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

MANIPULACE S OBJEKTY POMOCÍ P5 GLOVE

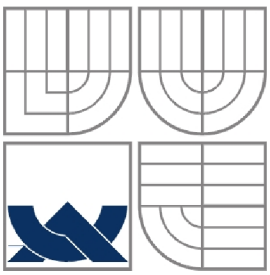
OBJECT MANIPULATION USING P5 GLOVE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

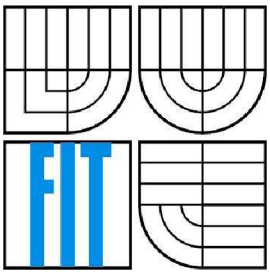
AUTOR PRÁCE
AUTHOR

Bc. RADOVAN ČAPEK

BRNO 2011



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

MANIPULACE S OBJEKTY POMOCÍ P5 GLOVE

OBJECT MANIPULATION USING P5 GLOVE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADOVAN ČAPEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL ŽÁK

BRNO 2011

Abstrakt

Práce se zaměřuje na popis základních principů virtuální reality, na využití datových rukavic ve virtuální realitě a implementuje aplikaci s využitím vstupního zařízení P5 glove. Implementace obsahuje mimo jiné různé vizuální techniky OpenGL a také ukazuje práci v současnosti s populárním fyzikálním enginem Bullet Physics. Teoretické základy těchto technik jsou v práci rozebrány. Výstup práce tak poslouží jako zdroj informací jak zájemcům o znalost problematiky datových rukavic a jim podobných periférií, tak i zájemcům, kteří se chtějí dozvědět něco o realizaci počítačové grafiky a tvorbě fyzikálních modelů.

Klíčová slova

Bullet Physics, detekce kolizí, kloubní soustavy, OpenGL, P5 glove, počítačová grafika, skeletální animace, virtuální realita

Abstract

The thesis is focused on the description of the basic principles of the virtual reality, on utilization of data gloves in the virtual reality and it implements the application with usage of the input device P5 glove. The implementation contains among the others various visual techniques of OpenGL and also shows up-to-date work with the popular physics engine Bullet Physics. The theoretical bases of these techniques are analysed in the thesis. So the output of the thesis will serve as a source of information to people interested in the knowledge of the data gloves problems and other similar peripherals, but also to interested people who want to learn more about the realization of the computer graphics and creation of physicist models.

Keywords

Bullet Physics, collision detection, articulated structure, OpenGL, P5 glove, computer graphics, skeletal animation, virtual reality

Citace

Bc. Radovan Čapek: Manipulace s objekty pomocí P5 glove. Brno, 2011, diplomová práce, FIT VUT v Brně.

Manipulace s objekty pomocí P5 glove

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Pavla Žáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Rád bych poděkoval konzultantovi Ing. Miroslavu Švubovi za pomoc při vypracování této práce.

Obsah

1 Úvod

Počítače a výpočetní systémy používá stále více a více lidí ať již v zábavní či komerční sféře. Spolu s tím se mění i nároky a požadavky na uživatelská rozhraní. Člověku je nejbližší jemu přirozený pohyb v trojrozměrném prostředí a s tím spojené třírozměrné vnímání okolí, pohyb v prostoru a manipulace s objekty. Ve světě počítačů se tomuto modelu vnímání a chápání světa nejvíce blíží virtuální realita.

Systémy virtuální reality se většinou skládají z podobných vstupních a výstupních zařízení. Často se jedná o přilbu pro virtuální realitu zajišťující zrakové a sluchové vjemy a datovou rukavici sloužící jako vstupní, v případě zpětné vazby, i výstupní zařízení. Právě použití datové rukavice je člověku nejpřirozenější způsob interakce s virtuálním okolím.

Tato práce se zabývá studiem a implementací aplikace, která pro svůj vstup právě takové rukavice využívá. Aplikace na příkladech demonstruje možnosti užití rukavice P5glove. Práce zahrnuje společná témata z oblasti počítačové grafiky, kloubních soustav, detekcí kolizí a jiné.

Druhá kapitola dokumentu pojednává o pojmech a technikách užívaných ve virtuální realitě. Kapitola tři je zaměřená na kloubní soustavy a řešení přímé a inverzní kinematiky v informatice. Obsahem čtvrté kapitoly je návrh programových řešení aplikace a z teoretického pohledu popisuje některé poznatky a knihovny užité pro implantaci, jejímž stručným popisem se zabývá kapitola pět.

2 Virtuální realita

Slovo „virtuální“ je definováno jako zdánlivý, možný, potenciální, neskutečný nebo v tento okamžik neexistující. Pojem virtuální realita pak lze chápat jako počítačem realizované prostředí, které simuluje skutečné vnímání světa. Jinými slovy můžeme říct, že se jedná o uživatelské rozhraní napodobující skutečné vnímání prostoru. V poslední době je tento název používán především jako označení nového a velice perspektivního oboru, který se zabývá využitím různých moderních technických zařízení.

Podle možností interakce můžeme virtuální realitu rozdělit následovně: o simulaci pasivní se jedná, jestliže pozorovatel simulace nemůže průchod prostředím žádným způsobem ovlivnit, příkladem může být např. film promítaný stereoskopickými brýlemi. V aktivní aplikaci dovoluje virtuální realita uživateli prostředí zkoumat. Je možno se v prostředí volně pohybovat, chybí však jakákoliv hmatová zpětná vazba a možnost zasáhnout do takového světa. Využití těchto aplikací je především v prezentacích, vizualizacích a různých vědních oborech. Nejpokročilejším stupněm virtuální reality jsou aplikace interaktivní. V nich má uživatel možnost svým zásahem pomocí speciálních vstupních zařízení prostředí modifikovat, případně dostávat zpětné hmatové odezvy od okolí a interagujících předmětů.

2.1 Senzory

Senzor je zařízení, které měří určitou fyzikální veličinu a převádí ji na analogový nebo digitální signál[3].

2.1.1 Atributy senzorů

- *záběr a dosah*: záběr senzoru je oblast, kterou senzor pokrývá. Záběr senzoru se většinou udává ve stupních. Je důležitý např. u fotografování. Dosah senzoru je vzdálenost, do které může senzor přinášet spolehlivá měření.
- *přesnost a rozlišení:* přesnost určuje, jak správná jsou data naměřená senzorem. Rozlišení určuje po jakých krocích je senzor schopný měřit.
- *spotřeba:* spotřeba udává spotřebu energie určitého senzoru. Spotřeba je zvlášť důležitá pro mobilní zařízení, která nemohou být stále připojena do rozvodné sítě. Pasivní senzory mají většinou menší spotřebu než aktivní.
- *hardwarová spolehlivost:* Senzory mají často fyzická omezení své funkce, např. pokud napájení sonaru klesne pod 12V, jeho měření nebudou přesná.
- *velikost:* Velikost a hmotnost senzorů ovlivňuje jeho vhodnost pro použití v určitém typu zařízení. Jestliže má být simulace virtuální reality věrná, nesmí nás datový oblek, rukavice či

helma zásadně omezovat při pohybu (například první helma pro virtuální realitu byla tak těžká, že musela být uchycena na stropě místnosti).

- *výpočetní složitost:* výpočetní složitost udává, kolik kroků musí algoritmus nebo program provést. Výpočetní složitost při zpracování údajů ze senzorů se dnes stává méně důležité, kvůli velkému pokroku v oblasti procesorů. Může však být stále problémem v malých zařízeních.
- *spolehlivost interpretace:* návrhář musí brát v potaz, jak spolehlivý je senzor.
- *aktivní a pasivní:* aktivní senzor je např. sonar, pasivní senzor je např. CCD kamera.

2.1.2 Snímání polohy

V současné době se ke snímání polohy využívá pěti fyzikálních principů: mechanického, ultrazvukového, magnetického, optického a analýzy obrazu. Některé metody vyžadují speciální obleky, které však zároveň mohou poskytovat i zpětnou vazbu[4].

a) Mechanické snímání – Ivan Sutherland, který je považován za vynálezce přilbového displeje, navrhl a vyzkoušel dvě metody snímání polohy. První z nich využívá mechanickou konstrukci připojenou k přilbě a ke stropu. Údaje o pohybech ve spojích armatury jsou přenášeny do počítače, který na jejich základě aktualizuje zobrazení. Mechanické snímání polohy a směru je ze všech metod nejpřesnější, ale zároveň nejvíce omezuje uživatele ve volném pohybu.

b) Ultrazvuk – V tomto případě je na vrcholu displeje umístěn malý generátor ultrazvukových impulsů, které jsou zachycovány čtyřmi mikrofony na stropě. Signály do nich přicházejí s malým časovým zpožděním. Porovnáním signálů lze vypočítat polohu a natočení přilby. Tento systém poskytuje uživateli dostatek volnosti, i když nevyniká přesností. Jeho funkce se též zhoršuje, pokud se nachází mezi mikrofony a vysílačem překážka, nebo pokud je vysílač příliš vzdálen od mikrofonů.

c) Magnetické snímání – Mechanický ani ultrazvukový systém nelze použít ve stísněném a hlučném prostředí. Pro taková prostředí byla vyvinuta metoda magnetického snímání, která se nakonec také nejvíce rozšířila. Elektrický proud protékající cívkou vytváří magnetické pole ve směru osy cívky. Naopak v cívce, která je v magnetickém poli, se indukuje proud. Jeho velikost závisí na intenzitě pole a na poloze os magnetického pole a cívky. Pokud umístíme tři cívky tak, aby jejich osy byly vzájemně kolmé, a jiné tři cívky se budou pohybovat v jejich poli, bude se v těchto cívkách indukovat náboj, který závisí na poloze a natočení cívek. Z těchto údajů je možné spočítat polohu a směr soustavy cívek. I tento systém má však dva nedostatky: setrvačnost a nepřesnost. Setrvačnost je dána zpožděním mezi pohybem cívky přijímače a okamžikem zachycení a vyhodnocení této informace. Nepřesnost měření je dána použitím střídavých magnetických polí, které dnes většina magnetických sledovačů používá. Magnetické pole je zachycováno nejen cívkovou soustavou přijímače, ale i jinými vodivými objekty v blízkosti. V nich vyvolává vířivé proudy, které jsou zdrojem sekundárních

magnetických polí. Řešení tohoto problému jsou dvě. Zaprvé lze využít počítačové predikce sekundárních polí a jejich eliminace při zpracování signálů z přijímacích cívek. Druhým způsobem je použití stejnosměrného napětí.

d) Optické sledovače – V původních systémech byl na přilbě upevněn zdroj světla (LED dioda) a její pohyb byl snímán kamerami. U novějších systémů se využívá postupu opačného. Tento systém má také svá omezení: uživatel nesmí stát příliš blízko stěn a váha přilby přesahuje čtyři kilogramy. Toto je však vykoupeno vysokou citlivostí (kolem 2 cm) a vysokou rychlostí obnovení údajů (20x – 100x za sekundu).

e) Sledovače založené na zpracování obrazu – Tato metoda je ze všech výpočetně nejnáročnější, ale je také nejdokonalejší a pro uživatele nejjednodušší. Nemusí nosit na hlavě žádné vysílače ani přijímače signálu; jeho polohu sleduje jedna, nebo více kamer, které jsou na něj zaměřeny. Z těchto informací je možno získat všechny údaje o natočení a poloze hlavy a rukou. Počítačové zpracování obrazu je sice mimořádně náročné, ale vzhledem ke svým přednostem je tento systém nadějný pro použití ve virtuální realitě.

2.1.3 Detekce polohy prstů

Detekce polohy prstů je důležitá právě pro manipulaci s objekty ve scéně. K detekci se využívají speciální rukavice, které podobně jako v případě snímání polohy mohou pracovat na různých principech. Nejčastější jsou právě principy založené na měření mechanických, optických nebo elektromagnetických veličin.

Detekce prstů pomocí optických senzorů pracuje na principu měření změn intenzity světla při ohybu optického vlákna. Pro každý kloub může existovat samostatný senzor. U takových zařízení je nutná rekalibrace pro každého uživatele, která se provádí sejmutím krajních poloh (zaťaté pěsti, natažené prsty).

Časté je také využití Hallových sond pro měření Hallova napětí[1], z kterého se pak určuje úhel ohnutí v kloubech. Toto měření je přesnější než metody optické.

2.2 Výstupní zařízení

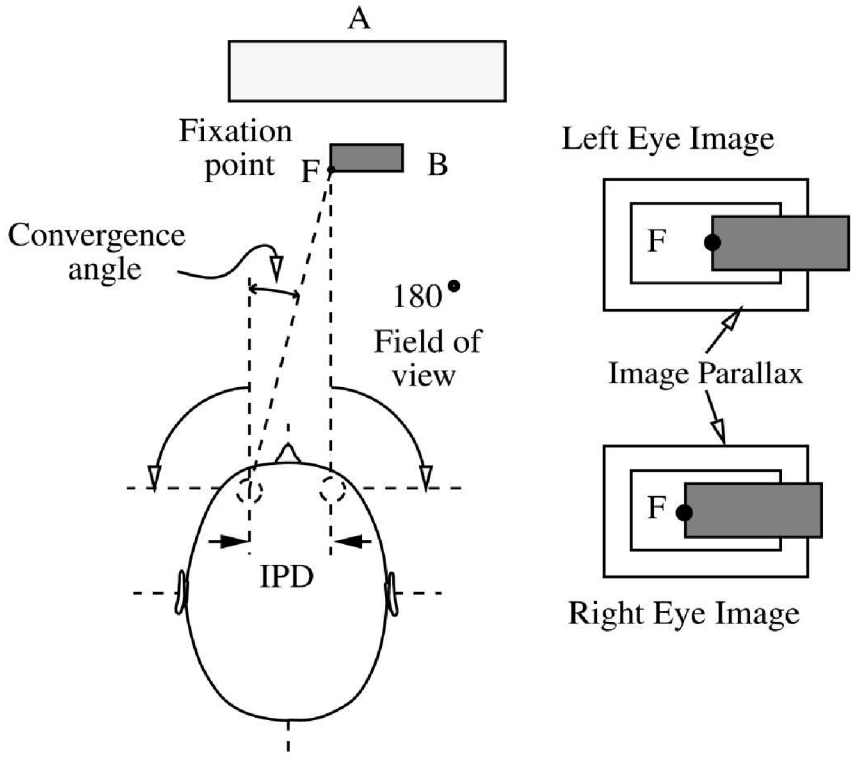
Cílem virtuální reality je poskytnout informace o 3D prostředí všem smyslům. Uspokojivě se to zatím daří pouze v případě zrakových a sluchovým podnětů, částečně i v případě hmatu. Pracuje se například i na vývoji zařízení pro reprezentaci čichových vjemů. Problém vývoje těchto zařízení ale spočívá v subjektivnosti vjemu zbylých dvou smyslů a nemožnosti měřit pach a chuť jako fyzikální veličinu.

---

[1] Hallovo napětí - vložíme-li vodivou destičku tloušťky d, kterou protéká řídící elektrický proud I, do magnetického pole s magnetickou indukcí B_y, kolmou na směr proudu, pak ve třetím směru, kolmém na směr proudu a zároveň na směr magnetického pole změříme potenciálový rozdíl rovný Hallovu napětí.

2.2.1 Zrak

Přibližně ¾ informací z okolí vnímáme pomocí zraku, proto je i vizuální zprostředkování nejdůležitější složkou scény virtuální reality. V dnešní době se používají nejvíce dva přístupy tvorby trojrozměrného obrazu. Oba využívají principu stereoskopie, disciplíny zabývající se zobrazením scény prostřednictvím dvou obrazů a simulující tak prostorový vjem.



Obrázek 1: Princip stereoskopie [28]

Nejčastěji se pro zobrazování využívá technologie Head Mounted Display. Jedná se o přílbu se dvěma zabudovanými obrazovkami, každá pro jedno oko. Pro každé oko je tak obraz vygenerován zvlášť. Vlivem schopnosti mozku spojit si dva posunuté obrazy vzniká prostorový vjem. Zpočátku byly zobrazovacími jednotkami CRT monitory, dnes se využívá především LCD nebo OLED displejů. Zvláštní zařízení v přilbě předává počítači informaci o poloze hlavy a očí. Dojde-li ke změně, musí počítač v reálném čase nový pár obrazů vygenerovat a zobrazit je na displejích[6].

Mnoho další systémů pro virtuální realitu využívá k promítání obrazů zdi místnosti. Takovéto systémy se nazývají CAVE (Cave Automatic Virtual Environments). Uživatelé CAVE jsou vybaveni speciálními brýlemi. Většinou se jedná o podobné brýle, za pomocí kterých se vytváří dojem 3D obrazu na běžných vysokofrekvenčních monitorech pomocí polarizačních filtrů. CAVE systémy jsou však finančně náročnější a mají větší prostorové nároky[5].

Ve vývoji je mnoho dalších různých technologií pro zobrazování 3D scény, například perspektivní VRD - Virtual Retinal Displays – technologii, která kreslí rastrový obraz přímo na sítnici oka, a jiné.

2.2.2 Sluch

Zvuk je ve virtuální realitě řešen podobně jako obraz. Do přilby jsou zabudována sluchátka jako nezávislý zdroj sluchu pro každé ucho. Požadavkem je dokonalý prostorový interaktivní zvuk. To znamená, že výsledný zvuk musí být dokonalou syntézou všech různých zdrojů ve scéně a současně musí interagovat s uživatelem. Především výsledný zvukový vjem musí korespondovat s natočením hlavy uživatele, s faktem, zda se ke zdroji zvuku přibližuje, nebo zda se od něho vzdaluje, či s odstíněním zvuku částí uživatelova těla[6].

V případě CAVE-like systémů využívající soustavu reproduktorů je vyřešení interaktivity s uživatelem mnohem jednodušší.

2.2.3 Hmat

Hmat často poskytuje důležité informace o povrchu objektu, teplotě nebo o tlaku, který je na objekt vyvíjen, je proto důležitý pro interaktivní systémy. Výzkum se soustřeďuje především na ruce. Přímo klíčový význam má hmat například v telerobotice[2], chirurgii nebo při simulaci řízení vozidel a proudových letadel. Základní princip spočívá ve vytváření protitlaku na kůži, který vyvolává pocit, jako bychom se dotýkali skutečných předmětů, nebo tlaku působícímu proti pohybu článků prstů či celé končetiny, pokud dochází ke kolizi s virtuálními objekty. Podle toho rozlišujeme modely dotykové a silové zpětné vazby. Dotyková zpětná vazba bývá simulována prostřednictvím vzduchových polštářků, slitinami s paměťovým efektem nebo vibračními elementy na prstech a dlani, silová zpětná vazba pak pomocí různých mechanických pomůcek, pístů apod. [4][7].

Systémy zprostředkovávající takovou vazbu se nazývají haptické. Výraz pochází z řečtiny a označuje první smysl, který se vyvine u lidského plodu[19]. Hmat je klíčovým smyslem pro interakci člověka s okolím. Využití haptického rozhraní ve virtuální realitě dává nový rozměr dříve pouze vizuálnímu řešení. Většina těchto aplikací používá hmatový rendering založený na dotykovém hrotu, díky němuž může uživatel komunikovat s virtuálním světem.

Kromě dotekového hrotu jsou v systémech virtuální reality hojně využívány datové obleky a datové rukavice. Toto vybavení umožňuje právě navození pocitu doteku a zajišťují silovou zpětnou vazbu. U mechanických systémů je velmi důležitá latence a snímací frekvence. Kontakt s pružným materiálem dovoluje jistou latenci, protože tělo uživatele do jisté míry umožňuje deformaci objektu z tohoto materiálu. Naproti tomu vysoká pracovní frekvence a velmi nízká latence je nezbytná při kontaktu s tuhým materiálem, kdy je potřeba uživatelovu vstupu do virtuálního předmětu včas zamezit. Zatímco lidský zrak se nechá ošálit již zrakovým vjemem kolem 25 snímků za vteřinu a považuje takový pohyb za pohyb plynulý, hmatové a vibrační změny jsme schopni zaznamenat a rozlišit až v řádu kHz.

---

[2] Telerobotika – obor robotiky zabývající se konstrukcí robotů ovládaných na dálku (především pomocí bezdrátových technologií jako Wi-fi, bluetooth, …)

2.3 Rozšířená realita

Rozšířená realita (Augmented reality) [8] je rychle se vyvíjejícím odvětvím v oboru virtuální realita. V literatuře se setkáme se dvěma hlavními definicemi: první úzce spojuje pojem rozšířené reality s hardwarovými prostředky (průhledové obrazovky v náhlavních systémech), druhá je obecnější a nevylučuje využití běžných zobrazovacích zařízení.

Steven Feiner spojuje pojem rozšířené reality výhradně s průhledovým displejem. Postupně se však začaly objevovat nové implementace a definice vázaná na hardware již nestačila. Ronald Azuma definuje rozšířenou realitu jako systémy mající následující tři vlastnosti:

- kombinují reálné a virtuální
- jsou interaktivní v reálném čase
- jsou registrované ve 3D

Takováto definice není vázána na konkrétní technologie, ale dostatečným způsobem ohraničuje pojem rozšířené reality. Dle této definice není rozšířenou realitou vkládání jednoduchých 2D informací (např. informace o stavu zápasu při sportovním utkání či jednoduchá navigace, která se zobrazuje na průhledových brýlích pilotovi letadla). Naopak popisky objektů, které by byly zobrazeny v prostoru, již vyžadují 3D zpracování, a proto můžeme mluvit o rozšířené realitě.

Paul Milgram definuje mimo pojmu rozšířené reality také pojem rozšířené virtuality (augmented virtuality). Je definována jako vkládání obrazů reálných objektů do virtuálního světa. Termínem zastřešujícím jak rozšířenou realitu, tak rozšířenou virtualitu, je pak smíšená realita (mixed reality). Milgram též rozpracovává pojem centricity, který můžeme přeložit jako zaměření. V případě egocentrického zobrazení je pozorovací bod (viewpoint) shodný s umístěním pozorovatele (například při použití průhledových displejů) a pozorovaná scéna se mění dle pohybu pozorovatele. Naopak v případě exocentrického zobrazení je bod pohledu fixní (například kalibrovaná kamera umístěná v místnosti) a scéna se mění v rámci zorného pole tohoto pohledu.

2.4 P5 glove

P5 glove[1][2] je periferní zařízení podobné rukavici, založené na snímání stupně ohnutí prstů a na sledování polohy pomocí infračervených paprsků. Díky tomu poskytuje rukavice naprosto intuitivní interakci s 3D virtuálním prostředím jako jsou hry, CAD aplikace, vzdělávací software a jiné. V běžných aplikacích funguje jako 2D polohovací zařízení podobně jako myš. Zařízení se skládá ze dvou částí: ze snímací věže a z datové rukavice. Datová rukavice je spojena s věží kabelem a nepotřebuje tak žádný vlastní zdroj napájení. Věž je připojena k počítači přes rozhraní USB. P5 glove je kompatibilní s Microsoft Windows XP a Mac OS verze 9. Existují také neoficiální ovladače pro Linux.

Původně se jednalo o projekt firmy Abrams Gentile Entertainment s kódovým označením „Gauntlet“. Design pochází z dílny společnosti Nytric. Firma se zabývá okrajovými technologiemi a technickými koncepty a pomáhá je realizovat. Za design P5 Essential Reality glove v roce 2001 získal designový tým cenu Solidworks 2001 Grand Prize. O prodej a marketing P5 glove se starala firma Essential Reality, která vznikla právě speciálně za účelem prodeje a distribuce rukavice P5. Vývoj započal v roce 2000, na trh se zařízení poprvé dostalo roku 2002 za cenu přibližně 200 amerických dolarů. Cílovým segmentem měl být herní průmysl a zákazníky hráčská komunita. Prvním tahákem pro hráče měla být podpora hry Hitman 2. I přes několik pozitivních recenzí z počítačových a herních magazínů byla podpora a výroba přerušena skrz problémy s nedostatečnou kompatibilitou programů a dalšími problémy.

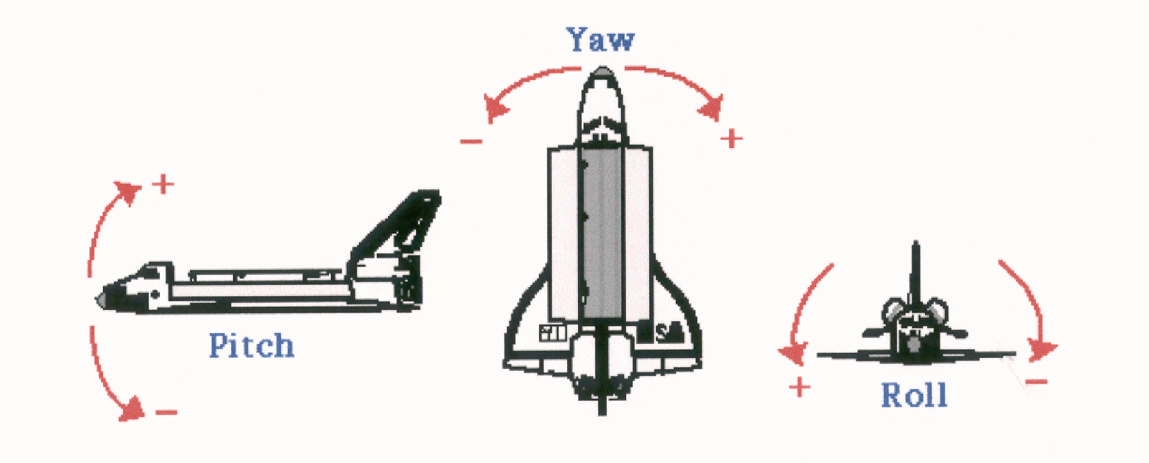
Kvůli špatným prodejním výsledkům nakonec došlo ke sloučení firmy Essential Reality se společností Alliance Distributer, která jakožto primární distributor P5glove firmu převzala a znovu začala s výrobou. Prodejcem se tak nově na krátký čas stala sesterská společnost Video Game Alliance, která toto zařízení prodávala za velmi příznivou cenu 50 dolarů. Po ní převzala prodej konkurenční společnost CyberWorld, která je jejím distributorem dodnes. Po té, co ustala distribuce ze strany Alliance Distributer, stal se CyberWorld oficiálním dodavatelem P5 glove. Současná oficiální cena činí 59 amerických dolarů, CyberWorld však často v minulosti poskytl značné velkoobchodní slevy[20].

Pokračovatelem se stala rukavice 5th Glove firmy Fifth Dimension Technologies, využívající optické detekce ohybu prstů a ultrazvukového snímání polohy, díky tomu má zařízení dosah přes dva metry od vysílače.

Obrázek 2: rukavice P5 glove[2]

2.4.1 Technické specifikace

Z mechaniky je známo, že poloha a orientace tělesa v prostoru je charakterizována šesti údaji. Většinou jsou to 3 hodnoty [x,y,z] souřadnic nějakého referenčního bodu tělesa v základním kartézském souřadném systému a 3 úhly [α, β, γ] natočení nějakého referenčního systému pevně s tělesem spojeného, vzhledem k tomuto základnímu souřadnému systému. Říkáme, že volné těleso má v prostoru 6 stupňů volnosti [3]. Proto polohovací zařízení P5 glove snímá pozici X, Y, Z kartézského souřadného systému a úhly náklonu ve 3 osách označovanými termíny Yaw, Pitch a Roll převzatými z letecké terminologie.



Obrázek 3: Typy náklonů umožňující 3 stupně volnosti[2]

Rukavice měří úhly v rozlišení po celých stupních s přesností +/- 1stupeň[3]. Osy X, Y, Z snímá s rozlišením 0.125 palce[4] a přesností 0.5 palce (asi 12.5mm) ve vzdálenosti 3 stopy[5]. Maximální dosah senzoru je 3-4 stopy, tedy něco mezi 90cm až 1.2m. Záběr senzoru je přibližně 45 stupňů ve všech osách.

Rukavice váží 0.13 kg a má pro každý prst nezávislý senzor míry ohnutí prstu v rozsahu 0-90 stupňů snímáno s rozlišením půl stupně. Obnovovací frekvence snímání polohy je 45Hz, obnovovací frekvence snímání ohybu prstů 60HzTato čísla jsou oficiální údaje uváděné výrobcem. Reálné rozsahy a parametry budou podrobněji rozebrány v kapitole pět.

---

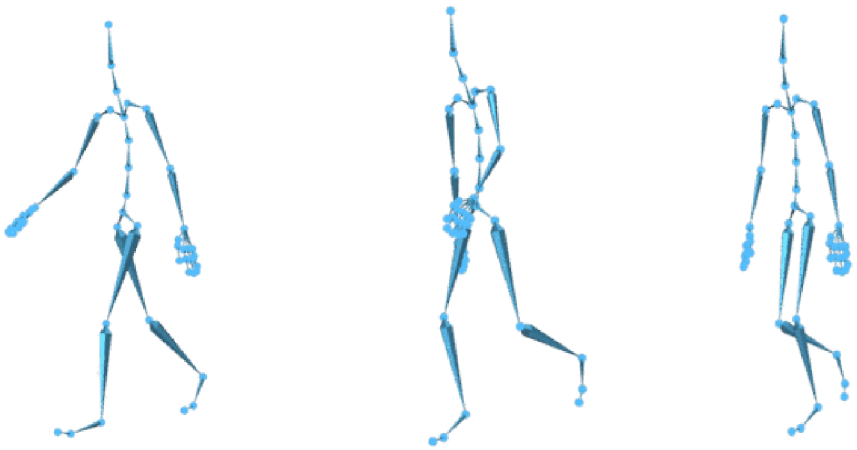
[3] zde se rozchází oficiální čísla a čísla uváděná v dokumentu k SDK, podle tohoto dokumentu má rozlišuje P5 glove po 3 stupních s přesností tří stupňů [25]

[4] Palec (inch) - stará americká a anglosaská jednotka pro měření délky. Definována přesně jako 25,4mm.

[5] Stopa (foot) - historická jednotka pro měření délky, existují různé definice její velikosti v závislosti na místě užívání. Jako platná jednotka má dnes význam jen angloamerická (imperiální) stopa o délce přesně 30,48cm

3 Kloubní soustavy

V počítačové grafice rozumíme pod pojmem kloubní soustavy komplexní model, který je možný animovat pomocí jeho kostry. Kostra modelu je stromová datová struktura složená z kloubů. Ty mohou provádět rotační nebo translační pohyb. Jejich možnosti pohybu definují počet stupně volnosti soustavy. Soustava má právě jeden kořen. Pro ostatní klouby platí, že jsou napojeny na rodičovský uzel (kloub) a jsou součástí podstromu daného uzlu. Toho se využívá při animaci. Proto bývá pojem kloubních soustav (articulated structure) často nahrazován pojmem skeletální animace a jeho anglickým ekvivalentem skeletal animation.



Obrázek 4: Ukázka kloubní soustavy a její animace [31]

3.1 Orientace tuhého tělesa v prostoru

3.1.1 Eulerovy kinematické rovnice

Chceme-li popsat natočení tuhého tělesa kolem jedné z os, potřebujeme k tomu tři nezávislé parametry. Počet těchto parametrů plyne ze symetrie matice ortogonality. Těmito parametry můžeme vytvořit soustavu tří úhlů (φ, θ, ψ) nazývanou Eulerovy úhly. Libovolné natočení tuhého tělesa lze složit ze tří otočení kolem vhodných os. Každé z těchto tří otočení je popsáno jedním z Eulerových úhlů[30]:

1. Precesní úhel φ - charakterizuje otočení kolem osy z kartézského systému souřadnic. Je z intervalu <0,2π). Určuje natočení tzv. uzlové přímky. Tuto rotaci popisuje matice **D**. $\vec{\Omega}^{D}$ je příslušný vektor úhlové rychlosti.

$$\mathbf{D} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

$$\vec{\Omega}^{D} = (0, 0, \varphi) \tag{2}$$

2. Nutační úhel θ - charakterizuje otočení kolem osy y', což je nová poloha osy x po předchozích otočeních. Je z intervalu <0,π). Určuje odchylku vlastní osy tuhého tělesa (kolem níž tuhé těleso rotuje) od svislého směru. Rotaci popisuje matice **C** s vektorem úhlové rychlosti $\vec{\Omega}^{C}$.

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \tag{3}$$

$$\vec{\Omega}^{C} = (\theta, 0, 0) \tag{4}$$

3. Rotační úhel ψ - charakterizuje otočení kolem nové polohy osy z. Je z intervalu <0,2π). Úhel φ a θ jednoznačně určují polohu osy tuhého tělesa, kolem níž těleso rotuje. Rotační úhel ψ pak určuje vlastní rotaci tuhého tělesa kolem jeho osy. Toto popisuje matice **B** a vektor úhlové rychlosti $\vec{\Omega}^{B}$.

$$\mathbf{B} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

$$\vec{\Omega}^{B} = (0, 0, \psi) \tag{6}$$

Výslednou rotaci můžeme popsat maticí A, která vznikne složením uvedených matic ve správném pořadí:

$$\mathbf{A} = \mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D} \tag{7}$$

Této matici pak odpovídá vektor úhlové rychlosti $\vec{\Omega}$.

$$\vec{\Omega} = \vec{\Omega}^{B}. + \mathbf{B}\cdot\vec{\Omega}^{C}. + \mathbf{B}\cdot\mathbf{C}\cdot\vec{\Omega}^{D} \tag{8}$$

Po dosazení a provedení patřičných úprav dostaneme vztah, který je maticovým vyjádřením Eulerových kinematických rovnic:

$$\begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} = \begin{pmatrix} \theta\cos\psi + \varphi\sin\theta\sin\psi \\ -\theta\sin\psi + \varphi\sin\theta\cos\psi \\ \psi + \varphi\cos\theta \end{pmatrix} \tag{9}$$

3.1.2 Gimbal lock

Gimbal lock[32] je název problému vznikajícího s užitím Eulerových úhlů. Protože výsledná rotační matice je závislá na pořadí násobení, rotace jedné osy se tak může mapovat na další rotační osy. Může tak nastat situace, kdy v požadované ose není možné objekt správně otočit. Toto se nazývá gimbal lock.

Předpokládejme, že objekt je otáčen podle os v pořadí z, y, x a že rotace v ypsilonové ose je 90 stupňů. V tomto případě se provede rotace kolem osy z jako první, a proto správně. Kolem osy y je těleso také natočeno správně. Po rotaci v ose y se však x-ová osa otáčí na ose z. Otáčíme-li tedy objektem v ose x, točí se ve skutečnosti v ose z. Řešením tohoto problému je užití quaternionů.

3.1.3 Quaterniony

Quaterniony [9][10] jsou zobecnění komplexních čísel v trojrozměrném prostoru rozšířením reálného čísla o tři imaginární složky.

Nechť H je množina quaternionů a q náleží H. Pak quaternion q může být zapsán jako uspořádaná čtveřice *(w, x, y, z)*. Častěji však bývá zapisován v úspornějším formátu *(w, v)*, kde *w* je skalární část a *v* je vektorová část. Pro počítačovou grafiku je nejvýznamnější jednotkový quaternion $|q| = 1$, který se dá zapsat jako:

$$q = (cos\ \Theta, \boldsymbol{v} \sin \Theta) \quad (10)$$

kde $\Theta \in (-\pi, \pi)$ a *v* je jednotkový vektor. Tento quaternion popisuje orientaci ve 3D prostoru. Jednotkový vektor *v* je osou rotace a Θ je dvojnásobek úhlu otočení kolem této osy. Platí:

$$\log(q) = \Theta * \boldsymbol{v} \quad (11)$$

Algebra tvoří ne-Abelovskou grupu $(H, +, .)$, kde + a . jsou operace scítání a násobení quaternionu. Nechť $q, q' \in H$, kde $q = [s, (x, y, z)]$ a $q' = [s', (x', y', z')]$. Operace sčítání je pak definovaná jako [33]:

$$q + q' \equiv [s\ , \boldsymbol{v}\ ] + [s', \boldsymbol{v}'] \equiv [s, (x, y, z)] + [s', (x', y', z')] \equiv$$

$$\equiv (s + \boldsymbol{i}\mathrm{x} + \mathbf{j}\mathrm{y} + \mathbf{k}\mathrm{z}) + (s' + \boldsymbol{i}\mathrm{x}' + \mathbf{j}\mathrm{y}' + \mathbf{k}\mathrm{z}') \quad (12)$$

V počítačové grafice využijeme především násobení quaternionů, které lze použít ke spojení dvou rotací vyjádřených quaterniony, a nalézt tak výslednou složenou rotaci. Máme-li opět quaterniony $q, q' \in H$ jako výše, pro jejich násobení platí:

$$qq' \equiv [s, \boldsymbol{v}][s', \boldsymbol{v}'] \equiv [s, (x, y, z)][s', (x', y', z')] \equiv (s + \boldsymbol{i}\mathrm{x} + \mathbf{j}\mathrm{y} + \mathbf{k}\mathrm{z})\,(s' + \boldsymbol{i}\mathrm{x}' + \mathbf{j}\mathrm{y}' + \mathbf{k}\mathrm{z}') =$$
$$ss' - (xx' + yy' + zz') + \boldsymbol{i}(sx' + s'x + yz' - zy') + \boldsymbol{j}(sy' + s'y + zx' - xz') + k(sz' +$$
$$s'z + sy' - yx' \equiv [ss' - \boldsymbol{v}\cdot\boldsymbol{v}', \boldsymbol{v}\times\boldsymbol{v}' + s\boldsymbol{v}' + s'\boldsymbol{v}] \qquad (13)$$

Máme-li jednotkový quaternion vyjadřující otočení tělesa v prostoru, můžeme ho následně převést do maticového tvaru:

$$\mathbf{Q} = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) & 0 \\ 2(xy + wz) & 1 - 2(x + z^2) & 2(yz + wx) & 0 \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (14)$$

Největší použití mají quaterniony při interpolaci orientace. Máme-li definovány dvě orientace pomocí transformačních matic M1 a M2, zjistíme, že možných interpolovaných orientací spojující trajektorii M1 a M2 je nekonečně mnoho, stejně jako je nekonečně mnoho možných trajektorií. Naproti tomu sférická interpolace dvou quaternionů dává nejkratší spojnici dvou quaternionů na povrchu hyperkoule[6]. V důsledku to znamená, že pohyb po takto vytvořené trajektorii zajišťuje minimální změnu úhlové rychlosti, a tím nejplynulejší spojnice. Postup při výpočtu trajektorie je následující:

1) Převedení transformačních matic na quaterniony
2) Interpolace
   a) Lineární interpolace mezi dvěma quaterniony:

$$Lerp(q_0, q_1, h) = q_0(1 - h) + q_1 h \qquad (15)$$

   b) Sférická lineární interpolace mezi dvěma quaterniony:

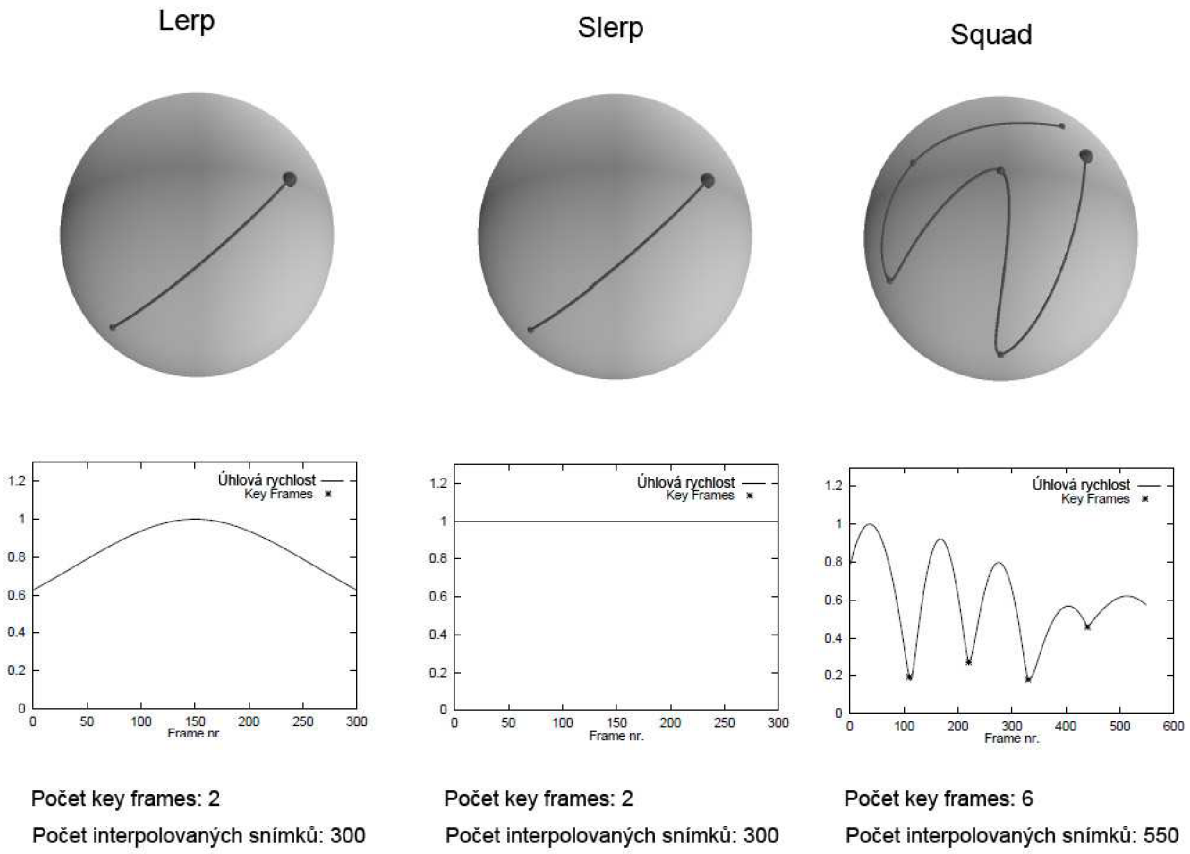
$$Slerp(q_1, q_2, u) = q_1 \frac{\sin(1-u)\Omega}{\sin\Omega} + q_2 \frac{\sin u\Omega}{\sin\Omega} \qquad (16)$$
$$\cos\Omega = q_1 q_2$$

c) Interpolace po hladké křivce mezi *n* quaterniony:

$$Squad(q_i, q_{i+1}, a_i, b_{i+1}, h) = Slerp(Slerp(q_i, q_{i+1}, h), Slerp(s_i, s_{i+1}, h), 2h(1-h)) \qquad (17a)$$
$$a_i = b_i = q_i \exp\left(\frac{-\ln\left(q_i^{-1} q_{i+1}\right) + \ln\left(q_i^{-1} q_{i-1}\right)}{4}\right) \qquad (17b)$$
$$q = (\cos\Theta, \sin\Theta\vec{v}),\ N(\vec{v}) = 1 \rightarrow \ln q = (0, \Theta\vec{v}) \qquad (17c)$$
$$q = (0, \Theta\vec{v}),\ N(\vec{v}) = 1 \rightarrow \exp q = (\cos\Theta, \sin\Theta\vec{v}) \qquad (17d)$$
$$q_i \in H, h \in [0,1]$$

3) Převedení interpolovaného quaternionu na transformační matici

[6] Hyperkoule - zobecnění kruhu a koule do vícerozměrného prostoru. Je definována jako množina bodů v dané n-dimenzionálním prostoru, které mají od daného bodu (tzv. středu) vzdálenost menší nebo rovnu poloměru r.



Tabulka 1: Rozdíly metod interpolace mezi quaterniony [33]

3.2 Přímá úloha kinematiky

Pokud známe parametry nastavení v jednotlivých kloubech, můžeme zcela jednoznačně určit pozici jednotlivých kloubů soustavy jednoduše výpočtem spojené transformace translačních matic. Výhoda tohoto přístupu spočívá v jednoduchosti implementace a přímé kontroly překročení limitních úhlů, na druhou stranu animátor si těžko umí představit výslednou pozici koncových uzlů při zadávání parametrů kloubů ručně.

Datové rukavice jsou vybaveny senzory ohybu prstů, jejich animace tak lze řešit přímou úlohou kinematiky, protože data, která obdržíme ze senzorů ohybu prstů, nám určují parametry kloubových proměnných.

3.3 Inverzní úloha kinematiky

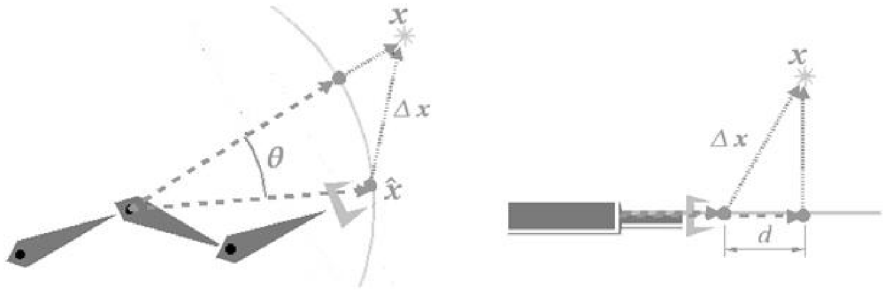
Častěji víme, kde chceme, aby se koncový bod soustavy nacházel. Výpočet hodnoty kloubových souřadnic z koncového bodu nazýváme inverzní úlohou kinematiky. Tato úloha je mnohem složitější než přímá úloha kinematiky a může mít více, nebo dokonce nekonečně mnoho řešení.

Způsoby řešení mohou stavět na algebraických, analytických nebo heuristických principech. Výhodou tohoto přístupu je přímá kontrola nad pozicí jednotlivých kloubů a výsledným uspořádáním modelu, výpočet je však mnohem náročnější.

3.3.1 Cyclic Coordinate Descent (CCD)

Základní myšlenkou CCD [16] metody je iterativní hledání úhlové proměnné každého kloubu tak, aby koncový bod mohl dosáhnout požadovaného místa v prostoru. Heuristickým[7] prohledáváním stavového prostoru se snažíme najít nejlepší množnou kombinaci úhlů. Za lepší výsledek je považován ten, který měnil pouze uzly v blízkosti koncového bodu. Stejně jako v případě lidského těla jsou kloubové proměnné limitovány jistými mezními úhly. Nemůžeme-li v daném uzlu dosáhnout lepšího výsledku, výpočet končí a přesouvá se na rodičovský uzel.

První věcí pro běh CCD metody je určení cílového bodu *X*. Poté se snažíme najít minimální vzdálenost ΔX mezi pozicí koncového bodu *X'* a chtěnou pozicí *X*. Toho docílíme navyšováním nebo snižováním parametru úhlové proměnné. Když je vzdálenost koncového bodu nejmenší, znamená to, že hodnota kloubové proměnné Θ dosáhla svého maxima v dané pozici. Tuto hodnotu uzlu zachováme a pokračujeme k rodičovskému uzlu, kde se postup opakuje. Příklad je znázorněn na obrázku č. 5:



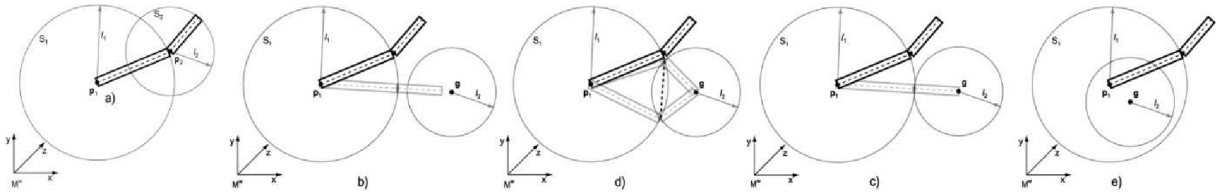
Obrázek 5: Cyclinc Coordinate Descent metoda pro a) rotační kloub b) translační kloub[16]

Cyclinc Coordinate Descent je rychlá metoda, která negeneruje chybné stavy, její zobecnění je použitelné i pro jiné účely než pouze skeletální animace.

3.3.2 Algebraicko-geometrický přístup

Algebraické metody animace kloubních soustav [17] jsou jednoduché metody pro řešení systémů až se šesti stupni volnosti. Každý kloub definuje kouli o poloměru kosti, která je na něho vázaná. Tato koule určuje pracovní neboli dosažitelný prostor kosti. Hledáme takový průsečík koulí, který nejvíce odpovídá dosažení cílového bodu. Obrázek č. 6 ukazuje možnosti, které při hledání mohou nastat:

[7] Heuristika - (z řečtiny – nalézt, objevit) znamená zkusmé řešení problémů, pro něž neznáme algoritmus nebo přesnější metodu. Heuristické řešení je často jen přibližné. První odhad se může postupně zlepšovat, i když heuristika nikdy nezaručuje nejlepší řešení. Zato je univerzálně použitelná, jednoduchá a rychlá.



Obrázek 6: Situace nastávající při použití metody algebraického přístupu[17]

Pokud je nová pozice dosažitelná, dojde k průniku koulí. V případě, že se má dosáhnout více různých bodů najednou, je možné pohnout rootovským uzlem tak, aby všechny koule měly průsečík se sousední koulí.

3.3.3 Analytický přístup

Analytickým přístupem [17][18] můžeme řešit nelineární rovnici (18) inverzní kinematiky:

$$\vec{X} = f(\vec{\Theta}) \qquad (18)$$

Tak zvanou linearizací převedeme tuto rovnici do lineární podoby. Poté spojitost mezi koncovým bodem a kloubovou proměnnou vyjádříme výrazem:

$$dX = J(\Theta).d\Theta \qquad (19)$$

Kde dX je změna pozice koncového bodu, $d\Theta$ změna kloubové proměnné. J je Jakobián, matice parciálních derivací cílových podmínek podle jednotlivých stupňů volnosti. Jinými slovy řešíme, jak se změní hodnota cíle v závislosti na změně jednoho stupně volnosti kostry (tzn. rotace nebo translace v jednom kloubu). Použitím této metody dostáváme matici Jakobiánů (20), která se řeší iterativními algoritmy a zpracování je výpočetně velmi náročné.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\Theta}_x \\ \dot{\Theta}_y \\ \dot{\Theta}_z \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \Theta_1} & \frac{\partial x}{\partial \Theta_2} & \cdots & \frac{\partial x}{\partial \Theta_n} \\ \frac{\partial y}{\partial \Theta_1} & \cdots & \cdots & \frac{\partial y}{\partial \Theta_n} \\ \vdots & \ddots & & \vdots \\ \frac{\partial \Theta_z}{\partial \Theta_1} & \cdots & \cdots & \frac{\partial \Theta_z}{\partial \Theta_n} \end{bmatrix} \begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \\ \vdots \\ \dot{\Theta}_n \end{bmatrix} \qquad (20)$$

4 Návrh

4.1 Nástroje programového řešení

Problém implementace můžeme dekomponovat do tří základních rovin, které pokryjí tři základní požadavky na aplikaci. V prvé řadě musí mít aplikace jistý grafický výstup. Zadruhé aplikace musí být interaktivní. Třetím požadavkem je, aby aplikace využívala P5 glove jako své primární vstupní zařízení.

V současnosti existují dvě grafická API vhodná pro tvorbu 3D scény v reálném čase. Jsou to OpenGL a DirectX. Rozhraní DirectX je vázáno na platformy společnosti Microsoft, tedy systém Windows a na Xbox 360. Originálně bylo určeno pro tvorbu her. Skládá se z mnoho komponent, které se vymezují podle svého účelu (pro vykreslování 3D grafiky jsou to například Direct3D a Direct3DX). K vytvoření projektu bylo zvoleno API OpenGL, které je multiplatformí a získává znovu na oblibě s příchodem nových operačních systémů pro mobilní zařízení, především pak pro rychle expandující Android a iOS. OpenGL je nízkoúrovňové API a existují k němu nadstavby pro rychlejší a snazší tvorbu programů. Známé a rozšířené jsou především dva vysokoúrovňové prostředky Open Inventor a OpenSceneGraph. Pro naše potřeby těchto prostředku není třeba, pro usnadnění práce s oknem byla však zvolena nadstavbová knihovna GLUT popsaná dále v textu.

Zatímco DirectX obsahuje komponenty DirectSound a DirectMusic zajišťující funkce pro přehrávání zvuku, OpenGL je API čistě grafické. Nabízí se volba často užívaného doplňku k dotvoření audio stránky OpenGL programů OpenAL neboli Open Audio Library. V projektu je však užito knihovny FMOD, která obsahuje širokou podporu zvukových formátů a velmi často se využívá v komerčních softwarových produktech, je ale dostupná i v nekomerční licenci.

Samotná grafická scéna nezahrnuje žádnou možnost interakce. Je nutné si uvědomit, že námi přirozené vnímání světa, kdy těleso má neoddělitelně spjatou svoji fyzikální a vizuální složku, ve světě počítačů neplatí. Prostředí grafické scény se může běžně měnit na základě dvou principů. Jednak je to příjmem pokynů od uživatele, jednak na základě pravidel chování objektů ve scéně. Většinou je žádoucí kombinace obou principů. Pravidla chování objektů ve scéně se snažíme většinou nastavit tak, jak jsme zvyklí z reálného prostředí, proto je objektům vhodné přiradit fyzikální veličiny jako jsou hmotnost, rychlost a jiné na základě potřeby, jaký okruh chování těles chceme simulovat. Vzniká tak fyzikální model scény. Aby tělesa mohla navzájem interagovat mezi sebou, je jeho významnou složkou detekce kolizí. Detekce kolizí je z výpočetního hlediska relativně snadným problémem, pokud se jedná o základní primitiva typu čára (paprsek), koule a kvádr. Výpočty kolizí ostatních útvarů (především pak konkávních) nejsou ani zdaleka triviální. Složitost fyzikálního modelu pak mimo kolizí určuje především to, do jaké míry chceme, aby byly interakce přesné, jaké chování chceme přesně simulovat (nemusí jít nutně o mechaniku) a jak komplexní má fyzikální model být. Celkově je

tato problematika velmi složitá. Z tohoto důvodu je v programu využito fyzikálního enginu Bullet Physics, který pokrývá veškeré požadavky na fyzikální model aplikace, tedy jak detekci kolizí, tak reakce na ně.

Pro komunikaci s P5 glove byla odzkoušena všechna dostupná SDK napsaná v jazyce C. Jednoznačnou volbou se staly ovladače Dual Mode Beta 3 od Carla Kennera.

Jako vývojové prostředí bylo zvoleno Microsoft Visual Studio 2008, později byl projekt vyvíjen v novější verzi 2010. Implementačním jazykem byl zvolen jazyk C++.

4.2 OpenGL

OpenGL (*Open Graphics Library*) [11] [12] je průmyslový standard specifikující multiplatformní rozhraní (API) pro tvorbu aplikací počítačové grafiky. Používá se při tvorbě počítačových her, CAD programů, aplikací virtuální reality, vědeckotechnické vizualizaci, či jiných grafických aplikací.

Základní funkcí OpenGL je vykreslování do obrazového rámce (framebufferu). Umožňuje vykreslování různých základních primitiv (bodů, úseček, mnohoúhelníků) v několika různých režimech, ale je možno také na něj pohlížet jako na stavový automat. Poskytuje nám možnost měnit vlastnosti primitiv nebo scény (např. barva, buffery, antialiasing, textury, světla, materiály, translace…) a toto nastavení zůstane zachováno, dokud ho explicitně nezměníme.

Rozhraní OpenGL je založeno na architektuře klient-server, program (klient) vydává příkazy, které grafický adaptér (server) vykonává. Díky této architektuře je například možné, aby program fyzicky běžel na jiném počítači než na tom, na kterém se příkazy vykonávají, a příkazy se předávaly prostřednictvím počítačové sítě.

OpenGL bylo postaveno jako 3D renderující systém, který může být HW akcelerován. Operace se tedy provádí buď programově (vykonává procesor CPU), nebo se předá ovladači 3D akcelerátoru (vykonává grafický procesor GPU).

Aby zůstalo platformě nezávislé, neobsahuje žádné funkce pro práci s okny ani s GUI (Graphical User Interface), stejně jako neumí pracovat ani se zvukem, tiskem, myší a klávesnicí či jinými vstupními zařízeními. Díky tomu je dnes OpenGL dostupné na většině operačních systémů (Windows, Unix, Linux, Irix, Sun). K velkému rozšíření OpenGL však nemalou měrou přispěla nejen platformní nezávislost, ale také možnost programování aplikací v mnoha jazycích jako jsou Ada, C, C++, Fortran, Java, ObjectPascal, Python, Perl, assemblery apod. OpenGL v rámci platformní nezávislosti přináší i své vlastní datové typy jako GLbyte, GLint nebo GLdouble.

4.2.1 Pohled do historie

V 80. letech bylo psaní aplikací používající grafický HW neskutečně obtížné. SW vývojáři psali API a ovladače pro každé zařízení zvlášť. To vedlo k neustálému předražování vývoje a duplicitě kódu, kdy mnozí psali něco, co už dávno před nimi napsal někdo jiný pro jiné zařízení. Na počátku 90. let byla v segmentu 3D grafických pracovních stanic firma SGI (Silicon graphics). Jejich IrisGL API se díky svému kvalitnímu návrhu a snadnému používání stalo de-facto průmyslovým standardem. Konkurenti SGI (společnosti jako IBM, HP, SUN) používali pro svůj 3D hardware konkurenční standard PHIGS, který byl považován za částečně zastaralý. Následkem toho se pozice firmy SGI na trhu neustále oslabovala s narůstajícím počtem konkurentů používajících standard PHIGS. Za této situace přichází SGI s revolučním řešením a rozhodne se proměnit IrixGL na otevřený standard. Vyústěním těchto snah bylo uvedení OpenGL standardu společností SGI, odvozeného od IrisGL.

OpenGL sjednotila přístup k HW a přesunula odpovědnost za obsluhu HW od vývojářů aplikací směrem k výrobcům HW. Dnes se tento revoluční krok projevuje například existencí ovladače ke grafické kartě a všichni berou ovladače jako samozřejmost. Přesto je to inovace, za niž vděčíme právě vzniku OpenGL. Jednotná řeč pro všechen možný grafický HW měla velký pozitivní vliv na vývoj 3D aplikací. V roce 1992 SGI podpořila vznik uskupení *OpenGL architectural review board* (*OpenGL ARB*), tedy jakousi asociaci společností, která měla zajistit vývoj OpenGL do budoucna. Jednou z mnoha společností tohoto uskupení byl i Microsoft, který jej ovšem v roce 2003 opustil. Od roku 2006 spravuje OpenGL skupina *Khronos group*. Mezi nejznámější členy patří např. AMD, Creative Labs, Intel, ID Software, Nvidia, Sony Computer Entertainement, Sun Microsystems a mnoho dalších.

4.2.2 Rozšíření OpenGL

Standard OpenGL dovoluje výrobcům implementovat na jejich zařízení další funkce korespondující k novým technologiím a umožnit k nim přístup skrze rozšíření. Rozšíření je potom dodáváno ve dvou částech. První je hlavičkový soubor, který obsahuje prototypy funkcí extenze, a druhý je ovladač zařízení. Více výrobců může přistoupit k implementaci stejné funkce, tato může být poté „posvěcena“ ARBem a stane se z ní součást nového standardu. První takovou extenzí byla GL_ARB_multitexture, která se z pozice nepovinného rozšíření stala standardem počínaje OpenGL API 1.4.
Nad OpenGL je postaveno několik knihoven, které ho doplňují:

OpenGL Utility Library (GLU) - umožňuje využívat tesselátory (rozložení nekonvexních polygonů na trojúhelníky), evaluátory (výpočet souřadnic bodů ležících na parametrických plochách) a vykreslovat kvadriky (koule, válce, kužely a disky) Nastavuje matice pro speciální pohledové a projekční orientace. Příkazy knihovny GLU začínají prefixem *glu*.

The OpenGL Utility Toolkit (GLUT) - tvoří doplněk ke grafické knihovně OpenGL. Základem této nadstavbové knihovny je podpora pro práci s okny (včetně zpracování událostí), vyskakovacími menu a písmem. Tyto činnosti totiž nejsou v knihovně OpenGL přímo podporovány - důvodem je snaha o co největší zachování platformové nezávislosti. Funkce pro práci s okny či menu, které jsou systémově závislé, se dříve, tj. v době, kdy knihovna GLUT neexistovala, musely naprogramovat pro každý operační systém (resp. jeho grafickou nadstavbu) zvlášť, což od vývojáře aplikace vyžadovalo podrobnou znalost funkcí daného operačního systému, grafické nadstavby a správce oken. Příkazy knihovny GLUT začínají prefixem *glut*.

OpenGL User Interface Library (GLUI) je C++ knihovna uživatelského rozhraní založená na OpenGL Utility Toolkitu (GLUT), která přináší do OpenGL aplikací prvky jako tlačítka, checkboxy, radio buttony, apod. Je systémově nezávislá, spoléhá ve všech systémových závislostech (jako je obsluha okna nebo periférií) na GLUT.

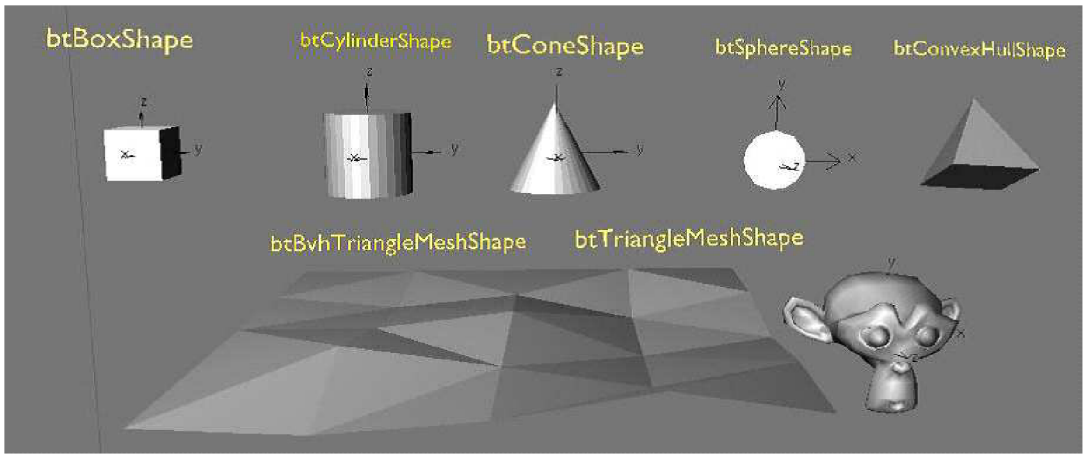
4.3 Bullet physics

Fyzika Bullet (Bullet physics) [13][14] je profesionální open-sourcová dynamická knihovna obsahující především algoritmy detekce kolizí, dynamiku tuhých a „měkkých“ těles. Spadá pod licencí ZLib, její užití je tak i pro komerční užití zdarma. Projekt založil v roce 2003 bývalý vývojář fyzikálního enginu Havok Erwin Coumans. Cílem projektu bylo vytvořit fyzikální zázemí pro tvůrce her a pro akademické účely, například pro simulování chování tuhých těles. V současnosti se jedná o třetí nejpoužívanější fyzikální knihovnu (první dvě příčky dnes drží NVidia PhysX a Havok). Využití Bulletu již překonalo původní záměry a setkáme se s ním kromě herního i v průmyslu filmovém.

Bullet podporuje jak grafické API OpenGL, tak DirectX a pracuje tak na většině současných platforem včetně Androidu, Linuxu, PC (Windows), Playstation 3, Wii, XBoxu 360, Mac OsX a iPhone. Navíc vzniká mnoho pluginů do nástrojů pro tvorbu modelů a animací jako je Blender, Cinema 4D, LightWave, Maya a mnoho dalších. Knihovna je napsána v jazyce C++ včetně podpory více programových vláken pro vícejádrové systémy a vznikl i port do C# podporující Windows a Xbox 360 XNA.

4.3.1 Detekce kolizí

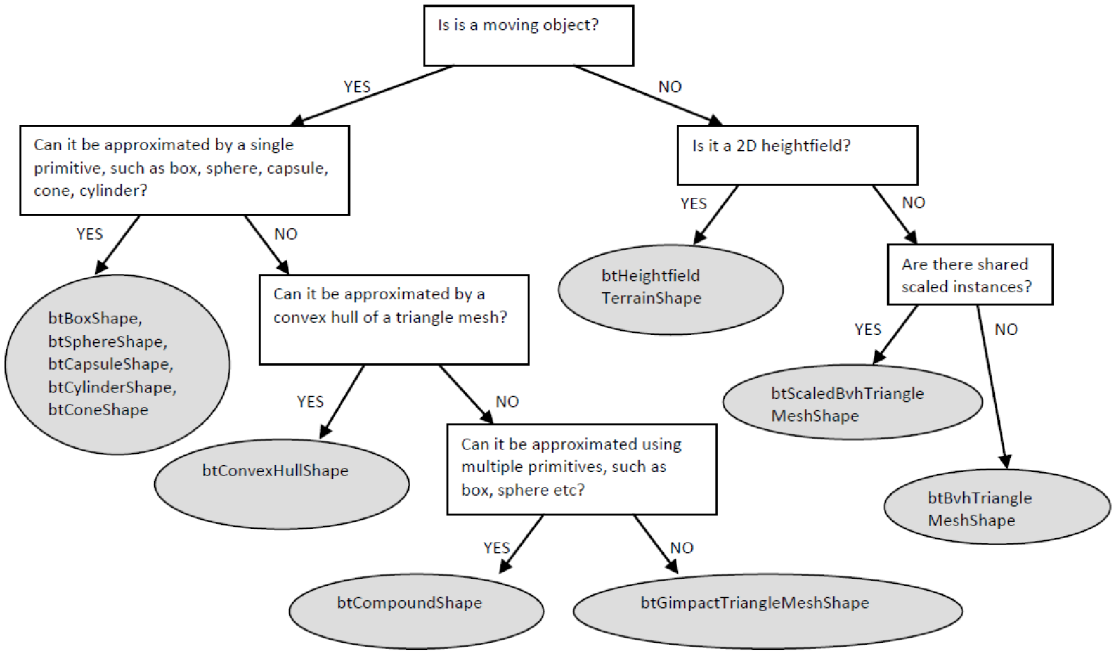
Bullet podporuje velké množství kolizních tvarů a umožňuje přidávat tvary vlastní. Kolizní objekty knihovny je možné kategorizovat do tří základních skupin. Zaprvé se jedná o základní primitiva, konkrétně o kvádr a kouli a o něco složitější válec, kužel a kapsle. Poslední tři jmenované tvary mají i své modifikace podle os X a Z. Druhou skupinu kolizních objektů tvoří tvary složené. Pomocí nich je možné vytvořit konkávní tělesa, která se skládají z primitiv první skupiny nebo jednodušších konvexních dílů. Tyto díly označujeme jako potomky a každý potomek má vůči složenému tvaru, ke kterému náleží, svou relativní pozici a lokální transformaci. Poslední skupinou jsou zcela univerzální, zato velmi výpočetně náročné, konkávní a konvexní trojúhelníkové sítě. Typickým příkladem užití je pokrytí členitého terénu kolizní zónou. Složité konkávní útvary je vhodné dekomponovat do dílčích konvexních tvarů nebo aproximovat kolizní tvar objektu vytvořením konvexní obálky[8] kolem objektu. Konvexní obálka vznikne spojením nejvíce vyčnívajících bodů objektu do prostoru podobně jako bychom objekt obalili pružnou membránou.



Obrázek 7: Základní kolizní tvary fyzikálního enginu Bullet Physics [14]

Pro lepší výkon aplikace a spolehlivější detekci kolizí je doporučeno držet se při výběru kolizních útvarů schématu, který je zachycen na obrázku č. 8.

[8] Konvexní obálka - souboru bodů X v reálném vektorovém prostoru V takových, že povrch obálky je minimální.



Obrázek 8: Rozhodovací schéma pro výběr kolizních tvarů enginu Bullet Physics [13]

Tyto vytvořené objekty Bullet uchovává ve dvou BVT stromech. BVT strom[15] je zkratka pro binární vertex toleranční strom. Jedná se o objekt, jenž je užíván pro dekompoziční metody objektově-prostorových databází. Jak název napovídá, strom pracuje s vertexy, jejich umístěním v prostoru a tolerančními vzdálenostmi mezi nimi, na jejichž základě strom restrukturalizuje. Právě výpočtem toleranční vzdálenosti se BVT stromy liší od jiných podobných stromů (BLG,…) .

Do jednoho z BVT stromu jsou ukládány statické nepohybující se objekty. Do druhé pak objekty pohybující se. Porovnávány jsou poté navzájem jen objekty, u kterých může dojít ke kolizi - tedy pohybující se s pohybujícími a pohybující se se statickými. Neporovnávají se však původní kolizní útvary. Pro rychlé vyloučení objektů, mezi kterými určitě ke kolizi v následném okamžiku nedojde, pracují tyto BVT stromy pouze s AABB[9] boxy, jejichž výpočty kolizí jsou výpočetně nenáročné. Tyto bounding[10] boxy obalují původní kolizní těleso a jsou rozměrově o něco větší. Hodnotu tohoto přesahu (angl. margin) je možné explicitně měnit, není to však doporučeno. Nad BVT stromy lze provázet dotazy na kolizi mezi stromem a stromem (Tree-Tree), stromem a částí stromu (Tree-Volume), stromem a paprskem (Raycast) a stromem a uživatelským vstupem (Tree-User). Až na základě pozitivní detekce je hledán přesný průsečík mezi jednotlivými objekty. Algoritmy k tomu použité jsou závislé na kolidujících tvarech. Pár kvádr-kvádr, koule-koule a koule-kvádr využívá standardních algoritmů pro řešení kolizí těchto těles. Pro ostatní konvexní tělesa má Bullet vlastní implementaci algoritmu GJK[11]. Rozšířen je navíc o kombinaci s algoritmem EPA[12].

---

[9] AABB box - angl. zkr. axis-aligned bounding box, osově orientované kolizní boxy

[10] Bounding – z angl. bounding – ohraničující

[11] GJK algoritmus - algoritmus sloužící k určení nejmenší vzdálenosti dvou konvexních sad, pojmenován byl po jeho autorech Gilbertovi, Johnsonovi a Keerthim.

[12] EPA – zkr. Expanding Polythope algorithm – stejně jako u GJK se jedná o iterativní algoritmus využívající koncept Minkowského sumy k výpočtu vzdálenosti, na rozdíl od GJK však řeší i případ překrytí [21]

4.3.2 Dynamika tuhých těles

Základní modul pro detekci kolizí je možné rozšířit o nadstavbový modul dynamiky tuhých těles, který rozšiřuje problematiku kolizí o působení sil, hmotnost, setrvačnost, rychlost a o závislosti mezi jednotlivými tělesy. Bullet rozlišuje základní tři typy objektů:

- *Pohybující se tuhá tělesa* – tělesa s kladnou hmotností, jejichž pozice je aktualizována v každém snímku simulace. Jedná se o většinu těles v simulovaném světě a o prakticky všechna tělesa, se kterými je možná fyzikální interakce.
- *Statická tuhá tělesa* – tělesa s nulovou hmotností, která se nepohybují a neplatí na ně působení gravitačních ani jiných sil, avšak kolidují s jinými objekty. Těmito objekty je možné simulovat zemský povrch aj.
- *Kinematická tuhá tělesa*. Kinematika je část mechaniky, která se zabývá pouze popisem pohybu, ne však jeho příčinami. U kinematických těles nás stejně jako u dynamických zajímá jejich poloha a rychlost, ale na rozdíl od dynamických nesledujeme veličiny, jako je například hybnost, protože ta je přímo úměrná hmotnosti. V Bullet enginu mají tedy taktéž nulovou hmotnost. Tyto objekty můžou a předpokládá se, že budou animovány uživatelem. Jejich působení na svět je však pouze jednosměrné. Kinematická tělesa působí na okolí (konkrétně na pohybující se tuhá tělesa, se kterými přijdou do kontaktu), ale okolí na ně nemá žádný vliv.

Kromě tuhých těles existují ve většině fyzikálních enginů i soft bodies[13] tělesa. Jejich dynamika je obsažena v dalším modulu, který je nadstavbovým modulem k modulu s dynamikou tuhých těles. Dynamika soft body těles napodobuje chování oděvů, látek, zavěšeného lana a podobných věcí. Na rozdíl od tuhých těles nelze soft body tělesa popsat jedinou pozicí, ale každý vertex takového tělesa má svou vlastní pozici.

4.3.3 Kloubní soustavy a omezení pohybu

Engine nabízí několik funkcí pro řešení vzájemných závislostí tuhých těles a funkcí pro tvorbu kloubů a kloubních systému. Je možné pomocí nich spojit dvě tuhá tělesa a nastavit jim pravidla jejich vzájemné interakce.

- *Bodové omezení pohybu* - třída *btPoint2PointConstraint* umožňuje spojení dvou tuhých těles v určitém bodě a vytvořit tak jejich závislost pohybu nebo navázání pouze jednoho tělesa k určitému bodu otáčení.

[13] Soft body – na rozdíl od pojmu Rigid body (česky tuhé těleso) není v češtině pro tento výraz vhodný ekvivalent. Jedná se o objekty, jako jsou látky, lana apod., jejichž fyzika je složitější než u běžných tuhých těles

- *Osové omezení pohybu* - s prvkem připojeným pomocí třídy *btHingeConstraint* je možno manipulovat kyvadlově dvěma směry. Těleso se může otáčet pouze podle jedné osy a to tzv. osy zavěšení. Tento typ uchycení je užitečný pro reprezentaci pantů u dveří nebo například kol, protože jejich pohyb je též svázán s jednou osou. Explicitně je možné upravovat limity rozsahu pohybu. Tato třída navíc umožňuje rozšířit osu otáčení o motor, který může na těleso působit, upravovat tak působení vnějších sil na těleso a umožnit změnu jeho pohybu.
- *Posuvné omezení pohybu* - posuvník dovoluje tělesu vykonávat translační pohyb ve směru jedné z os.
- *Kuželové omezení pohybu* – především při vytváření modelu lidské postavy se můžeme setkat s potřebou vytvoření kloubu, který by uchycoval těleso (kost) v jednom bodě (kloubu) a umožňoval jeho pohyb v pomyslné kuželové výseči. K tomu dobře poslouží kuželové omezení pohybu. Uchycené těleso se kroutí podél jedné osy nazývané osa kroucení.
- *Obecné omezení pohybu se šesti stupni volnosti* - vycházíme-li z faktu z kapitoly 2.4.1, že těleso v třírozměrném prostoru má šest stupňů volnosti, z toho tři translační osové volnosti a tři rotační volnosti, můžeme každou z této volnosti limitovat a vytvořit tak pro těleso libovolné omezení pohybu. Těleso může být v každém ze stupňů volnosti volné, limitované nebo zamčené. Při vytvoření nového spojení jsou všechny stupně volnosti implicitně zamčené, po vytvoření může být každý stupeň nezávisle přizpůsoben. Některé kombinace nastavení spojení můžou být problematické nebo neproveditelné.

4.4 Ovladače P5 glove

Snímací zařízení P5 glove obsahuje dva infračervené senzory. Na rukavice je pak umístěno 8 LED diod. Ty jsou označeny čísly 0 až 7 a jejich rozmístění bylo voleno tak, aby bylo při jakémkoliv úhlu natočení rukavice viditelných co nejvíce LED. Dochází ale i k situacím, kdy je zakryto všech osm diod. Data z optických senzorů prstů a data z tlačítek jsou odesílána prostřednictvím kabelu spojující snímač a rukavici.

4.4.1 Oficiální ovladače

Rukavice vstoupila na trh roku 2002. V té době k ní existoval ovladač ve verzi v1.0 vyvinutý Av Utukurim a Igorem Borysovem. V této fázi ovšem oficiální vývoj a podpora ze strany výrobce již téměř skončila. I přestože je zařízení dalších deset let v prodeji, nedočkalo se nikdy žádného oficiálního softwarového vylepšení. Původní SDK bylo určeno pro operační systém Microsoft Windows XP a Mac OS ve verzi 9.0 nebo nižší, definovalo strukturu P5Data a třídu CP5DLL se 14 metod pro komunikaci a získávání dat ze zařízení. Struktura P5Data obsahuje kromě pomocných (především charakteru charového řetězce) a informačního proměnných celkem devět proměnných

důležitých pro ukládání polohových a stavových dat. Jedná o 6 floatových hodnot pokrývajících stupně volnosti, dvě jednorozměrná pole, jedno pětičlenné s údaji o míře ohnutí prstů a jedno čtyřčlenné odpovídající čtyřem tlačítkům na horní části rukavice a jedno dvojrozměrné pole o rozměrech 3x3 popisující matici pro inverzní kinematiku. Orginální SDK sice obsahuje průměrovací (average filter) filtr a filtr ztráty signálu (deadbend filter), tyto filtr ale nepracují dobře, což je poměrně velký problém, protože snímací věž často ztratí signál některé z LED diod a reaguje náhlou změnou pozičních údajů. Zařízení tak vykazuje velké výkyvy v údajích i při neměnící se poloze rukavice, rotační proměnné obsahují až téměř náhodné hodnoty. V druhém čtvrtletí 2003 se SDK dočkalo verze 2.0. Toto SDK však nepřineslo nic nového kromě několika příkladů jak s ním pracovat. Ovladač zůstal ve stejné verzi. S těmito ovladači P5 glove plní maximálně svoji základní funkci a to levného herního hardwaru schopné nahradit počítačovou myš více intuitivním ovládacím prvkem. K takovému využití není nutná instalace ovladačů, dokonce ani přítomnost dynamické knihovny p5dll.dll v systému, protože P5 glove má hardwarově zabudováno jevit se systému jako myši podobné zařízení a systém pro práci s ní užívá své standardní ovladače pro práci s touto periférií.

Jelikož byla P5 glove první skutečně cenově dostupná alternativa datové rukavice, vysloužila si velkou oblibu ve vývojářské komunitě. Díky tomu začaly vznikat neoficiální ovladače, především pak pro platformu Linux, který nebyl v oficiálním SDK podporován. Nejúspěšnějším ovladačem se stal Dual Mode, který naprogramoval Carl Kenner.

4.4.2 Kalmanův filtr

Kalmanův filtr[22] je soubor matematických rovnic, které poskytují efektivní výpočet minimální střední kvadratické chyby, na jehož základě lze odhadnout stav určitého procesu. Tuto rekurzivní metodu lineární filtrace dat představil v roce 1960 R.E.Kalman. O té doby se našlo mnoho uplatnění Kalmanova filtru v různých počítačových systémech, zejména v oblasti autonomní nebo asistované navigace. Filtr je velmi silný v několika aspektech: podporuje odhad minulých, přítomných, a dokonce budoucích stavů, a může takové odhady učinit i bez přesné znalosti povahy modelovaného systému.

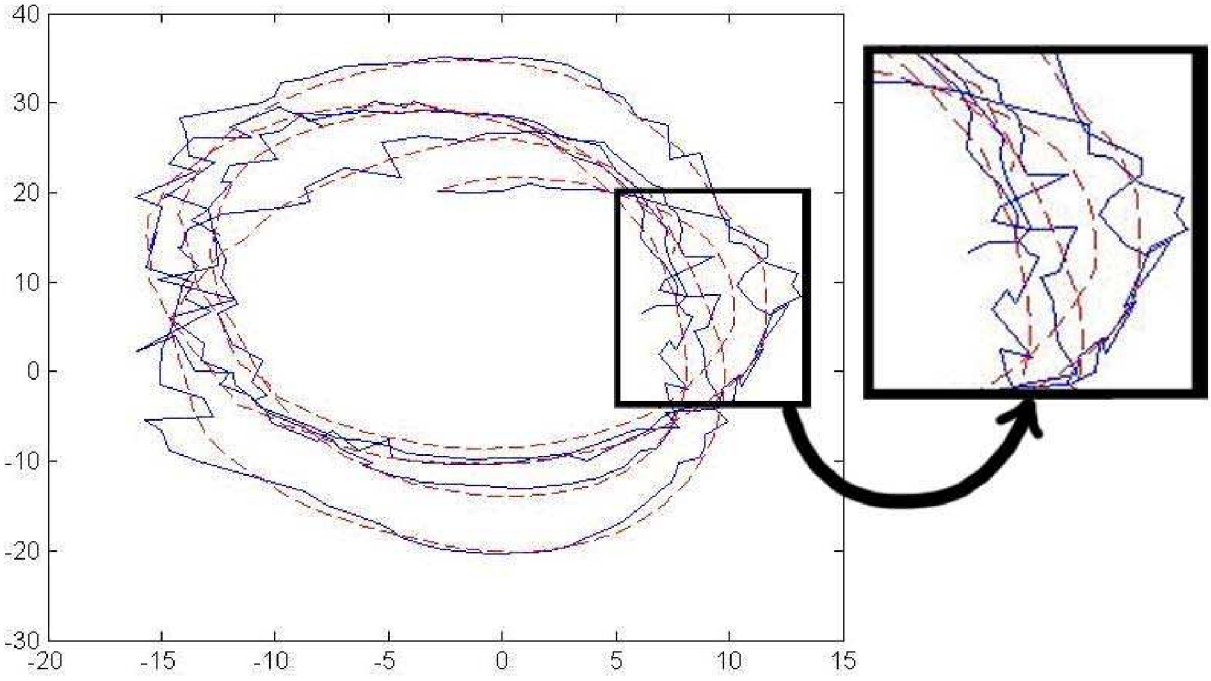
Kalmanův filtr upravuje své odhady na základě zpětné vazby. Filtr nejdříve provede odhad a pak získá zpětnou vazbu ve formě naměřené hodnoty. Tato hodnota je často zkreslena šumem měření (tento šum se snaží filtr potlačit). Kromě šumu měření vzniká také šum procesu daný nepřesným odhadem. Tyto dva šumy jsou navzájem korelované, ale v různých časových okamžicích jsou již nezávislé. Soubor matematických rovnic Kalmanova filtru můžeme rozdělit do dvou skupin. První skupinou jsou rovnice pro časový krok[14]. Tyto rovnice zajišťují predikci apriorní hodnoty příštího stavu přes kovarianci[15] šumů. Druhou skupinou jsou rovnice pro datový krok, které zajišťují zpětnou vazbu. Na základě porovnání predikce a skutečných údajů se snaží upravit odhad tak, aby filtr dosáhl

[14] České termíny pro rozdělení rovnic na časový a datový krok byly zvoleny na základě zdroje [23]. Původní názvy v anglickém jazyce jsou Time update (časový krok) a Measurement update (datový krok)

[15] Kovariace - míra vzájemné vazby mezi dvěma náhodnými veličinami

lepších výsledků. První skupinu rovnic tak často označujeme jako prediktor, druhou skupinu jako korektor.

O implementaci Kalmanova filtru pro P5 glove se zasloužil Richard Hachem. Implementace samotná nepřináší nové ovladače. Autor využívá ovladačů Dual Mode Carla Kennera. Soubor s filtrem je však možné vložit do projektu a využívat pro další filtraci. Na váze tomuto řešení ubírá fakt, že filtr je implementován pouze pro filtraci pozice. Filtrace rotace vyžaduje řešení s quaterniony, které je mnohem složitější. Bohužel právě nepřesné snímání rotací je největším problém P5 glove. Zajímavý je pohled na obrázek č. 9., kde vidíme výstup z MATLABu znázorňující grafem hodnoty vyhlazené Kalmanovým filtrem a hodnoty bez užití Kalmanova filtru. Musíme si uvědomit, že se jedná o hodnoty naměřené jako výstup ovladačů Dual Mode, které mají vylepšenou míru filtrace a mají i schopnost predikce. I přesto jsou hodnoty nepřesné a rozkmitané. Oficiální ovladače podávají výsledky ještě o mnoho horší.



Obrázek 9: Graf pozic X a Y filtrovaných Kalmanovým filtrem a bez filtrace [24]

4.4.3 Dual Mode beta 3

V současnosti je nejznámějším projektem Carla Kennera aplikace GlovePIE. Jde o programovatelný emulátor vstupu (Programmable Input Emulator) prvotně vyvíjen pro rozpoznání a emulaci gest datové rukavice P5 glove. Postupem času přibývala podpora dalších funkcí a dalších zařízení. Mezi ty nejznámější patří především periferie herních konzolí v popředí s Wii remote, Wii nunchuck, Dual Shock PS3, sixaxis, Xbox 360 controller nebo například novější typ datové rukavice 5DT data glove. V okamžiku kdy Microsoft oznámil práci na projektu Natal, dnes známém jako Kinect, přeorientoval se zájem Carla Kennera především na něj.

Při vývoji GlovePIE Kenner naprogramoval vlastní SDK pro P5 glove, které filtruje výstupy a podává nejlepší výsledky. Název Dual Mode souvisí se zpětnou kompatibilitou s originálními drivery, které pracují s relativní pozicí, Kenner však již v minulosti vytvořil ovladače vracející absolutní pozici rukavice vůči věži. Relativní pozice je vhodná právě do videoher či jako náhrada myši. Absolutní pozicování je vhodné pro kopírování pohybu a pozice ruky v prostoru. Dual Mode umožňují práci s relativním i absolutním souřadným systémem. Střed absolutního souřadného systému se nachází uprostřed spodní hrany snímací plochy věže.

Kenner rozšířil třídu CP5DLL o více než 40 nových metod, kvůli zpětné kompatibilitě zachoval původní třídu P5data, ale třídu rozšířil o další dvě nové struktury P5info sdružující výrobní informace rukavice a P5state, která je novou strukturou pro uchovávání všech dat rukavice. Ty se nyní dají dělit na poziční údaje, rotační údaje, údaj o viditelnosti, údaje prstů, tlačítek, údaje všech LED diod a viditelných LED diod. Hlavně poslední dvě zmiňované věci přinesly velkou změnu oproti oficiálnímu SDK, které neobsahují metody pro přímý přístup k datům z LED diod. Ty navíc lze nově vybrat k určování pozice (tedy aplikace nemusí pracovat ani v relativním nebo absolutním módu, ale přímo s konkrétní diodou[16]). Zajímavou novinkou je i rozšíření pozičních údajů a dat z prstů o rychlost a zrychlení a rotačních údajů o úhlovou rychlost a úhlové zrychlení. Následující seznam je shrnutím možností, které Kennerovo SDK přináší [26][27] :

- Zpětná kompatibilita s existujícím softwarem užívajícím jak v relativním, tak absolutním módu
- Nová API pro jazyky Delphi, Visual Basic, Java, C, C++
- Možnost volby relativního nebo absolutního snímání
- Výběr ze dvou filtrů (Averaging, Deadband) nebo jejich užití současně
- Možnost získat pozice jednotlivých LED a další nízkoúrovňová dříve nedostupná data
- Se snímací věží je možné manipulovat (přesouvat, otáčet)
- Odhad pro stav, kdy se rukavice dostane mimo dosah snímání
- Je možné nastavit počáteční orientaci rukavice v prostoru
- Výpočet rychlostí a zrychlení
- Měření chyb LED diod
- Absolutní hodnoty ohnutí prstů
- Rukavice může být používána levou rukou
- Možnosti nastavení senzitivity
- Volba jednotek[17]
- Umístění LED diod na rukavice může být měněno
- INI soubor umožňující uzpůsobovat chování ovladače pro různé aplikace zvlášť

[16] Může se jednat o jednu z deseti diod (Kenner nepracuje s osmi, ale deseti LED), nebo o „nejsvětlejší“ či „nejlepší“ diodu.

[17] Možnost volby jednotek přináší nezávislou volbu druhů jednotek pro poziční, rotační, rychlostní a časová data

- Levoruký a pravoruký souřadný systém
- Volitelný Eulerův úhel[18]
- Možnost zjištění, zda je rukavice mimo dosah nebo ne
- Data mají časové razítko
- Nové ovladače umí vše to, co uměly ovladače původní

4.5 MilkShape 3D

MilkShape 3D [29] je sharewerový nízkopolygonální 3D modelovací program vytvořený švýcarskou firmou Chumbalum Soft zaměřenou především na vývoj 3D nástrojů pro tvorbu her. MilkShape byl původně vytvořen jako program pro tvorbu modelů do enginu hry Half-life Metem Ciraganem. Časem do něho byla přidána podpora mnoha formátů. Jeho užívání je stále efektivní, protože program je jednoduchý a levný.

Nativní formát MilkShapu 3D je *.ms3d formát. Je vhodný právě pro modelování kloubních soustav a skeletálních animací, protože v sobě přímo zahrnuje data pro animaci. Struktura formátu je následující:

Header	MS3D000000 následovaná číslem verze (verze 3 nebo 4)
Vertex Data	Koordináty vertexů
Triangle Data	Ukazatelé na vertexy a normály povrchu
Group Data (object/mesh)	Jména shluků a jejich ukazatele na trojúhelníky
Material Data	Detaily rozložení barev
Bone Data	Data pro skeletální animaci

Tabulka 2: Struktura formátu ms3d

Hlavička (header) je důležitá pro ověření formátu a verze. Načítání vertexových (vertex data) dat a trojúhelníků (triangle data) probíhá zjištěním jejich počtu a následného postupného přecházení souborem a ukládáním načtených dat do struktur. Zajímavější je až načítání skupin trojúhelníků (group data), zde kromě počtu trojúhelníků a trojúhelníků samotných obsahuje soubor informace o jménu skupiny a nastavení jejích příznaků. Materiálová data (material data) obsahují informace o ambientních, difúzních, spekulárních, emisních hodnotách, shinness hodnotě a informace o textuře materiálu (cesta k souboru s texturou). Nejzajímavější pro tento formát jsou samozřejmě údaje pro animační kostru (bone data). Zde ze souboru získáme rychlost animace kostry, hodnotu aktuálního času animace, počet snímků animace. Pak se iterativně prochází jednotlivé klouby. Každý kloub má své jméno a index. Známe-li tyto informace, můžeme zjistit, zda má kloub rodičovský uzel, zjistit rotaci a posun kloubu a v závislosti na rychlosti animace a počtu snímků animaci celé kloubní soustavy provést.

[18] Eulerův úhel – určuje orientaci v prostoru vzhledem k soustavě souřadnic

5 Implementace

Tato část práce popisuje výsledek implementace. Je podobně strukturovaná jako předchozí kapitola. V první části bude rozebrána vizuální stránka 3D scény aplikace doplněná o screenshoty[19]. Druhá část bude zaměřena na tvorbu fyzikálních modelů, především pak modelu ruky. Poslední část obsahuje popis práce s SDK pro P5 glove, reálné možnosti P5 glove při manipulaci s 3D scénou a diskuse dosažených výsledků. Poté následuje rozbor možností užití P5 glove k analýze gest uživatele. Některé části textu jsou doplněny částmi zdrojových kódů pro snazší pochopení textu, a hlavně mají posloužit zájemcům, kteří by chtěli někdy s Bullet Physics nebo s SDK P5 glove pracovat.

5.1 Audio-vizuální zpracování

5.1.1 Pokročilá grafika

Datové rukavice jsou většinou určeny jako ovládací prvek virtuální reality, která je většinou spojována s vizuální simulací reálného prostředí. Proto i v případě aplikace pro P5 glove byla snaha o použití moderních prostředků počítačové grafiky pro vytvoření scény, která by spíše než grafickou hříčku připomínala reálnou scénu. K dosažení takového cíle bylo třeba se seznámit s pokročilými technikami OpenGL.

Základem pro reálné zobrazení scény je užití fotorealistických textur. Takové textury se nám běžně nepodaří vygenerovat, proto jedním z prvních kroků byla implementace třídy pro načítání textur. Pokud užijeme hlavičkový soubor *glaux.h*, která v sobě, mimo hlavičkových souborů OpenGL API, zahrnuje také *windows.h*, která definuje struktury pro práci s bitmapami, stává se načítání formátu BMP jednoduchou záležitostí. Proto je jedním z podporovaných formátů 24-bitový BMP. Protože aplikace pracuje s color bufferem typu RGBA, je posledních 8bitů doplněno hodnotou 255. Alfa kanál v RGBA bufferu je nutností, aby v aplikaci mohlo být využito blendingu. Blendingu je využito při tvorbě poloprůhledných materiálů v menu a tutoriálu. Největší benefit ale spočívá v možnosti vymaskování části textur. Textury určené pro maskování byly upravovány v grafických editorech Photoshop a GIMP. Přidáním čtvrtého kanálu do obrázku a nastavením pixelů, které se mají vymaskovat, na hodnotu 0, připravíme texturu. Jednoduchým formátem s podporou alfa kanálu je formát TGA. Kromě 32-bitové verze, která je pro blending nezbytností, byla implementována i podpora 24-bitové a 16-bitové verze. Získávání dat z TGA souboru je principielně stejné jako u formátu BMP, po přečtení hlavičky souboru, čteme hodnoty po osmi bitech a zapisujeme příslušné hodnoty. Mírně jiný postup je jen u 16-bitové verze. Protože targa a bitmapa jsou formáty značně

[19] Screenshot – anglický termín pro snímek obrazovky. Zkopírování aktuální podoby obrazovky a její uložení do paměti nebo na disk počítače v podobě grafického souboru.

neúsporné, bylo načítání formátu targa ještě rozšířeno o podporu algoritmu RLE[20]. Populární je často programování podpory pro formáty s vysokým kompresním faktorem, jako je JPEG a PNG. Vzhledem k užití složitých metod komprese se pro jejich načítání využívá většinou knihoven třetích stran, a jelikož datová úspora není v této aplikaci nezbytná, bylo od jejich implementace upuštěno.

Pokud nechceme nanášet textury na modely s jednoduchou geometrií, které jsme schopni vytvořit jen z primitiv OpenGL přímo v kódu, je potřeba naimplementovat modul pro načítání modelů. Je možné si vytvořit vlastní formát, například takový, že soubor s daty obsahuje na každém svém řádku tři souřadnice určující vrchol vykreslovaného trojúhelníku. Už jen takové řešení přináší možnost lepší představy o modelu a zamezení neúnosnému nafukování kódu. Lepším řešením je zvolit si některý z rozšířených formátů, pro které existují grafické editory. Dobrou volbou je výběr staršího formátu 3D Studia Max s koncovkou 3ds. Podíváme-li se na specifikace tohoto formátu, uvidíme, že obsahuje mnoho informací, které není potřeba načítat, protože nejsou pro aplikaci důležité, nebo už máme v aplikaci vlastní parametry těchto dat. Jedná se především o nastavení kamery, světel apod. Důležitá data pro nás jsou vertexy modelu a UV koordináty[21] textur. Pokud známe identifikátory ze specifikace formátu, můžeme najít kontejner, tzv. chunk, kde se požadovaná data nachází. Chunk 3ds formátu je vždy tvořen posloupností bytů takovou, že první dva byty uvozují typ chunku, následují 4byty označující celkovou délku kontejneru, tedy délku dat a všech zanořených subchunků. Data a subchunky následují za určením délky. Kontejner s vertexovými daty má identifikátor 0x4110, koordináty textur pak 0x4140. Je však potřeba projít všechny rodičovské chunky.

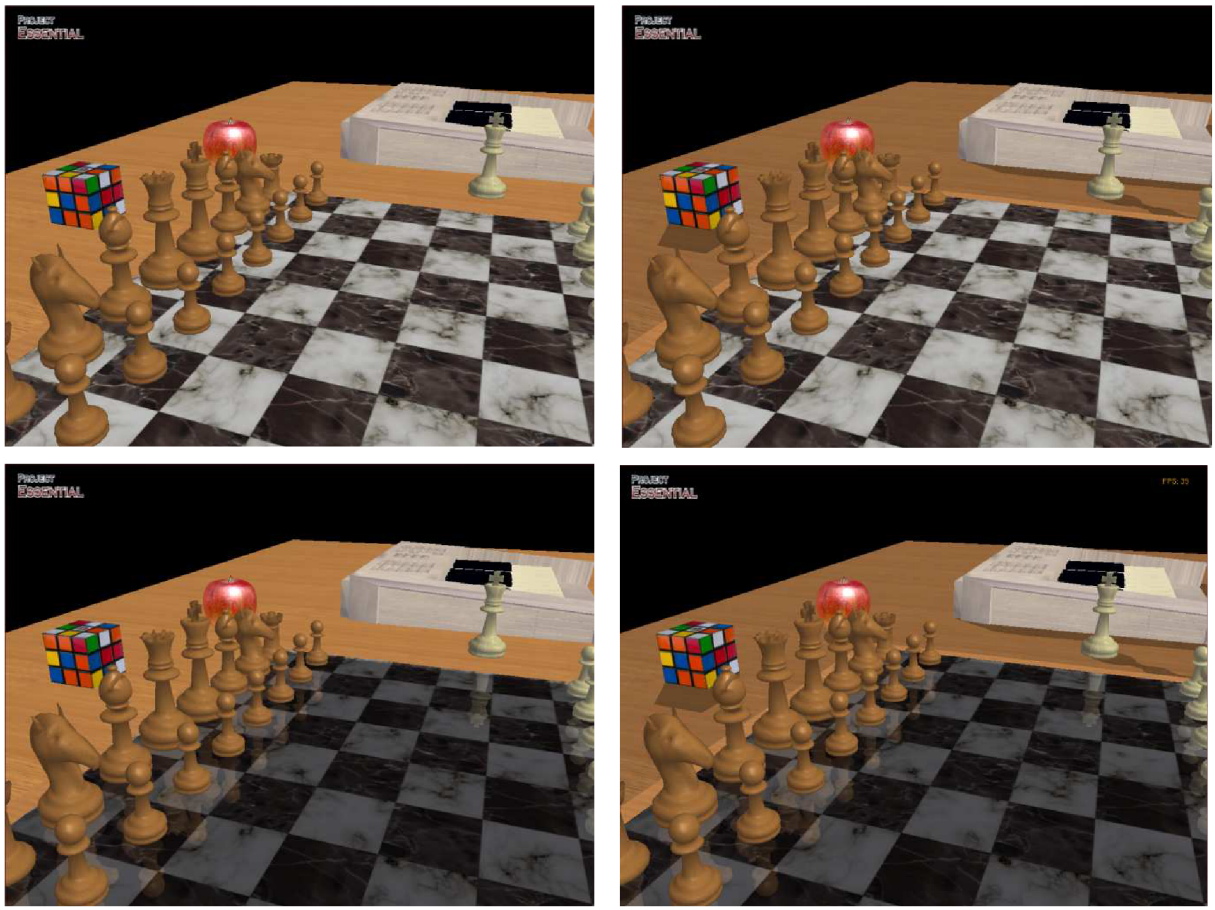
Existují různé jevy, které pomáhají lidskému zraku ve vnímání prostoru. Jedná se například o perspektivu, stíny, relativní velikost, vzájemná poloha, atmosférický vliv či pohybová paralaxa. Perspektiva je pro tvorbu 3D scény naprosto nezbytná a je samozřejmostí její implicitní přítomnost v OpenGL rozhraní. Samozřejmostí už nejsou stíny, které výrazně vnímání prostoru prohlubují. Stín napovídá, jak vypadá celkový tvar tělesa, můžeme díky němu vyvodit směr, ze kterého přichází světlo, a vztah, v jakém se nachází těleso vzhledem k podložce. Existuje mnoho metod, jak dotvořit projekci stínů do scény. Většinou platí, že čím složitější tato metoda je, tím lépe výsledek vypadá. Základní metodou je promítání stínů objektů scény do roviny. K implementaci této metody byl použit stencil buffer OpenGL. Rovinou příjemce v našem případě je rovina stolu, na kterém se nachází objekty k interakci. Po sestrojení stínové projekční matice je stín vykreslen v místech, kde je nastaven daný bit šablony, který byl zaznačen vykreslením tělesa do cílové roviny. Více o práci se stencil bufferem viz [34]. Výsledný dojem ze scény se po implementaci stínů o mnoho zlepšil a scéna působí mnohem plastičtěji. Jednou ze složitějších metod vrhání stínů je metoda stínových těles, anglicky označovaná „volume shadows“. Tato metoda by již značně přesahovala rozsah práce, ale byla by vhodným rozšířením, protože díky ní by bylo možné docílit reálných stínů vrhaných modelem ruky na předměty

[20] RLE – z angl. Run Length Encoding, je bezeztrátová komprese, která kóduje vstupní data tak jako dvojice *nd*, kde *n* je počet výskytů bezprostředně za sebou datové položky *d*.

[21] UV koordináty - jsou souřadnice v textuře v daném vrcholu

pod ní, a dosáhlo by se tak ještě lepšího výsledku při vnímání vztahu mezi pozicí ruky a okolními předměty.

Dostáváme scénu zobrazující otexturované modely, které vrhají stíny. Scéna vypadá dostatečně realisticky. Pro umocnění dojmu ze scény bylo renderování scény ještě rozšířeno o vykreslování zrcadlových odrazů těles. V našem případě se ve scéně nachází jeden zrcadlový povrch, a to mramorový povrch na šachovnici. Jako masku stencil bufferu tedy použijeme horní část šachovnice. Nastavením vhodné barvy a lesku materiálu docílíme vizuálního efektu, že povrch by skutečně odrážel světlo jako zrcadlo. Všechny předměty vykreslíme ještě jednou, ovšem nyní s převrácenou ypsilonovou souřadnicí (počítáme s horizontálním umístěním šachovnice - kdybychom počítali, že se bude se šachovnicí výrazně manipulovat, je vhodné převrátit pozici podle polohového vektoru šachovnice). Do výsledného obrazu zakreslíme pouze vymaskovanou část, odlesky se tedy budou nacházet jen na mramorové podložce. Vykreslení provedeme se zapnutým blendingem, alfa kanál nám určuje míru viditelnosti odlesku, čím vyšší zvolíme hodnotu, tím bude povrch lesklejší, naopak nízká hodnota alfa kanálu vytvoří efekt matného materiálu. Protože má povrch připomínat mramor, dáme odraženým objektům více modrý nádech, odrazový materiál tak bude působit chladně. Odlesky posunuly vizuální zpracování scény opět o stupeň výše. Rozdíly grafické úrovně aplikace v závislosti na vykreslení stínů a odlesků jsou patrné z následujících obrázků:



Obrázek 10: základní 3D scéna. Vlevo nahoře: vypnuté stíny, vypnuté odlesky. Vpravo nahoře: zapnuté stíny, vypnuté odlesky. Vlevo dole: vypnuté stíny, zapnuté odlesky. Vpravo dole: Zapnuté stíny, zapnuté odlesky.

5.1.2 Synchronizace grafického a fyzikálního modelu

Na rozdíl od skutečného světa nemá k sobě grafický a fyzikální model určitého objektu žádný vztah. Běžným postupem při renderování objektů, které se řídí fyzikálními zákony, je vypočítat v každém snímku pozici a orientaci pro všechny fyzikální modely scény a na základě těchto dat vykreslit grafické modely na patřičném místě. Bullet Physics programátorům tuto úlohu zásadně ulehčuje. Každé tuhé těleso[22] ve fyzikálním enginu Bullet má přidružený objekt třídy btMotionState, ve kterém jsou data o poloze v prostoru uložena. Volání metody *btMotionState::getWorldTransform* vrátí transformaci s pozičními daty. Používání MotionStatu přináší kromě usnadnění implementace i další výhody. U předmětů, které se nepohybují, se nové hodnoty MotionStatu zbytečně nepočítají.

Hodnoty MotionStatu odpovídají pozici těžiště předmětu, grafický model ale nemusí mít střed v těžišti tělesa. Z tohoto důvodu je potřeba u každého předmětu udržovat informaci o pozičním rozdílu mezi těžištěm a středem grafického modelu. Navíc mnoho modelovacích nástrojů nepoužívá pravotočivou verzi Kartézského souřadnicového systému. Proto byla vytvořena třída pro reprezentaci předmětu scény, která se všemi těmito aspekty počítá, a nese informace o grafické i fyzikální stránce modelu. Její deklarace vypadá takto:

```
class Object {

      int object_id;                              // ID předmětu
      int model_id;                               // ID grafického modelu
      int texture_id;                             // ID textury

      btCollisionShape* shape;                    // kolizní tvar
      btMotionState* motionState;                 // MotionState
      btRigidBody* body;                          // fyzikální model

      btVector3 trans;                            // grafický translační offset
      btQuaternion rot;                           // grafický rotační offset

public:

      Object::Object(int id, btCollisionShape* newShape, btMotionState*
      newMotionState, btRigidBody* newBody);
      Object::~Object();

      void setModel(int model);
      void setTrans(btVector3 * newOffset);
      void setRotation(btQuaternion * rotation);

      btMotionState* getMotionState();
      btRigidBody * getBody ();
      int getModel();
      btVector3 getTrans();
};
```

Údaj MotionState je v tomto případě ve třídě navíc, protože je možné jej získat z instance třídy *body*.

[22] Každé tuhé těleso nebo kolizní objekt, ukazatel na instanci třídy *btRigidBody** je přetypovatelný na *btCollisionObject **

Předmět ve scéně je renderován voláním následujícího kódu:

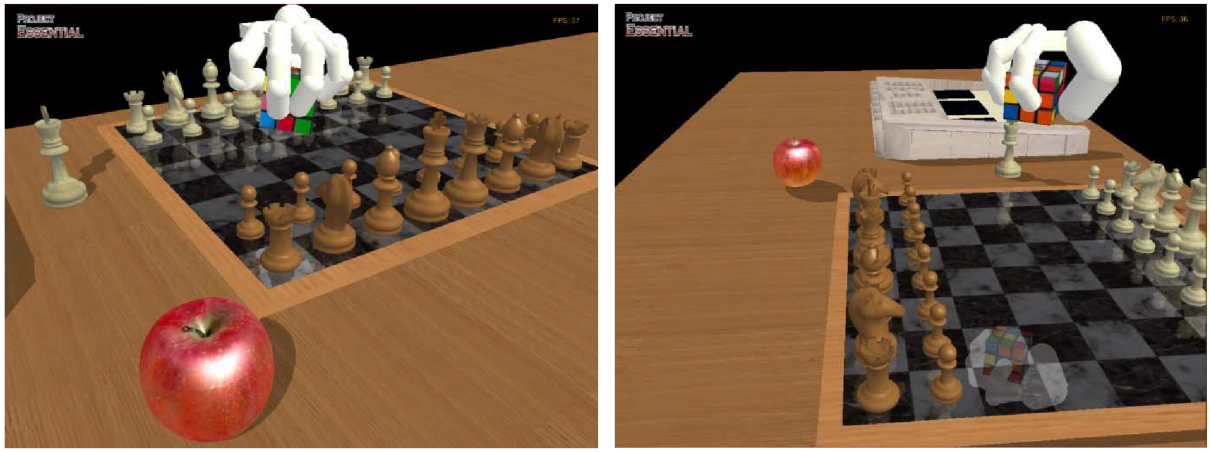
```
glPushMatrix();

      motionState = world_objects[item]->getMotionState();
      motionState->getWorldTransform(transform);
      origin = transform.getOrigin();
      graphicsOffset = world_objects[item]->getTrans();

      glTranslatef(origin.getX() + graphicsOffset.getX(),origin.getY()
      + graphicsOffset.getY(), origin.getZ() + graphicsOffset.getZ());

      rot = transform.getRotation();
      btVector3 axis = rot.getAxis();
      btScalar angle = rot.getAngle() * 180 /M_PI;

      glRotatef(angle, axis.getX(), axis.getY(), axis.getZ());

      glCallList(dListItem[world_objects[item]->getModel()]);

glPopMatrix();
```

Všimněme si zmiňované metody *getWorldTransform*. Bullet ji volá, pokud potřebuje získat transformaci nějaké tělesa. Pro kinematické těleso je metoda *getWorldTransform* volána v každém kroku simulace k zachování MotionStatu.

5.1.3 Animace ruky

Animace ruky je velmi komplexní problém. V této části budou popsaná pouze řešení z hlediska animace vizuální, fyzikální model a jeho animace budou popsány v kapitole 5.2.2.

Nejjednodušším řešením je využít MotionState popsaném v předchozí podkapitole. Tedy každému koliznímu objektu skeletu přiřadíme nějaký tvar nebo model a ten bude na základě pozic daných MotionStatem vykreslen. Výhodou kromě jednoduchosti tohoto řešení je přesné kopírování pohybů fyzikální modelu, což je opravdu zásadní plus. Proto tato možnost zůstala v projektu zakomponována, a stiskem mezerníku je možné přepínat mezi defaultní animací ruky reprezentující přesný fyzikální model a animací ruky vytvořenou milkShape modelem.

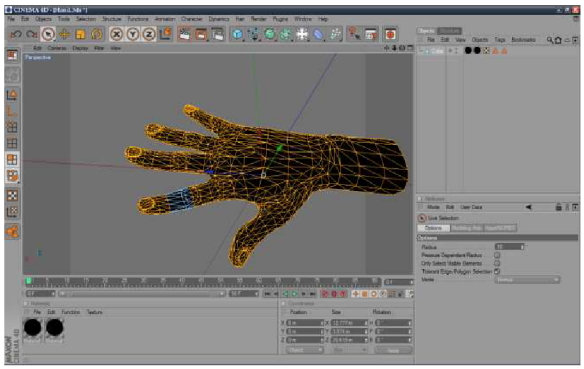


Obrázek 11: Grafická reprezentace fyzikálního modelu

Protože musel být fyzikální model upraven pro potřeby uchopování předmětů, jeho vizualizace se od skutečné lidské ruky liší. To vůbec nemusí být na škodu, protože například můžeme chtít simulovat ruku robotickou. Přesto se zaměříme i na tvorbu modelu lidské ruky. Jednak je to zajímavé téma a také chceme pomocí P5 glove přenášet na obrazovku pohyby lidské ruky uživatele, tedy chceme, aby simulace co nejvíce odpovídala skutečnosti. Navíc z důvodů uvedených dále v textu, dochází při pohybu fyzikálního modelu k výraznějšímu působení setrvačnosti na prsty, což vizuálně působí rušivě.

V první řadě si musíme uvědomit, že takovéto zpracování grafické reprezentace fyzikálního modelu bude klamavé. Tedy grafika nebude plně odpovídat fyzice, což může být matoucí. Na druhou stranu bude vykreslený model přesně odpovídat stavu ruky uživatele (samozřejmě v závislosti na přesnosti dat obdržených z P5 glove).

Začneme opět nejjednodušší variantou. Tou je nařezání grafického modelu na části, jejichž pozice je měněna podle dat z P5 glove. Vždy si uložíme modelovou matici a pohneme s celým modelem podle aktuální pozice rukavice, poté model orotujeme. Pro každý prst pak znovu uložíme modelovou matici. Každý článek prstu nejprve posuneme, poté, podle stupně ohnutí prstů, orotujeme. Až jsme na konci prstu, načteme modelovou matici zpět. Po vykreslení všech prstů vrátíme i matici uloženou před pohybem modelu. Toto řešení bohužel nevypadalo příliš hezky, protože při ohybu prstů byla příliš viditelná místa zlomu a vyčnívající polygony působily rušivě. Trochu tomuto řešení pomůže, umístíme-li do místa rotace článku prsu kouli o velikosti kloubu a vhodně ji otexturujeme. Místo modelu můžeme použít grafická primitiva, i když model není reálný, výsledek působí lépe. To že, při ohybu dochází k překryvu částí, nepůsobí již tak rušivě. Navíc přidáme-li opět funkcí *glutSolidSphere* či *gluSphere* dostatečně velké kulové kvadriky na místa ohybů, dojde ke schování tohoto nežádoucího jevu. Samozřejmě v tomto případě na sebe články nemusí přímo navozovat a nemusí k překryvům docházet vůbec. Bohužel tím se ale opět dostáváme do stavu, kdy nemáme věrnější kopii lidské ruky. Získali jsme řešení, které neodpovídá přesně fyzikálnímu modelu, ale vizualizuje přesný poziční stav rukavice P5 glove.



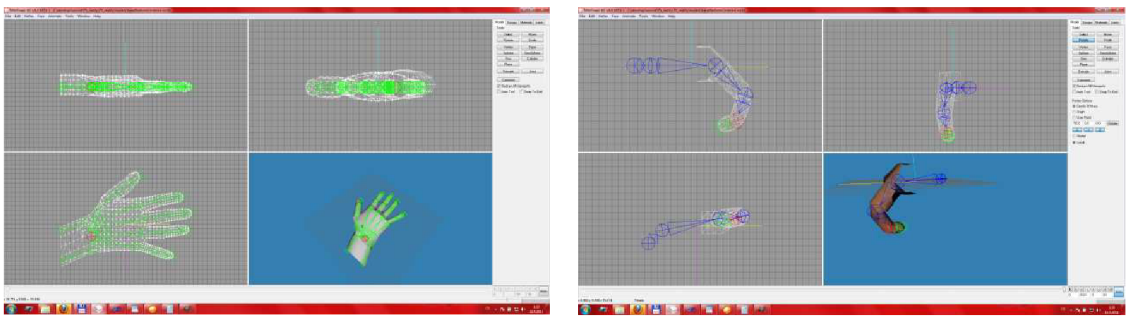
Obrázek 12: Řezání částí modelu v Cinama 4D

Nejúčinnějším metodou vizualizace je užití skeletální animace. Pro ni je potřeba vytvořit skeletální model objektu. K tvorbě modelu bylo využito nástroje MilkShape 3D, který se na modelování a tvorbu skeletálních animací úzce specializuje (viz kap. 4.5). Je možných více přístupů, jak model animovat. Je možné měnit kloubové proměnné a pozici kloubů přímo podle potřeby v aplikaci. Jednodušším přístupem je si požadovanou animaci dopředu v MilkShape 3D vytvořit. Problémem pak ovšem je, že animace je napevno daná a my se můžeme přemisťovat pouze mezi

snímky animace. Toho se využívá například při animacích, které se opakují a které je složité vytvořit přímo v aplikaci, protože se mění velký počet kloubových proměnných. Například máme-li běžící postavičku, animace je možné opakovat, dokud postavička běží, a vytvořit takovou animaci přímo v aplikaci je složité. V našem případě se nám též využijeme druhé řešení.

Nejdříve si uvědomme, co vlastně budeme animovat. Animace pohybu a otočení ruky docílíme pouhým posunem a rotací modelu, animovat tedy budeme jen prsty. Datová rukavice P5 nám vrací hodnotu od 0 do 63 pro každý prst, kde při hodnotě 0 je prst narovnaný a při 63 plně ohnutý. Vytvoříme tedy animaci o délce 64 snímků. V prvním snímku nastavíme kloubové proměnné tak, aby byla ruka zcela uvolněná, označíme tento snímek jako klíčový (keyframe). Naopak v posledním snímku animace nastavíme kloubové proměnné tak, aby byly prsty maximálně ohnuté. Opět snímek nastavíme jako klíčový. Ostatní snímky získáme na základě těchto dvou klíčových snímků pomocí interpolace.

Takto dostaneme model, který na základě toho, v jakém snímku se animace nachází, ohne stejně všechny prsty. My ale potřebujme, abychom animovali ohnutí prstu pro každý prst zvlášť. Není možné a ani by to nebylo nějak praktické, abychom vytvořili permutaci pro všechny vzájemné stavy všech pěti prstů. Navíc takto bychom přišli o výhodu plynulé animace. Model tedy byl rozdělen na pět části, čtyři prsty a dlaňová část s palcem. Animace palce je složitější než u ostatních prstů. Při uchopování totiž člověk palec pouze neohýbá, ale posouvá ho níže a přemisťuje ho tak, aby byl v opozici proti ostatním prstům, a umožnil tak úchop. Posledním krokem tvorby modelu je nanesení UV koordinát textur.



Obrázek 13: Tvorba skeletální animace v programu MilkShape 3D

Hotový model je třeba naimportovat do aplikace a získat z modelu animační data. Mete Ciragan, autor formátu, vydal přesnou specifikaci formátu ms3d napsanou přímo v jazyce C. Princip je podobný jako u jiných formátů, tedy lokalizovat vertexová a jiná data v souboru, rozparsovat je a zpracovat. Získáním animačních dat je možné nastavovat kloubové proměnné a animovat tak model. Není ale zajištěn hladký přechod mezi jednotlivými snímky. Tento problém se dá řešit pomocí interpolace quaternionů reprezentujících rotaci kloubu v minulém a aktuálním snímku animace. Postup výpočtu je popsán vztahy 15 až 17d v kapitole tři. Na tomto místě bych chtěl poděkovat Janu Kočímu, který je autorem tutoriálu pro načítání milkShape modelů v českém jazyce. Jeho kódy byly použity v projektu pro práci s formátem ms3d. Díky velmi dobrým komentářům v českém jazyce byla snadná

jejich modifikace pro potřeby této aplikace. Jan Kočí pro změnu děkuje Brettu Porterovi, který je pravděpodobně původním autorem tohoto algoritmu. Tento algoritmus je výjimečný právě užitím metody Slerp k interpolaci snímků. Díky tomu je dosaženo naprosto plynulé animace mezi snímky a výsledek po importu modelu do aplikace vypadá skutečně dobře. Každý snímač prstu P5 glove pak slouží jako posuvník animace (tedy jak dopředné, tak zpětné). Zcela korektní řešení by bylo užití velikosti kloubových proměnných fyzikálního modelu jako posuvníku. Uchopíme-li předmět, prsty po kontaktu s předmětem již nepokračují v ohybu. Ohnutí prstů tohoto grafického modelu však závisí na ohnutí prstů na P5 glove, tedy k zastavení ohybu při kontaktu s předmětem nedojde. Problém by řešilo užití metody *btHingeConstraint::getAngle*, která vrací aktuální úhel ohnutí v kloubu fyzikálního modelu. Bohužel fyzikální model, vlivem setrvačných a odstředivých sil, často překračuje vytyčené mezní úhly. Dostali bychom se tak na hodnoty, kde není animace definována. Tento grafický model není pro aplikaci vhodný. Pro správné uchopování předmětů ve scéně je pro uživatele důležité znát přesnou skutečnou pozici kolizních zón ruky. Je to ale vhodná demonstrace možností tvorby počítačové animace, navíc model přesně reflektuje stav uživatelovy ruky. Tvorba modelu ruky vyžaduje zkušeného modeláře. UV koordináty textur se nepodařilo na model nanést úplně přesně.

Obrázek 14: Grafická reprezentace ruky užitím MilkShape modelu.

5.1.4 Zvuk

Zvuk byl do aplikace přidán pomocí knihovny FMOD. Zvuk doplňuje některé animace menu. Je důležitou součástí základního tutoriálu učení gest. Doprovodný hlas k tutoriálům byl vytvořen elektronicky prostřednictvím programu IVONA, jehož funkce jsou on-line dostupné na domovském webu. Zvuky pro piano byly vytvořeny z reálné stupnice (část druhé a třetí oktávy) a rozkouskovány na krátké samply formátu WAV, aby zvuk odpovídal stisku jednotlivých kláves.

Potřebné zvuky jsou nahrané při startu aplikace, je možné je přehrát buď na základě identifikátoru samplu, nebo umístění v poli samplů. Přehrávání je možné na konkrétním kanálu, což přináší výhodu možnosti kontroly, zda není přehráván určitý zvuk. Toto je důležité především, aby se nekryly hlasové pokyny. Aplikace umožňuje deaktivace zvuku a změnu hlasitosti přehrávání.

5.2 Fyzikální model

5.2.1 Práce s Bullet Physics

Prvním krokem pro práci s Bullet Physics je vygenerování projektových souborů pro cílové vývojové prostředí přes CMake. Vygenerované moduly se podle potřeby (LinearMath a BulletCollision jsou esenciální) přidají do projektového řešení a je třeba stanovit projektové závislosti. Zdrojové texty Bulletu je třeba zkompilovat a ve formě statických knihoven přilinkovat do projektového řešení. Nyní je Bullet připraven k použití.

Při inicializaci fyziky je třeba vytvořit spojitý nebo diskrétní dynamický svět. Konstruktor objektu *btDiscreteDynamicsWorld* má 4 parametry. První je typu *btCollisionDispatcher*. V závislosti na typu kolizního objektu dispatcher zkoumá, zda nedošlo v kroku simulace ke srážce, a provádí výpočet kontaktních bodů. Dispatcher hledá kolize mezi všemi možnými dvojicemi objektů, složitost takového algoritmu je tedy $O(n^2)$. Výpočet je nutno akcelerovat.

Druhým parametrem je objekt typu *btBroadphaseInterface*, který pracuje právě jako akcelerátor dispatcheru. Broadphase provádí urychlovací výpočty, které slouží k rychlejšímu vyloučení páru. Ve verzi 2.76 existují 3 varianty, odzkoušeny byly dvě (třetí využívá akceleraci přes GPU CUDA[23] jádra). *BtDbvtBroadphase* broadphase založený na BVT stromu (kap. 4.3.1) podával v aplikaci výkonově lepší výsledek. Jednalo se řádově o jednotky framů.

Třetím parametrem konstruktoru je solver typu *btSequentialImpulseConstraintSolver* řešící působení gravitace, sil herní logiky, kolize a kloubní závislosti. Výpočty dispatcheru, broadphase i solveru mohou být na vícejádrovém systému paralelizovány. Posledním parametrem tvoří konfigurátor kolizí ovlivňující vlastnosti dispatcheru.

Krok simulace provedeme voláním metody *btDiscreteDynamicsWorld*::*stepSimulation* v každém snímku aplikace. Po každém kroku simulace sesynchronizujeme pozici grafických objektů. Volání *stepSimulation* ve vykreslovací smyčce aplikace s sebou přináší jednu příjemnou vlastnost. Metoda tak slouží jako omezovač framů, počítáme-li stav fyzikálního modelu šedesátkrát za vteřinu, dostaneme šedesát snímku, grafický akcelerátor se tak nezatěžuje zbytečnými výpočty navíc (bonusový snímek by byl stejně totožný s předcházejícím). Pokud výkon procesoru nezvládá vypočítat požadovaný počet kroků simulace, spolu s kroky simulace klesá i počet FPS.

V manuálu k Bullet Physics je uvedeno několik typů pro správnou práci enginu. Velikost pohybujícího se objektu nemá být menší než 0.2 jednotek (20 centimetrů), ve verzi 2.76 je tato minimální hodnota už 0.1. Protože prakticky všechny objekty ve scéně by tomuto neodpovídaly, bylo

[23] CUDA – angl. zkr. Compute Unified Device Architecture – paralelní výpočetní architektura vyvinutá nVidií. Umožňuje softwaru využít výpočetního enginu realizovaného na grafických akcelerátorech nVidia. Podpora CUDA přibyla do Bullet Physics ve verzi 2.73. Výkon CUDA demonstruje výpočet testovací scény pro 3 variace broadphase: CUDA (btCudaBroadphase) 6ms, OPCODE Array SAP 37ms, Bullet dynamic BVH (AABB tree, btDbvtBroadphase): 12ms

použito měřítko 1:10. To má za následek pomalejší pád objektů a lze to řešit úpravou hodnoty gravitace, není to však nutné. Tvůrci doporučují držet hmotnost objektů kolem čísla jedna, proto se hmotnost předmětů pohybuje většinou v rozmezí 1-3. Je dodržen i doporučený časový krok 1/60 vteřiny. Okraje kolizních objektů, důležité pro lepší výkon a stabilitu aplikace, by neměly být nastaveny na hodnotu nula. V tomto případě zůstala zachována defaultní hodnota 0.04. Není doporučeno použít *btHingeConstraint* a *btConeTwistLimit* pro tvorbu figurín. Místo toho radí autoři použít *btConeTwistConstraint*. Může se zdát, že model ruky částečně model figuríny připomíná. *BtConeTwistConstraint* byl ale přidán pro simulaci pohybu kloubů, jako jsou ramenní kloub nebo kyčel, proto pro tvorbu modelu ruky je užit *btHingeConstraint*, který se k tomuto účelu hodí lépe. Další tři doporučení z manuálu se týkají práce s mashes a nejsou pro projekt podstatné.

5.2.2 Fyzikální model ruky

V podkapitole 4.3.2 jsou popsány tři typy tuhých těles, které se v enginu Bullet vyskytují. Nabízí se několik možnosti, jak jich využít k sestrojení fyzikálního modelu ruky. Rovnou můžeme vyřadit užití těles statických. Naopak užití kinematického tuhého tělesa se přímo nabízí. Kdybychom totiž použili dynamického tělesa, museli bychom se vypořádat s gravitací, ale i toto řešení by mělo několik výhod. P5 glove je ale navíc datová rukavice bez zpětné odezvy, tedy tento případ přímo odpovídá modelu, kdy těleso působí na okolí, ale okolí nepůsobí na těleso. Uvažme tedy užití kinematického tělesa k sestrojení modelu ruky. Model ovšem nemusí být kinematicky celý. Následné variace byly implementovány, odzkoušeny a vybraná byla subjektivně nejlepší možnost.

První možností je udělat celou ruku, tedy dlaň i všechny články prstů jako kinematické. Toto řešení má jednu nespornou a velkou výhodu, a to, že model je absolutně přesný a programátor má stoprocentní kontrolu nad každou části takového objektu. Za nevýhodu pak může být považována nemožnost užití skeletálního systému. Kinematické těleso totiž nemůže být ovlivněno žádným pohybovým omezením, tedy ani kloubem (pokud jsou ovšem kinematická tělesa ve vzájemném kontaktu, ovlivňovat se mohou). Pozici každého článku prstu je pak potřeba nastavit v každém kroku zvlášť, postupem jako je například uveden v souvislosti s tvorbou grafického modelu. Z pohledu uživatele můžeme tuto nevýhodu považovat za irelevantní.

Uživatel ale pocítí druhou nevýhodu tohoto řešení. Stůl ve scéně je statický objekt. Objekty s nulovou hmotností, tedy statické a kinematické, mají v enginu vyšší prioritu a dynamická tělesa mají tendenci jim ustupovat. Dostane-li se pak dynamické těleso mezi dva takové objekty, může dojít k zatlačení objektu do obou takových těles. To má za následek grafické artefakty v obrazu. Dalším negativním efektem je odmrštění dynamického tělesa po uvolnění těchto sil působících na těleso. Dobře si ovšem tato varianta vede ve schopnosti držet předmět, protože nedochází k otvírání dlaně vlivem působením tíhy uchopeného předmětu. Jedná se o dobrý model, v jiných případech by se jednalo pravděpodobně o řešení nejlepší. Z důvodů nemožnosti příliš jemné manipulace s P5 a také

proto, že primárně je pod pojmem manipulace rozuměno uchopování než odstrkování předmětů, byl tento model upraven.

Úprava spočívala v ponechání dlaně jako kinematického tělesa, ale články prstů byly definovány jako dynamické. Na dynamické části můžeme použít pohybová omezení neboli constrainty (kap. 4.3.3). Nejblíže anatomickému modelu prstu odpovídá constraint *btHingeConstraint*, tedy osové omezení pohybu. Články lidských prstů se mohou ohýbat kolem horizontální osy v rozsahu přibližně 0 až 90 stupňů. Trochu jiná je situace u kloubů na hřbetu ruky, na které se prsty napojují. Ty umožňují i malý pohyb směrem do boku, především pak palec. To ale můžeme ignorovat, protože P5 glove nemá žádný senzor, kterým by tento pohyb snímala.

Dlaň ruky vytvoříme pomocí kolizního tvaru typu *btBoxShape*. Důležité je, nastavit ji jako kinematické těleso a zakázat její deaktivaci:

```
body->setCollisionFlags( body->getCollisionFlags() |
btCollisionObject::CF_KINEMATIC_OBJECT);
body->setActivationState(DISABLE_DEACTIVATION);
```

Články prstů je nejlépe reprezentovat tvarem kapsle. *BtCapsuleShape* se hodí nejlépe k tvorbě figurín (angl. ragdoll) a klouby spojených těles. Palec je tvořen dvěma tvary větší velikosti, ostatní prsty mají články tři. Po té, co konstruktor vytvoří a umístí všechna těla článků, následuje pro každý prst průchod cyklem, který sváže články do skeletu. Příklad svázání prvního článku s dlaní je vidět níže:

```
float fAngle = 4.1 * M_PI_4 * i / FINGER_NUM - (1.8 * M_PI_8);
if(i == 4) fAngle = 1.6 * M_PI * i / FINGER_NUM;
float fSin = sin(fAngle);
float fCos = cos(fAngle);

// joints between palm and first phalanxs
localA.setIdentity(); localB.setIdentity();
localA.getBasis().setEulerZYX(0,fAngle,0);
localA.setOrigin(btVector3(btScalar(fCos*palmSize), btScalar(fHeight),
btScalar(fSin*palmSize)));
localB = m_bodies[1+3*i]->getWorldTransform().inverse()
 * m_bodies[0]->getWorldTransform() * localA;

hingeC=new btHingeConstraint(*m_bodies[0],*m_bodies[1+3*i], localA,localB);
hingeC->setLimit(btScalar(limit), btScalar(limit));

m_joints[3*i] = hingeC;
m_ownerWorld->addConstraint(m_joints[3*i], true);
```

Proměnná *fAngle* určuje pootočení prstu kolem osy y. První část vzorce definuje mezery mezi prsty, část za znaménkem mínus míru odklonu od osy X. Je lepší zvolit vyšší diferenci mezi prsty, usnadní to uchopování předmětů. Vzdálenost prstů od středu dlaně je přepona pravoúhlého trojúhelníku vyjádřena vektorem `btVector3(btScalar(fCos*palmSize), btScalar(fHeight), btScalar(fSin*palmSize))`.

Důležité je, aby si umístění kloubů a kostí skeletu pozičně odpovídaly. Celý systém koordinát pro prst otočíme pomocí metody *setEulerZYX*. Tělo *m_bodies[0]* je dlaň, *m_bodies[1+3*i]* představuje první článek prstů. Kloubní spojení vytvoříme voláním konstruktoru třídy *btHingeConstraint.* Těla připojujeme přes čepy (pivot) nebo rámy (frame). Zde transformace *localA*, *localB* představují rámy, implicitně je *localB* umístěním kloubu. Rozšíříme-li volání konstruktoru *btHingeConstraint* o parametr true, bude kloubem transformace *localA*. V našem případě to znamená změnu otáčení o 180 stupňů. Metoda *setLimit* bude rozebrána podrobněji dále.

Nové kloubní spojení přidáme do dynamického světa, aby s ním mohl pracovat solver (viz 5.2.1), pomocí metody *addConstraint* . První parametr metody je typu *btTypedConstraint* * stejně jako prvky pole *m_joints*. *BtHingeConstraint** je na *btTypedConstraint** volně přetypovatelný. Za povšimnutí stojí hodnota true druhého parametru této metody. Hodnota true zde znamená, že nebudou řešeny vzájemné kolize mezi spojenými těly. Defaultně je tato hodnota false.

Konstruktorem třídy *Hand* vytvoříme model ruky. Animovat kinematické těleso je triviální a snadno proveditelné voláním metody *btMotionState::setWorldTransform*. Metoda je volána globální funkcí *getP5Data*, která ukládá data z P5 glove do objektu typu *Hand*, kde se data filtrují. Pro translaci jsou využita vyfiltrovaná data a vrácena zpět v podobě vektroru metodou *Hand::getCurrentPosition*. Vektor udává koncovou pozici modelu, tedy filtrovaná data z P5 posunutá o určitý offset a zvětšena multiplikátorem, aby rozsah snímače pokryl i rozsah scény.

Hodnota rotačního quaternionu je závislá na uživatelském nastavení. Defaultně jsou rotace vypnuté, protože P5 glove snímání rotací příliš dobře nezvládá a to působí nechtěné interakce se scénou. Pokud uživatel v menu pro P5 glove nastaví položku „Rotation“ na „roll only“, odemkne tím možnost rotace kolem osy X. Při odemčení všech os volbou „all rotation“ je nutné převést rotaci danou Eulerovými úhly na quaternion, se kterými Bullet pracuje. Převod rotací úhlů (zadaných ve stupních) na quaternion může být vykonán pomocí součinu dvou quaternionů. Všechny rotace úhlu jsou převedeny na úhly os obou quaternionů, které odpovídají Euklidovským osám. Úhly všech os jsou převedeny na quaterniony a vynásobeny [9]. Tím získáme quaternion reprezentující výslednou rotaci. Toto je velmi často potřebná operace při tvorbě počítačové grafiky, a proto je níže uveden její kód. Při programování aplikací se používá její zkrácený zápis, jehož obdoba v aplikaci vypadá následovně:

```
btQuaternion Hand::getCurrentRotation() {

float degree_to_rad = PI/180;

const float fSinPitch(sin((this->posData[ROLL]+180)*0.5*degree_to_rad));
const float fCosPitch(cos((this->posData[ROLL]+180)*0.5*degree_to_rad));
const float fSinYaw(sin(this->posData[YAW]*0.5*degree_to_rad));
const float fCosYaw(cos(this->posData[YAW]*0.5*degree_to_rad));
const float fSinRoll(sin((this->posData[PITCH]+180)*0.5*degree_to_rad));
const float fCosRoll(cos((this->posData[PITCH]+180)*0.5*degree_to_rad));
const float fCosPitchCosYaw(fCosPitch*fCosYaw);
const float fSinPitchSinYaw(fSinPitch*fSinYaw);

btQuaternion quaternion;
quaternion.setX(fSinRoll*fCosPitchCosYaw-fCosRoll*fSinPitchSinYaw);
quaternion.setY(fCosRoll*fSinPitch*fCosYaw+fSinRoll*fCosPitch*fSinYaw);
quaternion.setZ(fCosRoll*fCosPitch*fSinYaw-fSinRoll*fSinPitch*fCosYaw));
quaternion.setW(fCosRoll*fCosPitchCosYaw+fSinRoll*fSinPitchSinYaw));

return quaternion;
}
```

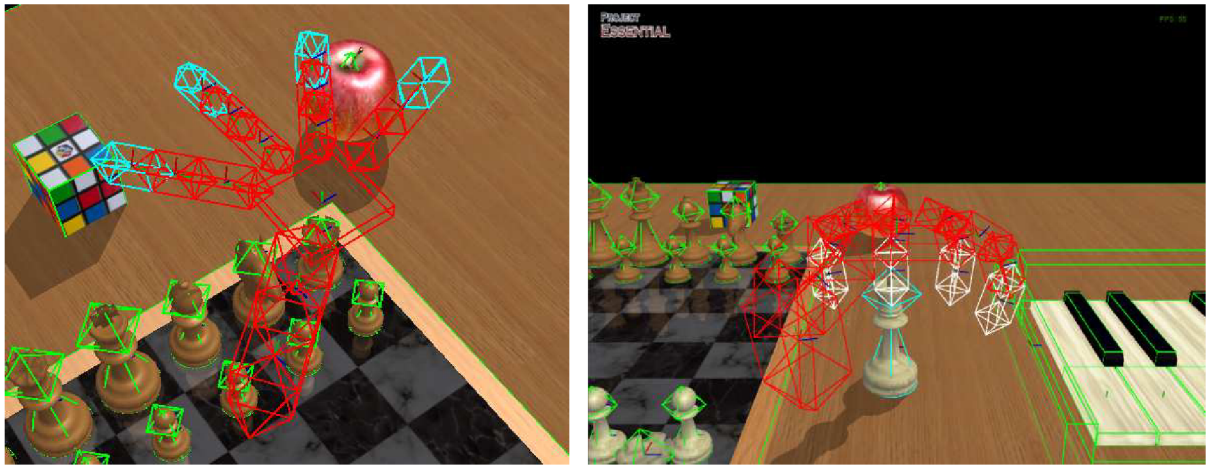
Tím je vyřešena translace a rotace modelu. Články prstů ruky, protože je model skeletálně propojen, následují dlaň. Zbývá dořešit jakým způsobem ohýbat prsty. Opustíme-li od možnosti měnit pozici článků přímo, nabízí se nám možnosti dvě. Jelikož Dual Mode drivery počítají i zrychlení a rychlost ohybu, je snadné užití metody *btHingeConstraint::enableAngularMotor*. Metoda má tři parametry: bool zda motor běží, skalární hodnotu rychlosti pohybu a skalár maximální velikosti impulzu motoru (síly motoru). Při rychlé změně mezi stažením a uvolněním prstů tato metoda reagovala místy zmatečně.

Naopak překvapila druhá možnost, která byla odzkoušena spíše jako alternativa. Jedná se o využití metody *btHingeConstraint::setLimit*. *SetLimit* má celkem pět parametrů z toho tři jsou přednastaveny, jeden je dokonce ponechán z důvodů kompatibility. Za experimentováním s těmito přednastavenými parametry byla snaha zpevnit kloub a zmírnit tak působení setrvačných sil na uchycené těleso. Metoda ale nejlépe pracovala právě s přednastavenými parametry, tedy ty nemá cenu měnit. Dva zbylé parametry, které je nutno zadat, určují velikost mezních úhlu pohybu – spodní a horní limit. Jestliže je hodnota spodního limitu menší než limitu horního, kloub poskytuje limitovaný pohyb v tomto rozsahu. Je-li hodnota spodního limitu větší než horního, kloub poskytuje v závěsné ose zcela volný pohyb. Jestliže se hodnoty rovnají, kloub se v tomto úhlu zamkne. Na základě dat o ohybu prstů tak můžeme nastavit tyto dva limity na hodnotu ohybu a články prstů zamknout v cílové pozici. Díky tomu, že se jedná o dynamická tělesa a působí na ně setrvačné síly, pohyb z jedné pozice zámku do druhé je plynulý. Tato možnost tedy byla pro animaci nakonec zvolena.

Shrňme tedy poznatky získané při tvorbě fyzikálního modelu. Je-li celý model kinematický, není ho možné skeletálně svázat, je však absolutně přesný. Při užití přesnější datové rukavice nebo při manipulaci s předměty větší velikostí, je tento model jasně nejlepší volbou. Dynamické části naproti

němu dovolují prudší náraz do objektu, protože se síla nárazu rozloží mezi narážející a zasažené těleso. Nedochází k prudkému odhazování předmětů. Uchopování drobnějších předmětů je tak snazší. Velkou nevýhodou je působení setrvačních sil, které způsobují pohyb prstů do stran. Neodpovídá to skutečnosti, degraduje to grafický model a občas způsobuje nechtěné kolize. U palce není problém, je tvořen dvěma velkými kvádry (resp. kapslemi), spojení tří článků za sebe již ale vytvoří dlouhý řetěz, kde je působení těchto sil znát. Dalším problém je volba hmotnosti takového článku. Čím vyšší hmotnost bude, tím budou nárazy při interakci drtivější, navíc bude i výraznější nechtěný setrvačný pohyb do boku. Naopak volba příliš nízké hmotnosti vede k tomu, že těleso působí na okolí příliš malou silou, ruka neudrží předměty nebo není dost silná na stisk klávesy piana. To se dá vyřešit právě užitím motoru s patřičně nastavenou silou, to ale přináší zase jiné nevýhody. Ideální by bylo zapracovat na prvním řešení a nějakým způsobem „změkčit“ sílu interakce. Stále je zde místo k experimentování.

Poslední věcí při tvorbě modelu, kterou bylo nutno vyřešit je rozložení prstů. Člověk při uchopování předmětu staví prsty v závislosti na tvaru cílového předmětu. Palec vždy působí silou proti zbylým prstům a umožňuje úchop. Živá tkáň je měkká a povrch prstu se přizpůsobuje hranám a povrchu tělesa. Navíc má kůže značný součinitel klidového a smykového tření. To jsou všechno věci, které nám usnadňují uchopování. Základem je udělat model s palcem v opozici a zbylé prsty hodně roztáhnout od sebe. Při sevření se tak vytvoří jakási klec, která předmět obejme. Toto ovšem neodpovídá anatomicky lidské ruce, proto si navzájem grafický a fyzikální model neodpovídají. Vylepšit by to mohlo oddělení animace palce od animace ostatních prstů. Užít na první článek palce místo *btHingeConstraint* omezení typu *btConeTwistConstraint* a palec při zvětšování míry ohybu postupně s ohýbáním do opozice teprve přesouvat. Implementačně však toto vyzkoušeno nebylo.



Obrázek 15: Fyzikální model ruky. Vlevo: Natažené prsty. Vpravo: Pokrčené prsty vytvoří klec obklopující uchopované těleso

Při použitém rozložení prstů se uchopování stalo poměrně dobře proveditelným, i když při vyvinutí příliš velké síly (tedy když uživatel i přesto, že už předmět drží, ještě více skrčí prsty) předmět často proklouzne mezi palcem a malíčkem. Zde výrazně chybí zpětná odezva ze strany P5 glove. I při ideálním tlaku vyvinutém na těleso docházelo dříve k postupnému vyklouzávání předmětu z úchopu ruky. To bylo vyřešeno zvýšením hodnoty tření (angl. friction) materiálu ruky. Fyzikální model k běžné interakci slouží dobře, při uchopování si vede dobře při použití klávesnice. Při použití P5 glove vyžaduje jistou míru cviku. Ovládání pomocí P5 je sice velmi intuitivní a pohodlné, ale uživatel si musí dát pozor, aby předmět zbytečně místo úchopu neodhodil prudkým pohybem ruky.

5.2.3 Piano

Piano ve scéně je vytvořeno voláním konstruktoru třídy *Piano*. Kolizní tvar obalující piano tvoří složený útvar typu *btCompoundShape*, na který jsou navázáni čtyři potomci *btBoxShape* tvořící základnu, zadní stěnu a dvě boční stěny piana. Na tom není nic výjimečného, ostatně všechny šachové figurky ve scéně jsou tvořené pomocí *btCompoundShape* mající dvě části: hlavičku šachové figurky tvoří kulový tvar *btSphereShape*, tělo figurky kužel *btConeShape*. Zajímavější je funkčnost piana. Klaviatura je složena ze sedmnácti *btBoxShape*. Původně bílé klávesy byly složené útvary, ale to pouze znesnadňovalo detekci kolizí. Jednotlivé klávesy jsou uchycené přes osový úchyt *btHingeConstraint* na svém konci. Odzkoušen byl i *btSliderConstraint*, klávesy se ale daly hůře stisknout. Na druhou stranu *btHingeConstraint* opět trpí svým neduhem bočního pohybu. Ten je umírněn tím, že klávesy nejsou navázány na tělo piana, ale jsou uchyceny bodově. Každá klávesa je třídy *Object*, má tedy svůj model, texturu apod. Černé klávesy mají hodnotu textury nastavenou na -1, tzn., že nejsou texturovány a k jejich vykreslení je použit pouze materiál.

Při běhu scény se v každém snímku kontroluje, zda nedošlo ke stisku klávesy. Opět se nabízí více možností, jak to udělat. Jednoduchým řešením by byla kontrola úhlu klapky vůči pianu. Toto řešení by ale z hlediska Bullet Physics nebylo zajímavé. Navíc je možné silou klapku zatlačit a přitom ji nenatočit. To by pro změnu dalo řešit dotazem na lokální transformaci klávesy. Téměř vždy je možné nalézt více řešení problémů, zaměřme se ale na užití Bulletu a detekce kolizí.

Základním způsobem jak zjistit, zda došlo v enginu ke kolizi, je prozkoumat manifold dispatcheru. Dispatcher provádí kontrolu kolizí mezi všemi páry, které propustí broadphase, a počítá konkrétní body kolize (viz kap. 5.2.1). Třídou pro ukládání kontaktní bodů je manifold. Průchodem všech kontaktní bodů, které obsahuje, je možné potvrdit nebo vyloučit kolizi dvou objektů. Mnohem efektivnějším způsobem je iterovat pouze skrz páry, které nás zajímají. To lze provést přes *btGhostObject*, který sleduje jen dráhy námi určených objektů. Alternativou je užití možnosti dotazování, která přibyla ve verzi 2.76. Místo dotazů na objekt typu *btCollisionWorld* nebo *btDiscreteDynamicsWorld* je nyní možné provést přímo kolizní test dvou objektů. Plusem tedy je, že se tyto dva objekty nemusí nalézat ve stejném dynamickém světě, nemusí být dokonce do tohoto typu

objektu přidány vůbec. Nesmíme ale zapomenout, že na rozdíl od předchozích případů se jedná o dotaz navíc, tedy o duplikaci dotazu. Proto je téměř nezbytné akcelerovat tyto dotazy užitím nějaké broadphase, např. *btDbvtBroadphase* nebo *btAxisSweep3*.

Pokud dotaz *btCollisionWorld::contactPairTest* je pozitivní, provede se volání callback funkce, která je třetím parametrem metody. Vytvoříme si tedy vlastní callback s užitím virtuální metody *addSingleResult* a zde si nadefinujeme kód, který má být v případě kolize spuštěn. V závislosti na tom, o jakou klávesu se jedná, přehrajeme patřičný zvuk. Aby se zvuk nepouštěl neustále dokola, nastavíme klávese, až do jejího uvolnění, flag disabled. Pro jistotu provedeme ještě kontrolu kanálu, na kterém je zvuk této klávesy přehráván. Jako kolizní pár můžeme testovat klávesu a prsty ruky, což je výpočetně náročnější varianta, a neodpovídá skutečnosti. Lepší je kontrolovat kolizi mezi klávesou a základnou piana, zvuk klávesy tak zazní, i když třeba na klaviaturu z výšky hodíme nějaký předmět ze scény. Nemusíme nutně promáčknout klávesu až na tělo základny. Metodou *contactPairTest* můžeme zjistit i vzdálenost mezi dvěma objekty, je-li vzdálenost nula, jsou tělesa v těsném kontaktu. Je tedy lepší nastavit jistou toleranci. Zvuk klávesy tak bude přehrán při stisku klávesy, ale klávesa nutně nemusí narazit až na tělo základny. V kódu pod textem představuje *BodyA* tělo piana, *bodyB* tělo aktuálně testované klávesy. Vzdálenost dvou nejbližších bodů těles *colObj0* a *colObj1* je uložena v proměnné typu btScalar *m_distance1* třídy *btManifoldPoint*.

```
struct btSoundResult : public btCollisionWorld::ContactResultCallback
{
   virtual   btScalar   addSingleResult(btManifoldPoint& cp,
   const btCollisionObject* colObj0,int partId0,int index0,
   const btCollisionObject* colObj1,int partId1,int index1)
   {
      btRigidBody * bodyA = piano->getCase();
      btRigidBody * bodyB = piano->getBody(active_key);

      if(bodyA == colObj0 && bodyB == colObj1 &&cp.m_distance1 < 0.1) {

         if(piano->isPressed(active_key) == false) {
            if(!sound->isPlaying(active_key + CHANNEL_OFFSET))
               sound->playSample(active_key + CHANNEL_OFFSET);
               piano->isPressed(active_key, true);
            }
      }
      . . .
      return 0;
   }
};

btSoundResult soundCallback;
m_dynamicsWorld->contactPairTest(colObjA,colObjB,soundCallback);
```

5.3 Využití P5 glove

5.3.1 Limitace P5 glove

Na úvod popisu programové práce s P5 glove rozebereme několik postřehů a zkušeností s touto datovou rukavicí. V kapitole 2.4.1 lze nalézt technické parametry tohoto zařízení. Oficiální čísla se od skutečných až tak o mnoho neliší. Poměrně mile překvapil dosah a záběr senzoru. V avizovaném dosahu 90-120cm má, při dobrém postavení, rukavice dobrý signál, a i za hranicí jednoho a půl metru je rukavice schopna pracovat bez ztráty signálu. Pokud v této vzdálenosti dojde k zakrytí klíčových diod, signál se ztratí a pro jeho znovuzískání je potřeba přiblížit rukavice zpět do vzdálenosti kolem jednoho metru. Klíčovými diodami budeme dále v textu rozumět dvojici diod nad malíčkem a prsteníčkem a diodu na hřbetu ruky. Jejich viditelnost je klíčová pro správné určování pozice. Každopádně hranice 120 centimetrů není nějak limitní. Kabel k zařízení je též dostatečně dlouhý a neomezuje uživatele v pohybu. Udávaný záběr 45 stupňů je ve skutečnosti také vyšší a zařízení je schopno snímat polohu rukavice na hranici úhlu přibližně šedesáti stupňů. To umožňuje chytat předměty ve scéně, které by byly běžně nedosažitelné, a musel by se tak zvyšovat pohybový násobič.

Tím výčet pozitivních zkušeností končí. Výrobce udává přesnost snímání pozice ve vzdálenosti 90 centimetrů asi 12.5 mm. Tato hodnota bohužel odpovídá skutečnosti a je těžké s takovou přesností snímání pozice pracovat. Navíc často dochází k tzv. „jitteringu“. Jittering označuje odchylku nebo posunutí některých částí pulzů vysokofrekvenčních digitálních signálů. U příjmu dat z P5 glove pod tímto termínem rozumíme kolísavé chvění kolem současné pozice, neschopnost ustálit se v určitém místě v prostoru. Podstatně větším problémem pak je náhle špatné přečtení pozice vlivem zakrytí nějaké z diod a skoková změna údajů. Toto uspokojivě řeší ovladače Dual Mode. Výrazně se zmenšila míra jitteringu. Přesnost určování pozice stále není dokonalá, ale je použitelná. Výrazně se s těmito ovladači zlepšila i přesnost určování rotací. Ta byla s původními ovladači taktéž nepoužitelná. Výrobce to zřejmě také věděl, a proto ani v aplikacích dodávaných s SDK, až na jedinou výjimku, rotace nepoužívá. Ani s novými ovladači se nedá na snímání rotací spolehnout. Příliš často dochází k obdržení naprosto náhodné hodnoty. Optické senzory ohybu prstů pracují přesně a bezproblémově. Bohužel celý dojem kazí nepoužitelný snímač palce. Pracuje ve velmi krátkém záběru, a vrací tak většinou jen mezní hodnoty. S tlačítky na těle rukavice není žádný problém a pracují správně. Uvedené skutečnosti musíme vzít v potaz a na jejich základě vyvíjet aplikaci. Přehledné shrnutí použitelnosti senzorů P5 glove je viditelné v tabulce.

Typ snímání	Přesnost snímání
Translace	Použitelná s výhradami
Rotace	Použitelná málo nebo vůbec
Snímání ohybu prstů (bez palce)	Výborná
Snímání ohybu palce	Použitelná málo nebo vůbec
Tlačítka	Výborná

Tabulka 3: Přesnost snímání P5 glove

5.3.2 Práce s SDK

V kapitole 4.4 této práce zazněla důležitá fakta o tom, jak vypadá třída pro práci s P5 glove. Nyní se podíváme a rozebereme si konkrétní rozdíl mezi prací s knihovnou originálních ovladačů a ovladačů Dual Mode na příkladu získání souřadnice X. V originálním SDK získáme souřadnici X následovně:

```
CP5DLL P5;
P5.P5_Init();
PosData[X] = P5.m_P5Devices[0].m_fx;
```

Struktura *m_P5Devices* je typu *P5data*. Všimněme si, že ovladače umí pracovat s více rukavicemi. Index pole nula označuje data z první připojené rukavice. Práce s SDK Dual Mode je mírně odlišná:

```
CP5DLL P5;
P5.P5_Init();

P5State *state;
state = P5_GetStatePointer(0);
PosData[X] =  state->x;                    // nefiltrovaná koordináta X
PosData[X] =  state->FilterPos[0];         // filtrovaná koordináta X
```

P5State je nová struktura, kterou Carl Kenner zavedl. *P5data* zůstala kvůli zpětné kompatibilitě zachována. *GetStatePointer(0)* opět značí data z první rukavice. Hlavní rozdíl při práci s Dual Mode spočívá v možnosti volat více než 40 funkcí, měnit velké množství nastavení a především struktura *P5State* je datově mnohem obsáhlejší než *P5data*. Též přístup k nízkoúrovňovým datům a hlavně jednotlivým LED je zásadní změnou.

Původně byla aplikace vyvíjena s originálními ovladači, tedy v relativním módu. Pomineme-li, že samotné ovladače byly nepřesné. Relativní mód se vůbec nehodí pro manipulaci 3D scénou. Naopak absolutní mód výborně odráží stav, kdy, za podmínky neměnné pozice přijímače, přemístí-li uživatel svoji ruku s P5 glove na určité místo, bude se vždy ve virtuální scéně nacházet ve stejném bodě. U relativního módu toto neplatí. Hlavním neduhem relativního módu je, že při rychlém pohybu nestíhá soustava snímat pohyb dostatečně rychle. Proto, urazíme-li stejnou vzdálenost jednou pomaleji a jednou rychleji, druhá hodnota naměřená snímačem v relativním módu bude o poznání menší. Relativní mód v Dual Mode pracuje o mnoho lépe, neumožňuje však zapnutí predikce. Navíc i autor

ovladačů doporučuje absolutní mód, proto byla aplikace dále vyvíjena v absolutním módu. Údaje o pozici obdržené aplikací od snímače nyní podstatně více odpovídají skutečnosti.

Rotace se získává ze struktury *P5State* nebo *P5data* podobně jako pozice. Dual Mode nabízí 16 možností, jak s daty rotací před uložením do *P5State* nakládat. Je-li hodnota tohoto nastavení rovna nule, Dual Mode vrací stejná data jako originální ovladače.

Hodnoty ohybu prstu se v datové struktuře *P5data* ukládala do pole proměnných typu unsigned char. Tento datový typ je pro další práci dosti nevhodný. *P5State* kromě do unsigned char zapisuje míru ohnutí prstů i do pole typu short. To, že byla aplikace původně vyvíjena s originálními ovladači, je důvodem, proč se místy v kódu stále objevuje tato nyní už zbytečná konverze.

Na horní části těla rukavice se nachází 4 tlačítka, označena abecedně písmeny A až D. Tlačítko D implicitně vypne či opětovně spustí komunikaci mezi věží a rukavicí. Explicitně lze však toto tlačítko použít pro potřeby aplikace. Informace o stavu tlačítek je opět udržována ve struktuře *P5State* nebo *P5data*. Při práci s tlačítky je třeba řešit pouze jedinou věc. Po celou dobu stisku nese flag stisku hodnotu true. To může vadit v případě, kdy musíme rozlišit, zda se jedná o jeden nebo opakovaný stisk. Tlačítko B v aplikaci většinou duplikuje funkci klávesy escape, jeho aktivace v menu působí opuštění aktuální nabídky. Změna nabídky změní stav menu, ve které ovšem B opět znamená posun o úroveň výše. Většinou není nutné se tímto zabývat, protože každá změna stavu menu je doprovázena animací, která po dobu svého průběhu zamkne příjem signálu z rukavice a klávesnice, a teprve po ukončení animace dojde ke změně stavu. Pro případ, kdy animace chybí, třída *Hand* obsahuje třírozměrné pole (funkce tlačítka D zůstala beze změny) s flagem určujícím, zda je tlačítko připraveno k použití. Flag je možné měnit voláním timeru. Drží-li uživatel tlačítko déle, než je hodnota timeru, lze to považovat za signál, že uživatel chce pokračovat v pohybu nabídkou nahoru. Jiné řešení je kontrolovat, jestli nebylo stisknuté tlačítko znovu uvolněno.

```
if(!hand->isEnabled(B) && !state->button[B]) hand->enableButton(B);
```

5.3.3 Ovládání 3D scény

Podle proměnných třídy *Hand* je v globální funkci *getP5data* nastavován stav fyzikálního modelu. O grafický model se stará metoda *Hand::drawHand* nebo *Hand::drawPhysicalModel* volaná ve vykreslovací smyčce. Fyzikální interakce jsou řešeny skrze metodu *stepSimulation* automaticky. Vše potřebné k ovládání 3D scény tedy je, aby funkce *getP5data* předávala údaje ze struktury *P5State* třídě *Hand*. Data zpracujeme tak, aby co nejvíce odpovídala našim potřebám. Pokud nechceme, aby fyzikální model ruky kopíroval pohyby v reálném světě v poměru jedna ku jedné, nastavíme poměr změny v reálném prostředí ku virtuálnímu.

Ovladače Dual Mode Beta 3 relativně dobře filtrují a průměrují snímaná data a přináší možnost predikce další pozice výskytu. Díky této predikci se velmi zvýšila přesnost a méně často dochází ke skokové změně pozice. Přesto k ní ale dochází. Jedná se o závažný problém, protože prudká změna pozice kinetického objektu způsobí odmrštění všech předmětů, se kterými přišel objekt do kontaktu a tedy k rozbití scény. Když se v dalším cyklu snímání vrátí kinematický objekt do původní pozice, nezbude často ve scéně už žádný předmět k interakci. Ani oprávněná rychlá změna pohybu není žádoucí pro uchopování a jemnou manipulaci s objekty. Proto třída *Hand* definuje hodnotu určující maximální chtěnou změnu. Pokud je v sekci menu týkající se nastavení rukavice zapnuta volba „Delta filtering“ (implicitně ano), dochází v každém kroku před uložením nové pozice ke kontrole, zda přijatá data neindikují změnu pozice o vyšší hodnotu, než je maximální povolená hodnota změny. Pokud ano, objekt se pohne ve směru změny, ne však o vypočtenou deltu, ale pouze o onu definovanou hodnotu. Porovnávání se děje až v třídě samotné, data s minulou pozicí jsou uloženy stejně jako data současné pozice v instanci třídy *Hand*. V každém kroku simulace pak virtuální model aproximuje blíže k pozici obdržené od přijímače. Jestliže se jednalo pouze o selhání a skokovou změnu pozice, nenaruší se tak scéna, jestliže se jednalo o chtěný prudký pohyb, virtuální model do cílové hodnoty v několika cyklech doaproximuje. Podobně funguje i deadband filtr ovladačů.

Tímto se možnost chycení předmětu ve scéně výrazně zlepšila. Problém je jak nastavit hodnotu změny, aby byla dost malá pro jemnou manipulaci a k uchopování, ale dost velká pro dostatečně rychlý pohyb po scéně. Pokud se zamyslíme nad tím, jak běžně chytáme věci na stole, dospějeme k závěru, že vykonáváme většinou rychlý pohyb v takové výšce nad stolem, kde nehrozí nechtěné shození předmětů, kdežto při samotném uchopování vykonáváme jemnější a pomalejší pohyb. Stejně Předmět napřed opatrně zvedneme, a poté, co je výškou nad úrovní ostatních předmětů, ho bez obav přemístíme. Tento model chování byl využit a zaimplementován tak, že velikost maximální změny je závislá na ypsilonové hodnotě pozice. Změna pozice v souřadném systému tedy probíhá na základě následujícího pravidla:

```
Pokud(abs(naměřená pozice-minulá pozice) < MAX_DELTA*(10-minulá výška))
      Současná pozice = naměřená pozice;
Jinak pokud (naměřená pozice - minulá pozice > 0)
     Současná pozice =  minulá pozice + MAX_DELTA * (10- minulá výška))
Jinak
     Současná pozice =  minulá pozice - MAX_DELTA * (10- minulá výška))
```

Má-li model vyšší ypsilonovou souřadnice než deset, umožňuje při změně pohybu překročit hodnotu delta, což nyní není na škodu, protože model je dost vysoko nad scénou, aby kdyby došlo ke skokové změně ve směru do scény, nedostane se model dost blízko k ostatním modelům během jednoho cyklu snímání. Tato úprava snímaných dat opravdu pomohla odstranit pocit prodlevy při rychlejších pohybech a zároveň zlepšuje schopnost uchopování.

Druhou metodou modifikující přijatá data je metoda upravující hodnotu stupně ohnutí palce. Zatímco ohyb ostatních prstů snímá rukavice spolehlivě, hodnota ohybu palce se většinou skokově pohybuje jen mezi 0 a 63 (maximální hodnota) a to tato hodnota ještě neodpovídá tomu, zda je palec skutečně skrčen nebo ne. Tento problém je nutné odstranit hlavně z toho důvodu, že palec je nezbytně důležitý pro uchopování. Nejde tedy ani tak o problém, kdy chceme mít palec natažený, ale rukavice odesílá data, že je skrčený. Skutečný problém nastane, když chceme uchopit předmět, ohneme palec, ale rukavice odesílá stále hodnotu nula. Metoda řešící tento problém pracuje jednoduše. Vezme data z ostatních čtyř prstů, a jestliže jejich míra ohnutí je u všech více než 35, znamená to, že se uživatel pravděpodobně snaží uchopit nějaký předmět. Pak jestliže hodnota ohybu v palci je méně než 35, vybereme nejnižší hodnotu ohybu z ostatních prstů a nastavíme míru ohybu palce na tuto hodnotu. Dostáváme tak hodnoty od 35 do 63. Tato úprava přijatých dat opět značně pomohla k mnohem lepší interakci.

Poslední co musíme udělat, je správně nastavit kameru, aby se scéna dobře ovládala a zároveň záběr dobře vypadal. Při loadingu aplikace jsou vytvořeny dvě instance třídy *Camera*, jedna pro 3D scénu, jedna pro menu. Kamera scény je s odstupem umístěna za modelem ruky a mění se společně s ní. Vzdálenost a úhel pohledu však nejsou zcela konstantní. Aby uživatel nekoukal na černá místa, pokud je ruka příliš vysoko, mění se pozice kamery v závislosti na výšce ruky nad stolem. Pokud je ruka hodně vysoko, kamera zabírá scénu z vrchu, aby byl vidět celý stůl, čím jde ruka níže, tím více se srovnává kamera do stejné výškové úrovně. Pokud svíráme předmět, kamera se na ruku zaměří blíže. Přiblížení kamery je řízeno průměrným ohybem všech prstů bez palce. Scéna je kompletní z grafického hlediska, je interaktivní a ovládaná přes P5 glove.

5.3.4 Rozpoznávání gest

Menu aplikace bylo navrženo tak, aby bylo graficky zajímavé. Základním konceptem jsou tři kostky v prostoru, na kterých se nachází volby. Průchod menu a změny položek jsou oživeny vizuálními efekty. Protože je aplikace navržena pro P5 glove, byla snaha, aby menu bylo ovladatelné skrze toto zařízení. Bylo tedy třeba implementovat gesta pro tyto operace: posun na další položku, posun na předcházející položku, výběr a návrat. Nejnižší úroveň menu je kvůli exaktnosti volby ovladatelná pouze skrze klávesnici.

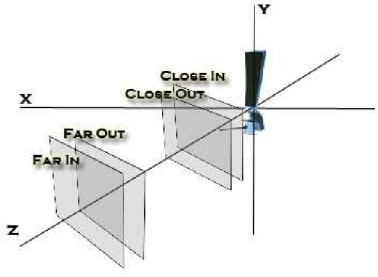
Zaměřme se nejdříve na tato čtyři základní gesta. Rozdělme si gesta podle toho, na jakém principu pracují na polohová, rotační, gesta s užitím prstů a gesta složená. Musíme si uvědomit, že pracujeme v absolutním módu. Při rozpoznávání gest to přináší nevýhodu při polohových gestech, kdy uživatel musí gesto provést v určité pozici vůči věži. Naopak hlavní nevýhoda relativního módu, že pro dva různě rychlé pohyby naměří dvě různé hodnoty, by se v tomto případě vytratila, dokonce je i přirozené, že čím rychleji člověk gesto provádí, tím více se rozmachuje. Problém by byl jinde. Zaprvé nevíme, kdy uživatel provádění gesta zahájil. Zadruhé by bylo třeba vytvořit zásobník se

zaznamenanými pozicemi, ty analyzovat a gesta v údajích vyhledávat. Zde má aplikace běžící v absolutním módu značnou výhodu. Zvolíme-li si totiž jednoduchá gesta, nemusíme takový zásobník konstruovat. Oproti relativnímu módu to má ještě jednu výhodu. Můžeme v určitém prostoru bezpečně rukou pohybovat, aniž bychom se museli bát, že příznak nějakého gesta spustíme.

Začneme gestem pro další položku v menu. Chceme naznačit odhození kostky (položky menu) doleva. Jednoduše stanovíme rovinu kolmou k ose X, která, když bude překročena, bude indikovat provedení gesta pohybu doleva. V následujícím okamžiku snímání, pokud by se uživatel nevrátil před danou rovinu, by došlo k další rozeznání toho samého gesta a k dalšímu posunu v nabídce. To je nežádoucí. Proto kromě současné pozice rukavice si budeme ukládat i data minulého měření. Podmínku tedy rozšíříme tak, že aby byla splněna, musí být nyní rukavice za hraniční rovinou, ale v minulém stavu musela být před ní. Toho, že by na hranici došlo k jitteringu, se bát nemusíme ze dvou důvodů. Díky zapnuté predikci snímač očekává další výskyt rukavice dále ve směru pohybu. Navíc první překročení této hranice způsobí spuštění animace odsunutí kostky, během této animace funkce getP5data nové údaje z rukavice ignoruje. Uživatel tedy dál pokračuje v pohybu za touto hranicí nebo se vrátí zpět. Gesto pro předchozí položku v menu je přesně zrcadlové.

Máchnutím rukou doleva nebo doprava měníme položky menu. Je třeba vytvořit gesto pro výběr položky. Využije gesta pracujícího s prsty. Výběr provedeme ukázáním na položku. Gesto je tedy provedeno natažením ukazováčku a pokrčením ostatních prstů. Hodnoty ohybu se pohybují mezi 0 až 63. Nastavíme-li hodnoty podmínky blíže mezním hodnotám, bude vyhodnocování gesta striktnější. Je lepší, když si dá uživatel s gestem více práce, než když ho provede omylem. Údaje ze snímače na palci jsou ignorovány, stejně jako v jiných gestech, protože jsou nespolehlivé.

Gesto pro návrat z nabídky je složeno z rotační a pohybové části. Protože je rotační část nespolehlivá, je možné, místo provedení gesta, stisknou tlačítko B. V animacích návratu do vyšší úrovně menu dochází většinou k pohybu kostek směrem ke kameře. Návrat tedy provedeme naznačením přitažení. Hřbet ruky musí směřovat směrem k zemi, ruka musí vykonat pohyb dozadu. Opět nastavíme hranici, která musí být překročena. Tato rovina bude opět striktní, velkou míru tolerance je ale dobré ponechat u rotace, minimálně 30 stupňů. Navíc si musíme dát pozor, že například hodnoty -5 a 355 znamenají stejnou míru rotace. Většinou volba kladného nebo záporného znaménka značí směr rotace, ale není to pravidlem.



Obrázek 16: Roviny pro rozpoznávání gest

Může se stát, že máme pohybová gesta, po kterých nenásleduje animace, a my nechceme spoléhat na to, že predikce zajistí hladký přechod přes hraniční rovinu. Nadefinujeme si tedy obecnou pozici ruky v prostoru. Pro každou osu můžeme nadefinovat tři stavy. Například pro osu X: vlevo, uprostřed, vpravo nebo pro osu Z: blízko, uprostřed, daleko. Pro každý přechod stanovíme dvě roviny, jednu označíme jako „in“, druhou „out“. Rovina „in“ je vždy dále od středu. Chceme-li se například dostat

z prostřední pozice do pozice označené jako „close“ (pozice blízko věži), musíme projít přes rovinu „close in“, která se nachází až za rovnoběžnou rovinou „close out“. Roviny jsou vždy kolmé na osy, ke kterým se jejich přechod má vztahovat. Máme-li udělat gesto odhození, jehož provedení vyžaduje udělat výpad proti věži z roviny „middle“ (prostřední rovina) do roviny „close“ (rovina blízká věži), musí se ruka nacházet napřed v obecné pozici „middle“ a přejít přes rovinu „close in“. Chceme-li gesto zopakovat, musíme napřed opustit obecnou pozici „close“ tím, že se dostaneme za hranice „close out“ a přejdeme tak do stavu „middle“. Nemusíme se tedy bát jitteringu. Využití přechodu přes dvě roviny je rovněž výhodné tam, kde se změnou obecné pozice změní způsob ovládání (viz např. galerie popsaná v 5.3.5.3). Nestane se tedy tak, že přejdeme do roviny „close“ a při gestu, u kterého provádíme například pohyb doleva, se nám obecná pozice lehce změní na „middle“.

Tabulka č. 3 obsahuje seznam základních gest, která aplikace rozeznává. V částech, kde se mluví o prstech, je vždy myšleno prsty mimo palec.

Gesto	Typ	Provedení	Užití
Move to left	translační	Ruka provádí pohyb doleva za hraniční rovinu	Posun na předchozí položku
Move to right	translační	Ruka provádí pohyb doprava za hraniční rovinu	Posun na následující položku
Push gesture	translační	Ruka provede pohyb směrem k věži	Odsunutí položky
Pull gesture	složené (translace + rotace)	Ruka provede pohyb směrem od věže, dlaň musí být natočená vzhůru	Zpět Přitažení položky
Pointing gesture	užití prstů	Ukazováček je natažený, zbylé prsty pokrčené	Volba položky Uchopení objektu
Rotate gestrure	složené (užití prstů + rotace)	Ukazováček a prostředníček jsou natažený, zbylé prsty pokrčené, ruka se otáčí podle osy Z	Otočení objektu
Grab gesture	složené (užití prstů + translace)	Všechny prsty jsou pokrčené, ruka provádí vertikální pohyb	Tažení objektu
Release gesture	užití prstů	Všechny prsty jsou natažené	Uvolnění uchopené objektu

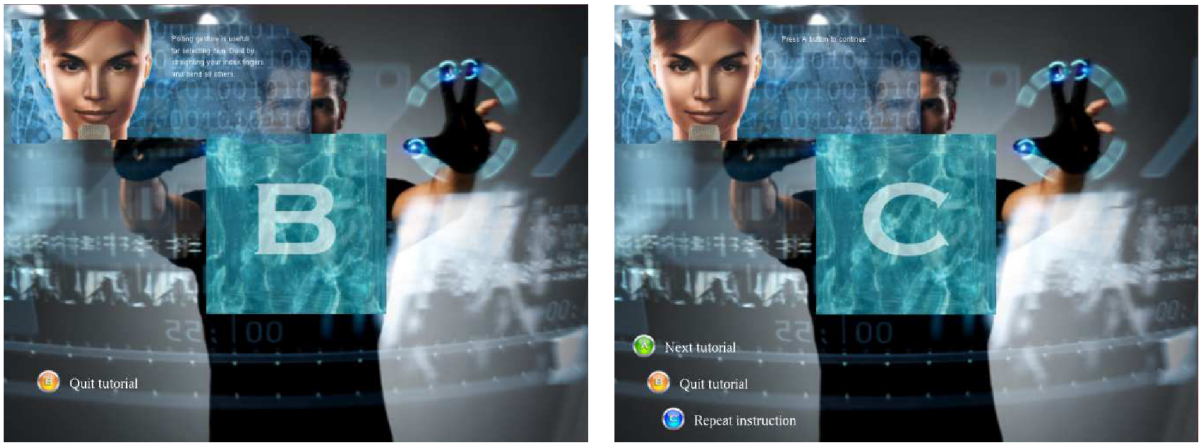
Tabulka 4: Přehled základních gest

5.3.5 Tutoriály

Rukavice P5 glove není zrovna ideálně přesná, ale ukázalo se, že pro práci s gesty funguje výborně. To bylo motivací ke vzniku několika miniaplikací, které demonstrují možnost využití P5 glove k rozpoznání gest. V okamžiku, kdy se člověk naučí provádět gesta správně, je ovládání přes P5 glove mnohdy pohodlnější než klávesnice či myš.

5.3.5.1 Základní tutoriál výuky gest

V základním tutoriálu pro výuku gest jsou s uživatelem projita a vysvětlena gesta pohybu na předcházející položku, pohybu na následující položku a gesto výběru. Uživatel má k dispozici tři kostky označené písmeny A, B a C, mezi kterými může vybírat, gesta se postupně odemykají. Instrukce jsou předkládány jak v hlasové, tak v obrazové formě. Za správný postup je uživatel pochválen. Hlas byl vytvořen elektronicky prostřednictvím programu IVONA. Tlačítkem B je možno kdykoliv tutoriál ukončit. Tlačítkem C si může uživatel nechat zopakovat instrukce.



Obrázek 17: Výuka základních gest

5.3.5.2 Práce s textovou informací

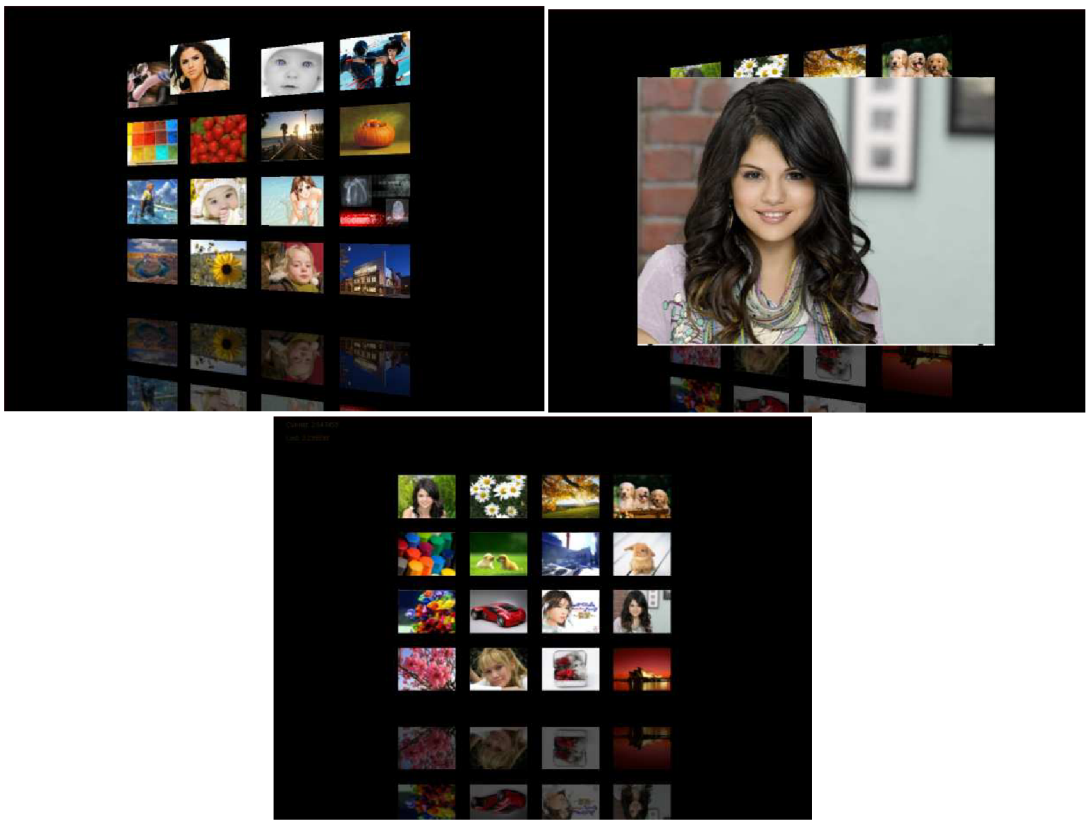
Uživatel je seznámen jak pracovat s textem. Gestem uchopení (grab gesture), které se provede zavřením dlaně (palec je opět ignorován), je možné uchopit text, následným pohybem nahoru nebo dolů dochází ke scrollování textu. K tomu se využívá pole obsahující minulou pozici P5 v třídě *Hand*. V okamžiku, kdy dojde k úchopu, se toto pole přestane aktualizovat. Posun textu je pak dán rozdílem mezi současnou pozicí a posledně uloženou pozicí v poli *lastPosData*. Při uvolnění prstů (gesto release) se informace o posunu zaznamená do instance třídy *Book*. Třída *Book* obsahuje informace o názvu knihy, autorovi, ID knihy, identifikátor textury obalu knihy, počet stran, informaci o tom, která strana je aktuálně prohlížena, a údaj o posunutí textu. Pokud je text stažen příliš směrem dolů, po uvolnění pěsti text plynule doscrolluje zpět. Pohybem ruky doleva a doprava je v knize možno listovat. Chceme-li změnit knihu, pomocí gesta odstrčení knihu odhodíme. Mezi knihami vybíráme opět gestem pohybu doleva a doprava. Máme-li vybráno, knihu otevřeme přitažení (viz tabulka č. 4).

Obrázek 18: Ukázka práce s textem

5.3.5.3 Galerie

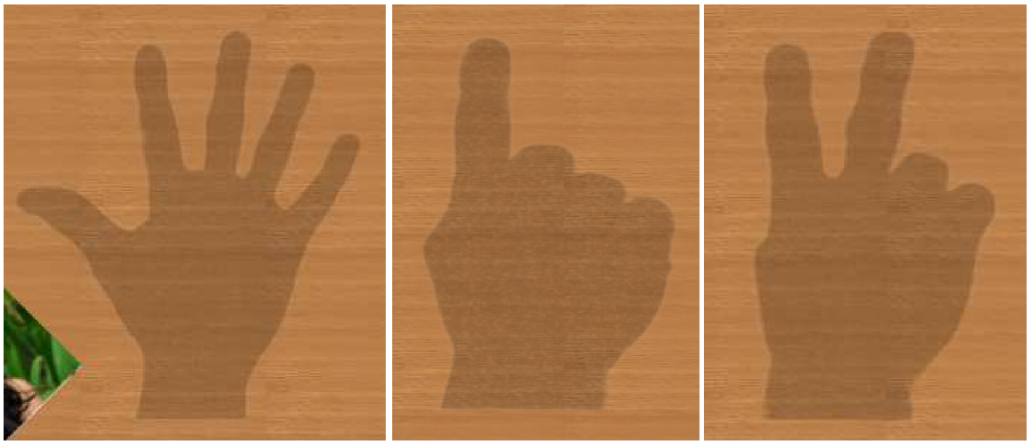
Další tutoriál ukazuje možnosti ovládání obrazové galerie pomocí P5 glove. Galerie je tvořena maticí obrázků 4x4. Galerie jsou k dispozici dvě, načtení 32 obrázků navíc zpomalovalo spouštění aplikace, proto jsou obrazová data pro galerie načtena až po spuštění toho tutoriálu. Pro lepší vizuální dojem je galerie lehce vytočená a umocňuje tak dojem z prostoru, směrem dolů se zrcadlí.

Výběr obrázku probíhá změnou pozice ruky. Představme si před snímačem virtuální matici 4x4, ukazováním na jednotlivé položky v této virtuální matici měníme pozici kurzoru v galerii. Obrázek, na který právě ukazujeme, poznáme podle toho, že vystupuje z matice směrem k nám. Chceme-li se na vybraný obrázek podívat zblízka, provedeme gesto výběru natažením ukazováčku a pokrčením zbytku prstů. Pro zrušení detailu odhodíme prohlíženou fotografii gestem pohybu doleva. Chceme-li vyměnit celou sadu obrázků (tedy přejít na jinou galerii), přiblížíme ruku k věži. Tím uchopíme galerii. Úchop je doprovázen animací, při které se galerie srovná do roviny. Nyní můžeme galerii táhnout do boku podobně jako v případě uchopení textu. Odtáhneme-li sadu obrázků za hraniční rovinu, přiletí z druhého směru galerie nová. Tedy za podmínky, že se tím směrem další galerie nachází, jinak se galerie vrátí původní pozici. Návrat je doprovázen animací. Stejný princip funguje i u boxů a knih. Chceme-li držení galerie uvolnit a vrátit se k prohlížení, odtáhneme ruku zpět od věže. Zde je typický příklad užití dvou rovin „close in“, „close out“. Jdeme-li ze středního pásma za rovinu „close in“, dojde k uchopení galerie. K uvolnění a přechodu zpět do středního pásma dojde až po návratu až za rovinu „close out“. Stiskem tlačítka A postoupíme na další typ galerie.

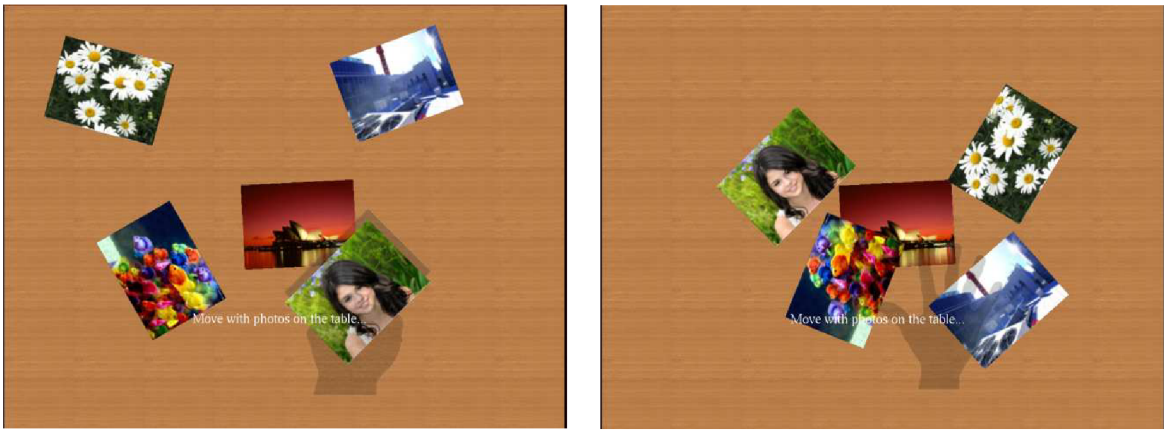


Obrázek 19: Galerie. Vlevo nahoře: Základní stav, kurzor ukazuje na druhý obrázek v první řadě. Vpravo nahoře: Detail obrázku. Dole uprostřed: Galerie je uchopena

Dalším typem galerie jsou fotky rozházené na stole. Proměnné třídy *Photo* nesou informaci o textuře obrázku, výšce obrázku, šířce obrázku, souřadnici z, která umožňuje vrstvit fotografie na sebe, a tři poziční údaje (osa x, osa y, rotace). Na obrázku č. 20 můžeme vidět tři stavy rukavice P5 glove, které sledujeme. Jsou-li prsty uvolněné, volně se pohybujeme po ploše stolu. Gestem výběru zvedneme fotografii, která se nalézá pod ukazováčkem, pokud taková existuje. Výběr není přesný, nepočítá se s rotací fotografie, ale s jejími rozměry ano. Pokud se na místě výběru nachází dvě fotografie, uchopí se ta vrchní. Uchopíme-li spodní fotografii, vytáhneme ji navrch. Algoritmus funguje tak, že vybrané fotografii přiřadí nejvyšší souřadnici z, a všechny fotografie, které měly vyšší souřadnici z než fotografie před úchopem, jsou posunuty o stupeň dolů. Po uvolnění se uloží nová pozice fotografie. Vztyčíme-li ukazováček i prostředníček (rotate gesture), můžeme vybranou fotografii orotovat. Stiskem tlačíka A opět postoupíme dále v tutoriálu.



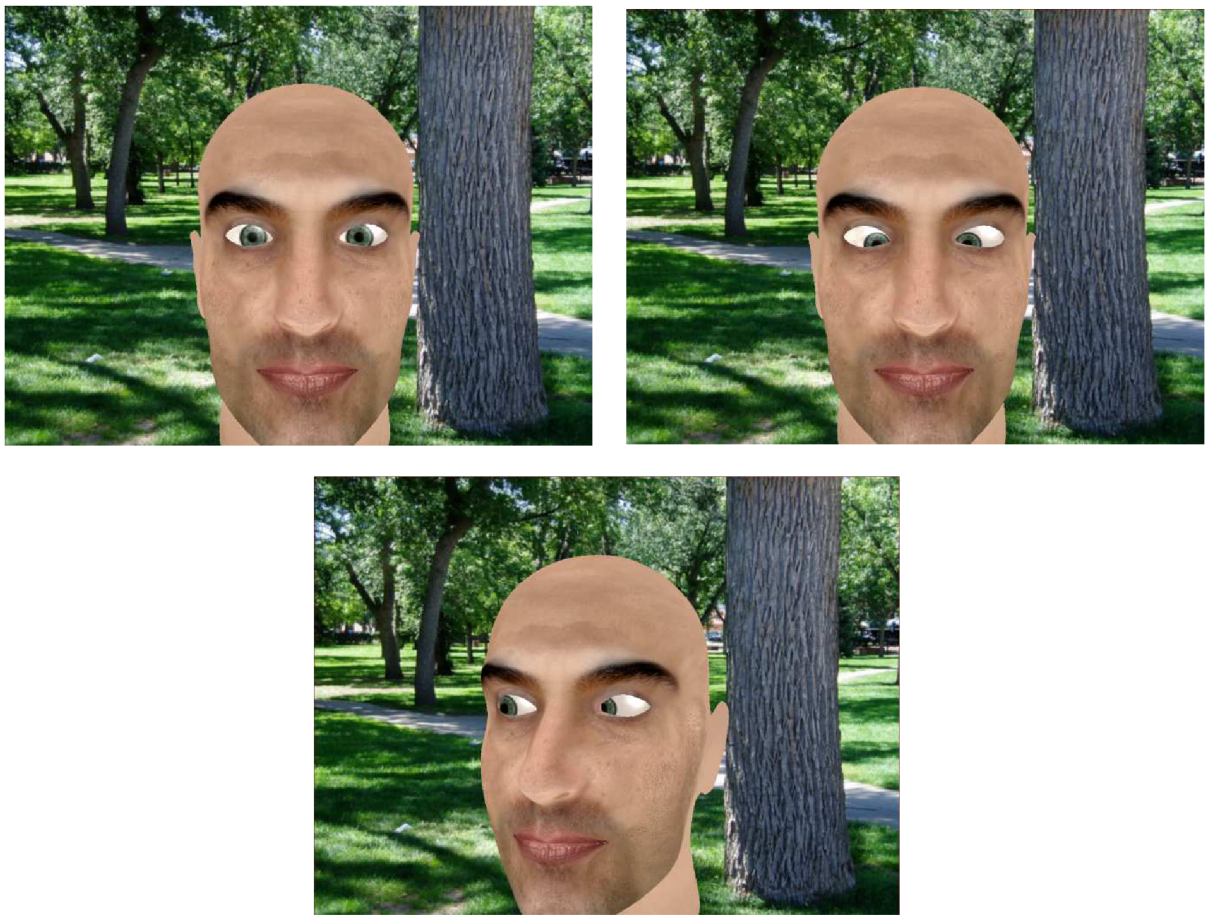
Obrázek 20: Tři stavy ruky - výběr, přesun, rotace



Obrázek 21: Vlevo: uchopení fotografie. Vpravo: Fotky po přeskládání

Poslední ukázkou práce s P5 glove je scéna s mužem, který sleduje uživatelovu ruku. Model s prázdnými očními důlky doplníme o dvě kvadriky typu *glutSphere*. Na ty naneseme pomocí sférického mapování textur texturu oka. Bohužel v tomto případě nemůžeme použít jednoduchou manipulaci s modelem. Ke změně pohledu tedy nehýbeme s modelem oka ale s texturou přepnutím zobrazovací matice z MODEL_VIEW módu na GL_TEXTURE. Koordinátu textury vypočítáme tím, že vypočítáme tangens úhlu, kde protilehlou stranou rozumíme vzdálenost mezi pozicí oka a

souřadnicí rukavice X, a přilehlou stranou rozumíme vzdálenost mezi pozicí oka a souřadnicí Z. Tím docílíme, že postava skutečně naši ruku sleduje. Přiblížíme-li ruku blízko k věži, postava dá oči k sobě, vzdalujeme-li ruku po ose Z dál od věže, postava postupně dává oči od sebe. Nastavíme chtěné mezní úhly. Jestliže by postava musela překročit mezní úhly, aby na ruku viděla, začne otáčet hlavou. Stiskem tlačítka B ukončíme ukázku.



Obrázek 22: Muž sledující vaši ruku Vlevo nahoře: Základní pozice. Vpravo nahoře: Rukavice P5 je v těsné blízkosti věže. Dole uprostřed: Rukavice je vlevo za mezním úhlem očí

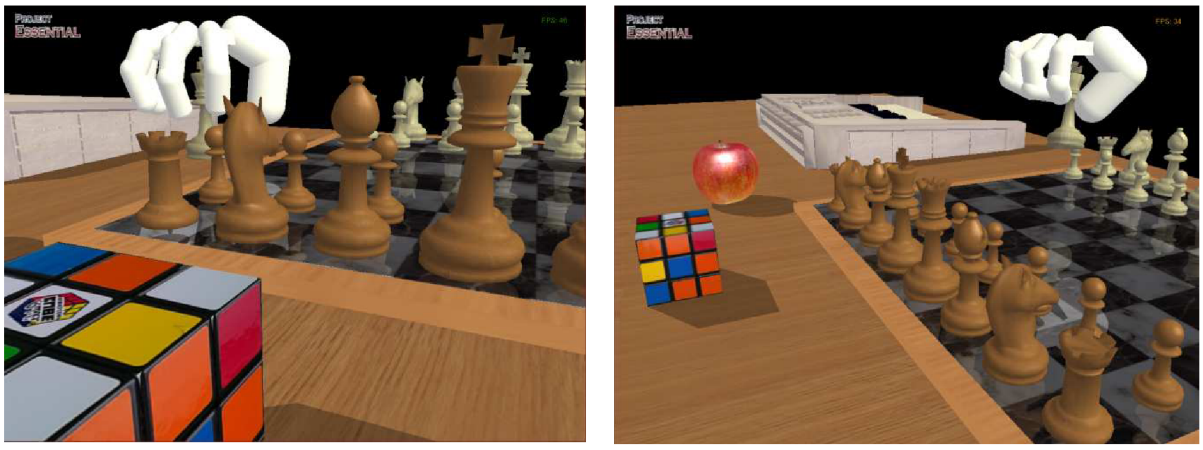
Mohli bychom vytvořit mnoho dalších ukázkových aplikací. Ovládání pomocí P5 je opravdu velmi intuitivní, abychom pokryli všechny data, která rukavice vrací, potřebovali bychom k tomu 26 kláves na klávesnici (10 pro ohyb prstů, 6 pro změnu pozice, 6 pro změnu rotace a 4 tlačítka na hřbetu rukavice).

5.4 Shrnutí výsledků a zkušeností

Hlavním zaměřením práce měla být rukavice P5 glove. P5 glove je dobré zařízení pro demonstraci možností datových rukavic, na tvorbu 3D scény ale není příliš vhodná. Problém spočívá v její velké nepřesnosti ve snímání pohybu a hlavně rotací. Pracujeme-li s enginem postaveným na skutečných fyzikálních zákonech, potřebujeme nástroj mnohem přesnější. Pokud už zvolíme jako vstupní zařízení P5 glove a chceme realizovat 3D scénu, je vhodné do scény umisťovat větší předměty. Uchopování předmětů realizované přes P5 glove je složitý úkol, rukavice se spíše hodí k manipulaci ve smyslu posouvání předmětů. Dalším doporučením vytvořit prostor scény velikostně odpovídající prostoru, který pokrývá senzor rukavice, tedy vytvořit poměr reálného světu k virtuálnímu jedna ku jedné. Zmírní se tak dopad nepřesného snímání. Také fyzikální model světa je lepší uzpůsobit více potřebám P5 glove než realitě.

Uživatel sám může funkčnosti P5 glove pomoci správným zacházením. Jde především o to, vykonávat s rukavicí pomalé pohyby a držet rukavici v pozici, kdy jsou dobře viditelné klíčové diody. To je dioda nad malíčkem, prsteníčkem a na hřbetu ruky. Programátor může posunout celý souřadný systém směrem dolů po ypsilonové souřadnici. Uživatel pak musí snímací věž umístit nad úroveň svého stolu, aby měl volný prostor pro manipulaci pod věží. Zajistí se tím ale lepší viditelnost klíčových diod. Právě umístění klíčových diod na přední části rukavice je jeden z důvodů, proč, pokud už je nějaké snímání rotace použitelné, je to právě rotace kolem osy Z.

Výstup implementační části práce je aplikace, která obsahuje velké množství implementovaných technik z oboru počítačové grafiky, obsahuje příklady skeletální animace, demonstruje možnosti užití populárního fyzikálního enginu Bullet Physics a ukazuje možnosti užití P5 glove jako vstupního zařízení. Navzdory tomu, že aplikace obsahuje 3D scénu postavenou na reálné fyzikální simulaci, manipulace se scénou pomocí P5 glove je celkem uspokojivá. Na různých mini-aplikacích jsou poté demonstrovány příklady užití, kde přesnost P5 glove postačuje. Získané poznatky jsou zmapovány v tomto dokumentu a spolu se zdrojovými kódy poslouží jako inspirace a zdroj informací potencionálním zájemcům, kteří se chtějí dozvědět něco práci s datovými rukavicemi.



Obrázek 23: Výsledná 3D scéna

6 Závěr

Tato práce se zaměřuje na periferní zařízení P5 glove, principy užití a velké množství poznatků se však vztahují i na jiné datové rukavice. Cílem implementace měla být 3D scéna, která by umožňovala interakci s objekty za pomocí rukavice P5. Tento cíl byl splněn. Audiovizuální stránka scény je na dobré úrovni, objekty ve scéně byly zvoleny tak, aby tvořily širší spektrum příkladů. K ovládání scény lze užít jak P5 glove, tak klávesnici. Ovládání pomocí P5 vyžaduje určitou míru cviku, a i tak není zcela přesvědčivé. Na rozdíl od klávesnice, pomocí které se scéna ovládá mnohem přesněji a lépe, vykazuje na druhou stranu P5 glove lepší intuitivnost, a díky zpřesňujícím řešením popsaných v předešlé kapitole, opravující některé nedostatky, snímání dat z P5, je tento způsob interakce i rychlejší.

Při implementaci menu aplikace se ukázala silná stránka P5 glove, a to je její využiti pro monitorování gest provedených uživatelem. Na základě této zkušenosti byl stávající program rozšířen o tutoriál čítající výuku základních gest, 3 různé typy galerií ovládaných gesty a tutoriál ukazující možnosti využití gest pro manipulaci s textovou informací. Komplikace spojené s gesty a zjištěné poznatky jsou rozvedeny v kapitole 5.3.4. Tím byla prozkoumána další množina možností, které zařízení nabízí. Užívání gest je i přes nedostatky testovaného zařízení rychlejší, intuitivnější a pohodlnější než obsluha přes klávesnici či myš.

Otázkou tedy je, zda je datová rukavice P5 glove smysluplné zařízení a zda má své místo na trhu. Rok 2010 vnesl do světa informačních technologií velmi zajímavý produkt společnosti Microsoft. Pohybový systém Kinect i přes to, že byl uveden pouze exkluzivně pro herní konzoli Xbox 360, se téměř okamžitě zapsal do Guinessovy knihy rekordů jako nejrychleji se prodávající zařízení v oblasti spotřební elektroniky. Kinect je zařízení pracující na bázi zpracování obrazu, jeho srovnání s P5 glove je však na místě, protože výrobky se nachází v podobné cenové hladině a dají se využít ke stejným účelům. Kinect oproti P5 glove přináší mnoho výhod. Pomineme-li jako výhodu jeho dobrou dostupnost a velkou softwarovou podporu, za výhody stěžejní můžeme považovat, že snímanou oblastí není pouze ruka, ale celé tělo, a hlavně pro komunikaci s počítačem si uživatel nepotřebuje nasazovat ani oblékat žádné zařízení a nic, co by při obsluze počítače uživatele omezovalo. Za nevýhodu lze považovat slabší přesnost snímání malých částí těla, jakými jsou prsty, a větší náchylnost ke ztrátě informace o pozici při částečném zakrytí snímače.

V posledních letech se ovládání přístrojů pomocí gest stalo velmi populárním a značně se tak ovlivnilo vnímání a vývoj komunikace mezi uživatelem a strojem. Dopomohlo k tomu především masové rozšíření přístrojů s kapacitními dotykovými displeji, přístrojů s mikrofony a přístrojů s kamerami umožňující právě rozpoznávání obrazu. Mají tedy datové rukavice a datové obleky budoucnost? Ano a velkou. Všem těmto výše zmíněným zařízením chybí jedna podstatná věc, která je pro intuitivní interakci uživatele s virtuální realitou podstatná, a tou je zpětná odezva. P5 glove nemá žádné

servomotory ani tlakové efektory, dražší haptické systémy ale ano. P5 glove tak dobře plní úlohu levného přístroje pro akademické účely a umožňuje si vyzkoušet práci s datovou rukavicí. Pro praktické aplikace by se pak použily dražší, přesnější modely se zpětnou vazbou.

Literatura

[1] *Wired glove – Wikipedia, the free encyclopedia*, [on-line], [cit. 28.12.2009]. URL http://en.wikipedia.org/wiki/Wired_glove

[2] *P5 glove – Virtual Reality glove,* [on-line], [cit. 28.12.2009]. URL http://www.vrealities.com/P5.html

[3] Orság, F., *Robotika*, *Studijní opora*, verze 2006-11-08, [on-line], [cit. 28.12.2009].

[4] Krotil, R., *Virtuální realita*, [on-line], [cit. 28.12.2009]. URL http:// gybu.webzdarma.cz/fyzika/paq/seminarka%20-%20virtualni%20realita.doc

[5] *The Virtual Reality Headset*, [on-line], [cit. 28.12.2009]. URL http://www.vrs.org.uk/virtual-reality/headset.html

[6] Brdička, B., *Virtuální realita*, [on-line], [cit. 28.12.2009]. URL http://it.pedf.cuni.cz/~bobr/ucspoc/virtreal.htm

[7] Beran, V., *PGP – Virtuální realita*, [on-line], [cit. 28.12.2009].

[8] Matěna, L., *Parametry systému pro rozšířenou virtuální realitu*, [on-line], [cit. 28.12.2009], Brno, 2007. URL http://is.muni.cz/th/60860/fi_m/xmatena_dp_v01.tex

[9] Kočí, J., Quaterniony, [on-line], [cit. 28.12.2009]. URL http://kengine.sourceforge.net/tutorial/vc/quaternion.htm

[10] Tobola, P., Matematické základy Quaterniony, [on-line], [cit. 28.12.2009], Brno, 2004. URL http://www.fi.muni.cz/~ptx/PA010/Slides/PA010_4.pdf

[11] *OpenGL vedle DirectX stále žije*, [on-line], [cit. 28.12.2009]. URL http://www.ddworld.cz/linux/opengl-vedle-directx-stale-zije-vychazi-opengl-3-5.html

[12] Boháček, P. *Esej na téma „Co a k čemu je OpenGL*, [on-line], [cit. 28.12.2009]. URL http://www.bohacek.info/doc/2005-OGL.pdf

[13] Coumans, E. *Bullet 2.76 Physics SDK*, [on-line], 2010, [cit. 8.5.2011]. URL http://www.bulletphysics.com/ftp/pub/test/physics/Bullet_User_Manual.pdf

[14] Coumans, E. *Bullet 2.55 Physics SDK*, [on-line], 2007, [cit. 8.5.2011]. URL http://code.google.com/p/bullet/source/detail?r=673&path=/trunk/Bullet_User_Manual.pdf

[15] Zhou, S., Jones, Ch., *Multi-Scale Spatial Database and Map Generalisation*, [on-line], Cardiff, United Kingdom, 2001, [cit. 8.5.2011]. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.3633&rep=rep1&type=pdf

[16] Habib, I.,*Cyclic Coordinate Descent (CCD)*, [on-line], [cit. 28.12.2009]. URL http://www.geekyblogger.com/2008/04/cyclic-coordinate-descent-ccd.html

[17] Fědor, *Animace kloubových soustav*, Illusion Softworks [on-line], [cit. 28.12.2009].

[18] Habib, I., *Inverse Kinematics*, [on-line], [cit. 28.12.2009]. URL http://www.geekyblogger.com/2008/04/human-motion-animation-with-inverse.html

[19] Tománek, T., *Nová média – Technologie – Hapitcká rozhraní*, [on-line], [cit. 5.5.2011]. URL http://novamedia.xf.cz/haptic.html

[20] *P5 Glove:History - Scratchpad Wiki Labs - Free wikis from Wikia,* [on-line], [cit. 8.5.2011]. URL http://scratchpad.wikia.com/wiki/P5_Glove:History

[21] Couman, E., *Collision Detection and Rigid Body Dynamics*, [on-line], 2010, [cit. 8.5.2011]. URL https://graphics.stanford.edu/wikis/cs248-11-winter/CS_248%3A_Interactive_Computer_Graphics?action=AttachFile&do=get&target=Erwin.pdf

[22] Bishop, G.,Welch, G., *An Introduction to the Kalman Filter*, [on-line], Chapel Hill, 2006, [cit. 9.5.2011]. URL http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[23] *Kalmanův filtr – Moderní teorie řízení*, [on-line], [cit. 5.5.2011]. URL http://support.dce.felk.cvut.cz/pub/roubalj/teaching/MTR/seminars/MTR_cv10_kf.pdf

[24] Hachem, R., *Kalman filtering for P5 glove [2005]*, [on-line], [cit. 5.5.2011]. URL http://rich99.com/opengl.htm#Kalman

[25] *P5 SOFTWARE DEVELOPMENT KIT INFORMATION BOOKLET*, [CD-ROM], 2002

[26] *ZZZ online|P5 community page*, [on-line], [cit. 5.5.2011]. URL http://www.zzz.com.ru/index.php?area=pages&action=view_page&page_id=11#P5DLL

[27] Kenner, C., *P5 Virtual Reality Glove Dual Mode Driver*, [CD-ROM], 2004

[28] Burdea, G., Coiffet, P., *Virtual reality technology*, New Jersey, John Wiley & Sons, 2003

[29] *MilkShape 3D – Wikipedia, the free encyclopedia*, [on-line], [cit. 14.5.2011]. URL http://en.wikipedia.org/wiki/MilkShape_3D

[30] Reichl,J.,Všetička,M., *Popis rotace tuhého tělesa, MEF :: Encyklopedie Fyziky*, [on-line], [cit. 20.5.2011]. URL http://fyzika.jreichl.com/index.php?sekce=browse&page=1298

[31] Kobbelt,L., *Computer graphics & multimedia - RWTH Graphics: Plugins*, [online], [cit. 20.5.2011]. URL http://www.openflipper.org/index.php?id=240

[32] *Matrix and Quaternion FAQ*, [on-line], 1998, [cit. 20.5.2011]. URL http://www.flipcode.com/documents/matrfaq.html

[33] Dam, Koch, Lillholm, *Quaternions, Interpolation and Animation*, [on-line], Denmark, 1998, [cit. 20.5.2011]. URL http://www.itu.dk/people/erikdam/DOWNLOAD/98-5.pdf

[34] Kilgard, M., *Creating Relections and Shadows Using Stencil* Buffers, [on-line], [cit. 20.5.2011]. URL http://www.opengl.org/resources/features/StencilTalk/

Seznam příloh

Příloha 1.: Ovládání programu

Ovládání pomocí klávesnice

Hlavní 3D scéna

Šipka nahoru/dolů, šipka doleva/doprava – pohyb modelu ruky v ose x a z

Escape – skok do menu

Space – změna fyzikálního modelu na milkShape a zpět

b/B – pohyb ruky směrem vzhůru

v/V – pohyb ruky směrem dolů

c/C – ohnutí prstů

x/X – uvolnění prstů

w/W – změna kamery

r/R – restart scény

Menu

Šipka doleva/doprava – pohyb na předchozí/následující položku

Šipka nahoru/dolů – vertikální pohyb v menu

Enter – výběr položky

Escape – návrat z nabídky

Vývojářská kamera je ovládána myší, levé tlačítko slouží pro rotaci kamery, pravé tlačítko pro zoom.

Ovládání pomocí P5 glove

Hlavní 3D scéna

Translace – pohyb modelu ruky v ose x, y a z

Rotace – otáčení modelem ruky (není doporučeno)

Ohyb prstů – ohyb prstů modelu ruky

Tlačítko A – skok do menu

Tlačítko B – restart scény

Menu

Move to left/ move to right gesture - pohyb na předchozí/následující položku

Pointing gesture – výběr položky

Pull gesture/tlačítko B – návrat z nabídky

Basic tutorial

Move to left/ move to right gesture - pohyb na předchozí/následující položku

Pointing gesture – výběr položky

Tlačítko B – návrat z tutoriálu

Tlačítko C – zopakování instrukcí

Text manipulation tutorial

Tlačítko B – návrat z tutoriálu

Otevřená kniha:

Move to left/ move to right gesture – listování v knize: předchozí/následující strana

Grab gesture – scrollování textu

Release gesture – uvolnění scrollovaného textu

Push gesture – zavření knihy

Zavřená kniha:

Move to left/ move to right gesture – přechod na předcházející/následující knihu

Pull gesture – otevření knihy

Galerie 1

Translace – pohyb vertikálního a horizontálního kurzoru

Pointing gesture – zobrazení detailu obrázku

Move to left gesture – zavření detailu obrázku

Přiblížení se k věži – uchopení galerie, následný pohyb doleva a doprava posouvá galerii

Tlačítko A – skok na galerii 2

Tlačítko B – návrat z nabídky

Galerie 2

Translace – pohyb ruky nad stolem, po výběru fotografie její přesun

Pointing gesture – uchopení fotografie

Rotate gesture – otočení uchopené fotografie

Release gesture – upuštění uchopené fotografie zpět na stůl

Tlačítko A – skok na galerii 3

Tlačítko B – návrat z tutoriálu

Galerie 3

Translace – muž sleduje pohyb rukavice

Tlačítko B – návrat z tutoriálu