Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Ohmatávání a modifikace geometrických modelů pomocí haptického pera

Plzeň, 2013 **Barbora Janská**

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10.května 2013 Barbora Janská

Poděkování

Tímto bych chtěla poděkovat vedoucí práce Prof. Dr. Ing. Ivaně Kolingerové za její trpělivost a podnětné rady a připomínky a Ing. Václavu Purchartovi za cenné rady a pomoc při vytváření práce.

Anotace

Cílem této práce bylo navržení a vytvoření aplikace pro ohmatávání geometrických modelů pomocí haptického pera. Práce se soustředí na ohmatávání 3D modelů lidských hlav poskytnutých Ing. Petrem Martínkem, a to pomocí dvou různých knihoven z knihovny OpenHaptics firmy SensAble.

První část je zaměřena na prozkoumání možností a výhod či nevýhod vyšší knihovny HLAPI, která umožňuje haptické renderování pomocí OpenGL geometrických primitiv. S pomocí této knihovny je vytvořena aplikace pro ohmatávání 3D modelů hlav včetně odlišení různých haptických materiálů pro vlasy, kůži a vousy.

V druhé části jsou pak popsány klady a zápory nižší knihovny HDAPI. Během vytváření aplikace pro ohmatávání 3D modelů hlav s použitím této knihovny bylo nutné vyřešit výpočet kolizí haptického pera s 3D modelem pro haptickou odezvu.

Dosažené výsledky byly otestovány uživateli. Zpracované výsledky testů jsou uvedeny na konci této práce.

Abstract

The goal of this thesis was to design and implement an application for touching geometrical models using haptic pen. The work concentrates on touching 3D models of human heads provided by Ing. Petr Martínek, using two different libraries from the library OpenHaptics of SensAble company.

The first part is focused on exploring the possibilities and advantages or disadvantages of the higher library HLAPI, that allows the haptic rendering using OpenGL geometric primitives. By using this library there has been created an application for touching 3D head models including differentiation of various haptic materials for hair, skin and beard.

The second part then describes the pros and cons of the lower library HDAPI. During the creation of application for touching 3D head models using this library it was necessary to resolve the calculation of the collisions of the haptic pen with the 3D model for haptic feedback.

The results were user tested. Processed results of the tests are included at the end of this thesis.

Obsah

1 Úvod

V současné době jsou aplikace haptiky a haptická zařízení v popředí zájmu mnoha výzkumných skupin a pracovišť. Vytvořené aplikace uživatelům umožňují umocnit dojem z interakce s počítačem či nějakým zařízením, a to přidáním dalšího smyslového vjemu - hmatu. Tím je umožněno posunutí hranice simulací i interaktivního ovládání na zcela novou úroveň. Například zapojení haptických zařízení v medicíně umožňuje simulovat chirurgické zákroky tak, jakoby byly opravdu prováděny. V dálkovém ovládání zařízení pak operátorům tato zařízení umožňují pocítit zpětnou vazbu efektorů daných zařízení.

Cílem této práce bylo navrhnout a realizovat aplikaci využívající haptického pera. Tato aplikace umožňuje ohmatávání 3D modelů lidských hlav, včetně zprostředkování rozdílných vjemů při ohmatávání různých povrchů (vlasy, vousy). Během návrhu bylo potřeba prostudovat možnosti existujících knihoven pro zpětnou vazbu, a poté po výběru té nejvhodnější knihovny provést vlastní implementaci.

2 Haptika

S použitím haptické technologie (haptiky) lze docílit ohmatávání virtuálních objektů uživatelem. Tato technologie je založena na hmatové zpětné vazbě. Umožňuje uživateli pomocí uplatňování sil, vibrací a pohybů pocítit hmatový kontakt s ohmatávaným objektem a případně s ním i pohybovat.

Jak pro výzkumné, tak pro komerční účely existuje mnoho haptických zařízení, která umožňují uživatelům hmatovou odezvu při dotyku virtuálních těles.
Pozn.: Slovo haptika je řeckého původu a znamená smysl pro dotek.

2.1 Použité haptické zařízení

Pro účely své bakalářské práce jsem použila haptické zařízení dostupné na katedře informatiky a výpočetní techniky na Západočeské univerzitě. Jedná se o zařízení PHANTOM Omni od firmy Sensable viz Obr.2.1.

Toto haptické zařízení, které lze nazvat haptickým perem, má šest stupňů volnosti pro snímání pozice a dvě integrovaná tlačítka na stylusu, která se dají uživatelem naprogramovat na různé události. Rozsah pohybu zařízení je omezen na pohyb ruky a její otáčení v zápěstí. Podrobné parametry zařízení je možné nalézt v Tabulce 2.1.

Obrázek 2.1: Haptické zařízení PHANTOM Omni [Zdroj: `www.sensable.com`]

Pracovní plocha zpětné vazby	> 160 W x 120 H x 70 D mm
Váha	3 lbs 15 oz
Nominální rozlišení polohy	> 0.055 mm
Tření	< 0.26 N
Maximální síla	3.3 N
Souvislá síla (24h)	> 0.88 N
Tuhost v ose X	> 1.26 N / mm
Tuhost v ose Y	> 2.31 N / mm
Tuhost v ose Z	> 1.02 N / mm
Setrvačnost	$\sim$ 45 g
Silová zpětná vazba	x, y, z
Rozhraní	IEEE-1394 FireWire port: 6-pin to 6-pin

Tabulka 2.1: Technické specifikace haptického zařízení PHANTOM Omni [Zdroj dat: `www.sensable.com`]

2.2 Dostupné knihovny

Pro vytvoření aplikace, která bude umožňovat ohmatávání virtuálních objektů pomocí manipulace se zvoleným haptickým zařízením, lze využít některou z dostupných haptických knihoven. V průběhu analýzy jsem uvažovala o následujících dvou knihovnách.

2.2.1 Knihovna CHAI 3D

CHAI 3D je multi-platformní opensource sada knihoven pro haptické renderování, která je využívána v mnoha výzkumných a výrobních projektech, jako například v simulátorech nebo lékařských aplikacích.

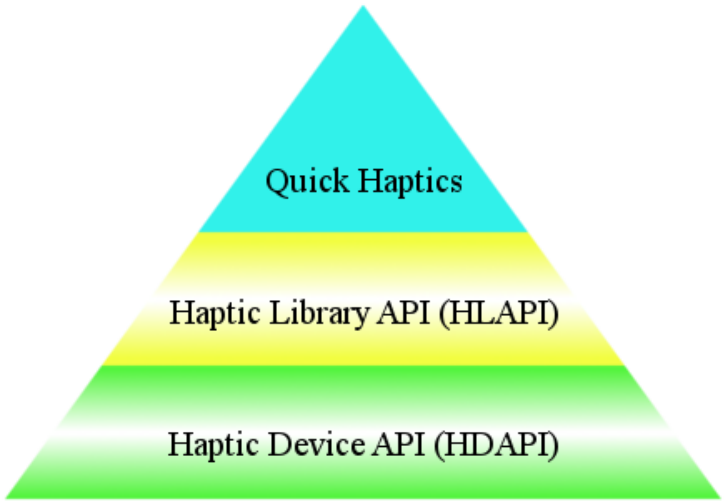
CHAI 3D je napsána v jazyce C++ a je určena pro vytváření aplikací, které kombinují 3D modelování se silovou zpětnou vazbou. Používá velké množství algoritmů pro renderování síly, které umožňují programátorům snadno vyvíjet sofistikované simulace se schopností integrované silové zpětné vazby. OpenGL grafické jádro poskytuje základy pro snadné vykreslování virtuálních prostředí pomocí speciálního 3D grafického akceleračního hardwaru.

Více informací o této knihovně lze nalézt na webových stránkách, viz [1].

2.2.2 Knihovna OpenHaptics

Tato knihovna je poskytována firmou SensAble (nejnovější verze 3.1) a skládá se z několika knihoven. Těmi jsou QuickHaptics micro API, Haptic Device API (HDAPI) a Haptic Library API (HLAPI). Jednotlivé knihovny jsou uspořádány dle Obr.2.2.

Podrobné informace o této knihovně je možné nalézt na stránkách firmy Sensable [4].



Obrázek 2.2: Uspořádání jednotlivých knihoven v OpenHaptics

QuickHaptics micro API

Tato knihovna je nejnovější ze zmíněných knihoven. Je implementována v C++. Umožňuje snadné a rychlé vytváření haptických aplikací nebo přidávání haptiky do již existujících aplikací. Díky vestavěným geometrickým parserům a inteligentním výchozím parametrům je možné vytvořit haptické/grafické aplikace s minimálním množstvím kódu. Definuje čtyři základní funkční třídy:

- DeviceSpace - pracovní prostředí, v němž se může haptické zařízení pohybovat

- QHRenderer - základní třída pro QHWin32 a QHGLUT. Vytvoření okna, které vykreslí tvary z pohledu kamery a umožní uživateli cítit tyto tvary pomocí hmatového zařízení.
- Shape - základní třída pro jeden či více geometrických objektů, které jsou renderovány jak graficky tak hapticky.
- Cursor - grafické znázornění koncového bodu haptického pera

Haptic Device API (HDAPI)

Tato knihovna poskytuje nízkoúrovňový přístup k haptickému zařízení a umožňuje programátorovi přímo generovat sílu a měnit konfiguraci runtime chování. Vyžaduje implementaci efektivních algoritmů pro renderování síly a algoritmů pro detekci kolizí. Zároveň se programátor musí starat o synchronizaci jednotlivých vláken. HDAPI neumožňuje implementaci reakcí na události.

Haptic Library API (HLAPI)

Tato knihovna poskytuje oproti HDAPI vysokoúrovňové haptické renderování. Využívá OpenGL API. Umožňuje použít OpenGL kód jak pro grafické, tak pro haptické renderování. Výpočet haptického renderování je založen na geometrických primitivech, transformacích a vlastnostech materiálu. Výrazně zjednodušuje synchronizaci grafických a haptických vláken. Dále umožňuje implementaci reakcí na události, což usnadňuje implementaci komplikovaných haptických interakcí, jako například dotýkání se geometrie, stisknutí tlačítek a pohyb s objektem.

Všechny tyto knihovny je možno v aplikaci propojit a využívat tak vlastností všech knihoven. HLAPI je postavena na HDAPI, proto může programátor při vytváření aplikace založené na HLAPI využít část funkčnosti z HDAPI. HDAPI je nutné použít pro inicializaci a konfiguraci popisovače haptického zařízení (HDD). Ten je pak využíván HL haptickým renderovým kontextem (HHLRC) pro rozhraní s haptickým zařízením. Tím je programátorovi umožněno řídit chování haptického zařízení.

3 Navržené řešení

Cílem této práce bylo navržení a vytvoření aplikace pro ohmatávání virtuálních geometrických modelů pomocí haptického pera. Pro testování této aplikace jsem měla za úkol použít 3D modely lidských hlav. Tyto triangularizované modely jsou výsledkem diplomové práce Ing. Petra Martínka [5]. Pro realizaci řešení jsem si vybrala knihovnu OpenHaptics od firmy SensAble, jejímž výrobkem je i používané haptické zařízení, a to pro více možností v rozdílnosti přístupů k haptickému renderování než tomu je u knihovny CHAI 3D.

Nejprve jsem se rozhodla využít možností vyšší knihovny HLAPI, s níž jsem již měla zkušenost z předmětu KIV/PRJ5, pro její vysokoúrovňový přístup k haptickému renderování a pro možnost využití OpenGL kódu. Výpočet síly zpětné vazby haptického zařízení je možné v tomto případě ovlivňovat pouze nastavením vlastností ohmatávaného materiálu, což s největší pravděpodobností nebude pro tuto aplikaci postačující vzhledem k tomu, že bude třeba napodobit různé materiály, jako jsou vlasy a vousy. Aby bylo možné implementovat vlastní výpočet síly, bude potřeba využít možností knihovny HDAPI.

Všechny aplikace byly vytvářeny ve vývojovém prostředí Microsoft Visual Studio 2010, pomocí nějž je možné kompilovat program i se všemi nutnými soubory z knihovny OpenHaptics verze 3.1.

3.1 Popis řešení pomocí knihovny HLAPI

Řešení se skládá z následujících částí. Nejprve je načtena trojúhelníková síť 3D modelu hlavy pomocí upravené třídy původně vytvořené Ing. Purchartem. Tuto trojúhelníkovou síť poté používám pro grafickou vizualizaci 3D modelu a též při haptickém renderování, kdy je pomocí knihovních funkcí uložena geometrie jednotlivých trojúhelníků trojúhelníkové sítě a tato síť je následně hapticky vyrenderována. Dále je zapotřebí vykreslit 3D kurzor, který znázorňuje polohu haptického pera ve virtuálním prostředí, kde je umístěn virtuální objekt odpovídající načtenému modelu.

3.1.1 Formát souborů s 3D modely hlav

Triangularizované modely jsou uloženy ve formátu **PLY** (Polygon File Format), který slouží pro ukládání grafických objektů popsaných jako soubor polygonů. Tento formát může obsahovat různé vlastnosti, jako je barva, normála nebo souřadnice textur. V hlavičce PLY souboru jsou vypsány všechny druhy parametrů, které jsou v něm obsaženy. Poté jsou po řádkách vypsány souřadnice vrcholů spolu s jejich vlastnostmi, následují výpisy jednotlivých stěn (polygonů), kde je u každé stěny popsáno, z kolika vrcholů se skládá a jaké jsou indexy těchto vrcholů. Hlavička souborů s modely hlav má následující strukturu. Tato struktura se u různých modelů liší pouze v počtu vrcholů a trojúhelníků:

```
ply
format ascii 1.0
comment VCGLIB generated
element vertex 32334
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar red
property uchar green
property uchar blue
property uchar alpha
property float texture_u
property float texture_v
element face 64544
property list uchar int vertex_indices
end_header
```

Soubor tedy obsahuje souřadnice vrcholů spolu s jejich normálami, barvami, průhledností, texturovými souřadnicemi a indexy vrcholů, z nichž se skládají jednotlivé polygony. Očekáváme, že všechny polygony, ze kterých se model skládá, jsou trojúhelníky. Jiná možnost v této práci není ošetřena.

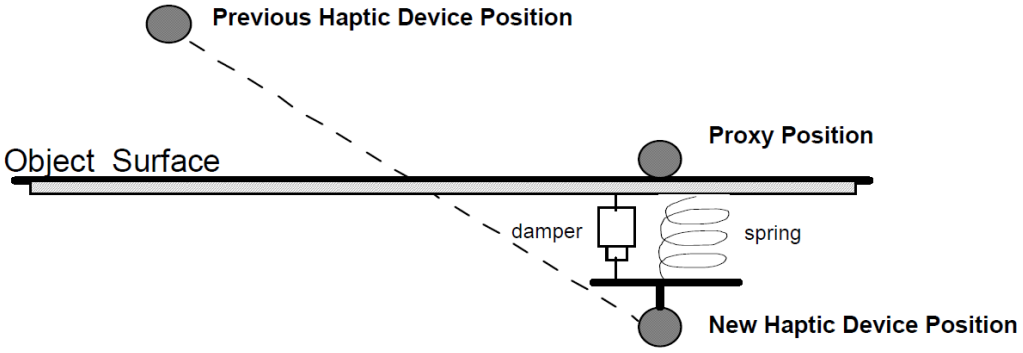
3.1.2 Načtení trojúhelníkové sítě (3D modelu)

Pro načtení 3D modelu ze souboru jsem použila třídu vytvořenou Ing. Purchartem v jazyce C++, kterou bylo potřeba upravit pro danou strukturu PLY souboru. Trojúhelníková síť je vytvořena ze samostatných trojúhelníků. Souřadnice jednotlivých vrcholů trojúhelníků jsou načteny ze souboru a jsou uloženy do připravené struktury. Aby bylo možné z jednotlivých vrcholů poskládat všechny trojúhelníky, je potřeba pro každý trojúhelník znát indexy (pořadové číslo) tří vrcholů, z nichž má být sestaven. Tyto indexy jsou také načteny ze souboru. Poté již stačí pouze použít několik OpenGL příkazů (více v kapitole 3.2.5), které pospojují jednotlivé vrcholy do trojúhelníků a trojúhelníky zobrazí. Pro grafickou vizualizaci je navíc načtena normála a barva vrcholu. U některých 3D modelů hlav jsou jednotlivé vrcholy obarveny podle umístění vlasů a vousů na modelu.

3.1.3 Vykreslení 3D kurzoru

Haptické renderování geometrie (v našem případě trojúhelníků) se provádí pomocí proxy metody. Proxy je zástupný bod haptického zařízení. Poloha tohoto bodu je omezena pouze na vnější plochu ohmatávaných objektů. Tento bod se stále aktualizuje tak, aby odpovídal poloze haptického zařízení, ale zároveň nezasahoval do vytvořených objektů. Zatímco skutečná poloha haptického zařízení může být uvnitř objektu, proxy bod bude vždy mimo. Pokud se haptické pero nedotýká tvaru, bude zástupný bod umístěn v poloze haptického zařízení. Při kontaktu s objektem však poloha haptického zařízení pronikne skrz plochu, ale proxy zůstane na jejím povrchu. Síla zpětné vazby, která je odesílána na haptické zařízení, je vypočítána natahováním virtuální soustavy pružina-tlumič mezi polohou haptického zařízení a pozicí proxy - viz Obr.3.1, kde *Proxy Position* je pozice proxy bodu, *New Haptic Device Position* je aktuální poloha haptického zařízení a *damper-spring* je soustava pružina-tlumič.

Pro vykreslení 3D kurzoru je zapotřebí pouze načíst souřadnice proxy bodu, kde má být kurzor vykreslen, a určit měřítko kurzoru, protože bude vykreslen jako 3D objekt ve scéně.



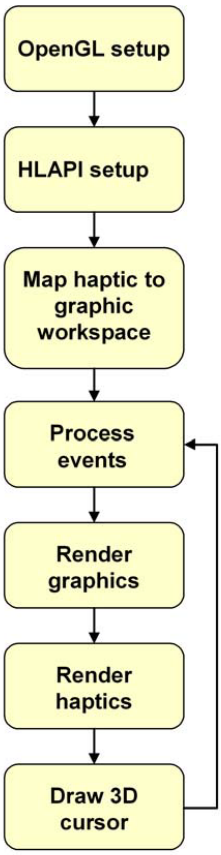
Obrázek 3.1: Znázornění proxy bodu [Zdroj: `http://geomagic.com/files/4013/4851/4367/OpenHaptics_ProgGuide.pdf`]

3.2 Struktura programu používajícího knihovnu HLAPI

Aplikace je implementována v jazyce C++. Součástí programu pro haptické renderování je i upravená třída pro načtení PLY souboru.

Haptická část programu a grafická vizualizace jsou obsaženy v souboru *Touch.cpp*. Typický HL program má strukturu znázorněnou na Obr.3.2.

Program lze tedy kromě načtení PLY souboru rozdělit na několik dílčích částí. Těmi jsou nastavení OpenGL a HLAPI, namapování pracovního prostředí haptického zařízení na grafické prostředí a pak opakovaně probíhající vykreslení grafiky, haptiky a 3D kurzoru.



Obrázek 3.2: Typická struktura HL programu [Zdroj: `http://geomagic.com/files/4013/4851/4367/OpenHaptics_ProgGuide.pdf`]

3.2.1 Struktura třídy pro načtení PLY

Načtení 3D modelu zajišťuje třída, jež se skládá ze souboru *PlyFile.cpp* a příslušného hlavičkového souboru *PlyFile.h*. Původní třída poskytnutá Ing. Purchartem umožňovala jen načtení PLY souboru, který obsahoval kromě hlavičky pouze souřadnice vrcholů a indexy vrcholů každého polygonu. Bylo zde zajištěno i prohození *y-ových* souřadnic se *z-ovými* souřadnicemi pro potřeby OpenGL souřadnicového systému a také seřazení podle hloubky (*z-ové* souřadnice) pomocí algoritmu *quicksort* - viz [6]. Soubor obsahoval i normalizaci souřadnic vrcholů.

Pro načtení PLY souboru s 3D modelem bylo potřeba upravit tuto třídu tak, aby kromě načtení souřadnic vrcholů umožňovala i uložení jejich normál, barev a

texturových souřadnic. Barvy vrcholů určují umístění vlasů a vousů na obličeji. Alpha složka je u každého vrcholu rovna 255, není tedy nutné ji ukládat.

Zároveň jsem třídu modifikovala tak, aby bylo možné model uložit do stejného souboru s již přehozenými souřadnicemi y a z a seřazenými vrcholy podle hloubky. Tím bylo docíleno větší rychlosti při načítání modelu, protože vzhledem k většímu počtu vrcholů je doba jejich seřazení velmi dlouhá. Díky možnosti uložení již upraveného modelu poté stačí jen načíst ze souboru požadované hodnoty.

Pro výběr možnosti, zda chceme model uložit nebo pouze načíst, slouží parametr konstruktoru `action`:

```
static const int ACTION_LOAD = 1;
static const int ACTION_SAVE = 2;
PlyFile(std::string path, int action);
```

Souřadnice vrcholů, jejich barvy a texturové souřadnice jsou pro každý vrchol uloženy do struktury vertex:

```
typedef struct {
    float x,y,z;
    float nx,ny,nz;
    int r,g,b;
    float u,v;
} vertex;
```

kde parametry x, y a z jsou souřadnice daného vrcholu, nx, ny a nz souřadnice jeho normály, r, g, b složky jeho barvy a u a v jeho texturové souřadnice.

Všechny vrcholy uchované v této struktuře jsou postupně přidávány do kontejneru jazyka C++ typu **vector** pojmenovaného `vertices`.

Načítání parametrů vrcholů ze souboru a jejich následné ukládání do kontejneru probíhá v metodě `loadFile(std::string path)`, která se automaticky vykoná po zadání akce načtení do konstruktoru a jejímž atributem je cesta k souboru s modelem. Po načtení všech vrcholů je v metodě zajištěno i načtení indexů polygonů. Indexy vrcholů každého trojúhelníku jsou načteny do struktury `int3`:

```
typedef struct {
    int a,b,c;
} int3;
```

kde parametry a, b a c jsou jednotlivé indexy vrcholů daného trojúhelníka. Pro každý trojúhelník jsou takto uložené informace o jeho vrcholech postupně přidávány do kontejneru **vector** s názvem `faces`.

V metodě `loadFile` je nejprve načtena hlavička s informací o počtu vrcholů a počtu trojúhelníků do proměnných `vertexCount` a `faceCount`. Dále již probíhá postupné načítání parametrů vrcholů a indexů trojúhelníku podle výše zmíněného postupu, které vypadá takto:

```
int alpha;

for (int i=0; i<vertexCount; i++) {
    //vrcholy
    vertex *v = new vertex;
    f >> v->x >> v->y >> v->z;

    //normaly
    f >> v->nx >> v->ny >> v->nz;

    //barvy
    f >> v->r >> v->g >> v->b;

    //skip alphy
    f >> alpha;

    //textur. souradnice
    f >> v->u >> v->v;

    this->vertices.push_back(v);
}

int vertCount;
```

```
for (int i=0; i<faceCount; i++) {

    //pocet vrcholu
    f >> vertCount;
    if(vertCount != 3) {
        cout << "vertCount: "<< vertCount << endl;
    }

    int3 *face = new int3;
    f >> face->a >> face->b >> face->c;
    this->faces.push_back(face);
}
```

Do proměnné `vertCount` se načítá první hodnota z řádky, která informuje o počtu vrcholů polygonu. Je zde provedena kontrola, zda jsou všechny polygony trojúhelníky. Pokud tomu tak není, pouze se vypíše počet vrcholů daného polygonu.

3.2.2 Inicializace OpenGL

Ve svém programu jsem nejprve provedla inicializaci OpenGL, jako je nastavení velikosti okna, nastavení glut funkcí a nastavení barvy pozadí. V programu je vytvořena struktura udržující informace o rozměrech a textu okna, o poloze přední a zadní ořezávací roviny a o pozorovacím úhlu - viz následující kód.

```
typedef struct {
    int width;
    int height;
    char* title;

    float field_of_view_angle;
    float z_near;
    float z_far;
} glutWindow;
```

Mezi glut funkce, které jsou v programu nastavovány patří nastavení okna, kde jsou použity atributy struktury `glutWindow`:

```
glutInitWindowSize(win.width,win.height);
glutCreateWindow(win.title);
```

Dále je inicializován počáteční zobrazovací režim pomocí funkce `glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH)` , kde *GLUT_RGB* znamená zobrazení všech tří složek barvy, *GLUT_DOUBLE* použití okna s dvojitým bufferem a *GLUT_DEPTH* umožní použití depth (hloubkového) bufferu. Dalšími funkcemi, které jsou nastaveny jsou `glutDisplayFunc(render)`, kde je jako zobrazovací funkce nastavena funkce `render`, která obsahuje vykreslení grafiky a haptiky, a `glutReshapeFunc(glutReshape)`, kde je jako obnovovací funkce nastavena `glutReshape`. V této metodě je nastavena část okna, do které se bude kreslit, pomocí `glViewport()`. Dále je zde nastavení maticového režimu projekce pomocí `glMatrixMode(GL_PROJECTION)`, perspektivy pomocí `gluPerspective()` a pohledu na scénu pomocí funkce `gluLookAt()`.

Inicializace OpenGL probíhá především v metodě `initGL`, kde je pro vykreslování povolen hloubkový buffer a nastavena funkce pro tento buffer, nastaven způsob výpočtu perspektivy, určena hodnota hloubky použitá, když je tento buffer vymazán a nastavena barva pozadí - viz následující kód.

```
glDepthFunc(GL_LEQUAL);
glEnable(GL_DEPTH_TEST);

glHint( GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST );
glClearDepth( 1.0f );

glClearColor(0, 0, 0, 0);
```

Je zde umístěno i nastavení osvětlení, které je nejprve nutné povolit. Poté je již nastavena poloha světla.

```
glEnable(GL_LIGHTING);
```

```
float lightPos[4] = {0, 0, 0.5, 0.0};
glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

glEnable(GL_LIGHT0);
```

3.2.3 Nastavení HLAPI

Dále je v programu provedeno nastavení HLAPI v metodě `initHL`, kde je inicializován popisovač haptického zařízení a následně předán haptickému renderovému kontextu - viz následující kód.

```
ghHD = hdInitDevice(HD_DEFAULT_DEVICE);
ghHLRC = hlCreateContext(ghHD);
hlMakeCurrent(ghHLRC);
```

3.2.4 Namapování haptiky na grafické prostředí

Pro definování jednotného namapování pracovního prostoru na pozorovací souřadnice grafické vizualizace tak, aby tento prostor zcela obklopoval rozsah projekce, tedy aby vše, co je viditelné, bylo také hmatatelné, je použita funkce `hluFitWorkspace()` s projekční maticí, která definuje rozsah projekce - viz následující kód.

```
hlMatrixMode(HL_TOUCHWORKSPACE);
hluFitWorkspace(projection);
```

3.2.5 Vykreslení grafiky

Pro vykreslení trojúhelníkové sítě 3D modelů hlav jsem v programu použila příkazy knihovny OpenGL. V programu je implementována struktura udržující informace o souřadnicích vrcholu:

```
struct VertexCoords{
    GLfloat x;
    GLfloat y;
```

```
    GLfloat z;
};
```

Všechny souřadnice vrcholů jsou do pole těchto struktur s názvem `vertexArray` postupně pro každý vrchol načteny ze struktury `vertex` nacházející se ve třídě *PlyFile*. Z této struktury jsou zároveň načteny normály vrcholů do pole s názvem `normals` a barvy do pole s názvem `colors`. Indexy vrcholů jednotlivých trojúhelníků jsou načteny ze struktury `int3` do pole s názvem `indexes`, které je typu `GLuint` (OpenGL integer bez znaménka).

Vykreslení trojúhelníkové sítě je pak uskutečněno pomocí OpenGL vertex array, normal array a color array, kdy jsou jednotlivé pointery nastaveny na pole vrcholů, normál a barev. Poté je pomocí funkce `glDrawElements()` s použitím pole indexů vykreslena celá trojúhelníková síť.

Kód pro vykreslení celé trojúhelníkové sítě vypadá takto:

```
//Povoleni vertex, normal a color array.
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_COLOR_ARRAY);

glColorPointer(3, GL_FLOAT, 0, colors);
glNormalPointer(GL_FLOAT, 0, normals);
glVertexPointer(3, GL_FLOAT, 0, vertexArray);


glDrawElements(GL_TRIANGLES, triCount*3, GL_UNSIGNED_INT, indexes);

// Deaktivace vertex, normal a color array po vykresleni.
glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_NORMAL_ARRAY);
glDisableClientState(GL_COLOR_ARRAY);
```

3.2.6 Haptické renderování

Veškeré haptické renderování je obsaženo v haptickém rámci mezi příkazy `hlBeginFrame()` a `hlEndFrame()`. První z těchto příkazů zjistí aktuální stav haptického renderování. Jakmile je haptika vykreslena, příkaz konce rámu synchronně vyhodnotí dynamické změny pozice haptického zařízení.

Pro haptické vykreslení ploch a objektů skládajících se z různých geometrických primitiv je nutné použít tzv. renderování tvarů. Geometrie jednotlivých tvarů je specifikována pomocí OpenGL příkazů a musí být ohraničena příkazy `hlBeginShape()` a `hlEndShape()`. HLAPI pak používá tuto geometrii pro haptické renderování. To se může dít dvěma různými způsoby. Buď s použitím depth bufferu nebo feedback bufferu. Pokud HLAPI zachytí geometrii použitím OpenGL depth bufferu, bude se moci uživatel ve výsledné aplikaci dotýkat pouze viditelných ploch objektů. Pokud v programu nastavíme použití OpenGL feedback bufferu pro zachycení geometrie, může uživatel ve výsledném programu ohmatávat například i tunel. Ve svém programu používám feedback buffer, protože umožňuje použití OpenGL vertex array.

OpenGL příkazy pro vykreslení haptiky jsou téměř stejné jako pro vykreslení grafiky, pouze bez použití normál a barev vrcholů.

Pro nastavení vlastností ohmatávaného materiálu jsem využila fakt, že vrcholy modelu jsou obarveny podle umístění vlasů a vousů. V oblasti 3D modelu hlavy, kde nejsou vlasy ani vousy, mají vrcholy šedou barvu. Vrcholy v oblasti vlasů jsou obarveny červenou barvou a v oblasti vousů zelenou barvou. Aby bylo možné pro každou z těchto tří oblastí nastavit jiné vlastnosti materiálu, bylo třeba rozdělit trojúhelníky podle barevného rozlišení. To jsem zajistila rozdělením pole `indexes` na tři další pole. V cyklu, který postupně prochází indexy vrcholů pro každý trojúhelník, je zjištěno, jakou barvu daný trojúhelník má. Trojúhelník se může skládat z vrcholů maximálně dvou různých barev, jelikož oblasti vlasů a vousů spolu nesousedí. Pro každý trojúhelník je spočtena převládající barva jeho vrcholů a tento trojúhelník, přesněji indexy jeho vrcholů, je následně zařazen do jednoho z kontejnerů typu **vector** s názvy `vecIndexesS` (šedivé trojúhelníky), `vecIndexesR` (červené trojúhelníky), `vecIndexesG` (zelené trojúhelníky).

Typ **vector** je zde zvolen kvůli možnosti dynamické změny jeho velikosti, jelikož není známo, kolik trojúhelníků které barvy je, jinak by bylo třeba celý cyklus zopakovat. Z těchto "vectorů" jsou pak indexy vrcholů trojúhelníků pouze překopírovány do polí s názvy `indexesS`, `indexesR` a `indexesG`, aby bylo možné je použít pro vytvoření trojúhelníkové sítě pomocí OpenGL příkazu `glDrawElements`.

Každá oblast modelu hlavy musí být zvlášť uzavřena mezi příkazy pro definování tvaru (`hlBeginShape()` a `hlEndShape()`), aby bylo možné každé oblasti nastavit jiný materiál. Proto jsou navíc vygenerována jedinečná id s názvy `ShapeIdS`, `ShapeIdR` a `ShapeIdG`, které je nutné použít v příkazu `hlBeginShape()`.

V haptickém rámci jsou tedy pro každý tvar nastaveny jiné vlastnosti tak, aby co nejvíce připomínaly kůži, vlasy či vousy. Vlastnosti pro šedivé trojúhelníky, tedy kůži, jsou nastaveny takto:

```
hlMaterialf(HL_FRONT_AND_BACK, HL_STIFFNESS, 0.5f); //tuhost
hlMaterialf(HL_FRONT_AND_BACK, HL_DAMPING, 0.9f); //tlumeni
hlMaterialf(HL_FRONT_AND_BACK, HL_STATIC_FRICTION, 0.17f); //treni
hlMaterialf(HL_FRONT_AND_BACK, HL_DYNAMIC_FRICTION, 0.95f);
```

Materiálové parametry červených trojúhelníků, tedy vlasů, jsem nastavila následujícím způsobem.

```
hlMaterialf(HL_FRONT_AND_BACK, HL_STIFFNESS, 0.5f); //tuhost
hlMaterialf(HL_FRONT_AND_BACK, HL_DAMPING, 0.9f); //tlumeni
hlMaterialf(HL_FRONT_AND_BACK, HL_STATIC_FRICTION, 0.7f); //treni
hlMaterialf(HL_FRONT_AND_BACK, HL_DYNAMIC_FRICTION, 0.5f);
```

Materiálové vlastnosti zelených trojúhelníků, které znázorňují umístění vousů, jsou nastaveny takto:

```
hlMaterialf(HL_FRONT_AND_BACK, HL_STIFFNESS, 0.5f); //tuhost
hlMaterialf(HL_FRONT_AND_BACK, HL_DAMPING, 0.9f); //tlumeni
hlMaterialf(HL_FRONT_AND_BACK, HL_STATIC_FRICTION, 0.94f); //treni
hlMaterialf(HL_FRONT_AND_BACK, HL_DYNAMIC_FRICTION, 0.1f);
```

3.2.7 Vykreslení 3D kurzoru

Po vykreslení haptiky a grafiky je vykreslen 3D kurzor reprezentující polohu haptického pera. Kurzor je vykreslen na pozici proxy bodu, jehož polohu získáme pomocí následujících příkazů:

```
HLdouble proxyxform[16];
hlGetDoublev(HL_PROXY_TRANSFORM, proxyxform);
```

3.3 Zhodnocení dosažených výsledků implementace s využitím knihovny HLAPI

Výsledná aplikace umožňuje uživateli ohmatávat 3D model hlavy, a to bez proniknutí dovnitř modelu s výjimkou úst, kde dochází k proniknutí kurzorem znázorňujícím polohu haptického pera dovnitř modelu. Vzhledem k možnosti nastavení pouze čtyř vlastností materiálu není výsledný dojem blízký realitě, což bylo zároveň otestováno uživateli, viz kapitola 4. Vlastnosti, které je možno nastavit, jsou pouze tuhost, tlumení a statické a dynamické tření. Pomocí těchto parametrů lze nastavit pouze odpor či přilnavost modelu. Nelze tak napodobit vystupující materiál, jako jsou například vousy. Z tohoto důvodu jsem aplikaci pro ohmatávání 3D modelů hlav implementovala i s použitím nižší knihovny HDAPI.

3.4 Popis řešení pomocí knihovny HDAPI

Na rozdíl od vyšší knihovny HLAPI musí být v programu vytvořeném s použitím HDAPI implementován vlastní výpočet detekce kolizí haptického pera s modelem a výpočet síly. Grafická vizualizace modelu hlavy je řešena stejně jako u aplikace, která využívá HLAPI, a to pomocí OpenGL - viz kapitola 3.2.5. Načtení 3D modelů hlav je implementováno také stejným způsobem jako v programu využívajícím HLAPI s použitím třídy obsažené v *PlyFile.cpp* a *PlyfFile.h* - viz kapitola 3.2.1.

3.4.1 Vykreslení 3D kurzoru

3D kurzor je vykreslen na pozici haptického pera v prostředí, ve kterém se nachází virtuální objekt. Pro vykreslení 3D kurzoru je zapotřebí pouze načíst transformační matici haptického pera, která shromažďuje informace o poloze a natočení haptického zařízení v jeho vlastních souřadnicích. Proto musí být OpenGL matice přenásobena maticí, která transformuje OpenGL souřadnice do souřadnic zařízení, na něž je pak aplikována zmiňovaná transformační matice haptického zařízení. Jakmile je známo, kde má být kurzor vykreslen, je určeno měřítko kurzoru pomocí převodní matice mezi OpenGL souřadnicemi a souřadnicemi haptického zařízení.

3.4.2 Haptická smyčka

Jak již bylo zmíněno, v programu je třeba implementovat vlastní detekci kolizí haptického pera s virtuálním objektem a vlastní výpočet síly. To je zajištěno v haptické smyčce, která je plánovačem spouštěna s vysokou frekvencí, tedy velmi často tak, aby byla spočítána zpětná síla i při sebemenší změně polohy haptického pera. Pokud by tomu tak nebylo, rozdíl mezi silami by byl moc velký a dojem z ohmatávání virtuálního objektu by nebyl dostatečně hladký.

Detekce kolizí haptického zařízení s trojúhelníkovou sítí byla v této práci řešena dvěma různými způsoby. Část programu související s nastavením HDAPI a haptickou smyčkou je však shodná u obou řešení a bude proto popsána pouze jednou.

Počáteční řešení detekce kolizí

Počáteční přístup k řešení detekce kolizí spočíval v počítání průsečíků paprsku, který je vytvořen na základě znalosti pozice haptického pera a jeho směrového vektoru, se všemi trojúhelníky tvořícími model hlavy. Pozici haptického pera i jeho natočení v prostoru je možné získat z jeho transformační matice. Tu bylo však potřeba převést do stejného souřadnicového systému, ve kterém je umístěn virtuální model hlavy. Pokud byly nalezeny nějaké průsečíky paprsku s trojúhelníky, bylo zjištěno, jak daleko se nachází od pozice haptického pera. Nacházel-li se průsečík dostatečně blízko, byla pro něj spočítána výsledná síla, která měla směr normály trojúhelníku. Velikost této síly byla určena v závislosti na vzdálenosti pera od průsečíku. Čím blíže se pero nacházelo, tím větší byla silová zpětná vazba. Postupné zvětšování síly při přibližování haptického pera k povrchu modelu je nutné, aby haptické pero nevyvinulo najednou velikou sílu a neuskočilo proti uživateli.

Tento způsob detekce kolizí však neumožňuje dotýkání se modelu stranou haptického pera, jelikož pero a tím pádem i paprsek jsou natočeny mimo model hlavy a nejsou nalezeny žádné průsečíky. Teprve jakmile se haptickým perem pronikne skrze model, je nalezen průsečík, výsledná síla je spočítána jako maximální a pero uskočí. Zároveň kvůli vysoké frekvenci spouštění výpočtu haptické smyčky a vysokému počtu trojúhelníků, pro které je počítán průsečík s paprskem haptického pera, je haptické renderování velmi pomalé a nestíhají se tak počítat průsečíky dost často. Proto bylo toto řešení optimalizováno tak, aby se počítaly průsečíky paprsku pouze s trojúhelníky, jejichž první vrchol (uložen v poli `indexes` jako první) se od pera nacházel do nějaké maximální vzdálenosti.

Vylepšené řešení detekce kolizí

Kvůli již zmíněným problémům s prvním řešením jsem se rozhodla pro implementaci odlišného způsobu detekce kolizí. Ten je založen na počítání vzdáleností těžišť jednotlivých trojúhelníků celého modelu od pozice haptického pera, která musí být opět převedena do stejného souřadnicového systému, jako je virtuální model. Pokud je některá z těchto vzdáleností dostatečně malá a haptické pero se tedy nachází v blízkosti daného trojúhelníku, je spočítána kolmá vzdálenost od

roviny, v níž trojúhelník leží.

Tato vzdálenost je poté opět použita pro výpočet silové zpětné vazby. Směr síly je určen normálou, která je spočítána pomocí barycentrických souřadnic, pro jejichž výpočet je nutné spočítat kolmý průmět polohy haptického pera do roviny trojúhelníku (tedy vlastně průsečík této roviny s přímkou určenou pozicí pera a normálou trojúhelníku.) Barycentrické souřadnice určují váhu normál jednotlivých vrcholů, které již známe a z nichž je počítána výsledná normála. Pokud se průsečík nachází mimo trojúhelník, je směr síly určen normálou nejbližšího vrcholu trojúhelníku.

3.5 Struktura programu používajícího knihovnu HDAPI

Tato aplikace je, stejně jako program vytvořený ve vyšší knihovně HLAPI, implementována v jazyce C++. Součástí programu pro haptické renderování je již dříve zmíněná třída pro načtení PLY souboru a také třída pro základní operace s vektory. Haptická část programu a grafická vizualizace jsou opět obsaženy v souboru *Touch.cpp*.

Kód v tomto souboru lze rozdělit do několika dílčích částí. Jak již bylo řečeno, nastavení OpenGL a vykreslení grafiky jsou stejné jako u aplikace používající vyšší knihovnu HLAPI. Dalšími důležitými částmi programu jsou nastavení HDAPI, namapování pracovního prostředí haptického zařízení na grafické prostředí, vykreslení 3D kurzoru a implementace haptické smyčky a plánovače této smyčky.

3.5.1 Třída pro operace s vektory

Tato třída mi byla poskytnuta Ing. Purchartem a je rozdělena do souborů *Vector3.cpp* a *Vector3.h*. Definuje datový typ `Vector3`, který se skládá ze tří souřadnic vektoru a pro nějž jsou implementovány různé operace, včetně sčítání vektorů, násobení vektoru konstantou, skalárního součinu vektorů, vektorového součinu či normalizace vektoru.

3.5.2 Nastavení HDAPI

Abychom mohli použít možnosti knihovny HDAPI, je třeba ji nejprve inicializovat. To je provedeno v metodě `initHD`, kde je inicializováno haptické zařízení,

povolena silová zpětná vazba zařízení a je zde také spuštěn plánovač haptické smyčky:

```
HHD ghHD = hdInitDevice(HD_DEFAULT_DEVICE);
hdEnable(HD_FORCE_OUTPUT);
hdStartScheduler();
```

3.5.3 Namapování haptiky na grafické prostředí

Pro správné namapování pracovního prostoru haptického zařízení na pozorovací souřadnice grafické vizualizace tak, aby zcela obklopoval rozsah projekce, je použita funkce `hduMapWorkspaceModel()`, která vypočte transformační matici pro převod mezi souřadnými systémy grafické vizualizace a haptického pera. Parametry této metody jsou projekční matice, modelview matice a matice, do níž chceme uložit výsledek:

```
hduMapWorkspaceModel(modelview, projection, workspacemodel);
```

Je zde nastaveno i měřítko 3D kurzoru, pro jehož výpočet je nutné použít viewport matici a transformační matici spočítanou v předchozím kroku. Zajišťuje to metoda `hduScreenToWorkspaceScale()`, jejíž parametry jsou uspořádány podle následujícího kódu.

```
screenTworkspace = hduScreenToWorkspaceScale(
        modelview, projection, viewport, workspacemodel);
```

3.5.4 Vykreslení 3D kurzoru

3D kurzor je vykreslen na pozici haptického pera v prostředí, ve kterém se nachází virtuální objekt. Toho docílíme tak, že vynásobíme aktuální OpenGL matici transformační maticí `workspacemodel` a tím převedeme grafický souřadnicový systém na systém haptického pera. Poté stačí již jen vynásobit maticí haptického zařízení `state.transform`. Výslednou polohu, na níž je kurzor vykreslen, nastavíme tedy pomocí následujících příkazů:

```
glMultMatrixd(workspacemodel);


glMultMatrixd(state.transform);
```

3.5.5 Implementace haptické smyčky a plánovače

Haptická smyčka je volána v plánovacím vlákně, které zasílá haptickému zařízení aktualizace síly s frekvencí přibližně 1000 Hz. Proto má toto vlákno velmi vysokou prioritu. Pokud je v aplikaci nutné zjistit stav haptického zařízení, jako jeho polohu, natočení či nastavenou sílu, je třeba tento dotaz provést uvnitř tohoto plánovacího vlákna, jelikož stav zařízení se mění velmi často.

Ve svém programu zjišťuji tento stav zařízení pomocí typické "callback" metody, jež je plánovačem volána. Kód struktury udržující informaci o stavu zařízení je následující:

```
struct DeviceState
{
    hduVector3Dd position;
    HDdouble transform[16];
    hduVector3Dd force;
};
```

Struktura obsahuje informaci o pozici, transformační matici a aktuální nastavenou sílu haptického zařízení.

Metoda pro zjištění tohoto stavu v plánovacím vlákně vypadá takto:

```
HDCallbackCode HDCALLBACK DeviceStateCallback(void *pUserData)
{
    DeviceState *pState = static_cast<DeviceState *>(pUserData);

    hdGetDoublev(HD_CURRENT_POSITION, pState->position);
    hdGetDoublev(HD_CURRENT_FORCE, pState->force);
    hdGetDoublev(HD_CURRENT_TRANSFORM, pState->transform);

    return HD_CALLBACK_DONE;

}
```

Návratová hodnota metody je `HD_CALLBACK_DONE`, což znamená, že metoda proběhne pouze jednou a již není plánovačem přeplánována. Tato metoda je volána

u již zmiňovaného vykreslování 3D kurzoru, kde je potřeba právě transformační matice zařízení:

```
DeviceState state;
hdScheduleSynchronous(DeviceStateCallback, &state,
                          HD_DEFAULT_SCHEDULER_PRIORITY);
```

Metoda `hdScheduleSynchronous` zajišťuje volání metody pro zjištění stavu plánovačem synchronně, což znamená že aplikační vlákno čeká, dokud volání této metody není dokončeno. Naproti tomu asynchronní volání se vrací okamžitě, jakmile je naplánováno.

K asynchronnímu volání dochází u haptické smyčky:

```
gSchedulerCallback = hdScheduleAsynchronous(
        ForceCallback, 0, HD_DEFAULT_SCHEDULER_PRIORITY);
```

Hlavička metody haptické smyčky vypadá následovně:

```
HDCallbackCode HDCALLBACK ForceCallback(void *data)
```

V metodě je nejprve nastaven popisovač haptického zařízení pomocí příkazu:

```
HHD hHD = hdGetCurrentDevice();
```

Poté je spočítána a nastavena výsledná síla zařízení a nakonec je vrácena hodnota `HD_CALLBACK_CONTINUE`, která zajišťuje, že je smyčka přeplánována a poběží znovu během dalšího tiku plánovače.

Samotný výpočet a nastavení síly zpětné vazby haptického zařízení je uzavřen do haptického rámce mezi příkazy `hdBeginFrame(hHD)` a `hdEndFrame(hHD)`. V tomto rámci je zajištěn neměnný stav haptického zařízení, aby s ním mohlo být pracováno a následně nastavena nová síla. Obsah haptického rámce je tedy odlišný pro různé způsoby detekce kolizí a výpočtu výsledné síly.

3.5.6 Počáteční postup

Jak již bylo předesláno v kapitole 3.4.2, tento přístup spočívá v počítání průsečíků paprsku haptického pera se všemi trojúhelníky modelu hlavy.

Směr a pozice pera

Pro určení paprsku haptického pera je třeba znát jeho polohu a směr, a to ve stejných souřadnicích jako je model hlavy. Tyto údaje o haptickém zařízení je možné získat z jeho matice, vypovídající o aktuálním stavu. Abychom tuto matici převedli do stejného souřadnicového systému jako jsou souřadnice modelu, je třeba transformační matici `workspacemodel`, která zajišťuje převod mezi souřadnými systémy, vynásobit maticí haptického zařízení. Z výsledné matice již dostaneme polohu i směrový vektor haptického pera podle následujícího kódu.

```
GLdouble *result = multMatrices(workspacemodel,transform);

//poloha pera
position = Vector3(result[12], result[13], result[14]);

//smerovy vektor pera
direction = Vector3(result[8], result[9], result[10]) * (-1);
```

Výpočet průsečíku paprsku s trojúhelníkem

Pro každý trojúhelník je nalezen průsečík s paprskem haptického pera, pokud nějaký existuje. Tento výpočet probíhá v metodě `getIntersect()`, jejímiž parametry jsou vrcholy daného trojúhelníku a směr a pozice pera. Způsob výpočtu průsečíku je převzatý z [7].

Nejdříve je potřeba zjistit, zda paprsek protíná rovinu, v níž leží daný trojúhelník. Pokud ne, neprotíná paprsek ani trojúhelník. Je-li však nalezen průsečík, stačí jen zjistit, zda leží uvnitř trojúhelníku.

Nejprve je vypočtena normála trojúhelníku jako vektorový součin vektorů hran trojúhelníku:

```
Vector3 u = V1 - V0;
Vector3 v = V2 - V0;
Vector3 n = Vector3::cross(u, v);
```

Poté je pomocí skalárního součinu normály se směrovým vektorem haptického pera určeno, zda paprsek protíná rovinu.

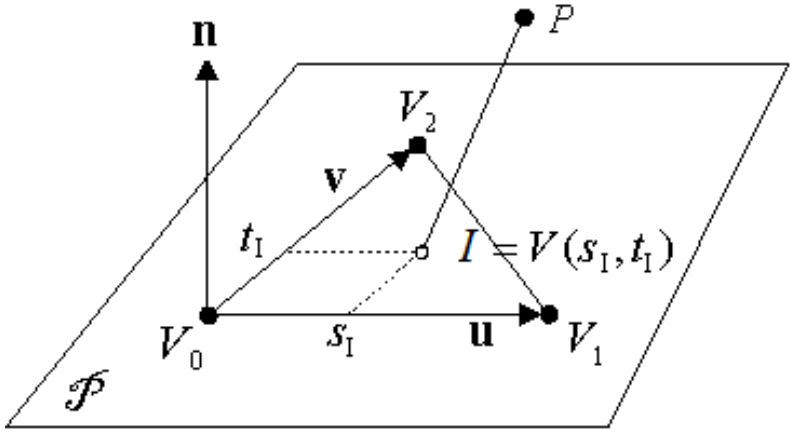
```
b = Vector3::dot(n,direction);
if (fabs(b) < SMALL_NUM) {...}
```

Pokud je výsledná hodnota nulová (menší než velmi malé číslo - vyhnutí se přetečení při dělení), znamená to, že je směrový vektor kolmý k normále trojúhelníku, tedy rovnoběžný s rovinou trojúhelníku.

Jakmile je zjištěno, že paprsek protíná rovinu, vypočte se průsečík následovně:

```
w0 = position - V0;
a = - Vector3::dot(n,w0);
r = a / b;
Vector3 I = position + (direction * r);
```

Dále již zbývá pouze zjistit, zda vypočtený průsečík leží uvnitř trojúhelníku. Definujeme si vektor w jako $(I - V0)$ a spočteme barycentrické souřadnice s a t, přičemž $w = su + tv$. Odvození polohy průsečíku I pomocí barycentrických souřadnic s a t je znázorněno na Obr.3.3, kde V_0, V_1 a V_2 jsou vrcholy daného trojúhelníku, n normála trojúhelníku, u a v vektory hran trojúhelníku a bod P je poloha haptického pera.



Obrázek 3.3: Znázornění výpočtu průsečíku

Barycentrické souřadnice vypočteme podle vzorců

$$s = \frac{(u.v)(w.v) - (v.v)(w.u)}{(u.v)^2 - (u.u)(v.v)}$$
$$t = \frac{(u.v)(w.u) - (u.u)(w.v)}{(u.v)^2 - (u.u)(v.v)} \tag{3.1}$$

Pokud $s \geq 0$, $t \geq 0$ a zároveň $s + t \leq 1$, průsečík leží uvnitř trojúhelníku. V programu je výpočet barycentrických souřadnic implementován následovně.

```
float    uu, uv, vv, wu, wv, D;
uu = Vector3::dot(u,u);
uv = Vector3::dot(u,v);
vv = Vector3::dot(v,v);
w = I - V0;
wu = Vector3::dot(w,u);
wv = Vector3::dot(w,v);
D = uv * uv - uu * vv;

float s, t;
s = (uv * wv - vv * wu) / D;
```

```
if (s < 0.0 || s > 1.0)   {      // I je vne T
    return 0;
}

t = (uv * wu - uu * wv) / D;
if (t < 0.0 || (s + t) > 1.0)  { // I je vne T
    return 0;
}
```

Pokud je zjištěno, že existuje průsečík paprsku s trojúhelníkem, je uložen do kontejneru typu **vector** s názvem `intersections`.

Výpočet parametru t

Po dokončení cyklu pro výpočet průsečíků se všemi trojúhelníky modelu je pro všechny nalezené průsečíky, pokud nějaké existují, vypočten parametr t podle parametrické rovnice přímky:

$$X = A + t.\overrightarrow{u} \quad , \tag{3.2}$$

kde X je nalezený průsečík, A pozice pera a $\overrightarrow{u}$ směrový vektor pera. Rovnici stačí spočítat pouze pro x-ovou souřadnici, parametr t musí být pro všechny souřadnice stejný. Z vypočítaných parametrů průsečíků je vybrán nejmenší a otestován, zda je menší než definovaná konstanta `EPS_T`, která je rovna 0.02 a která označuje největší možnou vzdálenost pera od povrchu modelu, pro kterou se provede výpočet silové zpětné vazby. Pokud parametr t splňuje podmínku, je použit pro výpočet síly.

Výpočet síly

Její směr je určen normálou trojúhelníku, která musí být normalizována. Výslednou sílu vypočteme podle vzorce:

$$\overrightarrow{F} = \overrightarrow{n}.(EPS\_T - |t|)^2.\frac{1}{(EPS\_T)^2} \quad , \tag{3.3}$$

kde $\overrightarrow{n}$ je normála daného trojúhelníku a EPS_T je již dříve definovaná konstanta,

která je maximální hodnotou parametru $|t|$, pro který se provádí výpočet síly. Tím je zajištěno postupné zvyšování síly při přibližování haptického pera k modelu a to se zvyšující se rychlostí.

Optimalizace výpočtu síly

Výsledkem dosavadního postupu bylo velmi pomalé počítání průsečíků a tedy i výsledné síly. To jsem vylepšila třemi různými způsoby.

První vylepšení spočívá v tom, že pokud byl v předešlém počítání průsečíků nalezen nějaký dostatečně blízký průsečík, je uložen do paměti trojúhelník, ve kterém se průsečík nachází, a normála, která byla pro tento trojúhelník spočítána. Při dalším hledání průsečíků se nejprve zjistí, zda se paprsek pera protíná se stále stejným trojúhelníkem, a také, zda je parametr t, vypočtený pomocí nového průsečíku, menší než `EPS_T`. Pokud ano, není již třeba počítat průsečíky paprsku s ostatními trojúhelníky a sílu lze vypočíst pouze na základě znalosti předešlé normály a nového parametru t.

Další optimalizací je úplné vynechání výpočtu síly na 70 iterací haptické smyčky, kdy zůstane zachována předešlá síla. Také pokud během výpočtu průsečíků není žádný nalezen, je k počtu iterací, které se mají vynechat, přičteno 200 iterací.

Poslední optimalizace spočívá v tom, že je před výpočtem průsečíku pro každý trojúhelník zkontrolováno, zda jeho první vrchol v pořadí spadá do sféry o poloměru 0.05 se středem na pozici haptického pera. To znamená, že je pro každý z těchto vrcholů zjištěna jejich vzdálenost od pozice haptického pera, a pokud je menší než daný poloměr, je pro náležející trojúhelník proveden výpočet průsečíku s paprskem pera.

3.5.7 Vylepšené řešení

Druhý způsob řešení detekce kolizí spočívá ve výpočtu vzdáleností těžišť všech trojúhelníků od polohy haptického pera.

Výpočet těžišť trojúhelníků

Těžiště všech trojúhelníků jsou spočítána pouze jednou na začátku aplikace při načítání modelu, a to podle vzorce:

$$T = \frac{V0 + V1 + V2}{3} \quad , \tag{3.4}$$

kde $V0$, $V1$ a $V2$ jsou vrcholy daného trojúhelníku.

Vzdálenost bodů

Poté je pro každý trojúhelník spočítána vzdálenost jeho těžiště od pozice haptického pera podle vzorce pro Eukleidovskou vzdálenost:

$$D = \sqrt{(T_x - P_x)^2 + (T_y - P_y)^2 + (T_z - P_z)^2} \quad , \tag{3.5}$$

kde T je těžiště trojúhelníku a P poloha pera.

Ze zjištěných vzdáleností je vybrána ta nejmenší. Pokud je tato hodnota menší než konstanta `EPS_T`, která je zde rovna 0.025, je pro daný trojúhelník vypočtena výsledná síla.

Výpočet parametru t

K výpočtu síly je opět potřeba vypočíst parametr t, který je zde určen jako kolmá vzdálenost haptického pera od roviny trojúhelníku. Abychom mohli tuto vzdálenost spočítat, je třeba nejprve vytvořit vektor z bodu trojúhelníku k bodu polohy haptického pera. Poté je již možné vzdálenost spočítat jako skalární součin vytvořeného vektoru s normálou roviny trojúhelníku. Velikost této vzdálenosti

bude rovna maximálně EPS_T. Aby se síla postupně zvyšovala i po proniknutí skrze model, kde je parametr t záporný, je k němu přičtena konstanta EPS_T, takže hodnota výsledného parametru je v rozmezí $\langle 0,\ 2\ \texttt{EPS\_T}\ \rangle$.

Výpočet normály

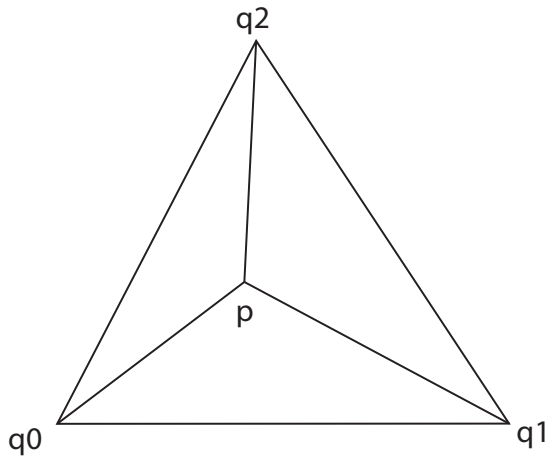
Aby byla hmatová odezva co nejhladší, je normála, která je použita k výpočtu směru síly, vypočtena pomocí barycentrických souřadnic trojúhelníku, které určují váhu normál všech tří vrcholů.

Nejprve je třeba vypočítat kolmý průmět polohy haptického pera do roviny trojúhelníku, tedy vlastně průsečík roviny s přímkou určenou pozicí pera a opačnou normálou roviny. Pro tento účel jsem použila již dříve použitou a implementovanou metodu pro výpočet průsečíku paprsku s trojúhelníkem.

Pokud výsledný průsečík neleží uvnitř trojúhelníku, je jako normála určující směr síly použita normála vrcholu, který leží nejblíže průsečíku. Pokud se však průsečík nachází uvnitř trojúhelníku, jsou spočteny barycentrické souřadnice podle vzorců:

$$\begin{aligned} a_0 &= \frac{A(p, q_1, q_2)}{A(q_0, q_1, q_2)} \\ a_1 &= \frac{A(p, q_2, q_0)}{A(q_0, q_1, q_2)} \\ a_2 &= \frac{A(p, q_0, q_1)}{A(q_0, q_1, q_2)} \quad , \end{aligned} \tag{3.6}$$

kde $A(p, q_i, q_j)$ je obsah trojúhelníku tvořeného vrcholy p, q_i a q_j, viz Obr.3.4. Jiný výpočet barycentrických souřadnic než v kapitole 3.5.6 je zde zvolen z důvodu, že je potřeba tří souřadnic, z nichž každá je použita jako váhový koeficient normály jednoho z vrcholů trojúhelníka při výpočtu výsledné normály. Výsledkem předchozího výpočtu barycentrických souřadnic byly pouze dvě souřadnice a sloužily jako váhové koeficienty vektorů hran trojúhelníku pro určení polohy bodu uvnitř nebo vně trojúhelníku.



Obrázek 3.4: Ukázka rozdělení trojúhelníku pro výpočet barycentrických souřadnic

Tyto souřadnice určují relativní polohu průsečíku vzhledem k vrcholům trojúhelníku. Jsou to tedy váhové koeficienty, které jsou použity pro výpočet výsledné normály:

$$\overrightarrow{n} = a_0\overrightarrow{n_0} + a_1\overrightarrow{n_1} + a_2\overrightarrow{n_2} \tag{3.7}$$

Výpočet síly

Silová zpětná vazba je spočtena podle vzorce (s max. velikostí síly 1.6 N):

$$\overrightarrow{F} = \overrightarrow{n}.(2EPS\_T - t)^2.\frac{1}{(2EPS\_T)^2}.1.6 \quad , \tag{3.8}$$

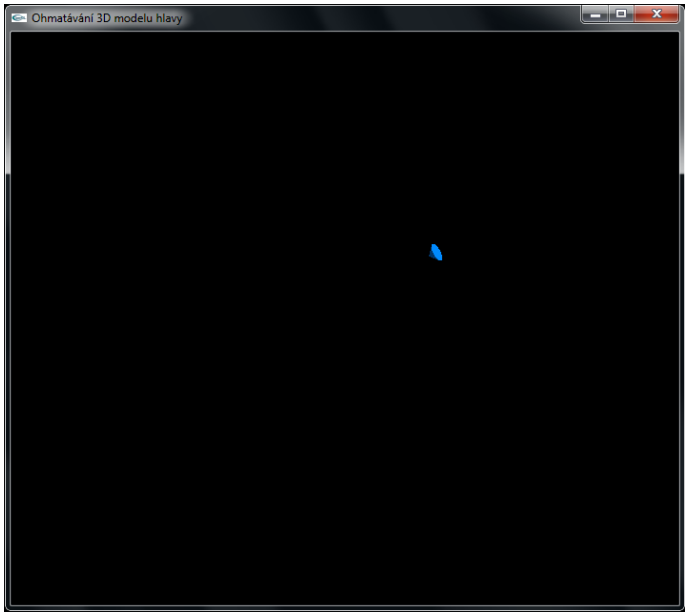
kde $\overrightarrow{n}$ je normála daného trojúhelníku a $2EPS\_T$ je maximální hodnota parametru t.

Optimalizace

Aby byl výpočet síly rychlejší, je v programu provedeno následující vylepšení. Pokud není žádná z vypočítaných kolmých vzdáleností haptického pera od roviny trojúhelníku menší než `EPS_T`, přičte se k iteracím, kdy se nepočítá nová síla, 200 iterací. Po spočítání síly jsou také přidány tři další iterace, během kterých se nepočítá nová síla.

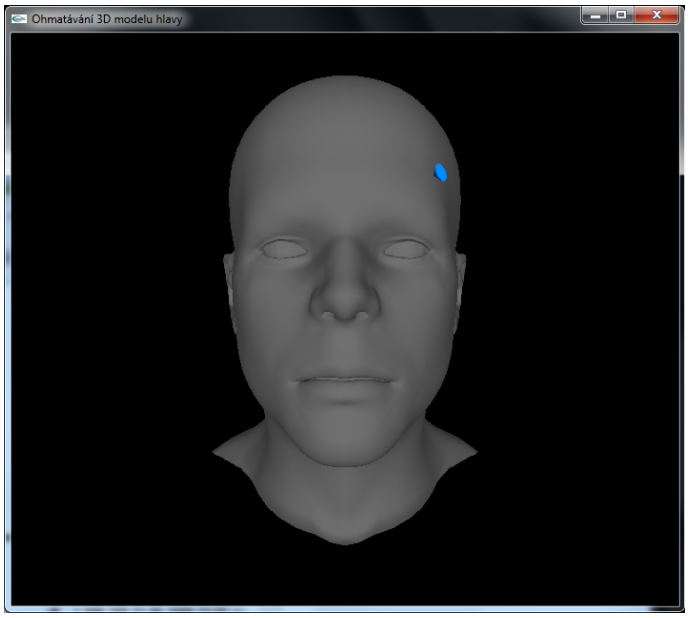
4 Uživatelské testy

Výsledky své práce jsem předložila k otestování skupině uživatelů, kteří zároveň vyplnili připravený dotazník - viz příloha. Testy zahrnovaly převážně ukázky z vyšší knihovny HLAPI. Nejprve byl uživatelům spuštěn program bez grafické vizualizace, pouze s vykresleným 3D kurzorem, aby měl uživatel přibližnou představu, v jaké části OpenGL projekce se haptické pero nachází - viz Obr.4.1. V tomto programu byl buď hapticky vyrenderován deformovaný elipsoid nebo lidská hlava s obličejem. Uživatelé měli poznat, zda je ohmatávaný model modelem hlavy či ne, a určit, do jaké míry je to poznat.



Obrázek 4.1: Ukázka aplikace určené pro uživatelské testy bez grafické vizualizace, pouze s vykresleným 3D kurzorem

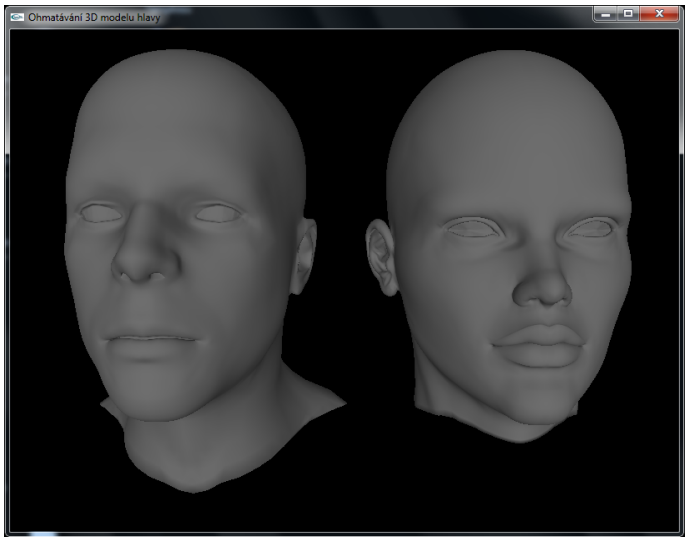
V další části testu již byl uživateli zobrazen model hlavy a uživatel měl na základě hmatového dojmu určit, zda má tento model vlasy a vousy - viz Obr.4.2. Pro každý z těchto materiálů pak byla v dotazníku položena otázka, na kolik je materiál podobný realitě.



Obrázek 4.2: Ukázka aplikace určené pro uživatelské testy, kde je zobrazen a ohmatáván model lidské hlavy

Dále byla uživatelům najednou spuštěna dvě okna. V jednom byly zobrazeny dva modely, model hlavy muže a model hlavy ženy - viz Obr.4.3. V druhém okně byl zobrazen pouze kurzor a byl zde hapticky vyrenderován jeden ze zobrazených modelů z prvního okna. Uživatelé měli poznat, který ze zobrazených modelů ohmatávají, a opět určit, nakolik je to poznat.

Poslední částí uživatelského testu bylo ohodnocení věrohodnosti ohmatávání modelu hlavy v aplikaci vytvořené za použití nižší knihovny HDAPI a detekce kolizí pomocí počítání těžišť.

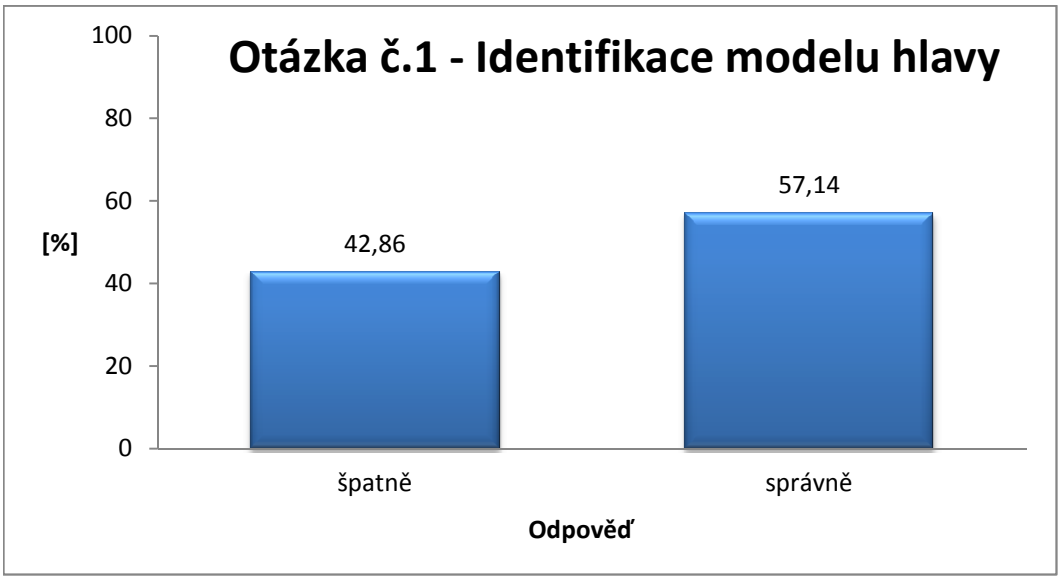


Obrázek 4.3: Ukázka aplikace určené pro uživatelské testy, kde jsou zobrazeny modely hlav muže a ženy

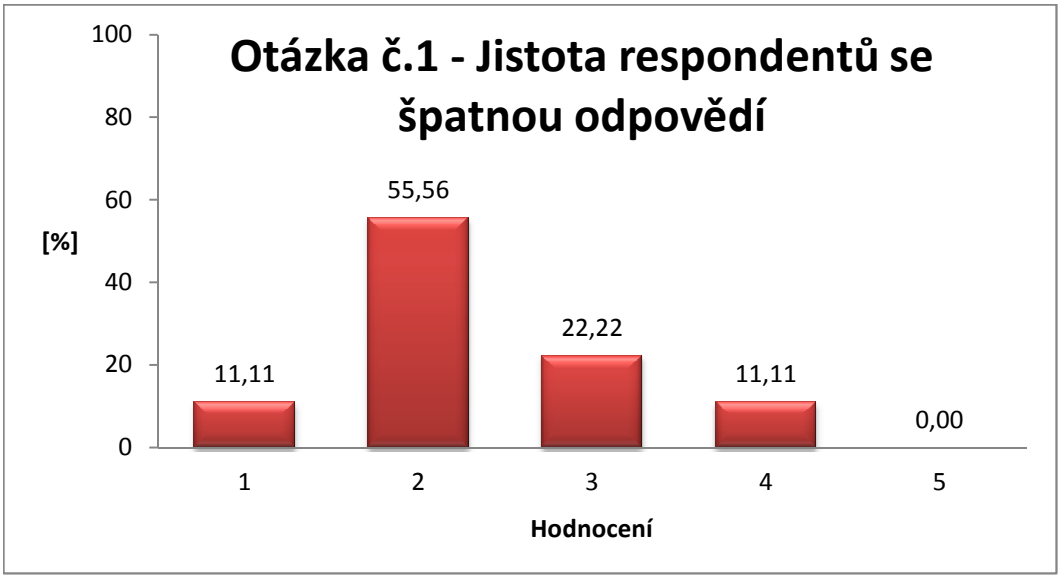
4.1 Výsledky

Uživatelských testů se zúčastnilo celkem 21 uživatelů. Každý účastník vyplnil připravený dotazník. Vyhodnocením všech dotazníků jsem dospěla k následujícím závěrům.

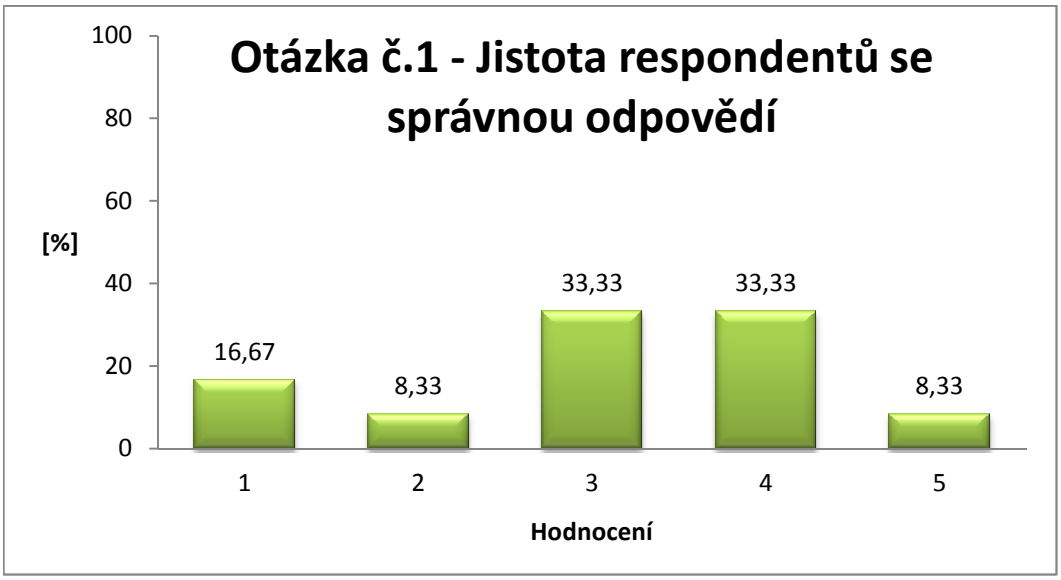
V první aplikaci, kdy uživatelé neviděli ohmatávaný model, byl virtuálním objektem deformovaný elipsoid. Poznalo to přibližně 57% všech dotázaných - viz Obr.4.4. Takto malý počet správných odpovědí může být způsoben tím, že někteří uživatelé si pod modelem lidské hlavy mohli představovat pouze její tvar bez přítomnosti obličeje. Druhou otázku k této aplikaci, kde měli uživatelé určit, do jaké míry je poznat, co ohmatávají, jsem vyhodnotila zvlášť pro uživatele se správnou a se špatnou odpovědí, aby bylo zřejmé, která skupinka si svou odpovědí byla více jistá. Z Obr.4.5 a Obr.4.6 je patrné, že většina uživatelů se špatnou odpovědí ohodnotila míru své jistoty číslem 2. Naproti tomu uživatelé se správnou odpovědí většinou zakroužkovali číslo 3 nebo 4, z čehož je možné usoudit, že si svou odpovědí byli o něco jistější.



Obrázek 4.4: Při ohmatávání deformovaného elipsoidu dosahovala úspěšnost přibližně 57%

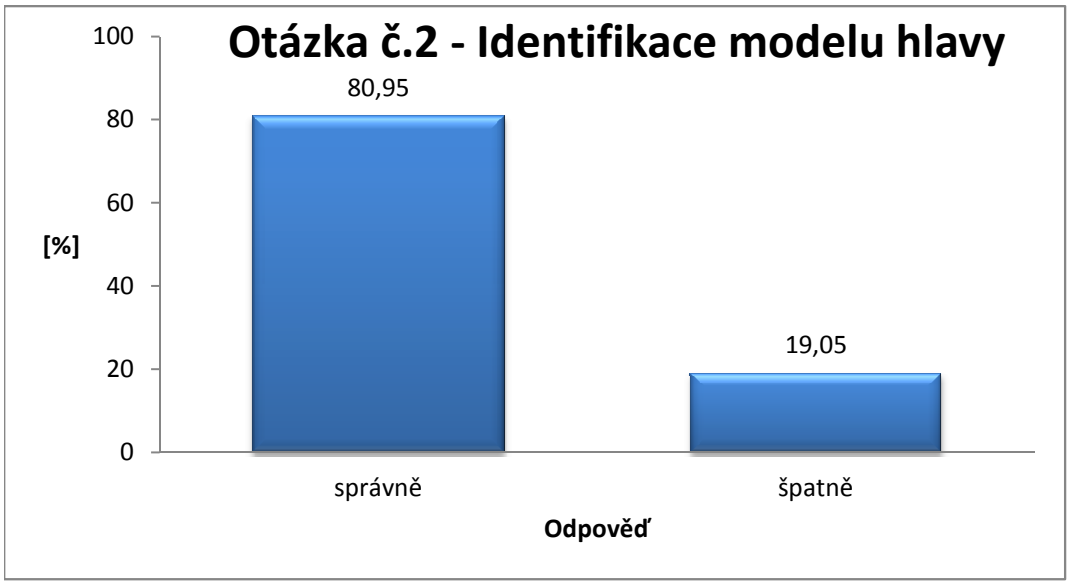


Obrázek 4.5: Percentuální rozdělení respondentů podle míry jejich jistoty při špatné odpovědi na první otázku, přičemž *1 - málo*, *5 - velmi*

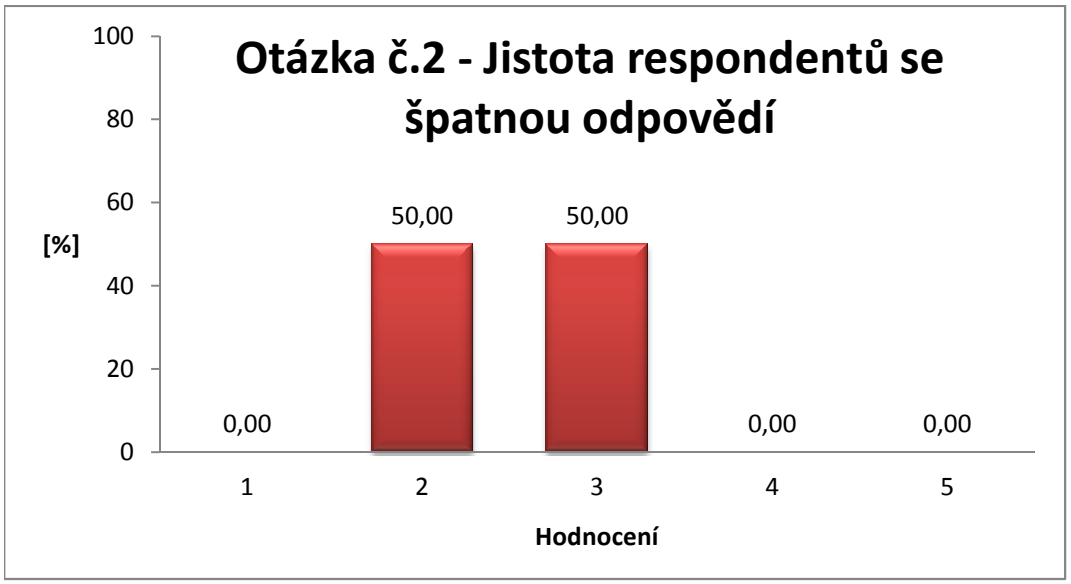


Obrázek 4.6: Percentuální rozdělení respondentů podle míry jejich jistoty při správné odpovědi na první otázku, přičemž *1 - málo*, *5 - velmi*

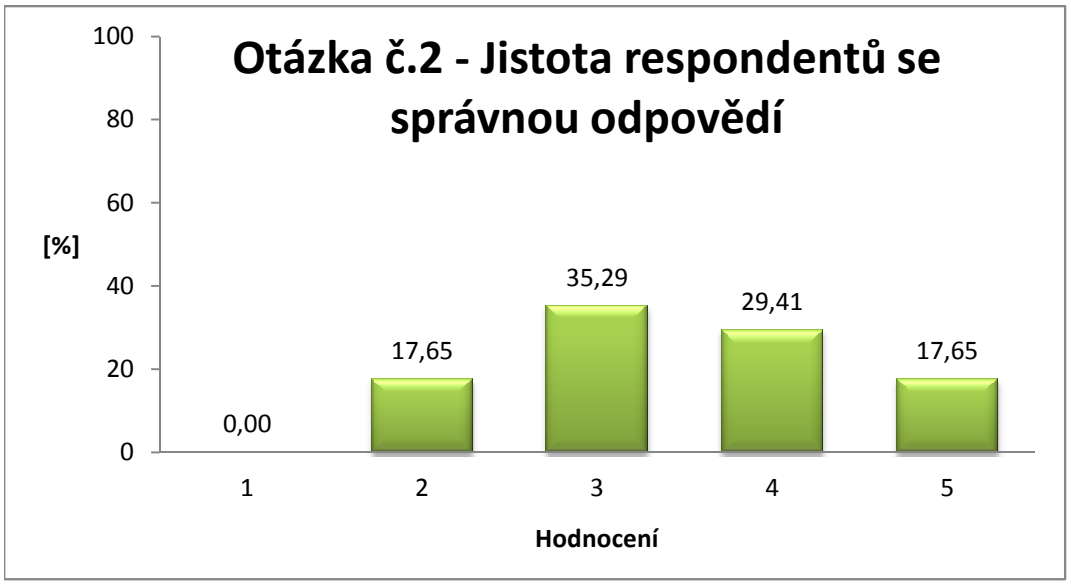
Z výsledků otázky týkající se druhé aplikace, která byla úplně stejná jako ta první s jediným rozdílem, že uživatelé tentokrát ohmatávali model lidské hlavy, je patrné, že model poznalo již více uživatelů, a to téměř 81% - viz Obr.4.7. Velký rozdíl výsledků první a druhé aplikace dokazuje, že většina uživatelů si již u druhého modelu uvědomila, že součástí modelu lidské hlavy by měl být i obličej. Stejně jako u výsledků první aplikace jsem i zde rozdělila vyhodnocení otázky, kde měli uživatelé ohodnotit, do jaké míry je to poznat, mezi uživatele se špatnou a se správnou odpovědí. I zde můžeme usoudit na základě Obr.4.8 a Obr.4.9, že uživatelé, kteří odpověděli správně, si svou odpovědí byli více jisti, než ti, kteří nepoznali, že jde o model lidské hlavy.



Obrázek 4.7: Při ohmatávání modelu hlavy dosahovala úspěšnost téměř 81%

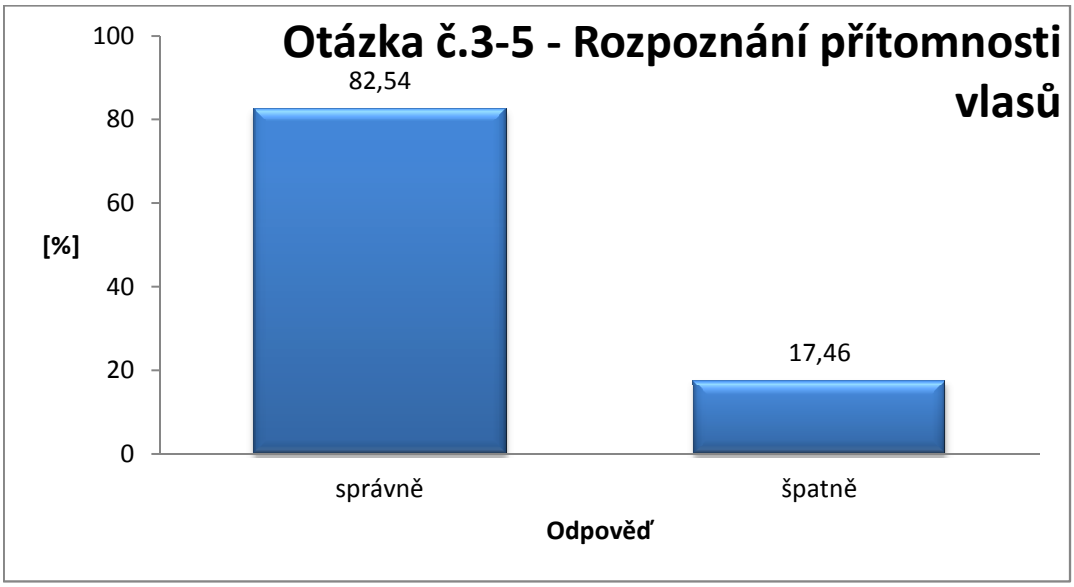


Obrázek 4.8: Percentuální rozdělení respondentů podle míry jejich jistoty při špatné odpovědi na druhou otázku, přičemž *1 - málo*, *5 - velmi*

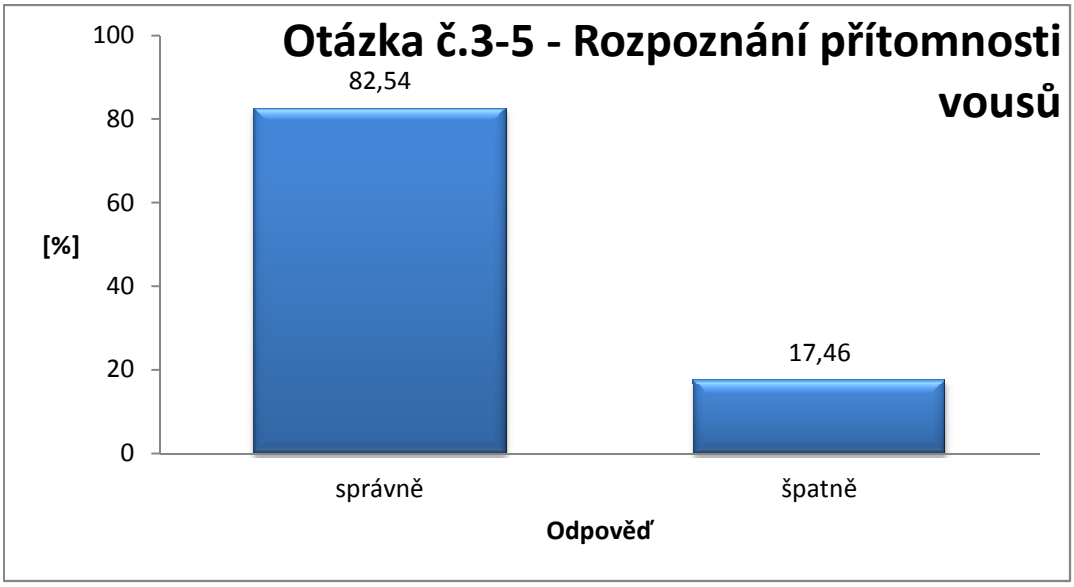


Obrázek 4.9: Percentuální rozdělení respondentů podle míry jejich jistoty při správné odpovědi na druhou otázku, přičemž *1 - málo*, *5 - velmi*

U třetí až páté aplikace měli uživatelé za úkol poznat, zda má ohmatávaný model vlasy a vousy. Výsledky jsem shrnula tak, že jsem uživatele rozdělila podle toho, zda u všech tří aplikací správně rozeznali přítomnost vlasů či vousů nebo ne. Výsledky jak pro vlasy tak pro vousy jsou překvapivě podobné. U všech tří aplikací poznalo správně přítomnost vlasů téměř 83% a vousů taktéž téměř 83% respondentů - viz Obr.4.10 a Obr.4.11.

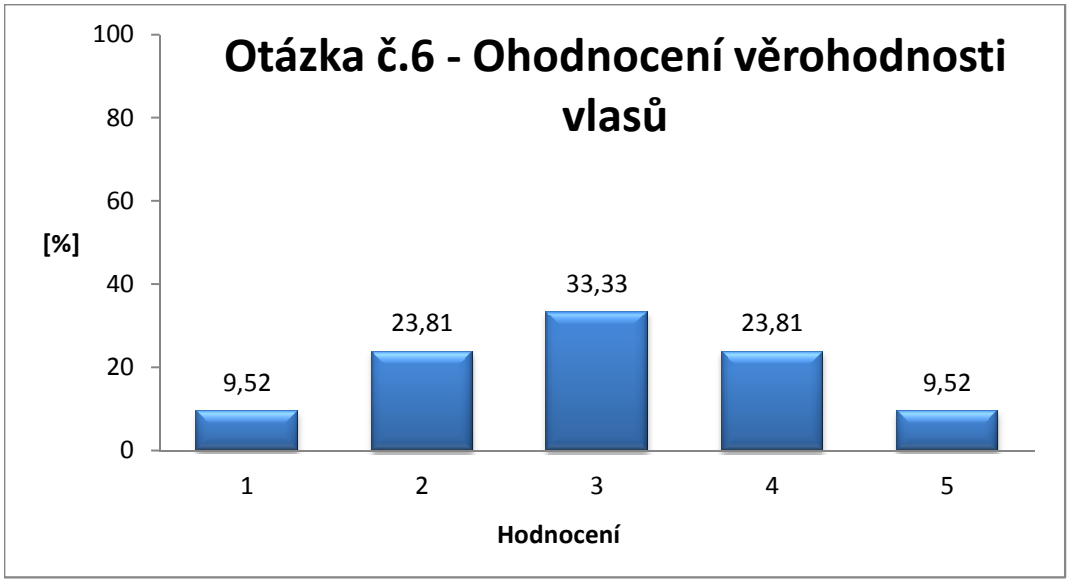


Obrázek 4.10: Přítomnost vlasů na modelu hlavy poznalo téměř 83% respondentů

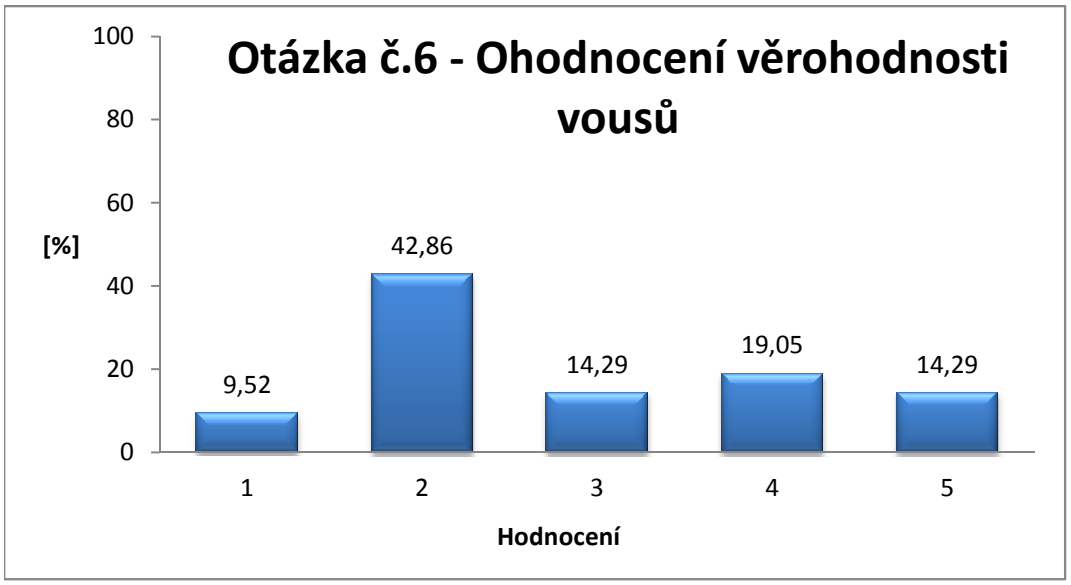


Obrázek 4.11: Přítomnost vousů na modelu hlavy poznalo téměř 83% respondentů

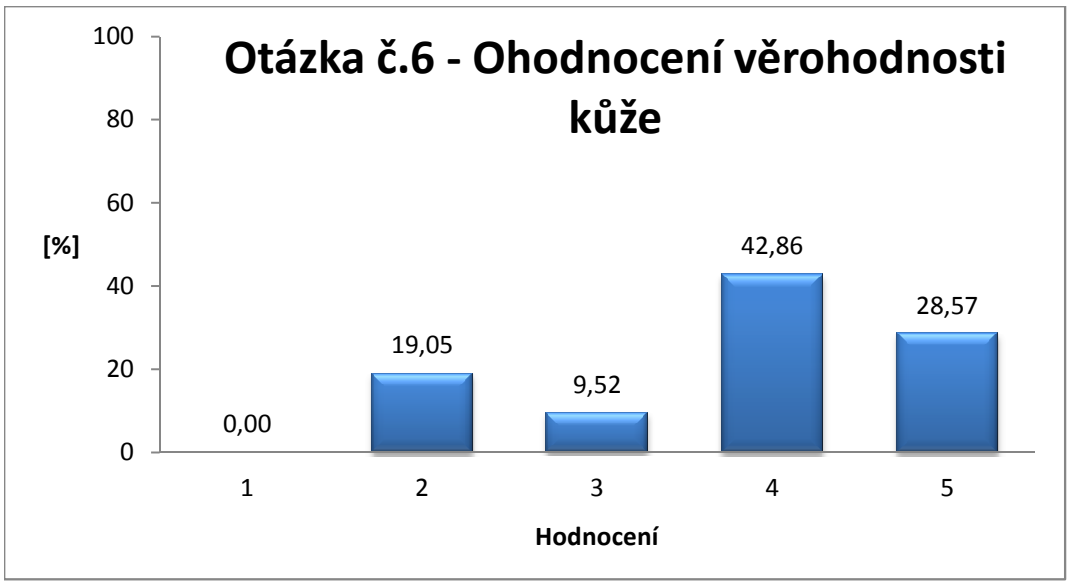
V šesté aplikaci měli uživatelé ohodnotit věrohodnost jednotlivých materiálů. Nejlépe byla hodnocena kůže - viz Obr.4.14. Výsledky hodnocení vlasů se podezřele podobají normálnímu rozdělení - viz Obr.4.12. Nejhůře však dopadlo ohodnocení vousů - viz Obr.4.13. Myslím, že se zde není čemu divit, jelikož materiály bylo možné nastavit pouze měněním čtyř různých vlastností.



Obrázek 4.12: Percentuální rozdělení respondentů podle hodnocení věrohodnosti vlasů, přičemž *1 - nejméně věrohodné*, *5 - nejvíce věrohodné*

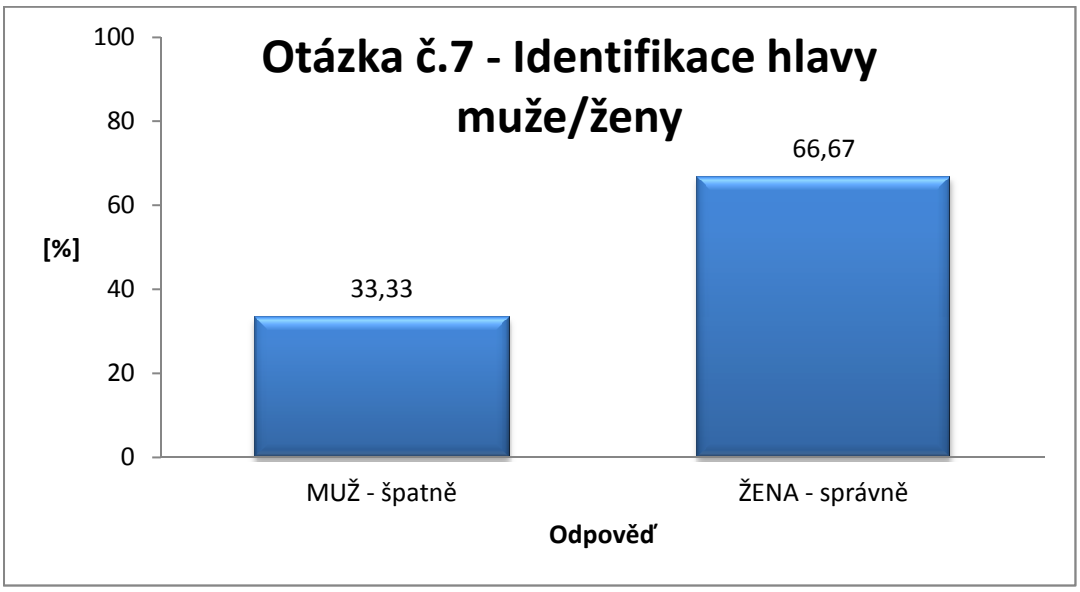


Obrázek 4.13: Percentuální rozdělení respondentů podle hodnocení věrohodnosti vousů, přičemž *1 - nejméně věrohodné*, *5 - nejvíce věrohodné*

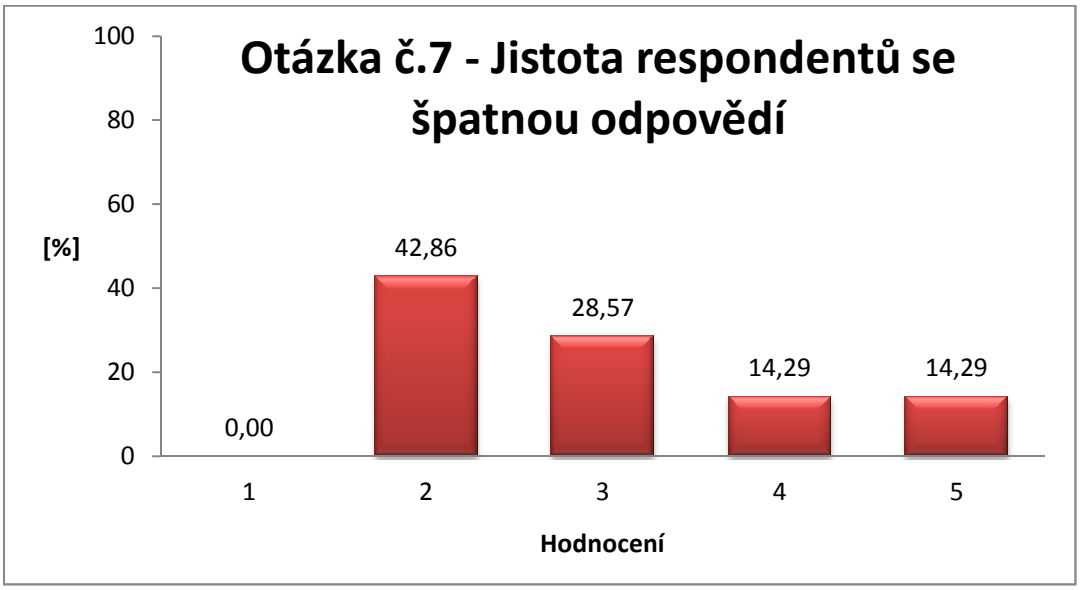


Obrázek 4.14: Percentuální rozdělení respondentů podle hodnocení věrohodnosti kůže, přičemž *1 - nejméně věrohodné*, *5 - nejvíce věrohodné*

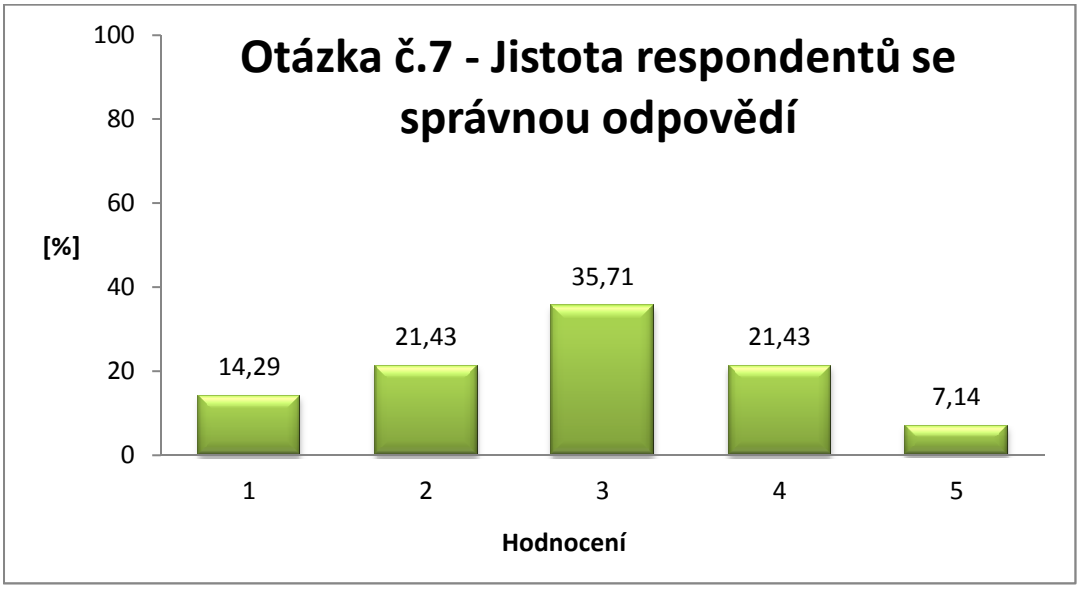
U sedmé aplikace uživatelé určovali na základě obrazového podkladu, zda ohmatávají hlavu muže nebo ženy. Správnou odpovědí bylo, že virtuálním modelem je model ženy. Poznalo to téměř 67% respondentů. Z Obr.4.16 a Obr.4.17 však vyplývá, že ať odpověděli uživatelé správně či špatně, ani v jednom případě si tím nebyli většinou příliš jisti.



Obrázek 4.15: Při ohmatávání modelu ženské hlavy dosahovala úspěšnost téměř 67%

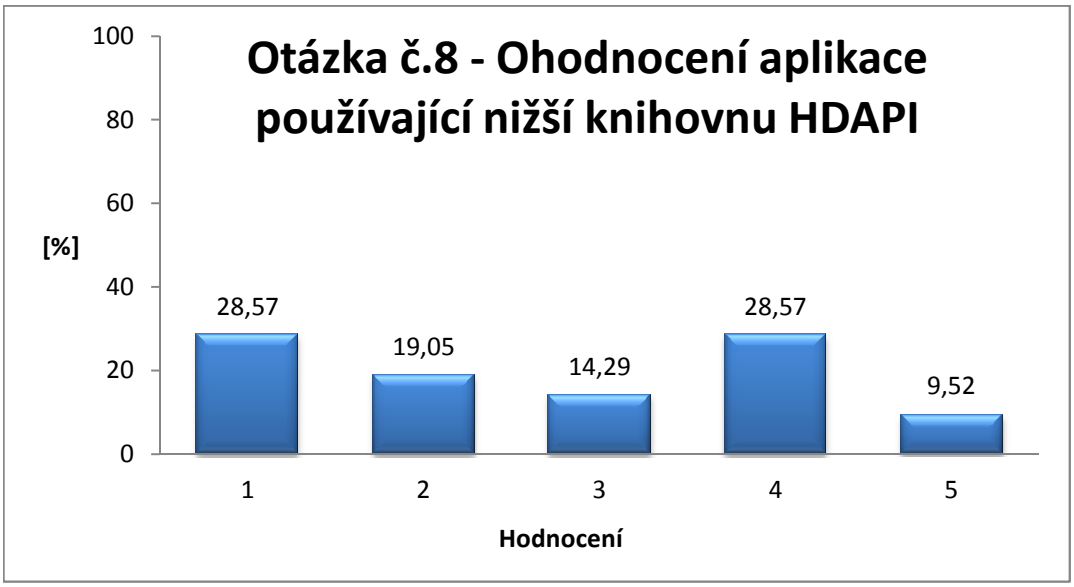


Obrázek 4.16: Percentuální rozdělení respondentů podle míry jejich jistoty při špatné odpovědi na sedmou otázku, přičemž *1 - málo*, *5 - velmi*



Obrázek 4.17: Percentuální rozdělení respondentů podle míry jejich jistoty při správné odpovědi na sedmou otázku, přičemž *1 - málo*, *5 - velmi*

U osmé aplikace uživatelé hodnotili věrohodnost aplikace používající nižší knihovnu HDAPI. Z výsledků není patrné, že by se uživatelé na ohodnocení shodovali - viz Obr.4.18.



Obrázek 4.18: Percentuální rozdělení respondentů podle hodnocení věrohodnosti aplikace používající knihovnu HDAPI, přičemž *1 - nejméně věrohodné*, *5 - nejvíce věrohodné*

5 Závěr

Výsledkem mé práce jsou tři různé aplikace pro ohmatávání 3D modelů hlav. Jedna za použití knihovny HLAPI a druhé dvě za použití nižší knihovny HDAPI pouze s rozdílnou detekcí kolizí a výpočtem silové zpětné vazby. Po srovnání těchto aplikací jsem došla k závěru, že nejvěrohodnějšího ohmatávání dosahuje první aplikace. Je to dáno tím, že jsou v HLAPI již vestavěné efektivní algoritmy pro detekci kolizí a výpočet síly. Proto jsem se mohla věnovat i nastavení vlastností materiálů. Nižší knihovna má podle mého názoru však vysoký potenciál v tom, že je zde možné nastavení vlastní síly, a tedy snad i věrohodnější hmatové odezvy materiálů.

Aplikace vytvořené v nižší knihovně se také velmi liší. V programu obsahujícím výpočet těžišť je haptická smyčka mnohem rychlejší, probíhá proto častěji a výsledný dojem silové zpětné vazby je mnohem spojitější než u druhé aplikace.

Výslednou aplikaci v nižší knihovně by bylo možné pro dokonalejší hmatovou odezvu vylepšit následovně. Bylo by třeba zde vytvořit systém, který nedovolí uživateli proniknout haptickým perem dovnitř modelu. Vhodným řešením by byla implementace systému pružiny a tlumiče, jako je tomu u vyšší knihovny HLAPI, viz kapitola 3.1.3.

Seznam obrázků

Reference

[1] CHAI 3D. [online]. 2012. [cit. 2013-05-10]. Dostupné na: http://www.chai3d.org/

[2] Force Dimension. *Chai3D overview* [online]. 2012. [cit. 2013-01-03]. Dostupné na: http://www.forcedimension.com/chai3d-overview

[3] Senasable Technologies. *PHANTOM Omni haptic device* [online]. 2012. [cit. 2013-01-03]. Dostupné na: http://www.sensable.com/haptic-phantom-omni.htm

[4] Senasable Technologies. *OpenHaptics Toolkit* [online]. 2013. [cit. 2013-05-10]. Dostupné na: http://geomagic.com/en/products/open-haptics/overview

[5] MARTÍNEK Petr. *Programové vybavení pro sestavování identikitů*, diplomová práce, Západočeská univerzita v Plzni, 2012

[6] WEISSTEIN, Eric W. *Quicksort.* [online]. 2013 [cit. 2013-05-10]. Dostupné na: http://mathworld.wolfram.com/Quicksort.html

[7] SUNDAY Dan. *Intersections of Rays and Triangles(3D).* [online]. 2013 [cit. 2013-05-10]. Dostupné na: http://geomalgorithms.com/a06-_intersect-2.html

[8] SensAble Technologies. *Openhaptics toolkit - Programmer's guide* [online]. 2013 [cit. 2013-05-10]. Dostupné na: http://www.geomagic.com/files/4013/4851/4367/OpenHaptics_ProgGuide.pdf

[9] CASILLAS Miguel. *Half-Space Test* [online]. 2010-06-15 [cit. 2013-05-10]. Dostupné na: http://www.miguelcasillas.com/?p=43

[10] MÁLKOVÁ Martina. *Multimorphing - Barycentrické souřadnice v n-úhelníku* [online]. 2013 [cit. 2013-05-10]. Dostupné na: http://home.zcu.cz/ mmalkov/index.php?stranka=multimorphing

A Příloha - dotazník

Věk: …. Pohlaví: M – Ž Student: ☐

Je váš obor informatika? ☐ ANO ☐ NE

Vyšší knihovna HL

1) Je ohmatávaný model modelem lidské hlavy? ☐ ANO ☐ NE

   Do jaké míry je to poznat? : 1 2 3 4 5
   (1 – málo, 5 – velmi)

2) Je ohmatávaný model modelem lidské hlavy? ☐ ANO ☐ NE

   Do jaké míry je to poznat? : 1 2 3 4 5
   (1 – málo, 5 – velmi)

3) Má ohmatávaný model hlavy vousy? ☐ ANO ☐ NE

   Má ohmatávaný model hlavy vlasy? ☐ ANO ☐ NE

4) Má ohmatávaný model hlavy vousy? ☐ ANO ☐ NE

   Má ohmatávaný model hlavy vlasy? ☐ ANO ☐ NE

5) Má ohmatávaný model hlavy vousy? ☐ ANO ☐ NE

   Má ohmatávaný model hlavy vlasy? ☐ ANO ☐ NE

6) Ohodnoťte věrohodnost materiálů (*1 – nejméně věrohodné, 5 – nejvíce věrohodné*):
   Vlasy: 1 2 3 4 5
   Vousy: 1 2 3 4 5
   Kůže: 1 2 3 4 5

7) Který ze zobrazených modelů myslíte, že ohmatáváte, mužskou nebo ženskou hlavu?

   ☐ MUŽE ☐ ŽENU

   Do jaké míry je to poznat? : 1 2 3 4 5
   (1 – málo, 5 – velmi)

Nižší knihovna HD

8) Ohodnoťte ohmatávání modelu hlavy (*1 – nejméně věrohodné, 5 – nejvíce věrohodné*):
   1 2 3 4 5