ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Daemon pro vstupní zařízení pro CAVE

Daemon for input devices for CAVE

Bakalářská práce

Studijní program: Softwarové technologie a management
Studijní obor: Web a Multimédia

Vedoucí práce: Ing. Zdeněk Trávníček

Markéta Karaffová

Praha 2014

Abstrakt

Rozvoj technologií umožnil vznik mnoha virtuálních prostředí. Po představení CAVE (Cave Automatic Virtual Environment) na konferenci SIGGRAPH v roce 1992 univerzitní týmy po celém světě sestavují vlastní CAVE. České Vysoké Učení Technické v Praze není výjimkou, CAVE je možné najít v Institutu Intermédií na Fakultě elektrotechnické. Bohužel rozšiřování jeho dalšího uplatnění brání software používaný pro dodávání dat aplikacím, trackd. Tato práce popisuje problémy současného stavu, zkoumá rozličné druhy vstupních zařízení vhodných pro uplatnění v CAVE, rozebírá přístupy ostatních dostupných frameworků a navrhuje řešení v podobně nového programu, který dokáže adekvátně nahradit trackd a spolupracuje s dosavadními grafickými aplikacemi. Kromě nahrazení navíc poskytuje možnost rozšíření pro budoucí potřeby CAVE, jako je možnost napsání vlastních výstupů a vstupů. Tímto využití CAVE již není omezené vstupem a může dále růst.

Abstract

Development of technologies allowed birth of various virtual environments. After introducing CAVE in 1992 university teams from all over the world form their own CAVE. Czech Technical University is not an exception, its CAVE can be found at Faculty of Electrical Engineering at Institute of Intermedia. Unfortunately expanding of its usage is discouraged by software used for collecting input data, trackd. This paper describes disadvantages of current state, explores various input devices suitable for using in CAVE, analyses approaches of other frameworks and as a solution suggests a new program, which can correspondingly replace trackd and cooperate with existing graphical applications. Furthermore it also allows extensions for possible future needs of CAVE, such as options of writing new inputs and outputs. This way CAVE is no longer limited by its input and can further evolve.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Poděkování

Ráda bych poděkovala vedoucímu práce ing. Zdeňkovi Trávníčkovi za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval. Mé poděkování též patří Institutu Intermédií za umožnění této práce.

Obsah

1 Úvod

CAVE, celým názvem Cave Automatic Virtual Environment, je rekurzivní zkratka a zároveň odkaz na Platónovo podobenství o jeskyni, ve kterém se probírá vnímání reality a iluzí.[1][2] Toto podobenství je více než výstižné pro automatické virtuální prostředí, jakým je CAVE. Více než jeskyni je ovšem CAVE podobný spíše kostce, jejíž stěnami jsou projekční plátna. Projekce na stěny kolem uživatele v kombinaci s 3D technologiemi při správném nastavení uživatele obklopují, tělesa vyplouvají ze stěn a obrazy z různých stěn kostky se skládají dohromady v jeden panoramatický. CAVE má minimálně tři stěny, přední a jednu na každé straně, maximální počet projekčních ploch může být až plný počet stěn kostky, a musí být umístěný v tmavé místnosti.

CAVE byl poprvé představen na konferenci SIGGRAPH v roce 1992 a za jeho vývojem stál tým z University of Illinois at Chicago. V letech 2006 až 2007 České Vysoké Učení Technické v Praze (ČVUT) po vzoru mnoha zahraničních univerzit sestavilo v Institutu Intermédií (IIM) na Fakultě elektrotechnické vlastní CAVE[3] a celý systém byl spuštěn v květnu 2007.[4] Tento CAVE má 4 projekční plochy, 3 stěny a podlahu, a každá plocha má rozlišení 800x600 a frekvenci necelých 120 Hz. Stěny řídí dva počítače s grafickou kartou NVIDIA QUADRO FX5800 a synchronizují se pomocí karty NVIDIA G-SYNC II.

Software pro CAVE můžeme rozdělit na dvě části – vstupní a zobrazovací. Vstupní má na starosti přijímání dat z ovladačů, zpracování a předání zobrazovací části. Ta vyhodnotí získané informace a podle nich vykreslí nový obraz, který pošle promítacím zařízení. V IIM se používá aplikace založená na knihovně CAVELib a o vstupní data se stará daemon trackd.

Cílem této práce je navrhnout aplikaci, která by dokázala z důvodů uvedených v kapitole 2 nahradit daemon trackd, posléze ji naimplementovat a nasadit do provozu.

2 Současný stav – trackd

V této kapitole je rozebrán framework používaný v současné době k získávání dat z ovladačů pro CAVE v IIM na ČVUT, trackd.[5]

2.1 O trackd

Trackd patří, stejně jako CAVELib, společnosti Mechdyne. Jedná se o komerční software s uzavřenou licencí. Trackd primárně funguje společně s aplikací postavené na CAVELib, ale není to nutnou podmínkou. Pro vývojáře aplikací existuje trackdAPI, odkaz sice není na oficiálních stránkách, ale dokumentace se nachází na stránkách Jackson State University[6].

2.2 Možnosti trackd

Trackd rovnou obsahuje ovladače na několik běžných typů vstupních zařízení, kromě toho poskytuje dokumentaci pro psaní vlastních ovladačů na další typy zařízení.[7] Daemon podporuje současné zapojení více zařízení a navíc je možné v konfiguračním souboru nastavit překrývání komponent dvou zařízení stejného typu, čímž výstup vidí jedno zařízení, které je průnikem obou. Tato vlastnost je důležitá například pro simulaci vstupních dat na dálku.

Data jsou čtena ze vstupních zařízení buď přímo přes sériový port, nebo UDP protokolem z trackd serveru. Tento server opět umí načítat data oběma způsoby, ale výstup má vždy síťový. Výstup samotného trackd je do sdílené paměti (anglicky shared memory, SHM).[8] V této paměti se data ukládají do dvou oblastí.

První oblast je velká 296 B pod klíčem 4127 a zde se ukládají informace o tlačítkách a osách zařízení (více o komponentách zařízení v kapitole 3). Data obsahují čas zápisu, počet tlačítek, počet os a dvě pole s hodnotami tlačítek a os.

Druhá část pod klíčem 4126 je větší, přibližně 4 kB, a slouží pro uložení hodnot trackerů. Nejprve se zapisuje hlavička s počtem trackerů, offset, velikost a čas zápisu, následují samotné trackery, u každého jsou dvě pole tří hodnot pro souřadnice pozice a pro rotaci trackeru, čas posledního zápisu, informace o kalibraci a číslo frame z posledního uložení.

Do obou oblastí se zapisují přímo stavy zařízení, data nejsou nijak interpretována.

2.3 Nedostatky trackd

Z předchozího odstavce vyplývají některé nedostatky současného frameworku, tyto i další více rozvádí následující část.

2.3.1 Licence

Uzavřená placená licence s sebou přináší mnohá omezení. Jelikož kód trackd není volně přístupný, co se děje uvnitř aplikace se dá zjistit pouze z dokumentací a zpětnou analýzou výstupů a chodu daemona. Navíc není možné ho jakkoliv vylepšit nebo rozšířit.

2.3.2 Načítání ovladačů

Trackd automaticky detekuje ovladače pouze na výchozí adrese, která je závislá na operačním systému. V Linuxu je cesta */dev/ttyS1*, pro IRIX zase */dev/ttyd2*, HPUX má */dev/tty1p0*[4] a konečně ve Windows je to *com2*. Pokud je připojeno více zařízení, nebo je zapojené jinam, je nutné nastavit cestu v konfiguračním souboru, který trackd načítá po spuštění. Kromě toho zařízení na těchto cestách musí být připojena už před startem daemona, dynamické načítání za běhu není možné a je vždy nutné daemona restartovat, pokud chceme, aby reagoval na změny připojení.

2.3.3 Více vstupních zařízení současně

Trackd sice podporuje současné používání více vstupních zařízení, ale data z nich sjednotí do jednoho výstupu a nepředává informaci o počtu zařízení ani které hodnoty patří kterému z nich. Například pokud je připojeno jedno zařízení s 10 tlačítky a zároveň druhé s 15, ve výstupu je jedno zařízení, které má tlačítek 25.

2.3.4 Čtení dat

Jak již bylo řečeno, při čtení nedochází k žádné interpretaci dat. Daemon jednoduše čte ve smyčce dokola současný stav vstupních zařízení a ten posílá na výstup, kam se dostávají redundantní informace v případě, že nedošlo od posledního načtení k žádné změně stavu zařízení. Frekvence čtení pravděpodobně závisí na možnostech počítače, na kterém trackd běží, bez znalosti kódu nelze zjistit přesnější informace.

2.3.5 Výstup dat

Samotný výstup dat také není ideální, používání sdílené paměti znamená, že na stejném stroji musí současně s trackd běžet i software, který data odebírá a zpracovává, nebo alespoň jeho část, která data posílá dál. Jelikož se čtením dat ze vstupu probíhá ihned i zápis, výrazně se zaměstnává stroj, na kterém trackd běží, a ubírá se tím výkon, který by mohl využít program ovládající CAVE. To se děje i v klidovém stavu, dokud daemon běží, tedy i pokud je druhá strana vypnutá a data z paměti nic nečte. Průměrná spotřeba CPU je 35 %.

Mimo tento problém také v případě, že zápis změny a návratu do předchozího stavu proběhne rychleji, než je interval mezi čteními z paměti, hrozí nebezpečí, že aplikace změnu nestihne zaznamenat, protože data budou přepsána dříve, než je stihne přečíst. SHM nese ještě jeden problém, přístup do kritické sekce. V době zápisu se sekce paměti nijak nezamyká a je možné číst i během zapisování. Tím vzniká riziko, že dojde k přečtení části starého a části nového zápisu během jednoho čtení.

2.3.6 Další

Konečně trackd nevyužívá plný potenciál ovladačů, například zcela opomíjí možnosti zpětné vazby vstupních zařízení.

3 Vstupní zařízení

Během mnoha let vývoje počítačových technologií vznikla široká paleta vstupních zařízení a vytvořit framework, který by dokázal obsluhovat velký počet těchto zařízení, se na první pohled zdá obtížné, proto je následující kapitola věnovaná rozboru typů vstupních zařízení a jejich výstupům. Dělí je na dvě kategorie: ovladače a trackery.

Kapitola úmyslně nepopisuje technické provedení zařízení, protože je irelevantní pro splnění cíle práce, tj. navržení frameworku pro zpracování vstupních dat. Při zájmu o tuto problematiku doporučuji diplomovou práci Vladimíra Paločka: *Generalized interactive techniques for Collaborative VR System*[10].

3.1 Stupně volnosti

Před rozebráním jednotlivých druhů zařízení je třeba vysvětlit si zásadní pojem v této oblasti. Stupně volnosti, anglicky *degrees of freedom* (zkráceně **DOF**), vyjadřují počet „různých základních směrů, kterými se může objekt pohybovat“[11]. Tyto základní směry se dělí na rotace a posun.

Posun v prostoru má tři základní směry. Představíme-li si v prostoru kartézskou soustavu souřadnic, základní směry jsou podél jejích os x, y a z. Z vektorů v těchto směrech dokážeme složit libovolný jiný vektor v soustavě, proto se jim říká základní. **Rotace** se řídí obdobnými pravidly, její tři základní směry jsou rotace podél jednotlivých os soustavy. Celkem tedy v prostoru můžeme mít 6 různých typů pohybu, tím je dáno i maximum 6 stupňů volnosti.

U vstupních zařízení stupně volnosti vyjadřují možnosti pohybu, které umí komponenta zprostředkovat, nebo schopnosti trackerů sledovat pohyb. Například pokud umí zařízení snímat pouze posun ve třech osách, říkáme, že sleduje 3 stupně volnosti. Obvyklé datové vyjádření výstupu pro trackery i ovladače je jedna numerická hodnota za každý stupeň,

které zařízení sleduje nebo poskytuje. Tato práce používá pro stupně volnosti zkratku DOF.

3.2 Ovladače

V této kapitole je používaný český výraz ovladač pro typ vstupního zařízení (controller), nikoli pro software, který zařízení zprostředkovává (driver). Ovladačem je myšleno takové vstupní zařízení, které vytváří výstupní data pouze z vědomých úkonů uživatele, jako je stisknutí tlačítka, pohnutí páčky, otočení kolečka a podobné.

Myš ani klávesnice nejsou vhodnými zařízeními na ovládání virtuálního prostředí, ale je potřeba je zmínit, protože z nich vychází další typy zařízení.

3.2.1 Klávesnice

Toto zařízení bylo dodáváno již s prvním osobním počítačem[12] a jeho původ se odvíjí od psacích strojů, podle kterých přebralo rozložení kláves. I po více než třiceti letech klávesnice patří mezi základní vybavení počítačů. Standardní klávesnice má minimálně 104 kláves, které mnohdy doplňují klávesy na ovládání zvuku, podsvícení nebo tlačítka rychlého spuštění. Herní klávesnice jsou navíc obohacené o programovatelné klávesy, kterým se dá přiřadit sekvence stisknutí ostatních kláves. Kromě tlačítek může klávesnice také obsahovat touchpad nebo trackball.

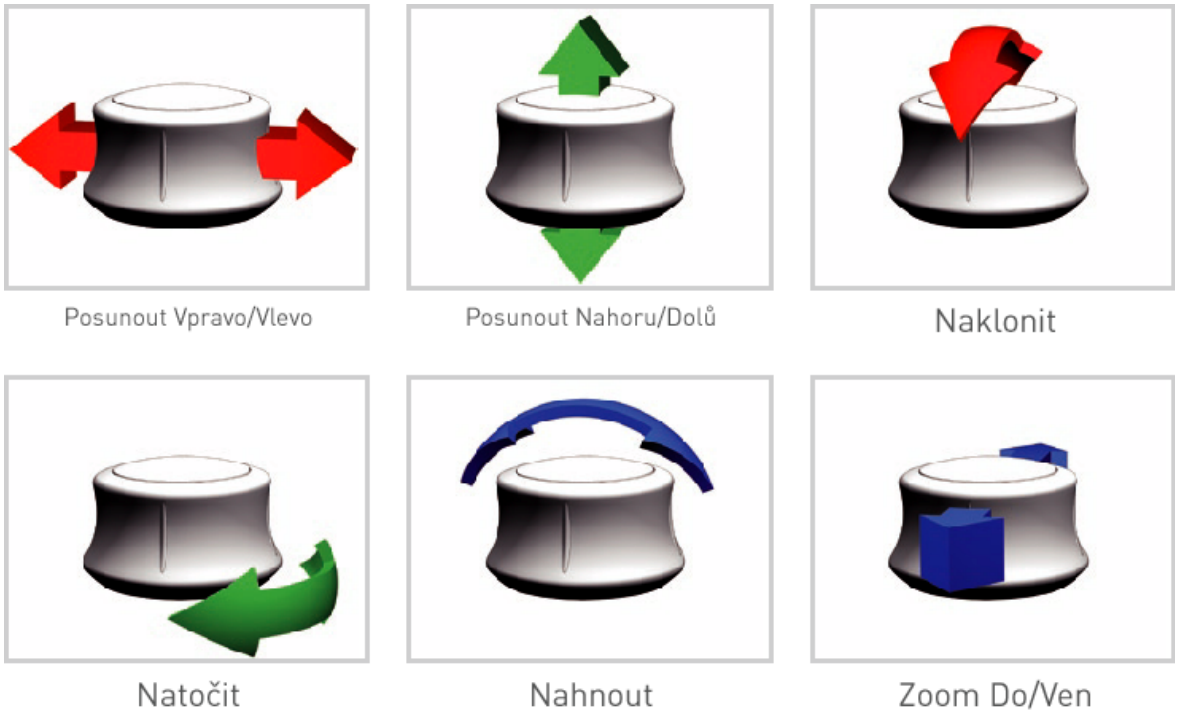
Tlačítko má základní dva stavy – stisknuto a nestisknuto. To svádí k reprezentací tlačítka jedním bitem, ale některá tlačítka mají i třetí stav opakovaného stisknutí, který vzniká při držení tlačítka. Tento stav ovšem lze vypustit, ale je potřeba s ním počítat při sběru vstupu z tlačítek. Touchpad je 2 DOF zařízení, proto je reprezentován dvěma číselnými hodnotami, kde jedna představuje polohu v horizontální ose a druhá ve vertikální. Pokud má možnost více dotyků najednou (*multitouch*), dává dvě numerické hodnoty na každý dotyk. Trackball může být také 2 DOF a tím mít stejný výstup, ale existují i 3 DOF trackbally.

3.2.2 Myš

I myš může mít různé varianty, od původní dvoutlačítkové až po dnešní s posuvným kolečkem a mnoha doplňkovými tlačítky. Samotná myš dává informaci o svém posunu,

tedy relativní pozici od poslední polohy při pohybu po ploše. Relativní pozice je stejně jako u trackballu reprezentovaná dvěma numerickými hodnotami. Posuvné kolečko dává informaci o relativní pozici v jednom směru a zároveň funguje jako tlačítko. Výstup dalších tlačítek je stejný jako u klávesnice.

Myš kromě tlačítek a kolečka může mít i trackball s 2 DOF nebo 3 DOF, speciální typy myší ovšem mají k dispozici 6 DOF vstup.[13] SpaceMouse Pro[14] má navíc komponentu, která umožňuje plný počet stupňů volnosti a taktéž je k dostání jako samostatný prvek nahrazující myš pod názvem SpaceNavigator.



Ilustrace 1: SpaceNavigator [40]

3.2.3 Gamepad

Gamepady jsou, jak název napovídá, primárně určeny ke hraní videoher, ale jejich bezdrátové verze jsou zároveň vhodné i pro virtuální prostředí. Standardní gamepad má větší množství tlačítek a alespoň jeden prvek *crossbutton*, jež získal název ze své podoby. Dává dvě numerické hodnoty, jednu pro každý směr. Speciální typ tohoto prvku ovládá tři směry a dává tedy celkem tři numerické hodnoty. Další nový prvek je *hat button* poskytující volný pohyb ve dvou osách a jeho výstupem jsou rovněž dvě numerické

hodnoty. Gamepady jako SpaceOrb 360[15] mají také 6 DOF prvek, volně otočnou kuličku, kterou je možné tlačit třemi směry.

3.2.4 Volant

Volant byl speciálně vyvinutý k simulacím řízení aut, což kromě autoškol našlo uplatnění také i v herním průmyslu. Zařízení proto vypadá zcela jako volant v autě, doplněný podle cenové relace o tlačítka a další prvky. Volant musí být uchycený například ke stolu, nebo má vlastní pevný stojan, a dává informaci o stupních rotace. Standardní rozsah rotace je 270 stupňů a tato hodnota stoupá až k 900 stupňů.[16] Tato rotace je reprezentována jednou numerickou hodnotou.

Kromě tlačítek na volantu muže být volant doplněn o další volitelné prvky pro věrnější simulaci. Nejčastěji to jsou pedály, výstup je numerická hodnota na jeden pedál, druhým často používaným prvkem je řadící páka, která dává numerickou hodnotu zařazené rychlosti.

U volantů je zcela běžná zpětná vazba zvaná *force feedback*, která způsobuje, že volant může při jízdě na hrubém terénu „drkotat“ a při otáčení klást odpor odpovídající skutečnému odporu volantu v autě v závislosti na podmínkách. Další typ zpětné vazby má Logitech G27 Racing Wheel[16], na samotném volantu jsou LED diody, které slouží jako indikátor otáček.

3.2.5 Joystick

Rozmanité druhy joysticků mají vždycky společné dvě části. Jsou jimi tlačítka a hlavní ovládací prvek – páka. Pákou lze pohybovat po směru dvou os, pohyb v každém z nich je reprezentovaný numerickou hodnotou, a v některých případech i otáčet kolem svislé osy. Jedno zařízení nemusí obsahovat pouze jen jednu páku, kromě další velké může mít i menší páčky, které se pohybují pouze jedním směrem. Pro všechny další páčky opět platí, že na každý směr je potřeba jedna numerická hodnota.

Specialitou jsou joysticky určené pro ovládání letadla. Jejich tvar odpovídá řídícím komponentám letadel a kromě pozice hlavní páky je možné sledovat rotaci kolem dvou os, první je stejná jako u volantu a druhá umožňuje naklánět páku od sebe a k sobě.

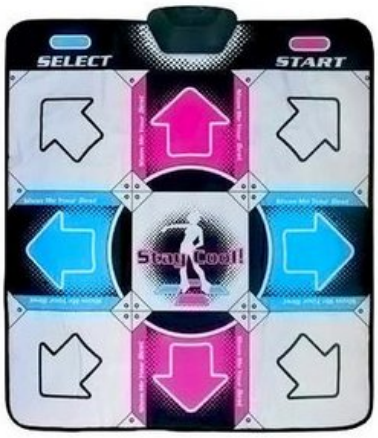
Za zmínku taktéž stojí, že některé joysticky mohou poskytovat zpětnou vazbu *force feedback.*

3.2.6 Taneční podložky

Dancepad, česky taneční podložka, je typ vstupního zařízení určeného pro taneční hry typu *Dance Dance Revolution*.[17] Principem těchto her je šlapání na šipky na podložce na zemi podle šipek, které se pohybují po displayi, za doprovodu hudby.

Běžná taneční podložka má čtyři až osm směrových tlačítek s šipkami a dvě další pro nastavení, existují ale i jiné podložky určené pro děti s jiným počtem tlačítek a vzhledem. V každém případě se tlačítka na podložce chovají jako tlačítka na klávesnici, navíc mohou i směry být popsané jako osy a dávat i výstup dvou numerických hodnot.

Dancepad by mohl být použitý v CAVE například pro pohyb v prostoru po ploše, další šipky se mohou využít pro třetí osu nebo rotaci, pokud jsou k dispozici.

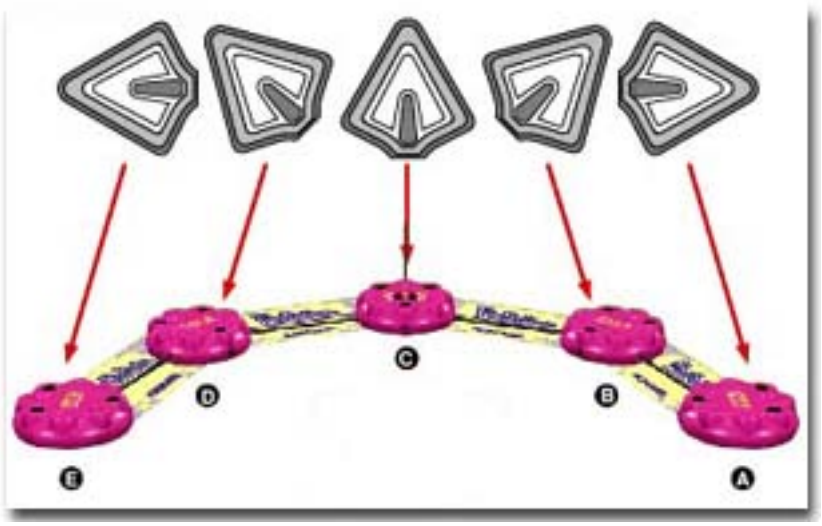


Ilustrace 2: Taneční podložka[41]

3.2.7 ParaParaParadise

Toto zařízení[18] slouží pouze ke hraní stejnojmenné hry, která je založena na podobném principu jako *Dance Dance Revolution,* a ačkoliv to vypadá, že snímá pohyb tanečníka, nedá se považovat za tracker. Zařízení je totiž pouze schopné zachytit přerušení paprsků infračerveného světla, které vysílají emitory rozestavěné kolem uživatele. Z toho je patrné, že nemá žádná data o pohybu, který jde mimo paprsky a tím se spíše než sledovacímu zařízení podobá taneční podložce.

Na výstup se dostane informace, jestli je daný paprsek porušený nebo ne. Její reprezentace je stejná jako u tlačítka a těchto hodnot je stejně jako počet paprsků. Vedle toho má zařízení ještě dvě klasická tlačítka.



Ilustrace 3: ParaParaParadise[42]

3.3 Trackery

Pojem tracker pochází z anglického slova *track* nebo-li sledovat. Zařízení tohoto typu mohou mít stejné prvky, jako ovladače popsané v kapitole 3.2, ale navíc zahrnují jeden nebo více senzorů na sledování pohybu. Velkou výhodou trackerů je, že sledování pohybu poskytuje nejen přirozené ovládání například úhlu pohledu a pohybu v prostředí, ale navíc má uživatel díky tomuto volné ruce na provádění složitějších úkonů.

Je třeba poznamenat, že typy sledování se liší nejen svým rozsahem částí, které sledují, ale i technickým provedením, na kterém zároveň závisí přesnost a cena. V této kapitole, stejně jako v předchozí, nebude z již zmíněných důvodů rozebírána technická stránka. V závislosti na technologickém provedení lze jednotlivé typy spolu kombinovat navzájem nebo doplnit o ovladače.

3.3.1 Wand

Zařízení typu wand nejsou ničím jiným než gamepady s jedním snímačem polohy 6 DOF. Kromě tohoto snímače mají také komponenty probírané v kapitole 3.2 jako jsou tlačítka, *trackball*, páčky a páky, nebo *crossbutton*.

3.3.2 Head tracker

Head tracker slouží k určení pozice a rotace hlavy. Tato data jsou důležitá pro určení úhlu pohledu, což pomáhá urychlení času vykreslení obrazu, neboť části mimo zorné pole není potřeba zobrazit ostře a detailně. Pozice navíc dodává informaci o pohybu uživatele v prostoru, případně jeho skrčení nebo skoku. Head tracker má jeden výstup s 6 DOF, tedy poskytuje šest numerických hodnot.

3.3.3 Helma

Helma je ve skutečnosti head tracker doplněný o display a sluchátka. Velkou výhodou je, že uživatel má plný audiovizuální vjem virtuálního světa. Toto zařízení ovšem není použitelné v kombinaci s CAVE, protože display v helmě CAVE nahrazuje.

3.3.4 Pohyb těla

Druhů snímání pohybu těla je mnoho, od těžké konstrukce na těle po pohodlný oblek nebo třeba jen pásek. Zařízení mají velké množství senzorů a každý z nich dává hodnoty své pozice. Orientace je určena buď pro každý senzor zvlášť, nebo pro skupinku 4 senzorů, v tomto případě má každý senzor jen 3 DOF a navíc jednotlivé skupinky dávají ještě tři numerické hodnoty pro orientaci.

3.3.5 Rukavice

Existuje mnoho typů rukavic s různými výstupními daty. Rukavice založené na ohybu prstů (*flex*) mohou mít jeden a více senzorů na každý prst, dále pak senzory na odtažení prstů od sebe, ohyb zápěstí a sevření dlaně. Tyto senzory dávají každý jednu numerickou hodnotu určující ohyb. K dostání jsou i rukavice založené na doteku dvou snímačů (*pinch*), které dávají stejné hodnoty jako tlačítka[19][20]. Pro oba typy platí, že některé rukavice mají

i snímání 6 DOF. Kromě zmíněných hodnot může být výstupem již rozpoznané gesto ruky.

Speciální typ rukavice, která kombinuje ohyb i dotek vytvořili na Brown University.[21] Senzory doteku lze v rukavici přesouvat, aby jejich pozice lépe vyhovovala požadavkům na gesta. Kombinace technologií přináší výhody obou a umožňuje používat každou z nich na jinou funkci, například stisk aktivuje mód a ohybem prstů se aplikace ovládá.

3.3.6 Eye tracking

Ačkoliv sledování pohybu očí je vhodné spíše jako alternativa klasických vstupů jako myš a klávesnice pro uživatele s tělesným postižením, není zcela vyloučené, že by jednou tato technologie mohla být využita v CAVE. Jedná se o snímání odrazu paprsku světla vysílaného do oka. Princip vypočítání úhlu pohledu je založený na různých vlastnostech odrazu a pohlcení světla oka.[22] Výstup zařízení jsou souřadnice pohledu, při pohledu na plochu to jsou dvě numerické hodnoty.[23]

Eye tracking má mnohé nevýhody, hlavním důvodem je kalibrace pro každého uživatele a také ne každý může tuto technologii používat (například uživatelé kontaktních čoček). Dalším faktorem je vysoká cena a také fakt, že pohyb očí je podvědomá akce a jako taková se špatně ovládá a tak může docházet k nechtěnému spouštění funkcí programu.[24]

3.3.7 Další trackovací zařízení

Různé aplikace pro virtuální prostředí podněcují k vytvoření specifických vstupních zařízení. V této kapitole pro ilustraci představím několik zařízení, které sestavili na Computer Science Department at Brown University[25].

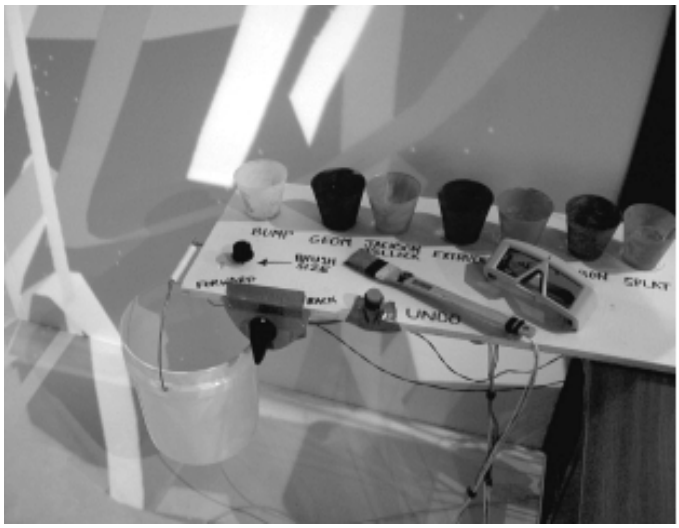
Prvním z jmenovaných je **FingerSleeve**, zařízení k navlečení na prst. Zařízení má jeden 6 DOF snímač a dvě speciální tlačítka. Tyto tlačítka nemají pouze dva stavy, jako všechna doposud zmíněná zařízení, ale lze je promáčknout a tím vzniká třetí stav. Tento princip podobně jako spoušť u foťáku, která po mělkém zmáčknutí zaostří scénu a fotí až po domáčknutí, rozšiřuje možnosti zařízení.

Ilustrace 4: FingerSleeve[25]

Dalším zajímavým zařízením jsou **interakční bačkory**.[26] Přes tendenci uvolnit uživateli ruce pro jiné úkony nejsou nohy často využívané, proto není toto ovládání obvyklé. Kromě pozice v prostoru CAVE má tato obuv na vnitřní straně snímače, které fungují na stejném principu jako u dotekových rukavic. Snímače umožňují dva různé typy doteků, které se na výstupu projevují jako stisk tlačítka.

Ilustrace 5: Interakční bačkory[25]

Posledním ale nemálo zajímavým zařízením je **CavePainting Table**. Jedná se o stůl přistavený na okraj CAVE, jehož pevnou součástí jsou kelímky se stylem malování, tlačítka a kolečka. Umístění stolu není nevýhodou, protože umělci při kresbách často odstupují od díla. Hlavní komponentou je štětec, který je vybavený 6 DOF senzorem a jedním tlačítkem, které ho vypíná a zapíná. Při „namočení" štětce z vodivého materiálu se uvnitř kelímku sepnutím obvodu simuluje stisk tlačítka.



Ilustrace 6: CavePainting Table[25]

3.4 Řeč

Řeč je dalším z možných typů vstupů a z pochopitelných důvodů ji nelze řadit mezi ovladače nebo trackery. Řeč je pro nás přirozená a povely nám nezaměstnávají ruce, proto se stává ideálním kandidátem na vstupní zařízení pro CAVE. Rozpoznávání povelů by ovšem muselo být na úrovni, kdy nebude docházet k omylům. Dalším problémem by mohlo být rušení snímání řeči zvuky okolí.

Záznam řeči může mít jako výstup rovnou záznam mikrofonu, nebo lépe již rozpoznané povely.

3.5 Shrnutí

Přes značnou variabilitu zmíněných zařízení lze vidět určité společné rysy a vytvořit kategorie, podle kterých se dá k zařízením přistupovat. Prvně je třeba uvědomit si, že i zařízení stejného typu mají různý počet rozdílných komponent, proto je lepší roztřídit komponenty a ne samotná zařízení na typy vstupu.

Základní komponentou je tlačítko. Kromě klasických případů do této skupiny také patří stisk u dotykových rukavic (3.3.5), přerušení paprsku u ParaParaParadise (3.2.7) a šlápnutí na šipku u taneční podložky (3.2.6).

Další jsou vícestavové komponenty, obvykle reprezentovány jednou celočíselnou hodnotou, například řadící páka u volantu (3.2.4) nebo přepínací páčky u joysticku pro simulaci letu (3.2.5). Také tlačítko z FingerSleeve (3.3.7) by se dalo do této kategorie zařadit.

Následující komponenty se dají rozřadit do kategorií, kolik DOF poskytují. Pohyb na jedné ose je 1 DOF a je to posuvné tlačítko myši (3.2.2), samotné otočení volantu (3.2.4), páky pohybující se analogově jedním směrem u joysticku (3.2.5) a ohyb rukavice (3.3.5). Pohyb po ploše neboli 2 DOF je posun myši (3.2.2), touchpad (3.2.1), některé trackbally (3.2.1), hatbutton a crossbutton u gamepadu (3.2.3), joystick (3.2.5) a eye tracker (3.3.6). Speciální typ crossbutton a část trackballů (3.2.3) jsou 3 DOF, stejně jako některé typy senzorů u trackerů. A konečně 6 DOF, které snímají trackery a umožňují SpaceOrb (3.2.3) a SpaceNavigator (3.2.2).

Samotnou kategorii také musí mít gesta rukavic (3.3.5) a verbální povely (3.4)

Všechna zařízení popisována v této práci se skládají z těchto typů komponent. Tabulka shrnuje, které komponenty mohou zařízení mít.

Zařízení	Povel/ gesto	Tlačítko	Stavová osa	1DOF	2DOF	3DOF	6DOF
Klávesnice (3.2.1)		x			x		
Myš (3.2.2)		x		x	x		x
Gamepad (3.2.3)		x			x	x	x
Volant (3.2.4)		x	x	x			
Joystick (3.2.5)		x	x	x	x	x	
Taneční podložka (3.2.6)		x			x		
ParaParaParadise (3.2.7)		x					
Wand (3.3.1)		x			x	x	x
Head tracker (3.3.2)							x
Helma (3.3.3)							x
Pohyb těla (3.3.4)						x	x
Rukavice (3.3.5)	x	x		x		x	x
Eye tracker (3.3.6)					x		
FingerSleeve (3.3.7)			x				x
Interakční bačkory (3.3.7)		x				x	
CavePainting Table (3.3.7)		x		x			x
Řeč (3.4)	x						

Tabulka 1: Vstupní zařízení

4 Frameworky pro zpracování vstupu

Pro zpracování vstupů zařízení jako je CAVE existují mnohé frameworky a často i zároveň řeší zobrazení dat ve výstupu. Mnoho jich vzniklo v rámci univerzitních projektů, ale s ukončenou prací na vývoji postupně zanikly. Jako zástupci byli vybraní **DIVERSE**, **Syzygy** a **VR Juggler** pro svoji rozšířenost a možnost je zdarma stáhnout na internetu. Poskytují podporu pro většinu obvyklých typů vstupů nebo dávají možnost dotvořit si vlastní ovladače na vstupní zařízení.

Kromě jmenovaných je zmíněna i sada knihoven VRPN. Následující kapitola se zaměřuje na postupy při zpracování dat z ovladačů.

4.1 DIVERSE

DIVERSE je projekt Virginia Tech, který začal již koncem devadesátých let. Zatím funguje na SGI IRIX a Linuxu, ale v plánu je rozšíření na další platformy. Program tvoří dvě části: The DIVERSE graphics interface for Performer (dgiPf) a The DIVERSE ToolKit (DTK)[28][29]. První zmíněný má na starosti grafickou část programu, druhý zbytek aplikace. Je důležité zmínit, že DTK může fungovat nezávisle na dgiPf, protože zařízení bez grafického výstupu dgiPF ani nepotřebuje. DIVERSE má defaultní nastavení, díky kterému aplikace funguje i bez složité konfigurace.

Vstupní zařízení jsou prezentována pomocí `device classes`. Implicitně je jich pět, ale je možné vytvářet nové podle potřeby jiných zařízení, což dává prostor pro rozšiřování frameworku. Základních pět `device class` tvoří: `locator` (6 DOF tracker), `keyboard` (pro klávesnice), `button` (pole typu boolean, maximálně 32 hodnot[29]), `valuator` (pole typu float) a `selector` (pole typu integer).

Každé zařízení je `polled` (dotazovací) – obvykle `locator` nebo `valuator` – nebo `queued` (tvořící frontu) – obvykle `button` nebo `keyboard`. První typ poskytuje pohled na současný stav, zatímco druhý řadí do fronty změny stavu. Abychom mohli vzít

události od všech zařízení, `polled` zařízení vytváří při změně stavu taktéž event, který se zařadí do fronty. Data se ukládají na sdílenou paměť DTK a bez žádosti jsou posílány příkazem `push` na daný seznam IP adres. Problém s přístupem do kritické sekce sdílené paměti je řešen zamykáním sekcí.

4.2 Syzygy

Framework Syzygy byl vytvořen v září roku 2000 jako snadno použitelná, na platformě nezávislá a odolná aplikace pro VR používající PC cluster. Je ho možno používat na Windows, Linuxu, MacOS X a Irix. Knihovny má psané v C++ a pro grafický výstup používá OpenGL.[30][31]

Data ze vstupních zařízení zpracovává v podobě vstupních událostí, které jsou tří typů. Prvním je matice 4x4 pro pozici a orientaci trackeru, druhým číslo s pohyblivou řádovou čárkou pro skalární vstupy a třetím integer, obvykle s hodnotami 0 nebo 1 pro tlačítka. Události jsou zabaleny do speciálních Syzygy zpráv v binárním kódu. Tímto způsobem probíhá komunikace i mezi počítači v clusteru. [32] Konfigurace může probíhat přímo za běhu a k dispozici jsou nástroje pro vývoj ovladačů na vlastní vstupní zařízení.[4]

4.3 VR Juggler

VR Juggler začal v roce 1997 jako projekt týmu studentů Iowa State University. Od roku 2000 je opensource a aktuální verze 3.0 je zdarma ke stažení na stránkách projektu. Kromě Linuxu a SGI tento framework funguje i na Windows a MacOS X.[34] Jeho výhodou je snadná instalace a také simulátor virtuálního prostředí, který je ovládán pomocí myši a klávesnice. Oproti DIVERSE postrádá defaultní nastavení a sami tvůrci přiznávají, že konfigurace je obtížná a je nutné pochopit proces nastavení.[33]

VR Juggler generalizuje typy vstupů na skupiny `input device`. Těchto skupin je zatím osm bez možnosti rozšíření uživatelem. Jsou to: `Analog` (float na uzavřeném intervalu <1.0,1.0>), `Command` (rozpoznané gesto nebo slovní příkaz), `Digital` (hodnoty on/off, obvykle odpovídá stavu tlačítka), `Digital glove` (různé kombinace prstů u rukavice), `Gesture glove` (rozpoznatelná gesta založená na úhlech ohnutí kloubů), `Positional` (poloha anebo orientace trackeru) a `String` (obvykle odpovídající mluvenému vstupu). Každému zařízení je přiřazena alespoň jedna `input device` třída, u většiny zařízení však jedna nestačí. Například pokud má zařízení čtyři tlačítka a jeden joystick, potřebuje čtyřikrát vstup typu `Digital` (jeden pro každé tlačítko) a dvakrát typu `Analog` (jeden

pro každou osu joysticku). Prostředníkem mezi `input device` a aplikací je `device proxy`, opět má různé typy podle skupiny vstupního zařízení a všechny typy mají metodu `getData`, která vrací data ze zařízení ve formátu podle skupiny vstupů.

4.4 VRPN

VRPN je sada knihoven, kterou vytvořil Russell M. Taylor II na University of North Carolina at Chapel Hill v roce 1998.[37] Je k dispozici na internetu zdarma ke stažení a funguje na platformách Windows, SGI Irix, Linux, Solaris, Hpux a AIX.[38]

VRPN má pro zařízení 3 typy rozhraní: `tracker` (pro 6 DOF senzory), `button` (tlačítka) a `analog` (numerické hodnoty). Pokud má zařízení více typů komponent, VRPN pro něj vytvoří více rozhraní pod stejným jménem zařízení. Například pokud joystick má tlačítka i páku, budou pro něj tedy vytvořena dvě rozhraní, jedno pro tlačítka a jedno pro analogové hodnoty. VRPN je pak vidí jako dvě samostatná zařízení.

5 Návrh daemona

Tato kapitola se věnuje návrhu daemona, implementaci pak popisuje kapitola 6. Před samotným návrhem je třeba určit požadavky na nový framework a stanovit si cíle. Dále následuje samotný návrh jednotlivých částí a jejich komunikace.

5.1 Požadavky

Hlavním požadavkem návrhu řešení je nahrazení stávajícího frameworku novým, který bude možné rozšiřovat o další varianty vstupů a výstupů. Bez možnosti rozšíření by byl omezený pouze na určité typy vstupních zařízení a bránil by použití nových technologií. Jinými slovy framework musí umět spolupracovat s CAVELib aplikací a zároveň podporovat další výstupy. Preferovaný operační systém je Linux a daemon bude otevřený software. Dalším požadavkem je srozumitelná dokumentace a API pro vytváření vstupních a výstupních modulů. Nové moduly musí jít integrovat bez zásahu do jádra aplikace.

5.2 Navrhovaná struktura

Navrhovaná aplikace má čtyři typy komponent. Hlavním a nejdůležitějším je jádro, komunikuje s uživatelem a koordinuje spolupráci ostatních. Jeho funkce nejlépe dokumentuje životní cyklus aplikace. Dalšími dvěma jsou vstupní a výstupní moduly, jež je třeba mít oddělené od aplikace, aby bylo možné vytvářet a připojovat nové podle požadavků na vstup a výstup. Poslední komponentou je ovladač zařízení (driver).

Po zvážení typů vstupních dat z kapitoly 3 byly komponenty rozděleny na následující kategorie: tlačítko, osa, tracker a povel. Toto členění není konečné, struktura umožňuje přidávání dalších, pokaždé je třeba přidat odpovídající typ události a dodat výstupům instrukce, co mají s novým typem dělat.

5.2.1 Životní cyklus

Při spuštění daemona startuje jádro a načte konfigurační soubor. Právě pomocí konfiguračních souborů, kde je napsáno které moduly spustit, dosáhneme snadného připojení nových modulů. První načte jádro výstupní moduly a odkazy na ně si uloží. Poté načte i vstupní moduly a opět si zaznamená ukazatele.

Po úvodní konfiguraci jádro čeká na příchozí události, které kromě vstupů mohou přicházet i z výstupů, například *force feedback*. Události ze vstupů pošle na všechny uložené výstupy a naopak. Také přijímá příkaz *refresh*, který načte znovu konfigurační soubor a podle něj upraví své vlastnosti. Pokud byl přidán nový vstup nebo výstup, spustí ho. Stávajícím modulům pošle jejich konfigurační údaje, aby provedly obdobnou kontrolu.

Když daemon obdrží příkaz k vypnutí, pošle ukončovací zprávu napřed na vstup a poté na výstup, počká na potvrzení ukončení a poté se sám vypne. Tímto dává možnost modulům uzavřít spojení nebo dokončit zápis do souboru.

5.2.2 Vstupní a výstupní moduly

Veškeré moduly mají jednotné rozhraní a drží si odkaz na jádro. Podle hodnot načtených ze souboru jádrem provedou vlastní konfiguraci, která může sloužit například ke kalibraci zařízení nebo nastavení frekvence čtení či psaní.

Základním výstupním modulem je výstup do SHM, protože je nutné, aby nový framework nahradil stávající. Rozhraní poskytuje možnost napsat si vlastní modul, který může posílat data po síti, zapisovat je do souboru pro pozdější použití nebo je předávat druhé straně podle jejích požadavků.

Předpokládaný nejčastější typ vstupu je přímo připojený k počítači, kde daemon poběží, ale jednotné rozhraní umožňuje přidat modul pro příjem po síti nebo modul, který bude simulovat příchod dat. Modul pro přímé připojení dostává od jádra z konfiguračního souboru informaci, kde nalezne připojená zařízení. Sám poté zařízení načte a vytvoří pro ně ovladače.

Hlavním účelem tohoto modulu je spravování ovladačů, na které má uložené v seznamu ukazatele, a zprostředkování komunikace mezi jádrem a ovladači. Při obdržení ukončovací zprávy ji pošle všem ovladačům a počká na jejich vypnutí.

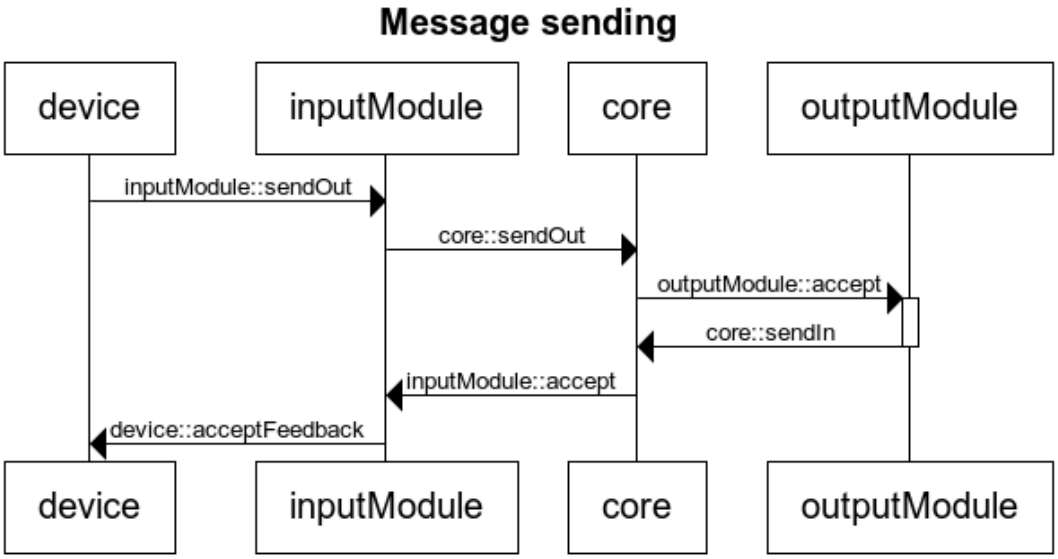
5.2.3 Ovladač

Ovladače mají stejně jako moduly jednotné rozhraní. Jejich úkolem je odchytávat události zařízení a posílat informace přes vstupní modul a jádro na výstup. Hned po inicializaci ovladač posílá zprávu s údaji o připojeném zařízení. Při odpojení zařízení vyšle zprávu stejným způsobem.

Úvodní zpráva je důležitá pro seznámení výstupů se vstupy, především pro výstupy, které zapisují celkový stav všech komponent zařízení a ne pouze události. Zpráva o odpojení zařízení taktéž slouží k informování výstupů, ale také říká vstupnímu modulu, že má ovladač ukončit.

5.2.4 Zprávy

Rozhraní pro všechny zprávy je rovněž jednotné, čímž usnadňuje komunikaci a zároveň dává možnost rozšíření. Zprávy putují z modulů přes jádro do příslušných modulů na druhé straně. Z výstupních může přijít pouze zpráva typu feedback, která kromě údajů o samotném feedbacku má časovou značku a identifikátor zařízení, pro které je určena.



Ilustrace 7: Schéma posílání zpráv

Zpráva ze vstupu je tvořena v ovladačích, obsahuje identifikátor ovladače a modulu svého původu, čas ve kterém došlo k události, typ události a její hodnoty. Hodnotami jsou informace o změně v datech, nepřenáší stav celého zařízení.

5.3 Řešení nedostatků trackd

Z požadavků vyplývá řešení problému s licencí trackd (2.3.1). Nový framework bude umět automaticky detekovat zařízení a připojovat je za běhu (2.3.2). Problém s neustálým čtením (2.3.4) bude vyřešen posíláním událostí. Tyto události budou obsahovat i identifikátor ovladače, který je vyslal, takto bude možné rozlišit, která data jsou od kterého zařízení (2.3.3). Pro integraci do současného systému se nedá změnit výstup, avšak do paměti už nebude zapisováno tak často, ale jen pokud přijde nová událost (2.3.5). Z výstupních modulů bude možné zaslat zpětnou vazbu vstupním (2.3.6).

5.4 Možnosti rozšíření

Z návrhu vyplývá, že nový framework bude možné flexibilně rozšířit o vlastní typy výstupů, vstupů, ovladačů a událostí bez zásahu do jeho jádra. Tím je otevřená cesta pro připojení vstupů a výstupů přes síť pomocí UDP nebo TCP. Rovněž není omezený počet možných vstupních zařízení, API a dokumentace na vytvoření dalších ovladačů a modulů umožní rozšíření podle budoucích požadavků.

6 Implementace

Tato kapitola popisuje implementační stránku návrhu z kapitoly 5. Popis začíná podpůrnými třídami a poté rozebírá hlavní části programu.

Při spuštění aplikace není třeba zadávat žádné parametry, k ukončení je nutné vyslat signály SIGINT nebo SIGTERM, jinak nedojde k řádnému ukončení.

6.1 Knihovny a externí komponenty

Aplikace využívá standardních knihoven C++ 11. K načítání konfiguračního souboru používá TinyXML[39], které je integrované do aplikace. Třída `defaultDevice` pro čtení používá knihovnu `linux/input.h`.

Po dohodě s vedoucím práce tato aplikace využívá v ovladači pro zařízení a ve výstupním modulu do sdílené paměti kódu, který byl předem připravený vedoucím práce. Hlavním účelem této práce je totiž navržení celé aplikace, nikoliv psaní vstupů a výstupů.

6.2 Konfigurační soubor

Před popisem vlastní implementace je nutné zmínit konfigurační soubor. Ten se musí jmenovat mainconfig.xml a být ve validním XML formátu, aby bylo možné jeho načtení. Při selhání načtení se aplikace sama ukončí, jelikož není možné bez tohoto souboru načíst jakékoliv moduly. XML formát byl vybrán pro možnost jednoduché úpravy uživatelem.

Formát souboru je daný nejen pravidly XML, kořenový prvek musí obsahovat dva prvky, první s výstupy, druhý se vstupy. Prohození by mělo za následek problémový chod, jelikož jádro by jednalo se vstupy jako s výstupy a obráceně. Komunikace směrem

k výstupům by nebyla takto možná, jelikož feedback se posílá jen určenému modulu. V obou prvcích může pak být libovolný počet prvků `module`.

Tento typ prvku může obsahovat pouze prvky, které uvnitř mají pouze text. Větší hloubka již nebude aplikací zaznamenána. `Module` musí nutně obsahovat dva prvky, `class` a `id`, bez kterých není možné modul spustit. `Id` musí pak být jedinečné mezi vstupy i výstupy. Jedinečnost není kontrolována a může způsobit nekorektní chování. `Class` obsahuje klíčové slovo, podle kterého se vytvoří určitý typ modulu. Stejnou hodnotu `class` může mít libovolný počet modulů. Veškeré prvky pod `module` jsou načtené do pole, kde název prvku je klíčová hodnota. Tímto je možné předávat libovolné hodnoty pro moduly.

6.3 Třídy typu builder

Tato skupina zahrnuje tři třídy, které mají pouze statické metody a slouží k vytváření objektů. Jsou jimi `deviceBuilder`, `eventBuilder` a `moduleBuilder`. Vytváří instance typu `device`, `eventMessage` a `module` a vrací jejich `std::shared_ptr`. Při rozšíření o další typy těchto komponent stačí změnit nebo přidat statické metody těchto tříd. `EventBuilder` má na každý typ zprávy jednu metodu, zatímco `moduleBuilder` převezme všechna data načtená z konfiguračního souboru a vytvoří objekt na základě prvku *class*.

Všechny třídy vrací `nullptr` v případě neúspěchu načtení.

6.4 Události

Každý typ události musí být potomkem `eventMessage` a implementovat její virtuální metody:

`getType` – vrací typ události z výčtu `eventType` z eventType.h. Typ je nastavený v konstruktoru,

`getTypeStr` – vrací textovou reprezentaci typu události jako `std::string`,

`getDeviceId` – vrací identifikátor ovladače původu/cíle zprávy,

`getModuleId` – vrací identifikátor modulu původu/cíle zprávy,

`setModuleId` – nastavuje `std::string` identifikátoru modulu z parametru,

setDeviceId – nastavuje `std::string` identifikátoru ovladače z parametru,

getTimestamp – vrací čas, ve kterém došlo k události. Čas se nastavuje v konstruktoru události.

K datu vydání této práce aplikace obsahuje 4 typy událostí: `NOTICE`, `NEW_DEVICE`, `FEEDBACK`, `DATA_UPDATE`.

6.4.1 EventMessageNotice

Tato třída typu `eventType::NOTICE` slouží k předávání jednoduchých textových zpráv. Kromě metod předka má metody `getData` a `setData` pro nastavení dat, která jsou typu `std::string`. Data ve tvaru BYE znamenají, že došlo k ukončení ovladače, ze kterého zpráva pochází.

6.4.2 EventMessageNewDevice

Tato třída typu `eventType::NEW_DEVICE` obsahuje kompletní údaje o nově připojeném zařízení. Kromě jména zařízení dodává ještě seznam komponent, které zařízení má. Ty jsou uloženy v `std::vector` jako pointery pod názvem `input` a jsou dostupné metodami `getInput` a `setInput`. Více o komponentách v kapitole 6.5.

6.4.3 EventMessageDataUpdate

Třída typu `eventType::DATA_UPDATE` nese informaci o změně stavu jedné komponenty. Součástí je jméno a kód komponenty pro její identifikaci a nová hodnota. Tyto tři atributy jsou přístupné přes metody get a set.

6.4.4 EventMessageFeedback

Třída typu `eventType::FEEDBACK` není k datu vydání práce implementována, jelikož výstup do sdílené paměti neumožňuje zpětnou vazbu. Je ale připravena pro budoucí rozšíření.

6.5 Vstupní komponenty

Komponenty musí být všechny potomkem třídy `ic` a implementovat její virtuální metody:

`getCode` – vrací číselný kód komponenty podle linux/input.h,

`getName` – vrací jméno komponenty jako `std::string`. Jméno je generované podle kódu,

`getType` – vrací typ komponenty z výčtu `inputType` z inputType.h.

K datu vydání této práce aplikace obsahuje 3 typy komponent: `BUTTON`, `REL_AXIS`, `ABS_AXIS`. Tlačítko a relativní osa, která dává relativní posun v prostoru, se vnitřní strukturou neliší, zatímco `ABS_AXIS` vyjadřuje absolutní polohu a k jejímu vyhodnocení je nutné dodat další informace. Jsou to minimum (min), maximum (max), fuzz, flat a rozlišení (res). Všechny atributy jsou dostupné přes get a set metody

6.6 Jádro

Jádro (třída `core`) při volání konstruktoru načítá konfigurační soubor, při neúspěchu se aplikace ukončí. Během načtení vytvoří `std::map` objekt, do kterého ukládá konfigurační hodnoty, celý objekt předá `moduleBuilderu` a uloží si návratovou hodnotu. Běh aplikace se spustí metodou `run` a končí při odchycení signálů SIGINT nebo SIGTERM, kdy jádro spustí `module::bye` pro každý z modulů a zastaví práci vláken.

Jádro má dvě vlákna, každé z nich obstarává jednu frontu událostí. V případě příchozích událostí vlákno vyjme událost z fronty a pošle ji všem odchozím modulům, u odchozí fronty událost pošle pouze modulu, pro který je zpráva určena. Přístup k frontě je zamykaný přes `std::unique_lock<std::mutex>`, který má každá fronta vlastní.

K frontám událostí se přistupuje přes `core::sendIn` a `core::sendOut`. Obě funkce přijímají `std::shared_ptr` s `eventMessage` a nedělají nic jiného, než že přidají ukazatel do fronty a probudí příslušné vlákno.

6.7 Moduly

Každý typ modulu musí být potomkem třídy `module` a implementovat její virtuální metody:

`bye` – příkaz k ukončení modulu, vrací 0 pokud bylo ukončení úspěšné,

`refresh` – v parametru dostává `std::map` s konfiguračními hodnotami,

`accept` – dostává jako parametr `std::shared_ptr<eventMessage>`,

`getID` – vrací `std::string` s identifikátorem modulu.

Dále má každý modul proměnnou `coreptr`, ve které je ukazatel na jádro. K datu vydání práce aplikace zahrnuje tři pomocné moduly. `SimpleModule` slouží jako ukázka pro vytváření vlastních modulů a lze ho použít pro vstup i výstup. `MockInModule` posílá v cyklu zprávy a lze ho použít pro simulaci vstupu. `MockOutModule` přijímá zprávy, vypisuje je a posílá zpět feedback zařízení, od kterého přišla zpráva. Hlavní dva moduly jsou popsány v následujících dvou podkapitolách.

6.7.1 StandardInputModule

Tento vstupní modul je určený pro správu zařízení přímo připojených do počítače s aplikací přes USB. Připojuje je do aplikace, sbírá od nich informace a dává příkaz k odpojení. Pro jeho správnou funkci je důležité obdržet z konfiguračního souboru řádek:

```
<devicepath>/dev/input/event*;</devicepath>
```

kde * je nahrazena číslem, pod kterým se zařízení připojuje. Je možné napsat více cest za sebou, je ale nutné je vždy ukončit středníkem, jinak není cesta platná. Při neplatné cestě nedojde k otevření spojení se zařízením.

Při volání metody `bye` zavolá modul nad každým `device`, třídou nad každým zařízením, metodu `device::close` a smaže na ně odkazy. Při `refresh` načte nové cesty k zařízením a vyžádá si od každého zařízení novou úvodní zprávu, protože mohlo dojít k připojení nových vstupů. Při obdržení zprávy metodou `accept` najde `device`, kterému je zpráva určena a předá ji.

Pro přijímání zpráv ze zařízení slouží metoda `sendOut`, která zprávu zařadí do fronty a probudí vlákno. Samotné vlákno kromě předávání zpráv jádru zjišťuje, jestli se nejedná o

zprávu o ukončení zařízení. V takovém případě nad zařízením, od kterého zpráva přišla, zavolá close a smaže si jeho odkaz. Poté zprávu pošle jádru.

Každá zpráva je při průchodu z modulu do jádra opatřena identifikátorem modulu. Kritické sekce se stejně jako v jádru zamykají pomocí `std::lock_guard<std::mutex>`.

6.7.2 TrackdModule

Výstupní modul, který zapisuje do sdílené paměti počítače ve stejném tvaru, jako trackd. Zprávy přijímá přes `accept` a zařazuje do fronty, kterou zamyká totožně jako jádro přes `std::unique_lock<std::mutex>`. Práce vlákna se spouští v konstruktoru a ukončuje se v příkazu `bye`. Uvnitř cyklu vlákna se zprávy zpracovávají.

Jelikož trackd umí rozeznávat pouze jedno zařízení a zapisuje stav všech komponent, drží si modul stav zařízení uložený ve dvou polích, jedné pro tlačítka, druhé pro osy. Tyto pole se naplní při příchodu `eventMessageNewDevice` a s každým dalším novým zařízením se pole ještě rozšiřují o další prvky. Při příchodu `eventMessageNotice` s obsahem „BYE“ se obě pole vyprázdní.

Pokud je tlačítek nebo os více než 32, zapíše se do paměti jen prvních 32 prvků daného typu. To je určeno omezením trackd. O pořadí rozhoduje číslo v linux/input.h. Inicializační hodnota je pro relativní osu a tlačítko 0, pro absolutní osu je to polovina intervalu mezi minimem a maximem osy.

`EventMessageDataUpdate` mění nastavené hodnoty v poli a po dokončení části cyklu volá metodu pro zápis do paměti.

6.8 Ovladače

Ovladače musí být potomky třídy `device` a implementovat její virtuální metody:

`getId` – vrací identifikátor ovladače,

`open` – zahájí činnost ovladače,

`close` – ukončuje činnost ovladače,

`acceptFeedback` – přijímá události typu feedback.

Dále musí mít každý atribut `out`, který je ukazatelem na nadřízený modul, kam bude posílat zprávy.

6.8.1 DefaulfDevice

DefaultDevice je konkrétní implementace ovladače pro vstup založený na linux/input.h. Pro jeho fungování je nutná dostupnost této knihovny a povolená práva pro čtení pro soubory /dev/input/event*.

Ovladač naslouchá událostem určitého eventu a rozhoduje, které pošle dál. Ty pak „obalí“ do zprávy a posílá svému modulu. Pokud ovladač přestane být schopný číst, vyšle zprávu s obsahem „BYE“ a modul v odpověď na to zavolá metodu ovladače `bye`, která ukončí činnost ovladače a připraví jeho vymazání.

6.9 Testování

Aplikace byla testována na vstupních zařízeních mnoha typů. Jmenovitě na klávesnici (Microsoft SideWinder X4 Keyboard), myši (Miscrosoft), gamepadech (Gamepad Genius MaxFire G-08XU a Elecom JC-U2812FBK), joysticku (Saitek Cyborg EVO Wireless Joystick), volantu (Microsoft SideWinder Force Feedback Wheel), grafickém tabletu (Wacom Bamboo MTE-450) a taneční podložce (Myflash 5in1 Deluxe Dance Pad Hard Foam). Ve všech případech bez běh bez závad, připojování i odpojování zařízení fungovalo taktéž bez potíží. Testování současného běhu více zařízení též neodhalilo problémy.

K testování výstupu byly použity dvě aplikace pro CAVE: Dark Bulb a Triangles.

Při běhu bylo zjištěno značné zlepšení oproti trackd, aplikace má pouze poloviční spotřebu paměti a při stejném čase reakce na událost (průměrně 0,2 ms) spotřebuje pouze méně než 0,01 % CPU. Potřeba CPU je tak o přibližně 35 % nižší, než u trackd.

7 Závěr

Cílem této práce bylo navrhnout, implementovat a nasadit aplikaci, která by mohla nahradit trackd v IIM na ČVUT v Praze. Výsledkem této práce skutečně je aplikace odpovídající požadavkům, která spolupracuje s CAVElib aplikacemi.

Přínosy práce jsou zřejmé. Aplikace netrpí nedostatky trackd a je schopná rozšíření, které je tolik nutné pro budoucí rozvoj CAVE aplikací. Oproti trackd běží framework při stejné rychlosti s průměrně o 35 % nižší spotřebou CPU a poloviční spotřebou paměti.

8 Literatura

1. KENYON, Robert V. Kenyon. THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT: CHARACTERISTICS AND APPLICATIONS. *Human-Computer Interaction and Virtual Environments,* [online]. 1995 [cit. 2013-12-29]. Dostupné z: http://www.cs.uic.edu/~kenyon/Conferences/NASA/Workshop_Noor.html
2. CAVE (Cave Automatic Virtual Environment). ROUSE, Margaret. [online]. 2011 [cit. 2013-12-29]. Dostupné z: http://whatis.techtarget.com/definition/CAVE-Cave-Automatic-Virtual-Environment
3. Virtuální realita a vizualizace: Zařízení typu CAVE na ČVUT v Praze. In: *Institut Intermédií* [online]. [cit. 2013-12-30]. Dostupné z: http://www.iim.cz/?id=69
4. TRÁVNÍČEK, Zdeněk. *Testování knihovny CAVElib* [online]. 2007 [cit. 2013-12-29]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/travnz1_2007dipl.pdf. Diplomová práce. České Vysoké Učení Technické.
5. Trackd THE Device Driver Software for Immersive Displays. *Mechdyne Corporation* [online]. [cit. 2013-12-26]. Dostupné z: http://www.mechdyne.com/trackd.aspx
6. VRCO, Inc. TrackdAPI™ Documentation. [online]. [cit. 2013-12-26]. Dostupné z: http://vrc.zid.jku.at/studenten/dokumente/trackdAPI.pdf
7. VRCO, Inc. Trackd Reference Manual: Version 5.5. [online]. 2005 [cit. 2013-12-26]. Dostupné z: http://vislab.jsums.edu/VIS/DOC/trackd_Reference_Manual.pdf
8. VRCO, Inc. Trackd User's Guide: Version 5.5. [online]. 2005 [cit. 2013-12-26]. Dostupné z: http://vislab.jsums.edu/VIS/DOC/trackd_User_Guide.pdf
9. VRCO, Inc. CAVElib: User and Reference Guide. [online]. 2004 [cit. 2013-12-26]. Dostupné z: http://vislab.jsums.edu/VIS/DOC/CAVELib3.1.1.pdf
10. PALOČKO, Bc. Vladimír. *Generalized interactive techniques for Collaborative VR System* [online]. 2011 [cit. 2014-01-03]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/palocvla_2011dipl.pdf. Diplomová práce. ČVUT v Praze.

11. LANG, Ben. An Introduction to Positional Tracking and Degrees of Freedom (DOF). In: *Road to Virtual Reality* [online]. 2013 [cit. 2014-01-04]. Dostupné z: http://www.roadtovr.com/introduction-positional-tracking-degrees-freedom-dof/

12. VÍTEK, Jan. Klávesnice a jejich historie. In: *Svět Hardware* [online]. 2006 [cit. 2014-01-03]. Dostupné z: http://www.svethardware.cz/klavesnice-a-jejich-historie/14552-2

13. BOWMAN, Doug. VE Input Devices. *Virginia Tech: Department of Computer Science* [online]. [cit. 2013-12-26]. Dostupné z: http://courses.cs.vt.edu/cs5754/lectures/input.pdf

14. SpaceMouse Pro: Superior 3D Navigation For Professionals. In: *3Dconnexion* [online]. [cit. 2014-01-04]. Dostupné z: http://www.3dconnexion.com/products/spacemousepro.html

15. CyberWorld, Inc. Virtual Reality Solutions: SpaceOrb 360 (demo units). In: [online]. [cit. 2014-01-04]. Dostupné z: http://www.cwonline.com/store/view_product.asp?Product=1108

16. G27 RACING WHEEL. In: [online]. [cit. 2014-01-03]. Dostupné z: http://gaming.logitech.com/cs-cz/product/g27-racing-wheel

17. Dance pad. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2014-01-03]. Dostupné z: http://en.wikipedia.org/wiki/Dance_pad

18. ParaParaParadise. In: *The International Arcade Museum* [online]. [cit. 2014-01-04]. Dostupné z: http://www.arcade-museum.com/game_detail.php?game_id=8978

19. BOWMAN, Doug A., Chadwick A. WINGRAVE, Joshua M. CAMPBELL a Vinh Q. LY. Using Pinch Gloves™ for both Natural and Abstract Interaction Techniques in Virtual Environments. [online]. 2001 [cit. 2013-12-26]. Dostupné z: http://eprints.cs.vt.edu/archive/00000547/01/PinchGlovePaper.pdf

20. PINCH Glove as VR Input Device. In: DAWE, Greg, Jim COSTIGAN, Hiroshi HAYAKAWA, Dave PAPE a Matthew SZYMANSKI. *Electronic visualization laboratory* [online]. 1993 [cit. 2013-12-26]. Dostupné z: http://www.evl.uic.edu/core.php?mod=4&type=1&indi=170

21. LAVIOLA, Joseph a Robert ZELEZNIK. Flex And Pinch: A Case Study Of Whole Hand Input Design For Virtual Environment. [online]. 1999 [cit. 2013-12-26]. Dostupné z: http://cs.brown.edu/people/jjl/pubs/flex.pdf

22. HO, Ray a Ayo OSITELU. Optical Tracking System. In: *ECE 4760: Optical Eye Tracking System* [online]. 2010 [cit. 2014-01-04]. Dostupné z: https://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2010/yh428_aoo34/eyetracking/

23. What is Eye tracking?. In: *Eye Tracking: Advanced Eye-Tracking Technology by LC Technologies* [online]. [cit. 2014-01-04]. Dostupné z: http://www.eyegaze.com/what-is-eye-tracking/

24. Disadvantages of Eye Tracking. In: *Eye Tracking* [online]. 2007 [cit. 2014-01-04]. Dostupné z: http://professionals306.blogspot.cz/2007/05/disadvantages-of-eye-tracking.html

25. LAVIOLA JR., Joseph, Daniel KEEFE, Robert ZELEZNIK a Daniel Acevedo FELIZ. Case Studies in Building Custom Input Devices for Virtual Environment Interaction. [online]. 2004 [cit. 2013-12-26]. Dostupné z: http://cs.brown.edu/people/jjl/pubs/vr2004_workshop.pdf

26. LAVIOLA JR., Joseph, Daniel KEEFE, Robert ZELEZNIK a Daniel Acevedo FELIZ. Hands-Free Multi-Scale Navigation in Virtual Environments. [online]. 2001 [cit. 2013-12-26]. Dostupné z: http://cs.brown.edu/people/jjl/pubs/i3d01_laviola.pdf

27. KEEFE, Daniel, Daniel Acevedo FELIZ, Tomer MOSCOVICH, David LAIDLAW a Joseph LAVIOLA JR. CavePainting: a fully immersive 3D artistic medium and interactive experience. [online]. 2001 [cit. 2013-12-26]. Dostupné z: http://cs.brown.edu/people/jjl/pubs/i3d01_cavepainting.pdf

28. KELSO, John, Lance E. ARSENAULT, Steven G. SATTERFIELD a Ronald D. KRIZ. DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments. [online]. 2003 [cit. 2013-12-26]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5614&rep=rep1&type=pdf

29. ARSENAULT, Lance, John KELSO, Ron KRIZ a Fernando DAS NEVES. DIVERSE: a Software Toolkit to Integrate Distributed Simulations with Heterogeneous Virtual Environments. [online]. 2001 [cit. 2013-12-26]. Dostupné z: http://eprints.cs.vt.edu/archive/00000534/01/DIVERSE.pdf

30. Syzygy: Cross-Platform, Open Source Virtual Reality on PC Clusters. UNIVERSITY OF ILLINOIS BOARD OF TRUSTEES. [online]. 2013 [cit. 2013-12-26]. Dostupné z: http://syzygy.isl.uiuc.edu/szg/index.html

31. CHAEFFER, Benjamin a Camille GOUDESEUNE. Syzygy: native PC cluster VR. [online]. 2003 [cit. 2013-12-26]. Dostupné z: http://80.ieeexplore.ieee.org.dialog.cvut.cz/stamp/stamp.jsp?tp=&arnumber=1191116

32. Syzygy Documentation: Input Framework. In: [online]. [cit. 2014-01-04]. Dostupné z: http://syzygy.isl.uiuc.edu/szg/doc/InputDevices.html

33. The VR Juggler Suite. [online]. [cit. 2013-12-26]. Dostupné z: http://vrjuggler.org/documentation.php

34. ANDÚJAR, Carlos. VR Juggler. *Universitat Politècnica de Catalunya BarcelonaTech.: Departament de Llenguatges i Sistemes Informàtics* [online]. 2007 [cit. 2013-12-26]. Dostupné z: http://www.lsi.upc.edu/~virtual/RVA/Course%20Slides/06.%20VR%20software%20-%20VR%20Juggler.pdf

35. BOWMAN, Doug. VE software support systems and standards. [online]. 2006 [cit. 2014-01-05]. Dostupné z: http://courses.cs.vt.edu/cs5754/lectures/software.pdf

36. Introduction to DirectInput. In: Windows Dev Center - Desktop: Dev Center - Desktop [online]. [cit. 2014-01-05]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/desktop/ee418273(v=vs.85).aspx

37. Credit and feedback for VRPN. [online]. [cit. 2014-01-05]. Dostupné z: http://www.cs.unc.edu/Research/vrpn/obtaining_vrpn.html

38. TAYLOR II, Russell M., Thomas C. HUDSON, Adam SEEGER, Hans WEBER, Jeffrey JULIANO a Aron T. HELSER. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. In: *University of North Carolina at Chapel Hill: Department of Computer Science* [online]. [cit. 2014-01-05]. Dostupné z: http://www.cs.unc.edu/Research/vrpn/VRST_2001_conference/vrst_vrpn_paper_reprint.pdf

39. Stránka projektu TinyXML: http://www.grinninglizard.com/tinyxml/

40. Zdroj obrázku: http://www.3dshop.cz/data/imgs/02813l.png

41. Zdroj obrázku: http://images.esellerpro.com/2261/W/922/0/scaleps2dancemat2.jpg

42. Zdroj obrázku: http://ps2media.ign.com/media/previews/image/ppp/con_in.jpg

Příloha A

Obsah přiloženého CD

Přiložené CD obsahuje elektronickou podobu této práce ve dvou formátech, dále pak zdrojové kódy výsledné aplikace.