménko mínus znamená, že síla vždy působí právě na opačnou stranu než je směr pohybu bodu.

Tření – vzniká mezi tělesy při jejich vzájemném pohybu. Existuje několik typů tření, ale všechny by se daly popsat jako:

$$F = -c \cdot F_0 \tag{3.3}$$

kde c je koeficient tření a F_0 je síla působící na těleso. Tření působí na rozhraní těles a má směr opačný než je směr pohybu.

Setrvačnost – se vyskytuje při pohybu hmotných objektů, které popisujeme jejich hmotností. Setrvačná síla se získává z Newtonova druhého pohybového zákona:

$$F = m \cdot a \tag{3.4}$$

kde m je hmotnost tělesa a a je zrychlení tělesa v daném okamžiku pohybu.

Časově závislé silové účinky:

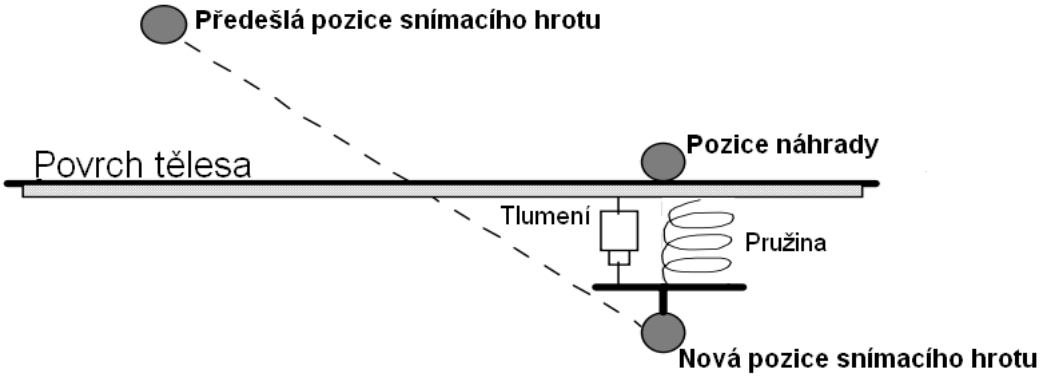
Konstantní – účinek má konstantní velikost i směr působení. Tento efekt se používá například pro simulaci zemského gravitačního pole, ale je samozřejmě možné simulovat pro snímací hrot stav beztíže i silné gravitační pole, několikrát silnější než na Zemi.

Periodické – síla má v čase se opakující průběh. Průběh je dán určeným vzorem, který je definován periodou a amplitudou. Aby bylo dosaženo korektního průběhu periodického účinku je nutné omezit jeho frekvenci podle Shannonova–Nyquistova–Kotelnikova teorému a to tak, že maximální frekvence účinku bude nejvýše poloviční něž frekvence obnovování haptického obrazu.

Impuls – působení síly po velmi krátký časový okamžik. Lidské vnímání síly je intenzivnější při nesouvislém průběhu než při ustáleném průběhu, proto se doporučuje iniciovat a vytvářet impulsy síly co nejostřeji. Impuls by proto měl mít velkou sílu po kratší dobu než použít opačné nastavení, aby se dosáhlo většího dojmu.

Již bylo řečeno, že v rámci této práce se zabývám pouze pevnými tělesy a i toolkit OpenHaptics je standardně stavěn především pro haptickou interakci s pevnými tělesy. Při dotyku virtuálního tělesa se proto musí zamezit proniknutí dovnitř tělesa. Knihovny toolkitu využívají při zamezení proniknutí metody náhrady[2].

Metoda náhrady při proniknutí do tělesa spočívá ve vytváření náhradního bodu na povrchu tělesa, který zachovává se skutečnou pozicí snímacího hrotu minimální energii. Tímto způsobem se neustále počítá pozice náhrady na povrchu tělesa a generuje se silová reakce. Silová reakce se počítá jako účinek virtuální pružiny a tlumení směrem k poloze náhrady (obr. 3.2).



Obrázek 3.2: Znázornění metody „náhrady“ (Zdroj [10])

Knihovny toolkitu OpenHaptics je možné spojit s grafickým rozhraním. Sám výrobce, společnost SensAble Technologies, doporučuje a nenápadně programátory nutí k použití knihovny OpenGL [19]. Hlavně knihovna HLAPI (kap. 3.1.3), je strukturou svých příkazů a systémem použití knihovny velmi podobná struktuře právě OpenGL. Společnost SensAble takto ovšem postupuje, aby bylo použití toolkitu co nejvíce intuitivní a aby si programátoři pracující již v minulosti s nějakým grafickým systémem rychle osvojili práci i s haptickým zařízením. Díky OpenGL umožňuje OpenHaptics toolkit zapracovat haptické zařízení jednoduše i do již stavajících aplikací, čímž mohou značně a hlavně efektně rozšířit jejich možnosti.

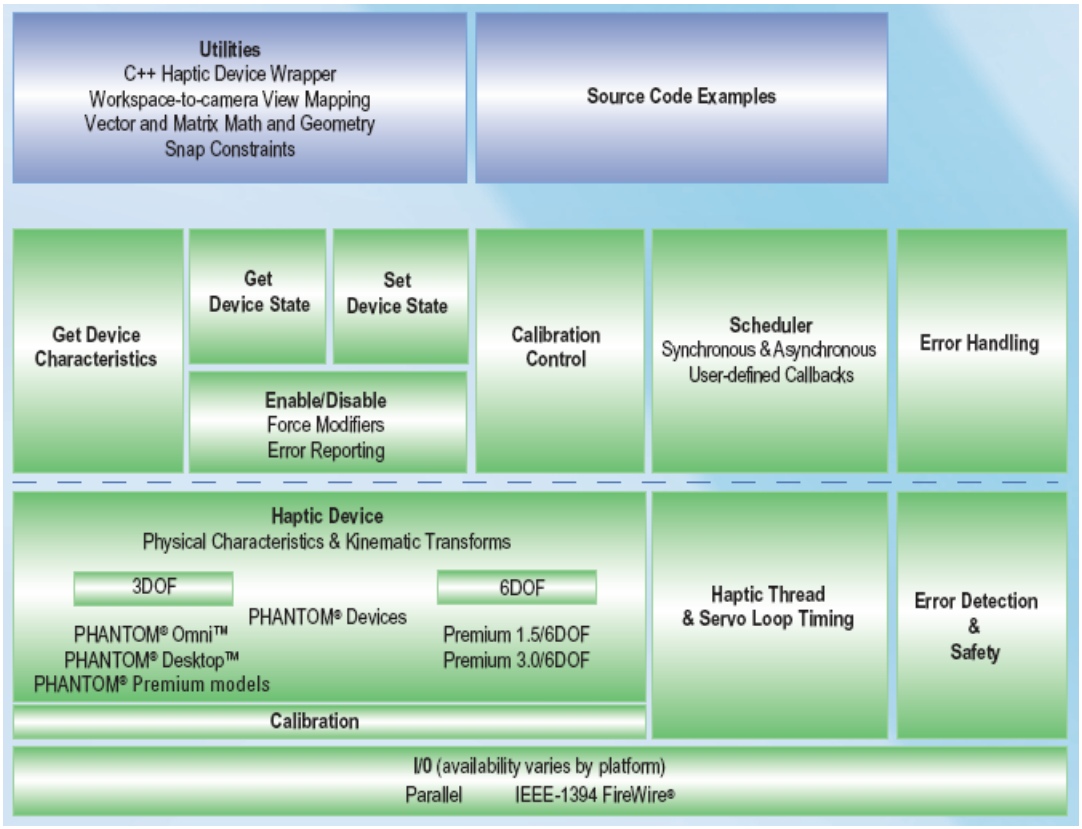
Součástí toolkitu jsou i ukázkové aplikace k jednotlivým knihovnám, které demonstrují základní použití a funkčnost částí knihoven. Z těchto příkladů jsem čerpal vědomosti o

[2] Angl. Proxy method – metoda náhrady, označuje se také jako SCP (Surface Contact Point).

správném použití funkcí. Ukázkových aplikací je velké množství a pokrývají téměř celou základní funkčnost knihoven HDAPI a HLAPI.

Přenositelnost toolkitu je teoreticky neomezená, neboť všechny části toolkitu jsou napsány v jazyce C/C++ a při vhodném nastavení vývojového prostředí je možné vyvíjet aplikace na jakékoli platformě. Tato vlastnost toolkitu a souvisejících knihoven je navíc umocněna právě použitím grafické knihovny OpenGL, která je také platformově nezávislá.

3.1.1 HDAPI



Obrázek 3.3: Struktura knihovny HDAPI (Zdroj [8])

Knihovna HDAPI (obr. 3.3, [10]) dovoluje tvůrci aplikace přistupovat k haptickému zařízení na nízké úrovni. Programátor může přímo určovat a generovat silovou odezvu, má přístup a kontrolu nad probíhajícím stavem zařízení a je schopen vytvořit vlastní obsluhy probíhajících stavů.

Hlavní dvě funkční části HDAPI jsou zařízení (Device) a plánovač (Scheduler). Zařízení dovoluje pracovat se všemi modely Phantom a vytvářet haptické aplikace. Tato část knihovny se stará o nastavení spojení se zařízením, nastavení parametrů a generaci silové odezvy. Architektura zařízení dovoluje spravovat více skenerů v rámci jednoho programu. Podle typu obsluhy je možné rozdělit funkce do těchto kategorií:

Inicializace zařízení Nastavení komunikace se skenerem, vytvoření spojení, nastavení síly a kalibrace zařízení.

Bezpečnost Funkce pro kontrolu silové odezvy nebo vnitřních mechanismů a stavů (např. příliš síly nebo teplota servo motorů).

Stav zařízení Funkce pro nastavování a získávání dat. Jedná se o správu stavů tlačítek, pozice snímacího hrotu, silové odezvy.

Plánovač umožňuje registrovat obslužné funkce, které volají příkazy v rámci vnitřního vlákna skeneru. Plánovač spravuje vnitřní vlákno s vysokou prioritou na velmi vysoké frekvenci, standardně 1000 Hz (kap. 3.1). Knihovna HDAPI dovoluje komunikaci s tímto vláknem a zabezpečuje konzistentnost dat.

Funkce knihovny HDAPI jsou na první pohled rozpoznatelné, jelikož mají názvy funkcí a parametrů začínají vždy sekvencí „hd“ resp. „HD“.

3.1.2 Struktura HDAPI aplikace

Na obr. 3.4 je znázorněna struktura obecné aplikace vytvořené s využitím knihovny HDAPI. Nicméně i tato struktura se dá ještě více generalizovat:

1. **Inicializace** – Vytvoří se a nastaví se spojení se zařízením. Provede se i nastavení možností zařízení, jako např. povolení generování sil a omezení jejich velikostí.
2. **Registrace** – Pro zpracování stavu zařízení a jeho interakci s prostorem a tělesy se registrují obslužné funkce, které získávají a nastavují atributy zařízení.
3. **Plánovač** – Po registraci se spustí plánovač, který bude volat obslužné funkce.
4. **Provádění** – Při běhu programu se periodicky volají z plánovače obslužné funkce a program provádí kýženou činnost.
5. **Ukončení** – Před ukončením aplikace se musí provést ukončení plánovače a uvolnění zařízení.

Inicializační fáze nastavuje spojení aplikace a haptického zařízení. Používají se tyto funkce:

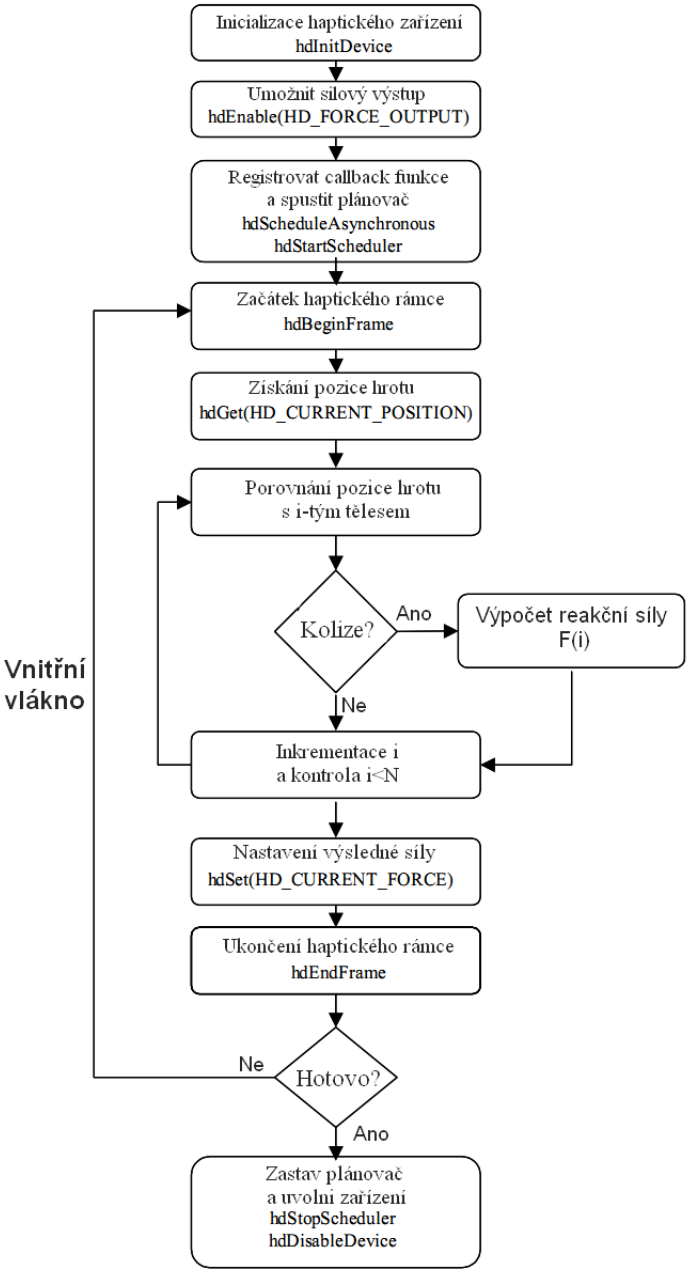
`hdInitDevice()` – Funkce nastaví spojení se zařízením, které je předáno jako parametr funkce. Obvykle se předává hodnota `HD_DEFAULT_DEVICE`. Návratová hodnota je handler pro zařízení.

`hdEnable()` – Zapnutí některých atributů HDAPI podle hodnoty předaného parametru. Zapíná se silová odezva (hodnota `HD_FORCE_OUTPUT`), omezení velikosti sil (`HD_FORCE_CLAMPING`) a další.

`hdStartScheduler()` – Zapnutí plánovače. Silová odezva se negeneruje dokud není spuštěn plánovač.

I v případě využití více haptických zařízení Phantom se spouští pouze jeden plánovač, i když musíme každé zařízení inicializovat samostatně.

Pokud bychom chtěli využít více zařízení je nutné při operaci specifikovat, pro které zařízení je operace určena pomocí příkazu `hdMakeCurrentDevice()` a předaného parametru, jímž je handler zařízení.



Obrázek 3.4: Obecné schéma HDAPI aplikace (Zdroj [10])

Jelikož je výstupem více vláknová aplikace je nutné dodržet konzistentnost dat. Abychom tak v aplikaci činili disponuje knihovna HDAPI haptickými rámci, což je úsek kódu ohraničen příkazy `hdBeginFrame()` a `hdEndFrame()`. Na začátku rámce je stav zařízení aktualizován a uložen, aby po dobu vykonávání rámce byla dostupná jednotná data. Na konci rámce jsou pak všechny změny zapsány do vnitřních dat stavu zařízení.

Obslužné funkce

Abychom mohli pomocí plánovače vykonávat operace a generovat silovou odezvu, je nutné registrovat obslužné (angl. callback) funkce. Prototyp takové funkce vypadá následovně:

```
HDCallbackCode HDCALLBACK FunctionName (void *userData);
```

Návratové hodnoty potom zároveň i určují, zda se funkce použije jednou nebo bude volána několikrát.

`HD_CALLBACK_DONE` – Funkce proběhne a již více se nevolá.

`HD_CALLBACK_CONTINUE` – Po provedení těla funkce se její volání znovu registruje v plánovači v jeho další smyčce.

Kromě návratové hodnoty je možné určit i typ obslužné funkce:

Synchronní volání – Po volání synchronní funkce aplikace čeká na dokončení celé funkce. Tento typ se využívá především pro získání současného stavu zařízení (pozice hrotu, velikost a směr současné síly, stav tlačítka).

Asynchronní volání – Při volání asynchronní funkce se předává vykonávání hlavní aplikaci, jakmile je funkce zaznamenána v plánovači. Nečeká se na její dokončení. Tento typ funkcí se používá hlavně pro správu haptického stavu zařízení (silové působení, které se aktualizuje v každé smyčce plánovače.)

Při registraci asynchronního volání se vrací handler, který následně slouží k případnému ovládání funkce. Pomocí handleru je možné asynchronní volání i ukončit.

Stav zařízení

Pro získání a nastavení parametrů zařízení se volají funkce ze skupin `hdGet*()` nebo `hdSet*()`. Názvy jednotlivých funkcí pokračují požadovaným typem proměnné, do které získáme nebo nastavíme hodnoty parametru. Několik ukázek, jak se tyto funkce používají a jaké parametry lze ovlivnit.

`hdGetDoublev(HD_CURRENT_POSITION, positionVec)` – získá se aktuální pozice snímacího hrotu do proměnné `positionVec` typu `double[3]`.

`hdGetString(HD_DEVICE_MODEL_TYPE, model)` – Získání názvu zařízení do proměnné `model` typu `string`.

`hdSetDoublev(HD_CURRENT_FORCE, forceVec)` – Nastavení generované síly přes `forceVec` typu `double[3]`, kde jsou nastaveny jednotlivé složky síly v osách X, Y a Z.

Správa chyb

V případě, že v průběhu aplikace dojde v zařízení k chybě, uloží se kód chyby na zásobník chyb, odkud se může chyba získat. Pokud nedošlo k chybě a zásobník je tedy prázdný, je při dotazu vrácena hodnota `HD_SUCCESS`.

Každá chyba je popsána:

- Kódem chyby.
- Vnitřním kódem chyby generovaným zařízením.

- Identifikátorem zařízení, které chybu nahlásilo.

Chyby se nemusí vyskytovat v průběhu aplikace vždy po provedení nějakého příkazu a není nutné ihned kontrolovat zásobník, např. chyba nastane během několikátého volání asynchronního volání funkce, ale je vhodné provádět kontrolu chyb v rozumné míře, nejlépe pomocí periodicky volané funkce.

V [10] je popsán univerzální postup kontroly chyb takto:

```
if(HD\_DEVICE\_ERROR(error = hdGetError()))
{
//zpracování chybového kódu error
}
```

Kalibrace

Kalibrace zařízení zajišťuje jeho schopnost snímat pracovní prostor určeným standardním způsobem a zamezuje například posunu pomyslného počátku souřadného systému nebo špatné interpretace snímaných úhlů.

Kalibraci jakéhokoli zařízení Phantom lze provést:

Hardwarový reset snímačů – Uživatel manuálně umístí zařízení do žádané polohy a spustí kalibrovací sekvenci. Nastavená kalibrace vydrží po celou dobu, kdy je zařízení připojeno do sítě. Kalibrace je vždy provedena do ortogonálního souřadného systému.

Kalibrace přes schránku – Schránka je otvor umístěný v podstavci zařízení a je možné provést kalibraci po vložení snímacího hrotu do schránky a spuštění kalibrační sekvence. Zařízení Phantom Omni uloženo ve schránce je vidět na obr. 2.10.

Automatická kalibrace – Zařízení používá vnitřní mechanismus a kontroluje kalibraci během práce se zařízením.

Více informací a podkladů je možno nalézt v [10]. Při vypracovávání této práce byl k dispozici model s automatickou kalibrací.

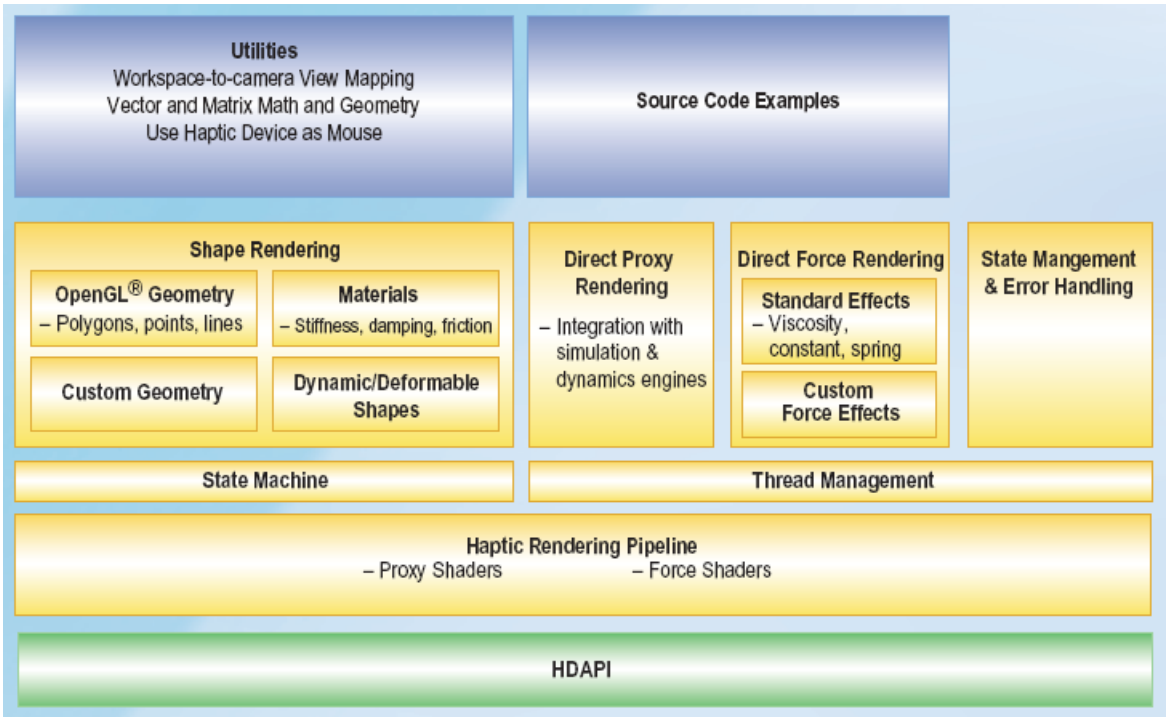
Ukončení aplikace

Před ukončením aplikace je nutné zastavit plánovač a uvolnit spojení se zařízením. Tato konečná fáze aplikace by měla mít následující průběh:

1. `hdStopScheduler()` – Ukončení běhu plánovače.
2. `hdUnschedule()` – Pokud byly použity asynchronní volání funkce v plánovači je nutné skrze získaný handler vymazat z plánovače. Funkci se předá právě handler volané funkce.
3. `hdDisableDevice()` – Ukončení spojení se zařízením.

3.1.3 HLAPI

Knihovna HLAPI (obr. 3.5, [10]) poskytuje vývojáři vyšší stupeň abstrakce než HDAPI. HLAPI využívá HDAPI, má velmi podobnou syntaxi příkazů jako OpenGL (názvy funkcí OpenGL začínají řetězcem „gl“ nebo u parametrů je to řetězec „GL“) a je velmi snadné



Obrázek 3.5: Struktura knihovny HLAPI (Zdroj [8])

rozpoznat názvy funkcí knihovny HLAPI (názvy začínají sekvencí „hl“ nebo „HL“). Navíc podobně jako OpenGL je HLAPI založena na systému stavového automatu, což znamená, že jednotlivé příkazy mění globální nastavení prostředí.

Knihovna HLAPI navíc umožňuje registrovat obslužné funkce jako reakce na nastalé události, jako např. stisk tlačítka, dotyk objektu a další.

Tato knihovna je navržena pro vytváření aplikací, které budou využívat zařízení Phantom, ale tvůrce je ušetřen nastavování některých atributů, například vzorců pro výpočet aktuální pružinové silové odezvy. HLAPI je také připravena spolupracovat s dalšími knihovnami, jako jsou grafické a fyzikální enginy[3] nebo enginy počítající kolize.

Jelikož haptické vykreslování virtuálních objektů vyžaduje podstatně rychlejší frekvence obnovování údajů než běžná grafická aplikace, haptický engine HLAPI vytvoří k hlavnímu vláknu aplikace další dvě vlákna: vnitřní (servo) vlákno a kolizní vlákno. Hlavní vlákno aplikace je potom v rámci [10] nazýváno klientským vláknem. V klientském vláknu se nastavují atributy pro práci s knihovnou a je z něj volána i většina příkazů. Většina jednoduchých haptických aplikací běží výhradně v rámci klientského vlákna a není potřeba se zbývajícími vlákny zaobírat. Pro pokročilejší aplikace je ale někdy nutné využít dalších vláken.

Servo vlákno zajišťuje komunikaci se zařízením. Získává pozici a orientaci snímacího hrotu, velikost, směr a další parametry generované síly na velmi vysoké frekvenci (obvykle 1000 Hz). Toto vlákno se podobá vnitřnímu vláknu, které využívá HDAPI (kap. 3.1.1), ale narozdíl od HDAPI je před programátorem skryto.

Kolizní vlákno počítá kolize objektů scény se snímacím hrotem zařízení. Toto vlákno běží

[3]Engine [14] je počítačový termín, který se překládá jako jádro programu nebo základní program, který je možné jednoduše rozšiřovat a následně využívat. Nejčastěji se hovoří o grafickém nebo fyzikálním enginu, ale existují i například enginy antivirů.

na nižší frekvenci než vnitřní servo vlákno, ale běží rychleji než klientské vlákno (obvykle 100 Hz). Vlákno zjišťuje, které objekty, vytvořené v klientském vlákně, se dotýkají s pozicí hrotu a vytvoří aproximaci objektů v okolí hrotu. Aproximace je předána vnitřnímu vláknu, které na základě tohoto odhadu okolí upraví generované síly. Díky tomuto přístupu je možné obnovovat vnitřní vlákno na velmi vysoké frekvenci i při vysokém počtu objektů ve scéně.

HLAPI je postavena tak, aby využívala OpenGL, je tedy možné přímo vytvářet aplikace pracující s virtuálním prostorem, tedy provádějící transformace objektů v závislosti na stavu haptického zařízení.

Podle [10] jsou tři způsoby jak pomocí HLAPI generovat silovou odezvu:

Tělesa – Programátor může definovat téměř libovolný tvar objektu a atributy jeho materiálu a knihovna automaticky generuje příslušné síly, aby se docílilo pocitu dotyku tohoto tělesa.

Silové efekty – Je dovoleno vytvářet globální silové efekty, které mohou působit v jakémkoli bodu prostoru.

Snímací hrot – Tvůrce aplikace může vytvářet vlastní generování síly při kontaktu s tělesy nebo vlastní detekci kolizí s objekty scény.

Jak již bylo popsáno v kapitole 3.1, knihovna využívá pro počítání pozice snímacího hrotu metody náhrady a při počítání pozice náhrady se tedy generuje i adekvátní silová odezva.

Při haptickém zpracovávání grafických těles ve scéně využívá knihovna HLAPI OpenGL příkazů pro vytváření a vykreslování rozličných geometrií. Díky tomuto přístupu je možné hapticky vykreslovat body, čáry, polygony určené pomocí `glBegin()` stejně jako obsahy display listů nebo předdefinované objekty. Zpracování geometrie tělesa se provádí uvnitř knihovny dvěma způsoby: pomocí depth bufferu nebo pomocí feedback bufferu.

Přístup depth buffer geometrií znamená, že jsou použity standardní OpenGL příkazy pro vytvoření geometrie v depth bufferu a HLAPI následně čte tento obraz a použije jej při generování silové odezvy.

Druhou možností je použít feedback buffer geometrie. Postup je podobný jako v případě depth bufferu, OpenGL příkazy vytvoří v bufferu obraz geometrie, knihovna následně podle tohoto obrazu generuje silovou odezvu (více v kap. 3.1.4).

3.1.4 Struktura HLAPI aplikace

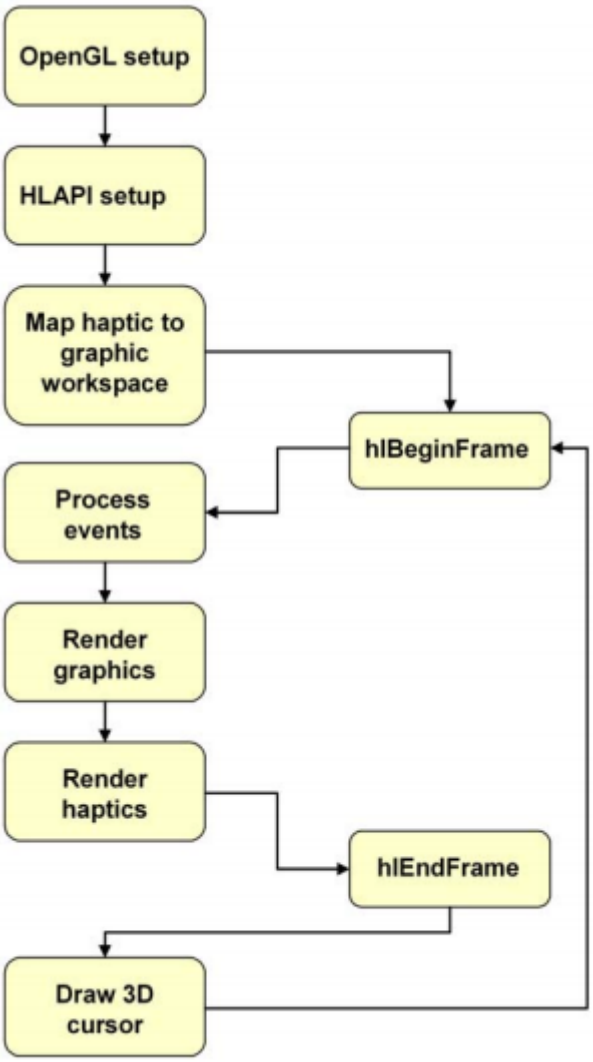
Na obr. 3.6 je nakreslena struktura typického programu vytvořeného pomocí knihovny HLAPI.

Prvním krokem je nastavení grafického systému. Obecně pro aplikace využívající OpenGL platí: inicializace knihovny GLUT [15] a nastavení OpenGL.

Následuje inicializace HLAPI, kdy se vytvoří spojení mezi haptickým skenerem a aplikací. Při této operaci se postupně inicializuje zařízení a vytvoří se haptický kontext, který obsahuje stav haptického vykreslování a slouží jako cílový objekt všech HLAPI příkazů. Používají se tyto příkazy:

`hdInitDevice()` – Přiřazení zařízení podle jeho jména. Pokud je v systému pouze jedno zařízení, většinou se předává parametr `HL_DEFAULT_DEVICE`. Funkce vrátí handler pro zařízení (stejné jako v kap. 3.1.2).

`hlCreateContext()` – Vytvoření kontextu pro předané zařízení.



Obrázek 3.6: Schéma jednoduchého programu vytvořeného v HLAPI (Zdroj: [10])

`hlMakeCurrent()` – Nastavení vytvořeného kontextu jako aktivního.

V dalším kroku se musí nastavit, jak se mají souřadnice snímané haptickým zařízením mapovat do virtuálního prostoru. Tento krok je důležitý, neboť na nastavení mapování prostoru závisí chování a spolupráce zařízení s grafickým prostorem. Je možné získat pomocí funkce `hlGetDoublev()` s parametrem `HL_WORKSPACE` pole skutečných mezí pracovního prostoru zařízení v milimetrech a následně tyto hodnoty použít při nastavení grafické projekce. Druhou variantou je obrácený postup, kdy se rozměr grafické projekce namapuje na pracovní prostor zařízení. Zde se využívá funkce `hluFitWorkspace()`, která pomocí předané projekční matice mapuje pracovní prostor.

Po skončení inicializační fáze přechází aplikace do smyčky, jenž pracuje se stavem zařízení a adekvátně se upravuje grafický výstup a silová odezva.

Jelikož má systém HLAPI více vláken, je nutné zachovat konzistenci dat podobně jako při vývoji aplikací s HDAPI (kap. 3.1.1 a 3.1.2). Aby se dodržela konstantnost dat během vy-

kreslování musí se všechny haptické vykreslovací funkce volat mezi příkazy `hlBeginFrame()` a `hlEndFrame()`. Takto se vytvoří haptický rámec, který umožňuje správnou vnitřní synchronizaci.

Funkce `hlEndFrame()` je haptickou obdobou funkce `glFlush()` z knihovny OpenGL a zajišťuje provedení všech změn najednou na konci rámce.

Standardně se v těle rámce nejdříve provede zpracování nastalých událostí (stisk tlačítka, dotyk objektu, pohyb, atd.). Následně se vykreslí všechny objekty scény graficky pomocí OpenGL. Pokud tvůrce aplikace nemá zvláštní úmysly, všechny grafické objekty se vykreslí i hapticky tak, aby bylo možné se jich dotýkat nebo s nimi jinak pracovat, ale aby grafická část odpovídala haptické.

Haptické vykreslení geometrických objektů se provádí pomocí příkazů `hlBeginShape()` a `hlEndShape()`, mezi kterými se volají klasické vykreslovací příkazy OpenGL, jenž určují geometrii tělesa.

Každý vykreslovaný tvar by měl mít svůj jedinečný identifikátor. Tento identifikátor se generuje pomocí funkce `hlGenShapes()` a na konci aplikace by se měl identifikátor uvolnit příkazem `hlDeleteShapes()`, aby mohl být opět využit. Získaný identifikátor se předává funkci `hlBeginShape()` pokaždé, když se tvar vykresluje společně typem tvaru:

`HL_SHAPE_DEPTH_BUFFER` – HLAPI používá k haptickému zpracování prostoru obrazy objektů vykreslený grafickými příkazy do OpenGL depth bufferu. Tento uložený obraz má nevýhodu, jelikož nedovoluje zaznamenávat body a čáry, mohou se vyskytnout problémy v haptické části při dotykovém modelu přitahování (kap. 3.1.5).

Důležitým prvkem při používání obrazu v depth bufferu je nastavení správné pozice kamery scény. Při základním nastavení jsou v bufferu uloženy pouze viditelné části objektů, což pro haptickou část znamená, že budete moci kolidovat pouze s určitými částmi objektů. HLAPI však obsahuje optimalizaci pohledu kamery, která je na začátku vypnutá a její zapnutí se provádí příkazem `hlEnable(HL_HAPTIC_CAMERA_VIEW)`. Po nastavení optimalizace HLAPI automaticky přizpůsobuje parametry pohledu podle pohybu hrotu a mapování prostoru zařízení.

Pracujete-li s depth bufferem je nutné korektně pracovat s resetováním a výměnou bufferů při vykreslování. Pokud bychom provedli klasický proces vykreslování scény: grafická část, haptická část a nakonec výměna bufferů, v OpenGL (příkaz `glSwapBuffers()`, vykresloval by se haptický pohled do okna aplikace a překrýval by skutečný grafický výstup. Správný postup tedy je: resetovat buffery (`glClear()`, grafické vykreslení, výměna bufferů, resetování depth bufferu, haptické vykreslení scény.

`HL_SHAPE_FEEDBACK_BUFFER` – Narozdíl od depth bufferu dovoluje feedback buffer zaznamenávat i body a čáry a hapticky je vykreslovat. Při použití feedback bufferu HLAPI automaticky alokuje prostor pro OpenGL feedback buffer a nastaví správný vykreslovací mód. Následně se uloží všechny objekty vzniklé grafickými příkazy a po ukončení haptického vykreslování příkazem `hlEndShape()` se získaný obraz použije pro generování silové odezvy.

Jak tedy vybrat správný přístup při haptickém vykreslování? Depth buffer je obecně výhodnější pro scény s vyšším počtem objektů a jeho využívání je efektivnější a alokuje méně paměti než feedback buffer. Depth buffer je však méně přesný, ale ve většině aplikací je rozdíl nepatrný. Tento jev je způsoben převodem geometrie v depth bufferu do 2D hloubkového obrazu.

Feedback buffer je určen především pro scény s malým počtem objektů. V takovém případě je jeho využití efektivnější a přesnější a navíc umožňuje vykreslovat body a čáry pro použití společně s modelem přitahování.

Po grafickém i haptickém vykreslení se zobrazuje 3D kurzor, což je grafické znázornění polohy snímacího hrotu nebo, v případě dotýkání se tělesa, polohy náhrady v prostoru.

Konec těla smyčky je určen voláním příkazu `hlEndFrame()`.

Ukončení celé aplikace může proběhnout až po provedení uvolnění všech navázaných proměnných a zařízení. Obecný průběh:

1. `hlDeleteContext()` – Smazání haptického kontextu. Předá se handler získaný při vytvoření.

2. hdDisableDevice() – Stejně v kapitole 3.1.2 ukončí se spojení se zařízením.

3.1.5 Další možnosti HLAPI

Dotykový model a vlastnosti materiálu

Aby byly možnosti interakce s objekty rozsáhlejší, nabízí knihovna HLAPI možnost nastavit dotykový model. Nastavit dotykový model je možné přes příkaz `hlTouchModel()`. Předané hodnoty parametrů následně určují způsob interakce:

`HL_CONTACT` – Při klasickém nastavení, kdy se nemění nastavení dotykového modelu počítá HLAPI kolize snímacího hrotu s objekty a generuje silovou odezvu, metoda náhrady (kap. 3.1). Je simulován dotyk s pevnými objekty. Toto nastavení je standardní a je účinné vždy od začátku inicializace.

`HL_CONSTRAINT` – Model přitahování[4] nastaví prostředí tak, že hrot je přitahován k tvaru objektu a pokud se nepůsobí příliš velkou silou, zůstává a pohybuje se v rámci definovaného tvaru. Navíc je k tvaru přitahován (chování podobné magnetickému poli) v závislosti na rozdílu vzdáleností pozice hrotu a povrchu tělesa.

Je samozřejmě možné nastavit vzdálenost působnosti modelu přitahování příkazem `hlTouchModelf(HL_SNAP_DISTANCE, distance)`, kde se za proměnnou `distance` dosadí hodnota vzdálenosti v milimetrech pracovního prostoru.

Právě v tomto dotykovém modelu je vhodné pracovat s body a čárami hapticky vykreslenými feedback buffer přístupem.

Podobně jako se mohou nastavit vizuální vlastnosti materiálu objektů, je možné nastavit i haptické vlastnosti materiálu. Vložením příkazu `hlMaterialf()` do programu, ovlivníme chování silové odezvy během dotyku s tělesem. Hodnoty parametrů funkce nastavují jednotlivé atributy materiálu:

Tuhost – parametr `HL_STIFFNESS` a hodnota nastaví tuhost materiálu. Tuhost značí, jak tvrdě se objekt jeví. Reálně se hodnota tuhosti použije při počítání síly při dotyku tělesa a dosadí se do Hookova zákona (vzorec 3.1) za konstantní tuhost pružiny. Vyšší hodnota tuhosti vyústí ve vyšší generovaný odpor.

[4] Je problematické přeložit angl. výraz „constraint model“ tak, aby opravdu vystihoval reálnou podstatu. Zvolil jsem překlad model přitahování, neboť podle mého názoru je výstižnější než doslovně model omezení.

Tlumení – parametr `HL_DAMPING`. Tlumení je složka materiálu, která reaguje na rychlost pohybu po tělese. Hodnota tlumení se použije ve vzorci 3.2 jako konstanta tlumení a reálně znamená, že čím prudčeji se budete snažit dotýkat, tím větší odpor získáte.

Tření – parametry `HL_STATIC_FRICTION` pro statické tření nebo `HL_DYNAMIC_FRICTION` pro dynamické tření. Tření způsobuje odpor při pohybu po povrchu objektu.

Statické tření je odpor, který cítíme při dotyku tělesa a počátečním posunu po povrchu.

Dynamické tření je generovaný odpor, který pociťujeme během pohybu po tělese.

V [10] se jako příklad udává led, který má vysokou hodnotu statického tření, při dotyku se led jeví jako lepivý povrch. Ale dynamická hodnota tření je nízká, protože při pohybu po povrchu už naše prsty jemně kloužou.

Propadnutí – parametr `HL_POPTHROUGH`. Hodnota propadnutí určuje jak velkou silou se musí působit na povrch tělesa než se ocitneme snímacím hrotem na druhé straně povrchu tělesa. Tento atribut materiálu se používá při dotykovém modelu, kdy nám dovoluje protlačit se objektem a dotýkat se jej zevnitř.

Všechny parametry je možné nastavit na reálnou hodnotu v intervalu $\langle 0; 1 \rangle$.

Silové efekty

I přesto, že nelze pomocí HLAPI přímo nastavit silové působení, existuje způsob, jak generovat sílu, přes silové efekty. Silové efekty slouží pro vytváření rozličných haptických vjemů, jenž mohou simulovat gravitaci, vibraci a další průběhy síly. Díky efektům se mohou generovat síly konstantní, tření a pružina.

Pro každý vytvářený efekt je nutné vytvořit unikátní identifikátor vygenerovaný funkcí `hlGenEffects()` a po ukončení využívání se musí proměnná uvolnit funkcí `hlDeleteEffects()`.

Silové efekty mohou být trvalé nebo budou působit po přednastavenou dobu. Působnost se nastavuje příkazy `hlStartEffect()` pro počátek a `hlStopEffect()` pro ukončení působení identifikátorem předaného efektu. Dočasné efekty se spouští příkazem `hlTriggerEffect()` a nevyžadují vytvoření identifikátoru.

Aby byl programátor schopen vytvářet širokou škálu efektů, nastavují se funkcí `hlEffect*()` vlastnosti efektů:

- **Velikost** – Hodnota působící síly.
- **Směr** – Určení směru působení konstantní síly.
- **Přírůstek** – Při generování se síla postupně zesiluje.
- **Poloha** – Simuluje-li se pružina, takto se určí pevný bod (bod uchycení pružiny).
- **Trvání** – Pro dočasné síly se určí čas působení v milisekundách.

Všechny příkazy pro nastavení parametrů sil a jejich začátek nebo konec působení se musí objevit v těle rámce, tzn. mezi příkazy `hlBeginFrame()` a `hlEndFrame()`, aby byla práce s efekty korektní a nedocházelo k problémům.

Nastavení silových efektů je možné i v průběhu měnit. Provede se, opět v těle rámce, nové nastavení parametrů a následně se zavolá funkce `hlUpdateEffect()` s příslušným identifikátorem efektu.

Události

I v aplikacích využívající knihovnu HLAPI se vyvolávají při některých operacích události, které je možné kontrolovat a obsluhovat pomocí registrovaných funkcí. Obslužná funkce musí mít následující prototyp:

`void HLCALLBACK CallbackFunctionName()`

Funkce má i povinné parametry (v jejich pořadí i s datovým typem):

`HLenum event` – Hodnota určující událost, na kterou se bude reagovat

`HLuint object` – Může se definovat pro který objekt se bude funkce volat.

`HLenum thread` – Specifikuje se vlákno, které bude volat funkci.

`HLcache *cache` – Je možné získat stav zařízení v okamžiku události. Předaný ukazatel je nutno použít s funkcí `hlCacheGet*()`.

`void *data` – Mohou se předávat i uživatelská data.

Takto definovaná funkce se následně registruje předáním ukazatele příkazu `hlAddEventCallback()` s těmito parametry:

`HL_EVENT_*` – Hodnota obsluhované události.

`MOTION` nastává při změně polohy snímacího hrotu, tedy při pohybu.

`TOUCH / UNTOUCH` je vyvolán při kontaktu nebo opuštění povrchu objektu. Událost dotyku nastává pouze při prvotním kontaktu. Pokud se budeme po povrchu objektu pohybovat, událost se již volat nebude, až do okamžiku opuštění povrchu, kdy nastane druhá událost.

`*BUTTONDOWN / *BUTTONUP` jsou události na stisknutí nebo uvolnění tlačítka. Zařízení Phantom Omni (kap. 2.5.2) disponuje dvěma tlačítky, některé modely však mají ještě třetí tlačítko, tzv. „safety button“.

`HL_OBJECT_ANY` – Definování objektu, na němž nastane událost. Většinou se předá právě tento parametr a reakce probíhá na všech tělesech, ale je možné předat identifikátor tvaru objektu (kap. 3.1.4), který určí specifický zdroj události.

`HL_CLIENT_THREAD` – Určení vlákna pro vyvolání obslužné funkce. Tento parametr je nejčastější a předává správu obsluhy klientskému vláknu, jelikož běží na menší frekvenci a zabírá méně výkonu. Kontrola stavu událostí se provede příkazem `hlCheckEvents()` a pokud některá událost nastala, zavolá se obslužná funkce.

I při provádění obslužných funkcí je třeba správně umístit jejich kontrolu, neboť pokud by obslužné funkce prováděli operace s tvary objektů (kap. 3.1.4) nebo úpravu silových efektů (tato kapitola výše) musí být v těle rámce, čili mezi příkazy `hlBeginFrame()` a `hlEndFrame()` (kap. 3.1.4).

Jestliže je potřeba opravdu okamžitá reakce na událost, aby se zamezilo nepřesnostem při haptickém působení, předá se parametr `HL_COLLISION_THREAD` a obsluha se bude vyvolávat právě z kolizního vlákna. Tento postup je vhodné použít v situaci, kdy chceme nastavit model přitahování vzhledem k pozici hrotu. Kdyby byla reakce lehce zpožděna, mohlo by se projevit malé uskočení nebo trhnutí k pozici, ve které byla událost vyvolána.

`&CallbackFunctionName` – Ukazatel na obslužnou funkci.

`NULL` – Poslední parametr slouží pro případná uživatelská data, tudíž se předá buď hodnota NULL nebo data přetypovaná na `void*`.

3.2 Demonstrační aplikace

Pro ukázku využití haptického zařízení Phantom Omni (kap. 2.5.2) a funkčnosti knihoven OpenHaptics toolkitu jsem vytvořil několik demonstračních aplikací.

Aplikace používají grafickou knihovnu OpenGL [19] pro vizualizaci haptických účinků v rámci scény. Knihovna OpenGL byla zvolena z důvodů provázanosti s knihovnami HDAPI a HLAPI (kap. 3.1.1 a 3.1.3).

Pro vytvoření okna aplikace využívám knihovnu GLUT [15], která je nadstavbou knihovnou OpenGL. Nastavení knihovny GLUT proběhne registrací obslužných funkcí pro překreslení obsahu okna, nastavení parametrů projekce a velikosti okna, reakce na stisk kláves.

Po nastavení knihovny GLUT se provede inicializační fáze pro OpenGL a haptickou knihovnu (HDAPI nebo HLAPI) a spustí se hlavní smyčka programu knihovny GLUT příkazem `glutMainLoop()`.

Uvolnění alokovaných zdrojů (zařízení, plánovač, funkce) proběhne vždy před ukončením aplikace v těle funkce registrované příkazem `atexit()`.

3.2.1 Aplikace Force field

První demonstrační aplikace, kterou jsem vytvořil využívá knihovny HDAPI (kap. 3.1.1). Aplikace simuluje silová pole několika bodů a jelikož silová pole jsou trvalé zdroje je vhodné použít plánovač a obslužné funkce plánovače pro generování stálé síly (kap. 3.1.2).

Tento příklad využití HDAPI je často používán a opravdu nejlépe demonstruje vlastnosti a funkčnost knihovny. V souboru ukázkových příkladů je velmi jednoduchá aplikace „Coulomb Field“, která demonstruje interakci dvou nábojů a působení elektrické síly Coulombova zákonu [12]:

$$F_e = k\frac{|Q_1 Q_2|}{r^2} \tag{3.5}$$

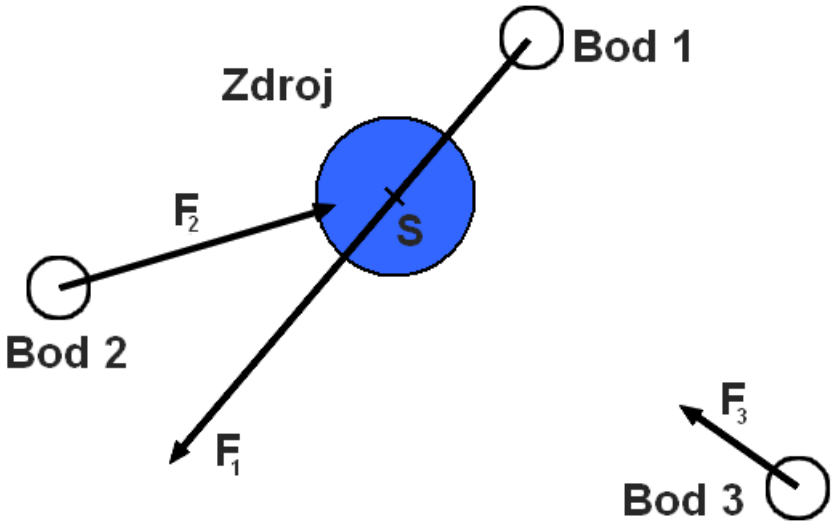
Za předpokladu, že velikosti obou nábojů Q_1 a Q_2 jsou neměnné, pak velikost elektrické síly je nepřímo úměrná druhé mocnině vzdálenosti bodů r.

Tuto aplikaci jsem použil jako zdroj inspirace a příkladu použití HDAPI. Vytvořená aplikace Force Field má rozšířenou funkčnost, aby lépe demonstrovala generování síly.

V případě ukázkové aplikace Coulomb Field se projevilo, že použití fyzikálních vzorců přesně podle definice je problematické, neboť generovaná síla má kvadratický průběh, což v praxi znamená, že působení skutečně citelné síly se projeví až při malé vzdálenosti obou bodů. Po diskuzi s Doc. Krškem jsme usoudili, že bude vhodné upravit vzorec síly tak, aby generovaná síla měla lineární průběh. Vzorec vytvářené síly:

$$F = \frac{C}{r} \tag{3.6}$$

Hodnota C je konstantní hodnota, vztaženo na Coulombův zákon jedná se o součin konstanty úměrnosti k a hodnot nábojů Q_1 a Q_2, a r je vzdálenost bodu působení od zdroje.



Obrázek 3.7: Znázornění působení silového pole zdroje ve třech různých bodech

Na obr. 3.7 je znázorněno několik poloh bodů a působení síly tvořené zdrojem silového pole. Jelikož zdroj působí přitažlivou silou má síla vždy směr do středu zdroje a velikost je ovlivněna vzdáleností bodu od zdroje, proto:

$$F_1 > F_2 > F_3 \tag{3.7}$$

Rozšířením oproti ukázkové aplikaci je simulace silového pole několika zdrojů, které mají i protichůdné vlastnosti, bod přitahují nebo odpuzují. Odpudivá síla má v daném bodě stejnou velikost, ale její směr je opačný. Pro obr. 3.7 by znázornění odpudivé síly leželo na stejné přímce jako nakreslené síly, ale směřovaly by od středu zdroje.

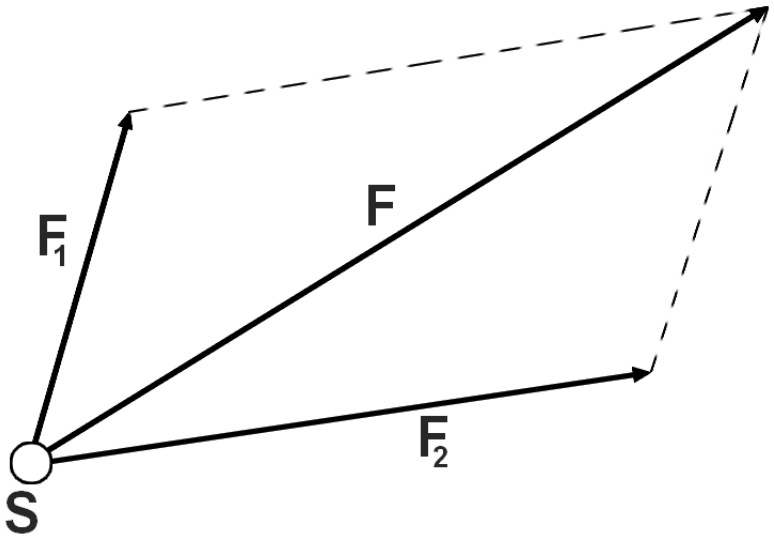
Zdroje silových polí budou mít pevnou pozici v prostoru a budou působit na objekt virtuálně umístěný na souřadnicích snímacího hrotu. Změnou polohy hrotu se bude měnit velikost i směr síly generované na objekt zdrojem pole podle vzorce 3.6. Objekt vázaný na polohu snímacího hrotu bude posunován (kap. 2.3.1) podle získaných souřadnic snímacího hrotu.

Při simulaci více zdrojů je nutné počítat skládání sil. Skládání sil je podle [12] proces, kdy nahradíme působící síly silou jedinou, jenž má na těleso stejný pohybový účinek jako síly, které skládáme. Skládáním sil F_1 a F_2 vznikne výslednice sil F (obr. 3.8).

Počítání s vektory sil bude prováděno pomocí objektů třídy `hduVector3Dd` z knihovny `hduVector.h` ze souboru pomocných knihoven toolkitu OpenHaptics (kap. 3.1).

Pro účely generování síly bodou registrovány dvě obslužné funkce plánovače (kap. 3.1.2). První synchronní volání bude zajišťovat získání současné polohy snímacího hrotu a velikost generované síly pro vykreslení pohyblivého objektu.

Druhá funkce bude asynchronně volána a bude provádět simulaci silového pole. Bude získávat současné souřadnice snímacího hrotu, z těchto hodnot počítat vektor výsledné síly a tuto sílu předávat do stavu haptického zařízení.



Obrázek 3.8: Znázornění skládání sil

3.2.2 Aplikace Simple haptics

Tento ukázkový program využívá knihovny HLAPI (kap. 3.1.3) a demonstruje základní použití knihovny pro vytvoření aplikace, jenž umožňuje základní interakci těles scény a 3D kurzoru haptického zařízení.

Struktura aplikace je shodná s popisem v kapitole 3.1.4.

Pro demonstraci základních možností HLAPI aplikace je schopna vykreslit objekty v různých dotykových modelech (kap. 3.1.5): dotykový nebo model přitahování a uživatel může i prakticky vyzkoušet rozdíly mezi tvary vykreslenými z depth bufferu nebo feedback bufferu. Uživatel má právo v každém okamžiku nastavení měnit a je informován o současném stavu.

Ovládání dovolí uživateli vykreslit i několik trojrozměrných objektů a každý objekt disponuje jinými vlastnostmi haptického materiálu tak, aby uživatel pocítil rozdílnosti jednotlivých atributů: tuhosti, tlumení a statického a dynamického tření.

3.2.3 Aplikace Cubes

Třetí ukázková aplikace Cubes provádí simulaci gravitačního pole. Pro haptickou obsluhu zařízení Phantom byla vybrána knihovna HLAPI (kap. 3.1.3) a především možnosti vytvořit silový efekt a získávat informace ze scény skrze obsluhy událostí vyvolané interakcí s objekty scény. Výsledkem působení na tělesa je jejich změna polohy ve scéně. Aplikace předpokládá simulaci dokonale pružného prostředí, kde neprobíhá tlumení sil vlivem prostředí.

Aplikace vykreslí místnost s objekty, ve které působí gravitační (tíhové) pole a tedy i tíhová síla [12]:

$$F_G = m \cdot g \tag{3.8}$$

Tíhová síla F_G je součinem hmotnosti tělesa m a tíhového zrychlení g.

Každé těleso má určenou hmotnost a váha objektu je znázorněna i jeho velikostí a barvou. Pro tvar těles jsem zvolil krychli, protože se s ní dobře pracuje v rámci počítání

polohy a kolizí, stejně jako s koulí, ale narozdíl od koule má lepší vlastnosti pro haptickou interakci s kurzorem zařízení.

Uživatel má možnost působit na objekt pomocí kurzoru haptického zařízení. Při vyvolání silového působení se vypočítává zrychlení udané tělesu silou podle Newtonova druhého pohybového zákona (vzorec 3.4, [12])

$$a = \frac{F}{m} \tag{3.9}$$

Tento jev je znázorněn na obr. 3.9. Na těleso je působeno silou, díky které vyvstává zrychlení a těleso se pohybuje po dráze s. Aplikace počítá dráhu jako posunutí objektu během krátkého časového úseku Δt, kdy lze působící sílu považovat za konstantní. V takovém případě lze za konstantní považovat i zrychlení a z něj vyplívající rychlost. Podobně jako v kapitole 3.2.1 se oprostím od přesných fyzikálních definic a pro výraznější interakci dosadím do vzorce pro výpočet okamžité rychlosti

$$v = a \cdot t \tag{3.10}$$

za hodnotu času t konstantu 1. Díky této úpravě je potom v aplikaci při počítání během velmi malého časového okamžiku Δt použit vzorec

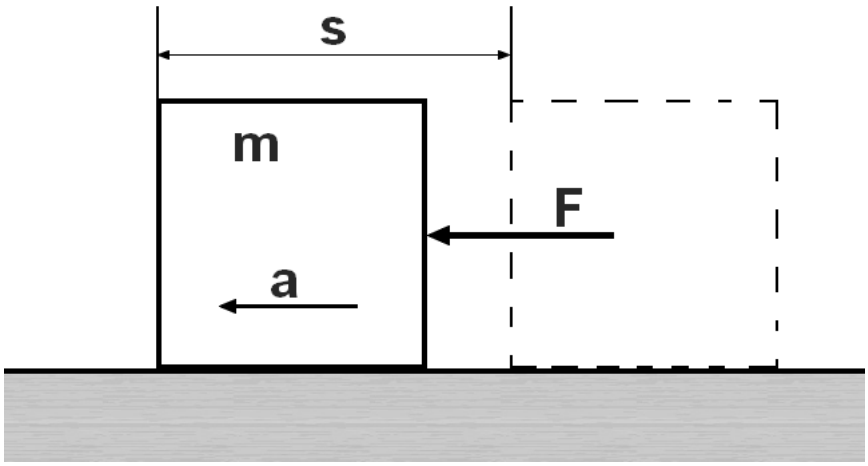
$$v = a \tag{3.11}$$

Výsledné posunutí tělesa (kap. 2.3.1 se získá ze vzorce pro výpočet dráhy při přímočarém rovnoměrném pohybu

$$s = v \cdot \Delta t \tag{3.12}$$

dosazením ze vzorce 3.11 proto získáme posunutí Δs během Δt jako

$$\Delta s = a \cdot \Delta t \tag{3.13}$$



Obrázek 3.9: Posunuti tělesa při působení silou F, která vytvoří zrychlení a

Jak vyplývá z předešlých odstavců pro každý objekt ukládám údaje o jeho rozměrech, aktuální pozici a rychlost. Z důvodů jednoduchého uložení polohy se u každého tělesa ukládá

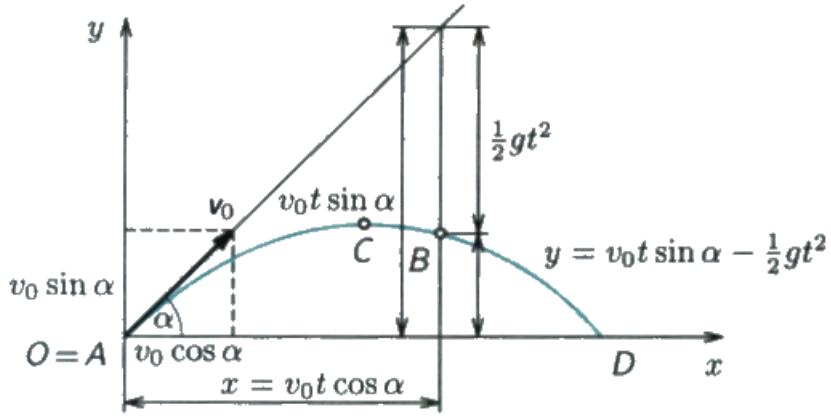
transformační matice, která obsahuje posunutí od souřadného počátku (kap. 2.1 a 2.3), což usnadní práci s objektem při grafickém i haptickém vykreslování. Rychlost je při každém tělese uložena jako vektor.

Kromě možnosti působit na těleso na podložce, může uživatel vybrané těleso zvednout, hodit jej a ovlivňovat trajektorii pádu tlačením na těleso stejně jako na podložce.

Zvednout těleso je možné stisknutím tlačítka haptického zařízení Phantom Omni při kontaktu objektu a 3D kurzoru. Těleso se připoutá ke kurzoru a zároveň se generuje silový efekt tíhové síly (vzorec 3.8) závisející na hmotnosti objektu.

Pokud uvolníme tlačítko zařízení zaznamená se rychlost udělená tělesu a těleso začne konat pád, přesněji fyzikální jev zvaný vrh šikmý.

Vrh šikmý popsaný v [12] je pohyb tělesa v gravitačním poli, kdy je tělesu udělena počáteční rychlost v_0, která svírá s horizontální rovinou úhel α. Podle obr. 3.10 těleso koná pohyb ve směru horizontální osy a zároveň koná pohyb ve směru vertikální osy, který je ovšem ovlivněn gravitačním polem a v každém bodě trajektorie vrhu se od vertikální složky odečítá dráha vykonaná při pádu tělesa.



Obrázek 3.10: Vrh šikmý vzhůru (Zdroj [12])

Aplikace ovšem pracuje s objekty v trojrozměrném prostoru. Získaný vektor rychlosti při uvolnění tlačítka má proto tři složky ve směrech souřadných os. Jelikož známe přímo jednotlivé složky rychlosti můžeme vzorce pro výpočet polohy ve vrhu uvedené v [12] upravit a rozšířit. V pracovním prostoru je horizontální rovina XZ a proto bude volným pádem ovlivněna pouze vertikální složka polohy Y. Jednotlivé souřadnice bodu v čase t (označení v_{0*} znamená složku rychlosti pro odpovídající souřadnou osu):

$$x = v_{0x} \cdot t \tag{3.14}$$

$$z = v_{0z} \cdot t \tag{3.15}$$

$$y = v_{0y} \cdot t - \frac{1}{2} g \cdot t^2 \tag{3.16}$$

Z těchto vzorců pro výpočet polohy je snadné vytvořit posunutí během časového intervalu Δt, pouhou záměnou t za Δt.

Při silovém působení na těleso během vrhu je podle vzorce 3.9 a následné úpravy 3.11 se získaný vektor rychlosti přičte k současnému vektoru rychlosti udělenému při hodu tělesa a takto se ovlivní trajektorie vrhu.

Pro lepší identifikaci těles se pro každý objekt vytvoří jedinečný identifikátor tvaru, který se následně předá obslužným funkcím pro události dotyku a stisknutí tlačítka (kap. 3.1.5). Takto je ihned při výskytu události známo ovlivňované těleso a mohou se na základě fyzikálních vzorců počítat transformace aplikované na objekt scény (kap. 2.3).

Abych zamezil zmizení těles během pohybů ve virtuální místnosti, aplikace využívá jednoduchý kolizní systém, kdy se při pohybu kontroluje poloha tělesa vzhledem k rovinám určenými stěnami místnosti. Kontrola se provádí mezi příslušnými souřadnicemi tělesa a konstantními hodnotami určující omezovací rovinu v dané souřadné ose. Pro tento přístup jsem se rozhodl, když jsem studoval detekci kolizí na [2].

Při detekci kolize se následně ovlivní další posunutí tělesa. Při vrhu, jestliže nastane kolize se stěnou, rychlost v daném směru souřadné osy je obrácena a pokračuje výpočet posunutí až do okamžiku dopadnutí na podložku, tedy kolize se spodní omezovací rovinou vertikální osy. Kolize se také počítají při zvednutí (tažení) tělesa skrze haptické zařízení. Pokud nastane kolize během této činnosti je do hodnoty posunutí dosazena maximální hodnota posunutí v daném směru, čímž se zamezuje, aby těleso opustilo místnost.

Kapitola 4

Implementace aplikací

Vytvořené demonstrační aplikace jsou napsány v jazyce C/C++ ve vývojovém prostředí Microsoft Visual Studio 2005. Grafické rozhraní aplikací zajišťují knihovny OpenGL [19] a GLUT [15].

Pro integraci haptického zařízení byl použit SensAble OpenHaptics toolkit verze 3.0 pro operační systém MS Windows.

Aplikace byly odladěny na počítači s operačním systémem Windows XP SP3 s připojeným haptickým zařízením SensAble Phantom Omni (kap. 2.5.2).

Všechny aplikace byly implementovány podle návrhu uvedeném v kapitole 3.2.

Nastavení grafické knihovny OpenGL a GLUT bylo naprogramováno podle zvyklostí a zkušeností získaných během studia na FIT VUT v Brně. Bližší detaily implementace je možné získat přímo ze zdrojových souborů aplikací nebo z dokumentace na přiloženém datovém médiu.

4.1 Demonstrační aplikace Force field

Aplikace používá schéma jednotlivých fází haptického programu popsané v kapitole 3.1.1 a fyzikálních vzorců podle 3.2.1. Během inicializační fáze je vytvořen vektor zdrojů. Každá položka vektoru obsahuje informace o umístění zdroje a hodnotu určující odpudivý nebo přitažlivý účinek. Je také registrována asynchronní obslužná funkce `ForceField()`, která nastavuje generovanou sílu vypočítanou funkcí `ForceComputeFunc()`.

Mapování prostoru se provede získáním reálných hodnot rozměrů pracovního prostoru zařízení Phantom a předáno funkci `glFrustum`, která nastaví projekční matici.

Funkce zobrazující grafický obsah okna `DisplayFunc()` vykreslí zdroje síly, objekt navázaný na polohu snímacího hrotu je vykreslen na souřadnice získané synchronním voláním obslužné funkce `DeviceStateFunc()` a jako poslední je vykreslena šipka znázorňující směr a velikost výsledné síly. Zdroje i pohyblivý objekt jsou vykresleny jako koule a jejich funkce jsou rozlišeny barvou:

- **Žlutá** – pohyblivý objekt reprezentující polohu hrotu zařízení.
- **Zelená** – přitahující zdroje.
- **Červená** – odpuzující zdroje.

Pro vykreslení šipky síly byla převzata a upravena funkce `drawForceVector()` z ukázkových příkladů.

Vypočítání výsledné síly ve funkci `ForceComputeFunc()` se provádí jako součet jednotlivých působících sil v objektu třídy `hduVector3Dd`. Z důvodů přílišného nepřetěžování zařízení se při přiblížení pohyblivého objektu ke zdroji na vzdálenost menší než dvojnásobek poloměru koulí opustí skládání sil a generuje se síla přímo závislá pouze na poloze těchto dvou objektů.

4.2 Demonstrační aplikace Simple haptics

Jelikož aplikace demonstruje základní funkčnost knihovny HLAPI, využívá se schéma aplikace popsané v kapitole 3.1.3, což bylo zmíněno i návrhu (kap. 3.2.2).

Pro mapování pracovního prostoru je použita funkce `hluFitWorkspace()`, které se předá získaná projekční matice.

Ovládání pomocí klávesnice nastavuje několik logických a číselných přepínačů, které následně ovlivní vykreslený obsah a haptické nastavení. Uživatel může nastavit dotykový model a buffer, ze kterého se bude číst haptický obraz a přepínat mezi 4 tělesy s různými grafickými i haptickými vlastnostmi.

Přepínače následně ve funkci `DrawHaptics()` určí haptické vlastnosti těles, model dotyku a buffer pro vykreslení tvaru. Ve funkci `DrawObjects()` přepínače určí vykreslovaný objekt.

Pro vykreslení nápovědy v okně aplikace a kurzoru haptického zařízení byly z ukázkových příkladů a dokumentace [10] převzaty a upraveny funkce `drawPrompts()` resp. `DrawCursor()`.

4.3 Demonstrační aplikace Cubes

Opět byla tato aplikace naprogramována podle struktury popsané v kapitolách 3.1.3 a 3.1.4. Znázorňované fyzikální výpočty a funkčnost jsou popsány v kapitole návrhu 3.2.3.

Během inicializační fáze je, podobně jako u aplikace Force field (kap. 4.1), vytvořen vektor pohyblivých objektů scény. Jednotlivé položky tělesa obsahují popis jeho tvaru, jeho virtuální hmotnost, transformační matice objektu a několik dalších proměnných určených především pro počítání polohy během vrhu (např. čas a aktuální rychlost tělesa).

Pro každý objekt jsou registrovány i obslužné funkce událostí dotyku tělesa a stisknutí tlačítka na jejich povrchu, které při výskytu události předají index tělesa.

Mapování prostoru je stejně jako v případě aplikace Simple haptics (kap. 4.2) uskutečněno funkcí `hluFitWorkspace()`. V této fázi se uloží i rozměry vykreslované místnosti, které v dalším průběhu slouží k detekci kolizí.

Funkce zobrazující obsah okna postupuje takto:

1. Kontrola událostí a podle stavu zařízení provedení úprav poloh těles.
2. Vykreslení místnosti, graficky i hapticky.
3. Grafické i haptické vykreslení těles na jejich současných souřadnicích.
4. Vykreslení haptického kurzoru.
5. Vykreslení nápovědy.

Kurzor i nápověda jsou vykresleny jako u aplikace Simple haptics (kap. 4.2).

Posouvání těles je prováděno funkcí `PushObject()`, která mění polohu tělesa o indexu získaným během reakce na událost dotyku.

Změna pozice tělesa, které je taženo kurzorem při stisknutém tlačítku, je zajištěna funkcí `updateDragObjTransform()`.

A v poslední řadě jsou souřadnice těles editovány během pádu, který simuluje funkce `UpdateObjectPosition()`, jenž je průběžně volána na objekty neležící na podložce (podlaze vykreslované místnosti) v těle funkce `IdleFunc()`.

Pro změny polohy se používají metody třídy `hduMatrix`, které tvoří transformace tak, jak byly popsány v kapitole 2.3.

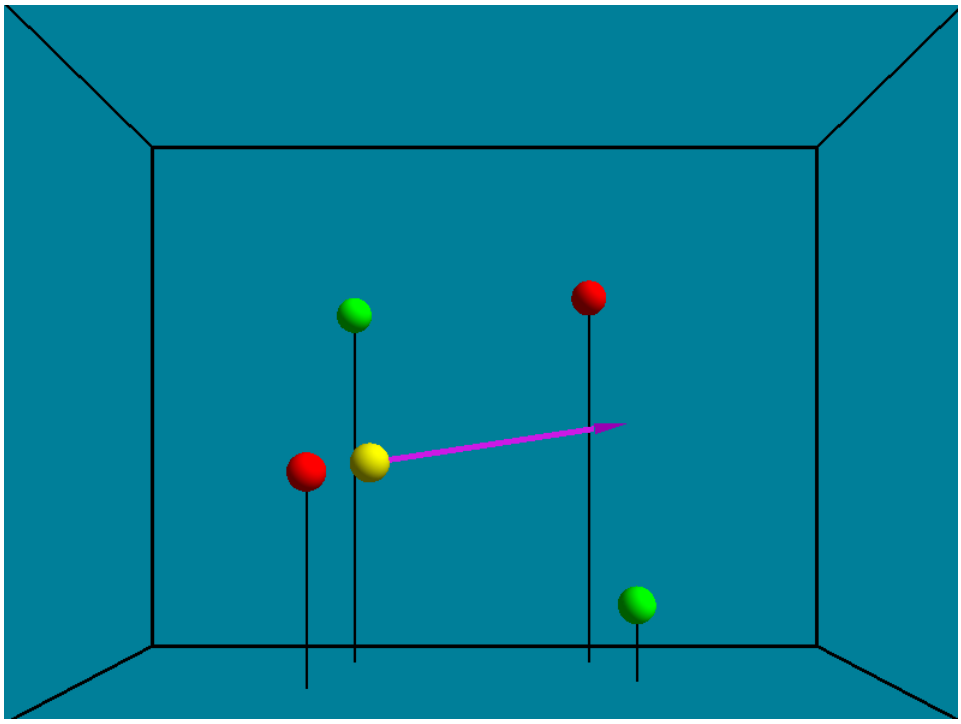
Pro zlepšení vizuální prezentace a přehlednosti byl do aplikace podle [3] přidán pro vykreslování těles Phongův osvětlovací model formou vertex a fragment shaderu. Pro přidání shaderů byla použita knihovna GLEW, také zmíněná na [3].

Nastavení knihovny GLEW se provede voláním funkce `glewInit()` a navázání shaderů se následně provede pomocí převzaté funkce `SetShaders()`, také z [3].

Kapitola 5

Výsledky

5.1 Force Field



Obrázek 5.1: Prostředí aplikace Force field

Aplikace Force field má jednoduché grafické prostředí (obr. 5.1), které je ovšem plně využité díky přítomnosti haptické interakce.

Navržený program pomocí funkcí HDAPI, a především využitím plánovače (kap. 3.1.1), generuje kontinuální silovou odezvu působící na pohyblivé těleso. Pohyb tělesa je svázán s aktuální polohou snímacího hrotu. Použití lineárního průběhu generované síly působí velmi názorně a bez potíží dovoluje pohrávat si interakcí ve vytvořeném silovém poli.

Díky použití vektoru jako datové struktury pro uložení zdrojů pole je možné minimální úpravou kódu změnit počet zdrojů, jejich polohu, silový účinek a tím ovlivnit průběh výsledné síly.

Velikost a hlavně směr výsledné síly jsou zobrazovány velmi názorně šipkou. Jelikož je zobrazován vektor aktuální síly, nastává v situaci, kdy se pohyblivé těleso ocitne v prostoru přitažlivého zdroje, postupné zmenšování síly, jež je dáno výpočtem síly v blízkosti zdroje (kap. 4.1). Síla je v tomto prostoru počítána přímo z aktuálního rozdílu poloh a jelikož je pohyblivé těleso přitahováno ke středu, síla se zmenšuje až dochází k ustálení v souřadnicích zdroje.

U přitahujících zdrojů silového pole je největší síla na rozhraní dotyku pohyblivého tělesa a zdroje. Narozdíl od odpuzujících zdrojů, kde se největší síla generuje při snaze proniknout tělesem do zdroje.

Právě na rozhraní dotyku tělesa a zdrojů se objevuje silný akustický jev, který vydává haptické zařízení. Jelikož zařízení v průběhu jevu nehlásí žádné chybné stavy, zřejmě se jedná o reakci na probíhající omezení výsledné síly.

5.2 Simple haptics

Obrázek 5.2: Prostředí aplikace Simple haptics

Základní funkčnost knihovny HLAPI demonstruje aplikace Simple haptics (kap. 3.1.3 a 3.1.4). Grafické prostředí je zobrazeno na obr. 5.2. Zobrazená nápověda udržuje uživatele v každém okamžiku informovaného o současném nastavení haptické knihovny a ovládacích klávesách, které tento stav změní.

Uživatel může vyzkoušet pomocí haptického kurzoru jednotlivá nastavení materiálů a především rozdíly mezi jednotlivými vlastnostmi haptického materiálu vykreslených těles. Nejcitelnějšími parametry jsou tuhost a tření materiálu. Tlumení materiálu i přes svou vysokou hodnotu neplní předpokládané chování a i při rychlém pohybu po povrchu se

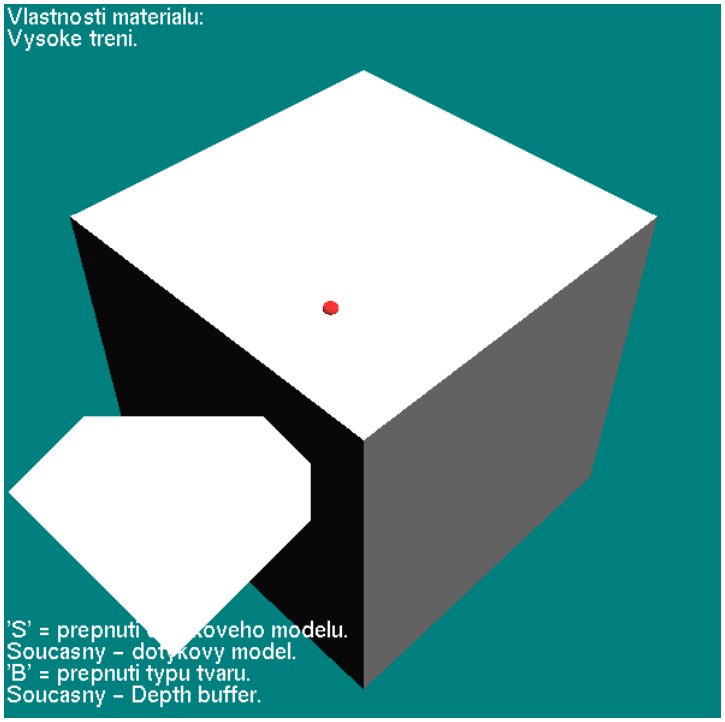
nějaké výraznější odezvy nedostává.

Nejdůležitější části je pozorování rozdílů při haptickém vykreslování tvarů použitím různých bufferů.

Depth buffer je prezentován jako velmi stabilní při dotykovém modelu. Pro kouli se jeví jako mnohem přesnější než feedback buffer, u kterého se při rychlém pohybu přes povrch koule projevuje propadnutí objektem. Naprostou výhodu pak depth buffer získává při interakci s drátovým modelem krychle, kde je uživatel schopen se dotknout jednotlivých čar, zatímco při použití feedback bufferu jsou čáry naprosto nepostřehnutelné a dotyk je nemožný. Ale i u depth bufferu se vyskytují při dotykovém modelu chyby. U tělesa tvaru konvičky se pro depth buffer komplikuje interakce například v oblasti ucha konvičky, kurzor přeskakuje a nekopíruje povrch. Pro stejnou situaci se feedback buffer chová přirozeně a nejsou žádné problémy při interakci.

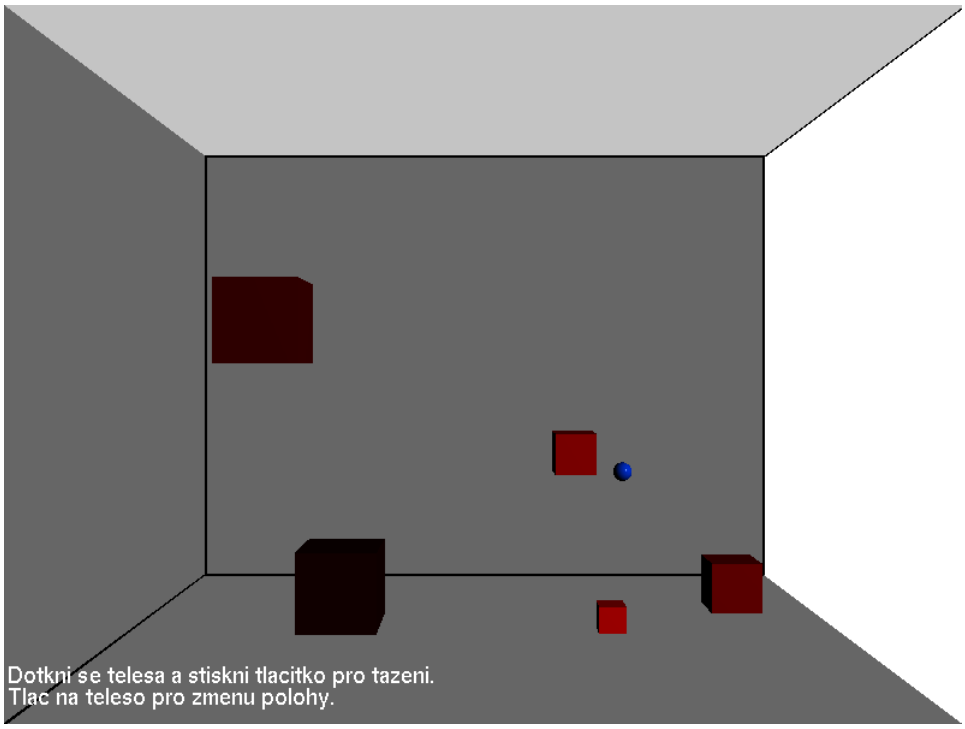
Skoro opačná situace pro hodnocení bufferů nastává při nastavení modelu přitahování. Feedback buffer se zdá obecně lepším způsobem vykreslování haptických tvarů při tomto dotykovém modelu. Největší rozdíly jsou cítit na drátovém modelu krychle, kde feedback buffer plní svou funkci naprosto bezchybně, zatímco na tvar vykreslený z depth buffer se kurzor občas přichytí, ale není schopen zůstat v linii čáry a z předpokládané trajektorie uskakuje.

Je třeba znovu zmínit i korektní práci s resetováním depth bufferu a výměnou OpenGL bufferů (kap. 3.1.4). Pokud se provede výměna až po vykreslení tvaru z depth bufferu provede se vykreslení obsahu bufferu do okna aplikace, jak je vidět na obr. 5.3.



Obrázek 5.3: Vykreslení depth bufferu v levé spodní části okna při nekorektním použití HLAPI

5.3 Cubes



Obrázek 5.4: Prostředí aplikace Cubes

Pro aplikaci Cubes jsem vytvořil grafickou scénu (obr. 5.4) obsahující místnost a tělesa, která jsou rozlišeny podle barvy a velikosti závisející na jejich virtuální hmotnosti.

Díky haptickému kurzoru je možné s tělesy pohybovat. Děje se tak buď tlačením na stěny těles a jejich posunováním nebo při uchopení tělesa tlačítkem na zařízení a jeho posunem nebo vržením.

Všechny fyzikální vzorce a jejich implementace popsané v kapitolách 3.2.3 a 4.3 jsou prováděny velmi dobře. Průběh jevů je věrohodný a splňuje předpokládanou funkčnost. Způsob interakce závisí pouze na uživateli. Po chvíli cviku je možné si s tělesem i „pohazovat“.

Detekce kolizí zajišťuje neustálé setrvání těles v prostoru místnosti a nedovoluje případy, kdy by těleso opustilo vymezenou místnost. Kurzor zařízení přímo nekopíruje detekci kolizí tělesa, jestliže je těleso uchopeno a při kolizi se může pohybovat dále než těleso směrem ke stěně.

Jelikož mapování pracovního prostoru s prostorem místnosti není dokonalé, je nastavena hodnota propadnutí haptického materiálu (kap. 3.1.4) na vysokou hodnotu, nicméně uživatel je schopen se pohybovat i mimo místnost a i při interakci s tělesy se často stává, že kurzor propadne místností. Jelikož se mi nepodařilo implementovat operace, které by vedly k nastavení pozice zařízení na začátku programu uvnitř místnosti, bylo nutné povolit propadnutí objekty.

Kapitola 6

Závěr

Obsah této diplomové práce tvoří představení a popis příkladů využití haptické technologie při manipulaci s 3D objekty. Dostupná haptická technologie je reprezentována zařízením Phantom Omni a toolkitem OpenHaptics. Cílem a výstupem práce je soubor aplikací, které demonstrují funkčnost a možnosti zařízení i získaných knihoven.

První kapitola uvádí problematiku manipulace ve virtuálním prostoru z pohledu matematického aparátu, jenž pokládá pevné základy pro jeho využití v počítačové grafice. Jelikož samotné matematické popisy nestačí, následuje popis operací při vykreslování virtuální scény a krátká historie haptické technologie. Minulost je svázána se současností představením společnosti SensAble Technologies a jejího produktu zařízení Phantom Omni.

Výstupem jsou aplikace vytvořené použitím toolkitu OpenHaptics. Popisu funkčnosti a možností jednotlivých knihoven, které jsou součástí toolkitu, a návrhu demonstračních programů se věnuje druhá kapitola. U jednotlivých knihoven jsou vysvětleny postupy pro naprogramování haptické aplikace a prostředky pro korektní ovládání a komunikaci s haptickým zařízením. Pro každou aplikaci jsou v této kapitole vysvětleny kýžené vlastnosti programu a fyzikální základy jevů, jenž se budou v rámci jednotlivých funkčností simulovat.

Následující dvě kapitoly se zabývají implementací demonstračních aplikací a souhrnem dosažených výsledků. Na začátku kapitoly implementace je popsána použitá i nutná softwarová výbava.

Cíl práce byl dosažen vytvořením tří demonstračních aplikací. Každá aplikace má specifickou funkčnost, aby bylo pokryto co nejširší pole funkčnosti nabízené knihovnami toolkitu. Aplikace ukazují na jednoduchých námětech základní použití funkcí. Zjištěné závěry jsou blíže popsány v kapitole výsledků.

Získaný toolkit OpenHaptics je velmi komplexním řešením vývojového nástroje pro haptické zařízení Phantom. Jednoznačnou výhodou knihoven toolkitu je struktura podobná OpenGL. Podobnost nesouvisí pouze s názvy funkcí knihoven HDAPI a HLAPI, ale především s použitým schématem stavového automatu, které je pro programátora, zvyklého vytvářet aplikace v OpenGL, velmi příjemné a intuitivní. Další výhodou toolkitu je dodaná kvalitní dokumentace a množství ukázkových příkladů.

I přes klady toolkitu je vývoj haptických aplikací spojen se subjektivním posuzováním vytvářeného chování. Nastavení hodnot některých vlastností materiálů a silových efektů je třeba vyzkoušet a upravit podle cílového uživatele.

Při dalším zájmu o haptické programování se zařízením Phantom a toolkitem OpenHaptics by mohlo být vhodné prozkoumat některé další možnosti obsažených knihoven, například nástroj Haptic mouse, který by měl umožnit použití zařízení jako klasickou 2D

myš nebo novinku v souboru knihoven pro tento rok, část QuickHaptics. Nebo se pokusit propojit některý existující fyzikální engine s haptickým zařízením.

Literatura

[1] Bělín J.: *Interaktivní manipulace s 3D objekty ve virtuálním prostoru s využitím 3D skeneru MicroScribe - 3D myš*. 2007, bakalářská práce. Brno. FIT VUT v Brně.

[2] Christopoulos D.: *NeHe Productions: OpenGL Lesson# 30* [online]. URL: http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=30, [cit. 2009–05–10].

[3] Fernandes A.: *GLSL Tutorial* [online]. URL: http://www.lighthouse3d.com/opengl/glsl/index.php?ogloverview, [cit. 2009–05–11].

[4] Immersion Corporation: *What is haptic?* [online]. URL: http://www.immersion.com/corporate/press_room/what_is_haptics.php, 2008 [cit. 2008-12–31].

[5] Jayarajan J., Singh M.: *Master–Slave manipulators: technology and recent developments* [online]. URL: http://www.barc.ernet.in/publications/nl/2006/200606-1.pdf, June 2006 [cit. 2009–04–30].

[6] Klaška J.: *Matematika I: Matice a determinanty* [online]. URL: http://mathonline.fme.vutbr.cz/, 13. 7. 2006 [cit. 2007–04–30].

[7] Mocek T.: *Efektivní manipulace s objekty ve 3D*. 2006, diplomová práce. Brno. FIT VUT v Brně.

[8] SensAble Technologies: *OpenHaptics Toolkit Data sheet* [online]. URL: http://sensable.com/documents/documents/OpenHaptics_datasheet_hi.pdf, 2007 [cit. 2009–05–01].

[9] SensAble Technologies: *OpenHaptics Toolkit version 3.0 API Reference Manual*. 5 December 2008 [cit. 2009–05–04].

[10] SensAble Technologies: *OpenHaptics Toolkit version 3.0 Programmer's guide* [online]. URL: http://sensable.com/documents/documents/OpenHaptics_ProgGuide.pdf, 16 December, 2008 [cit. 2009–05–01].

[11] Sochor J.: *Human computer interaction with force-feedback*. URL: http://is.muni.cz/predmety/predmet.pl?kod=PV160&fakulta=1433&lang=cs&obdobi=4724, 19. 1. 2009 [cit. 2009–04–30].

[12] Svoboda E. a kol.: Přehled středoškolské fyziky. Prometheus, 2003, ISBN 80-7196-116-7.

[13] WWW stránky: *3Dconnexion* [online]. URL: http://www.3dconnexion.com.

[14] WWW stránky: *Engine* [online]. URL: http://cs.wikipedia.org/wiki/Engine, 15. 10. 2008. [cit. 2008–12–31].

[15] WWW stránky: *GLUT – The OpenGL Utility Toolkit* [online]. URL: http://www.opengl.org/resources/libraries/glut/.

[16] WWW stránky: *Haptic technology* [online]. URL: http://en.wikipedia.org/wiki/Haptic_technology, 29. December 2008 [cit. 2008–12–31].

[17] WWW stránky: *Lineární algebra* [online]. URL: http://cs.wikipedia.org/wiki/Line%C3%A1rn%C3%AD_algebra, 22. 4. 2007 [cit. 2007–05–02].

[18] WWW stránky: *Matice* [online]. URL: http://cs.wikipedia.org/wiki/Matice, 28. 4. 2007 [cit. 2007–05–02].

[19] WWW stránky: *OpenGL* [online]. URL http://www.opengl.org.

[20] WWW stránky: *Remote manipulator* [online]. URL: http://en.wikipedia.org/wiki/Remote_manipulator, 7 April 2009 [cit. 2009-04–30].

[21] WWW stránky: *SensAble* [online]. URL: http://www.sensable.com, 2009 [cit. 2009–04–30].

[22] WWW stránky: *Tait-Bryan rotations* [online]. URL: http://en.wikipedia.org/wiki/Tait-Bryan_rotations, 4 may 2009 [cit. 2009–04–28].

[23] Žára J., Beneš B., Sochor J.: Moderní počítačová grafika. Computer Press, 2004, ISBN 80-251-0454-0.

Dodatek A

Manuál aplikací

A.1 Požadavky

Pro spuštění aplikací nebo překlad zdrojových souborů je nutné mít nainstalovaný především OpenHaptics toolkit verze 3.0, grafické knihovny jsou součástí toolkitu a mít k dispozici haptické zařízení Phantom Omni.

OpenHaptics toolkit nemohu vzhledem k licenčním podmínkám společnosti SensAble Technologies vložit na přiložené datové médium, ale není problém získat akademickou verzi toolkitu zadarmo po registraci na stránkách společnosti [21].

Pro spuštění aplikace Cubes je nutné mít do operačního systému integrovanou knihovnu GLEW, jenž je k dispozici na [3]. Verze knihovny OpenGL by měla být alespoň verze 2.0.

A.2 Ovládání aplikací

Jelikož výstupem aplikací jsou především generované silové vjemy, ovládání je velmi strohé.

Na obr. A.1 je znázorněno okno aplikace a popsány jednotlivé prvky.

Všechny tři vytvořené aplikace mají společné některé prvky ovládání:

Ukončení aplikace – uživatel může ukončit aplikaci pomocí kláves „Q“ a „Esc“.

Interakce – funkčnost aplikace závisí na pohybu kurzoru v prostředí. Kurzor zařízení je vždy vykreslen jako koule určité barvy:

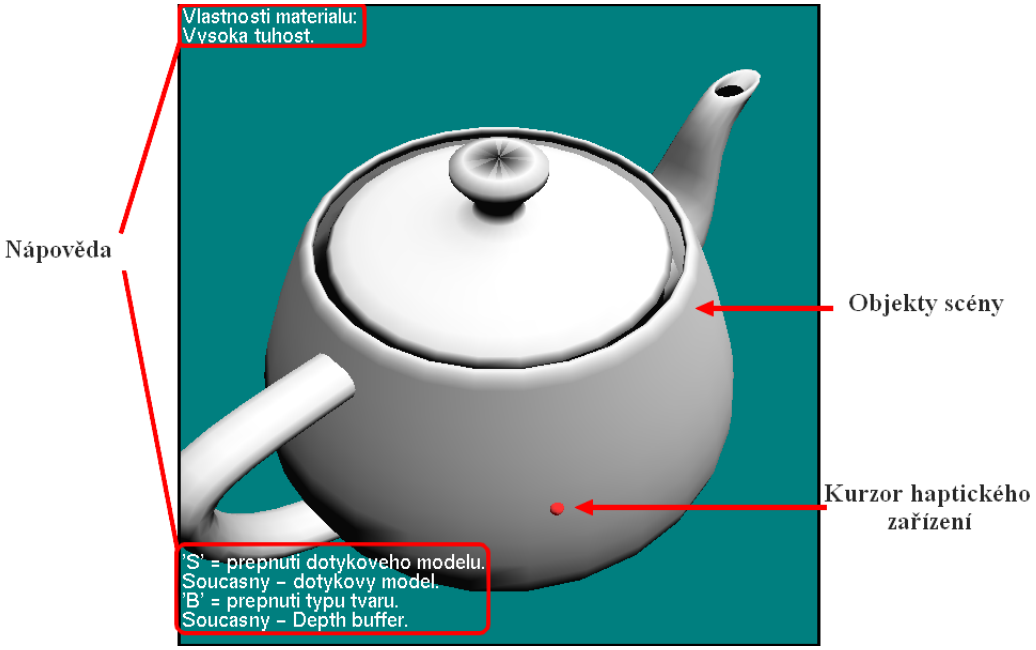
- Force field – žlutá barva
- Simple haptics – červená barva
- Cubes – modrá barva

Force field

Nejsou přidány žádné další ovládací prvky.

Pohybujte snímacím hrotem haptického zařízení v prostoru a pociťujte generovanou sílu podle pozice v silovém poli zdrojů.

Pozorujte průběh síly, který znázorňuje aktuální generovanou sílu.



Obrázek A.1: Popis prvků v okně DEMO aplikací

Simple haptics

Rozšířené ovládání přes klávesy:

1, 2, 3, 4 – Přepínání mezi jednotlivými tvary těles s různými haptickými vlastnostmi materiálu. Definované tvary jsou v daném pořadí koule, krychle, konvička a drátový model krychle.

S – Přepínání mezi dotykovým modelem a modelem přitahování.

B – Přepínání mezi vykreslením haptického obrazu z depth bufferu nebo feedback bufferu.

Interakce kurzorů s objekty dovoluje cítit virtuální tvary, jejich povrchy a případně se pohybovat „ve stěnách těles“ (model přitahování).

Cubes

Rozšíření ovládání dovoluje při stisku tlačítka haptického zařízení na povrchu tělesa, pohybovat s tělesem po virtuální místnosti a vrhat tělesa prostorem.

Pomocí kurzoru je možné působit na tělesa na posouvat je po podložce nebo ovlivňovat jejich pád.

Dodatek B

Obsah CD

- Elektronická forma této práce ve formátu PDF
- Zdrojové soubory této práce
- Spustitelné soubory aplikací
- Složky se zdrojovými kódy aplikací a kompilační soubory
- Dokumentace aplikací