České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Julie Partyková**

Studijní program: Otevřená informatika (magisterský)
Obor: Počítačová grafika a interakce

Název tématu: **Komunikační protokol pro VR systémy**

Pokyny pro vypracování:

Na základě rešerše ze semestrálního projektu specifikujte typy vstupních událostí používaných ve VR aplikacích. Dále navrhněte protokol pro obousměrnou komunikaci VR aplikace s daemonem abstrahujícím vstupní zařízení.
Navrhněte také rozšiřitelný systém adaptací mezi různými typy vstupních událostí. Implementujte SW knihovnu využívající navržený protokol a adaptace. Implementujte také demonstrační aplikaci využívající tuto knihovnu pro obsluhu vstupů.
Otestujte komunikaci mezi daemonem a testovací aplikací. Pro testování použijte alespoň 2 různá zařízení. V rámci testu bude aplikace vyžadovat vstupy alespoň 3 různých typů. U alespoň 2 vstupů bude změněn typ pomocí definovaných konverzí. Otestujte také alespoň jednu událost z aplikace do zařízení.
Zhodnoťte vliv protokolu na velikost latence v komunikaci. Implementaci realizujte v jazyce C++. Výsledná knihovna bude fungovat minimálně pod systémem Linux. Součástí práce bude i dokumentace API knihovny v anglickém jazyce.

Seznam odborné literatury:

V. Paločko, Generalized interactive techniques for Collaborative VR System, Diplomová práce, FEL ČVUT v Praze, 2011

Vedoucí: Ing. Zdeněk Trávníček

Platnost zadání: do konce letního semestru 2014/2015

L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 2. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Komunikační protokol pro VR systémy

Bc. Julie Partyková

Vedoucí práce: Ing. Zdeněk Trávníček

Studijní program: Otevřená informatika, Navazující magisterský

Obor: Počítačová grafika

11. května 2014

Poděkování

Ráda bych poděkovala Zdeňkovi Trávníčkovi za vedení této práce a užitečné rady.

Prohlášení

Prohlašuji, že jsem práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 11. 5. 2014 .....

Abstract

Virtual reality (VR) is constantly developing area and new control devices which can interact with VR applications are perpetually being generated. The problem with such a high diversity of input devices is the effectiveness of their connection to VR applications.

The goal of this work is to design and implement a protocol for communication with VR systems ensuring two-way communication between VR applications and input control devices. The protocol also allows the highest possible abstraction of input events through the use of adaptations of the input data.

Abstrakt

Virtuální realita (dále VR) je široce rozvíjená oblast a neustále vznikají nová ovládací zařízení, jimiž lze s aplikacemi VR interagovat. Problémem tak vysoké rozmanitosti vstupních zařízení je však efektivita jejich propojení s aplikacemi VR.

Cílem této práce je navrhnout a realizovat protokol pro komunikaci se systémy VR, který zajistí obousměrnou komunikaci mezi aplikacemi VR a vstupními ovládacími zařízeními. Protokol zároveň umožní co nejvyšší abstrakci vstupních událostí za pomoci systému adaptací vstupních dat.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Virtuální realita se v současné době velmi dostává do popředí zájmu. Nejen, že se využívá na poli her a zábavy, ale stále častěji se uplatňuje i v široké škále profesionálních oblastí jako je například architektura, medicína, vývoj průmyslových prototypů a v mnoha dalších odborných směrech vědy a výzkumu. Je pravděpodobné, že oblast virtuální reality čeká do budoucna další rozvoj a šíře jejího využití se bude ještě zvětšovat. Právě i tato perspektiva mě zaujala, abych se touto problematikou ve své diplomové práci zabývala a pokusila se přispět ke zlepšení možností ovládání aplikací virtuální reality.

Aktuálně vznikají stále nová vstupní zařízení, která lze využít k ovládání virtuální reality, ale zároveň s tím se výrazně zvyšuje také problematika propojení aplikací virtuální reality se zvětšující se množinou těchto dostupných zařízení. Každá z aplikací virtuální reality pracuje s určitými vstupními událostmi a ve stávající situaci musí tvůrci těchto aplikací vstupní události zpracovávat a modifikovat v rámci aplikace pro každé vstupní zařízení zvlášť, což není žádoucí stav. Stejně tak v případě, že aplikace vyžaduje zasílat na vstupní zařízení zpětnou vazbu, není ideální řešit v rámci aplikace, která konkrétní zařízení ji požadují a v kterém formátu. Sice již existují nějaká řešení této problematiky (podrobněji se tomu věnuji v podkapitole 2.5), ale abstrakce událostí v nich není dostatečně vysoká (případně není vůbec žádná) a také se příliš nezabývají možnostmi zpětné vazby.

Proto je potřeba způsoby komunikace ještě rozvíjet tak, abychom naplno využili potenciál dostupné interakce mezi periferními ovládacími prvky a aplikacemi virtuální reality bez větších komplikací při tvorbě aplikací. Právě to je důvodem, proč se ve své práci zabývám tvorbou komunikačního protokolu pro systémy virtuální reality a doufám, že to do budoucna usnadní práci s virtuální realitu a možná i zvýší její využitelnost.

Cílem této diplomové práce je tedy vytvořit návrh komunikačního protokolu, který bude zajišťovat vhodnou komunikaci mezi aplikacemi virtuální reality a různými vstupními ovládacími zařízeními. Významnou složkou tohoto protokolu je návrh rozšiřitelného systému adaptací mezi vstupními událostmi (daty), které přicházejí z ovládacích zařízení a výstupními událostmi určenými pro aplikaci, za účelem dosažení co nejvyšší abstrakce vstupních událostí. V rámci této práce byla implementována softwarová knihovna, která využívá tento navržený protokol a adaptace dat. Současně práce obsahuje i ukázkovou aplikaci, na které je demonstrována funkčnost této knihovny pro obsluhu vstupů.

Kapitola 2

Analýza

V této kapitole se podrobněji věnuji možnostem interakce s virtuální realitou. Nejprve se v obecné rovině zabývám otázkou, jakými způsoby lze s virtuální realitou interagovat a jaká vstupní (ovládací) zařízení se k tomu například dají použít. Dále jsou zde rozebrány typické události, kterých chceme obvykle ve virtuální realitě dosáhnout. Jedná se zejména o pohyby scénou, ale i další. Následuje krátké shrnutí přenosových protokolů a jazyků, které se používají pro komunikaci. V závěru této kapitoly je výčet již existujících projektů, zabývajících se totožnou, nebo velmi blízkou problematikou.

2.1 Interakce uživatelů s VR systémy

Ve virtuálním prostředí existuje několik základních druhů interakce. Prvním z nich je navigace, nebo-li pohyb v tomto prostředí (virtuálním světě). Dalšími jsou výběr (označení) objektů ve virtuálním prostředí a možnost manipulovat s těmito objekty (posun, otočení, změna měřítka).

Také může dojít k potřebě speciálních druhů interakce v závislosti na aplikaci. Speciální interakcí může být například střelba, házení objektů, otvírání/zavírání dveří, apod.

2.1.1 Navigace

Navigace, nebo-li pohyb virtuálním prostředím může probíhat dvěma způsoby. Při prvním z nich pohybujeme "sebou"(my chodíme světem) a tento způsob se nazývá Eye-in-hand (EiH).

V druhém případě pohybujeme virtuálním světem kolem nás a tomu se říká World-in-hand (WiH). [5]

Eye-in-hand

Pohyby odpovídají běžné chůzi prostorem - tj. když se chceme podívat více vlevo, použijeme posun vlevo a "popojdeme"tak prostředím doleva (ve skutečnosti se tedy prostředí posune doprava). Stejně tak s otáčením - naše otočení vlevo ve skutečnosti znamená otočení prostředí kolem nás doprava.

World-in-hand

Pohyby jsou oproti EiH převrácené - my jsme statickým bodem a pohybujeme prostředím kolem nás, tj. pokud se chceme podívat více vlevo, použijeme posun vpravo, který prostředím posune doprava. U otáčení taktéž, pokud se chceme otočit doleva, musíme použít otočení vpravo a otočit tak okolním prostředím doprava.

Směr a rychlost

Obecně nás při navigaci zajímají dvě vstupní informace - směr, kterým se chceme pohybovat, a rychlost tohoto pohybu.

2.1.2 Selekce

Selekce může být buď lokální (je možno označit objekty, na které by teoreticky dosáhla naše ruka) a nebo vzdálená (je možno označit i objekty, které jsou příliš daleko, než aby na ně naše ruka mohla dosáhnout). Vzdálená selekce je sice méně realistická, avšak praktičtější (rychlejší) především ve velkých virtuálních prostředích.[5]

Při provádění selekce se musíme postupně zabývat několika úkoly:

- jakým způsobem ukázat na požadovaný objekt
- jak selekci daného objektu aktivovat
- jak vyjádřit odezvu systému na provedení selekce (tzv. zpětná vazba)
- jak selekci daného objektu deaktivovat

2.1.3 Manipulace

Manipulace stejně jako selekce může být lokální či vzdálená. Manipulace přímo navazuje na selekci (nebo spíše se s ní částečně prolíná) a k úkolům doprovázejícím selekci doplňuje další:

- jakým způsobem přesouvat objekt
- jakým způsobem otáčet objektem
- jakým způsobem měnit velikost objektu
- odezva systému na provedení manipulace(tzv. zpětná vazba)

Metody manipulace [13]

- přímá uživatelská manipulace (rozpoznání gest, pohybu uživatele a další)
- nepřímá manipulace
  - fyzická - ovládací zařízení s tlačítky apod.
  - virtuální - virtuálně zobrazené ovládací prvky ve virtuálním světě, k jejich ovládání potřebujeme fyzický ovladač (zařízení s tlačítky apod.)
  - agentní - ve virtuálním světe je postavička - agent, kterému říkáme, co má dělat (například pomocí kombinace rozpoznání hlasu a pohybu)

2.2 Dostupná zařízení

Ovládací zařízení se vzhledem k jejich funkčnímu principu dají rozdělit na **trackovaná** a **netrackovaná**. Netrackovaná jsou ta, kde k ovládání dochází pomocí tlačítek a dalších ovládacích prvků bez závislosti na poloze zařízení (a uživatele) v prostoru. U trackovaných zařízení naopak poloha a pohyb v prostoru hraje zásadní roli. [3] [11] [16] [8] [10]

Dále následuje soupis některých aktuálně používaných ovládacích zařízení, spolu s informací o jejich ovládacích prvcích. Z těchto informací vyplývají typy dat, která z nich lze získat - v popisu uvedená jako output. U některých zařízení se také vyskytují prvky pro zpětnou vazbu (možnost vibrace, LED diody...) - v popisu uvedená jako input.

2.2.1 Netrackovaná zařízení

2.2.1.1 Klávesnice

Standardní klávesnice

- USB/bezdrátová
- 101 nebo 104 kláves

- output:
  - boolean pro každou klávesu

Klávesnice s touchpadem/trackballem

- USB/bezdrátová
- 2DOF
- output:
  - stejně jako standardní klávesnice - tj. boolean pro každou klávesu
  - 2 integer hodnoty pro pohyb na touchpadu/trackballu

Logitech Wireless Touch Keyboard K400 CZ (Obr. 2.1)

- bezdrátová
- 2DOF
- output:
  - vícedotykový touchpad -> navigace pomocí dotyků a gest
  - 2 tlačítka (levé a pravé)
  - 84 kláves

Obrázek 2.1: Logitech Wireless Touch Keyboard K400 CZ

Razer Orbweaver Elite Mechanical Gaming Keypad (Obr. 2.2)

- USB
- 1DOF
- output:
  - 20 tlačítek

  - 8-mi směrný thumb-pad

- input:

  - podsvícení
  - 3 diody

Obrázek 2.2: Razer Orbweaver Elite Mechanical Gaming Keypad

2.2.1.2 Myš

Standardní myš

- USB
- 2DOF
- output:

  - 2 hodnoty pro posun (x,y)
  - 1 hodnota pro scrollovací kolečko
  - boolean podle počtu tlačítek

Logitech G602 Wireless Gaming Mouse (Obr. 2.3)

- bezdrátová
- 2DOF
- output:

  - 1 hodnota pro scrollovací kolečko
  - 11 + 2 tlačítek

Obrázek 2.3: Logitech G602 Wireless Gaming Mouse

2.2.1.3 Trackball

Logitech Trackball Wireless M570 (Obr. 2.4)

- bezdrátový
- 2DOF
- output:
  - 4 tlačítka (levé a pravé, vpřed a zpět)
  - scrollovací kolečko
  - trackball (pozice)

Obrázek 2.4: Logitech Trackball Wireless M570

Kensington optický kulový ovladač Orbit (Obr. 2.5)

- USB
- 2DOF
- output:
  - 2 tlačítka (levé a pravé)
  - trackball (pozice)

Obrázek 2.5: Kensington optický kulový ovladač Orbit

Genius Ring Presenter (Obr. 2.6)

- bezdrátový "prstýnek"
- 2DOF
- zabudovaný laser
- output:
  - 5 tlačítek (jedno z nich je touch senzor, ostatní klasická tlačítka)
  - hardware přepínač (mezi ukazovátkem, in-air mouse a vypnutím)

Obrázek 2.6: Genius Ring Presenter

2.2.1.4 Joystick

Standardní joystick

- USB
- 2DOF
- output:
  - kontrolní páka pro pohyb ve 2 osách -> 2 hodnoty (x,y)
  - alespoň 2 tlačítka -> n-krát boolean (n = počet tlačítek)

Logitech Extreme 3D Pro (Obr. 2.7)

- USB
- 2DOF
- output:
  - 12 tlačítek
  - přesná otočná rukojeť
  - osmisměrný kloboučkový přepínač

Obrázek 2.7: Logitech Extreme 3D Pro

Thrustmaster T.Flight Stick X (Obr. 2.8)

- USB
- 2DOF
- output:
  - 12 tlačítek
  - POV hat switch
  - ostatní: plyn, směrovky, airbrake, rapid fire

2.2.1.5 Gamepad

Standardní gamepad

- USB /bezdrátový
- 2DOF
- output:
  - boolean pro každé tlačítko (obvykle podporuje vícetlačítkový stisk)
  - numerické hodnoty pro joysticky
- input:
  - vibrace (některé i světelné diody, zvuk)

Obrázek 2.8: Thrustmaster T.Flight Stick X

Sony PS4 DualShock 4 (Obr. 2.9)

- bezdrátový
- 6DOF
- output:
  - 12 herních tlačítek (4x směrové, 4x akční, PS, SHARE, OPTIONS, Pad, L1, L2, R1, R2, L3, R3)
  - 2 x Joystick
  - dvoubodový dotykový panel (klikací mechanismus, kapacitní typ)
  - šestiosý systém snímání pohybu (tříosý gyroskop, tříosý akcelerometr)
- input:
  - vibrace
  - světelný pruh - 3 LED diody
  - vestavěný monofonní reproduktor (+konektor na sluchátka)

Microsoft XBOX 360 Wireless Common Controller (Obr. 2.10)

- bezdrátový
- 2DOF
- output:
  - 11 tlačítek (A/B/X/Y, L/P spoušť, L/P přední tlačítko, Start, Guide, Back)

Obrázek 2.9: Sony PS4 DualShock 4

  - 8-směrný D-PAD
  - 2 analogové ovladače
- input:
  - vibrace

Obrázek 2.10: Microsoft XBOX 360 Wireless Common Controller

Microsoft XBOX One Wireless Controller (Obr. 2.11)

- bezdrátový
- 2DOF
- output:
  - 10 tlačítek (A/B/X/Y, L/P spoušť, L/P přední tlačítko, Guide, Back)
  - 4-směrný D-PAD
  - 2 joysticky
- input:
  - vibrace

Obrázek 2.11: Microsoft XBOX One Wireless Controller

Steam Controller (Obr. 2.12)

- bezdrátový
- 2DOF
- output:
  - 16 tlačítek
  - 2 trackpady (+ fungují i jako tlačítko)
  - dotyková plocha (+ funguje i jako tlačítko)
- input:
  - 2 lineární rezonantní servomotory - velká škála vibrací (přesná kontrola nad frekvencí, amplitudou a směrem svého pohybu)

Obrázek 2.12: Steam Controller

2.2.1.6 Taneční podložka

X-PAD Extreme Dance Pad (Obr. 2.13)

- USB
- output:
  - 10 aktivních polí (8 šipek + Select + Start)

Obrázek 2.13: X-PAD Extreme Dance Pad

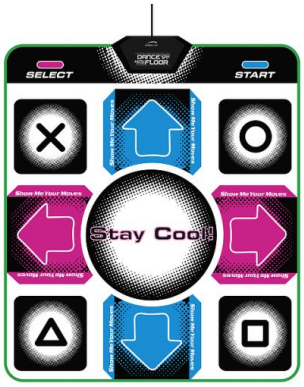
SPEED LINK Dance Mat (Obr. 2.14)

- USB
- output:
  - 10 aktivních polí (4 šipky + 4 tvary + Select + Start)

2.2.1.7 Volant

Thrustmaster Ferrari Red Legend Edition (Obr. 2.15)

- USB
- 2DOF
- output:
  - 11 akčních tlačítek



Obrázek 2.14: SPEED LINK Dance Mat

- D-pad
- 2 páčky sekvenčího řazení
- 2 pedály

Obrázek 2.15: Thrustmaster Ferrari Red Legend Edition

Saitek Pro Flight Yoke System (Obr. 2.16)

- USB
- 2DOF
- output:
  - letecký knipl - náklon a sklon
  - 14 tlačítek
  - POV Hat Switch
  - 3 trojpoziční přepínače
  - 3 páky na panelu ovládání motoru

Obrázek 2.16: Saitek Pro Flight Yoke System

2.2.2 Trackovaná zařízení

2.2.2.1 Rukavice

5DT Data Glove 5 Ultra (Obr. 2.17)

- USB
- 6DOF
- output:
  - ohyb prstu - 1 senzor pro každý prst (8-bitové rozlišení ohybu -> 256 pozic/prst)
  - orientace ruky (pitch and roll)

Obrázek 2.17: 5DT Data Glove 5 Ultra

The AcceleGlove (Obr. 2.18)

- USB
- 6DOF
- output:
  - pozice ruky tvořená tvarem a orientací ruky jako 12-byte vektor

Obrázek 2.18: The AcceleGlove

2.2.2.2 Trackování hlavy

TrackIR 5 (Obr. 2.19)

- USB
- 6DOF
- output:
  - poloha a otočení hlavy

Head-mounted display

- USB
- 6DOF
- output:
  - poloha a otočení hlavy

Obrázek 2.19: TrackIR 5

Occulus rift (Obr. 2.20)

- USB
- 6DOF
- output:
  - poloha a otočení hlavy

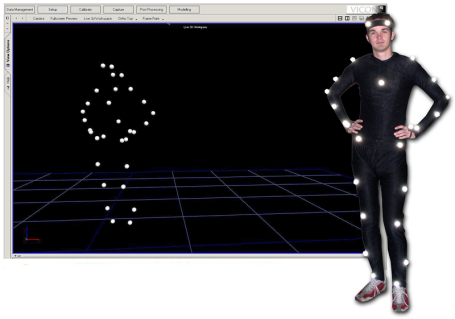
Obrázek 2.20: Occulus rift

2.2.2.3 Trackování pohybu

Vicon (Obr. 2.21)

- 3DOF/6DOF
- output:

  - pro každý marker pozice (3 číselné hodnoty), v případě 4 a více markerů i orientace (nesmí ležet ve stejné rovině nebo přímce)



Obrázek 2.21: Vicon

Kinect (Obr. 2.22)

- bezdrátový
- 3DOF
- output:
  - pozice bodů v prostoru

Obrázek 2.22: Kinect

Asus Xtion Pro (Obr. 2.23)

- USB
- 3DOF
- output:
  - infračervený senzor + detekce hloubky -> detekce gest a celého těla
  - pozice bodů v prostoru

Obrázek 2.23: Asus Xtion Pro

MTw Development Kit (Obr. 2.24)

- bezdrátový (USB přijímač pro komunikaci s počítačem)
- 6DOF
- 6 trackovacích modulů
- output:
  - pozice a otočení trackovacích modulů

Obrázek 2.24: MTw Development Kit

Stem System (Obr. 2.25)

- bezdrátový
- 6DOF
- 5 trackovacích modulů (STEMů) a STEM Controller
- obsahuje STEM Base, která modulům určuje stacionární referenci
- output:
  - pozice a orientace na všech 3 osách pro každý STEM modul
  - STEM Controller -> 9 tlačítek + joystick
- input:
  - možnost odezvy (ještě není uvedeno jaká)

Obrázek 2.25: Stem System

Leap motion (Obr. 2.26)

- USB
- 3DOF
- rozpoznání pohybů rukou ve 3D
- trackuje všech 10 prstů s přesností 1/100 milimetru
- output:
  - pozice bodů v prostoru

Obrázek 2.26: Leap Motion

Virtu Sphere (Obr. 2.27)

- 6DOF (samotná koule 3DOF)
- obvykle ve spojení s head-mounted displayem (člověk chodí uvnitř velké koule a na hlavě má HMD), mohou být zkombinována i další zařízení (pistole apod.)
- output:
  - pohybový senzor pod koulí -> pohyb člověka
  - senzor se 3 DOF nebo 6 DOF na HMD (také 6 DOF na zbrani pokud je v setu)

Obrázek 2.27: Virtu Sphere

2.2.2.4 Kouzelná hůlka

Nintendo Wii Remote Plus (Obr. 2.28)

- bezdrátový
- 6DOF
- output:
  - 8 tlačítek (A,B,+,-,power,HOME,1,2)
  - směrové tlačítko (D-pad)
  - infračervený senzor
  - akcelerometr a gyroskop
- input:
  - vibrace
  - zvuk

Obrázek 2.28: Nintendo Wii Remote Plus

Sony PS3 Move Motion Controller (Obr. 2.29)

- bezdrátový
- 6DOF
- output:
  - 9 tlačítek (+ reset)

  - ovladač spolupracuje s PlayStation Eye USB kamerou -> poloha
  - šestiosý systém snímání pohybu (tříosý gyroskop, tříosý akcelerometr)

- input:

  - dualshock vibrace

Obrázek 2.29: Sony PS3 Move Motion Controller

2.2.3 Shrnutí zařízení

Níže uvedená tabulka 2.1 shrnuje základní informace o výše uvedených zařízeních. Pro orientaci v tabulce je zde popis jednotlivých kategorií, které obsahuje:

- DOF - udává, kolik stupňů volnosti může uživatel ovládat jedním ovládacím prvkem ovladače
- Output - druhy informace, které můžeme ze zařízení získat
  - Tlačítka - počet tlačítek zařízení (vrací boolean)
  - Polohy - komponenty vracející informaci o poloze 2 DOF (např. touchpad, trackball)
  - Scroll - počet komponent vracejících rozsah
  - D-pad - počet směrů směrového tlačítka (skupiny tlačítek) pokud je přítomno
  - Jiné - výstupy nezařaditelné mezi druhy výše
- Input - schopnosti odezvy zařízení

Zařízení	DOF	Output					Input
		Tlačítka	Polohy	Scroll	D-pad	Jiné	
Standardní klávesnice	-	101/104	-	-	-	-	-
Klávesnice s touchpadem/trackballem	2	101/104	touchpad/ trackball	-	-	-	-
Logitech Wireless Touch Keyboard K400 CZ	2	84 + 2	touchpad	-	-	-	-
Razer Orbweaver Elite Mechanical Gaming Keypad	1	20	-	-	8	-	3 LED, podsvícení
Standardní myš	2	3	-	1	-	-	-
Logitech G602 Wireless Gaming Mouse	2	11 + 2	-	1	-	-	-
Logitech Trackball Wireless M570	2	4	trackball	1	-	-	-
Kensington optický kulový ovladač Orbit	2	2	trackball	-	-	-	-
Genius Ring Presenter	2	5	touch senzor	-	-	-	-
Standardní joystick	2	2+	páka	-	-	-	-
Logitech Extreme 3D Pro	2	12	páka	-	8	-	-
Thrustmaster T.Flight Stick X	2	12	páka	-	-	-	-
Standardní gamepad	2	4+	joystick	-	4 / 8	-	vibrace
Sony PS4 DualShock 4	6	12	2 joysticky	-	-	dotykový panel	3 LED, vibrace, zvuk
Microsoft XBOX 360 Wireless Common Controller	2	11	2 joysticky	-	8	-	vibrace
Microsoft XBOX One Wireless Controller	2	10	2 joysticky	-	4	-	vibrace
Steam Controller	2	16	2 trackpady	-	-	dotyková plocha	škála vibrací
X-PAD Extreme Dance Pad	-	10	-	-	-	-	-
SPEED LINK Dance Mat	-	10	-	-	-	-	-
Thrustmaster Ferrari Red Legend Edition	2	11	-	2 páčky, 2 pedály	4	-	-
Saitek Pro Flight Yoke System	2	14	knipl, hat switch	3 páky	-	3 trojpoziční přepínače	-
5DT Data Glove 5 Ultra	6	0	-	-	-	-	-
The AcceleGlove	6	0	-	-	-	-	-
TrackIR 5	6	0	-	-	-	-	-
Head-mounted display	6	0	-	-	-	-	-
Occulus rift	6	0	-	-	-	-	-
Vicon	3 / 6	0	-	-	-	-	-
Kinect	3	0	-	-	-	-	-
Asus Xtion Pro	3	0	-	-	-	-	-
MTw Development Kit	6	0	-	-	-	-	-
Stem System	6	9	joystick	-	-	-	-
Leap motion	3	0	-	-	-	-	-
Virtu Sphere	3 / 6	0	-	-	-	-	-
Nintendo Wii Remote Plus	6	8	-	-	4	-	vibrace, zvuk
Sony PS3 Move Motion Controller	6	9	-	-	-	-	vibrace

Tabulka 2.1: Tabulka vstupů a výstupů zařízení

V tabulce 2.1 vidíme, že existuje několik typů dat, která lze ze vstupních zařízení získat:

- boolean
- rozsah hodnot
- 1DOF - 1 hodnota vzdálenosti
- 2DOF - souřadnice 2D
- 3DOF - souřadnice 3D
- 6DOF - souřadnice 3D + otočení 3D

A také data, která lze do zařízení poslat:

- boolean
- rozsah hodnot

Z průzkumu dostupných zařízení vyplývá, že každé zařízení (nebo alespoň skupiny obdobných zařízení) potřebuje s aplikacemi VR komunikovat odlišně. Vzhledem k rozdílným ovládacím prvkům (a možnostem zpětné vazby) těchto zařízení je potřeba vytvořit způsob konverze dat z nich pocházejících na data potřebná pro aplikaci (a opačně). Tato konverze dat umožní sjednocení komunikace a také umožní určit, zda dané ovládací zařízení je pro ovládání dané aplikaci použitelné nebo ne. Čímž se dostáváme k tomu, že nezáleží pouze na vlastnostech ovládacích zařízení, ale také na vlastnostech - požadavcích - aplikací. Této problematice se věnuje následující kapitola.

2.3 Virtuální realita a její aplikace

2.3.1 Typické aplikace VR

Využití virtuální reality je opravdu velmi rozmanité a setkáme se s ním ve velkém množství odvětví ať už vědeckých, uměleckých či zábavních. [17] Zde je uveden výčet obvyklých oblastí využití:

- Architektura
  - využití při návrhu budov, silnic, mostů a dalších objektů a jejich umístění do okolí
  - studium monumentů
  - virtuální rekonstrukce poškozených nebo zničených památek

- Umění a design
  - vizualizace uměleckých děl a návrhů
- Vzdělání
  - využití real-time simulací kritických situací při trénování chirurgů
  - vzdělávací modely např. DNA apod.
- Inženýrství a prototypování
  - zobrazení prototypů produktů a simulace jejich použitelnosti dříve než se vyrobí
  - návrhy a virtuální "otestování"dopravních prostředků, strojů, součástek, atd.
- Hry a zábava
  - cave painting
  - nejrůznější hry
- Geografický informační systém (GIS) a vizualizace informace
- Matematika
- Vědecká vizualizace
  - v medicíně se uplatňuje především pro anatomický volumetrický rendering, simulace operací, neinvazivní prohlídky [6]
- Vizuální simulace
  - vizualizace částicových systémů, elektromagnetických polí, atd.
- Armáda
  - využití při návrhu kontrolního rozhraní pro ovládání bezpilotních letounů [15]

2.3.2 Typické případy užití

Na jednotlivé aplikace se také váže potřeba jejich ovládání. Různé aplikace mají různé požadavky v oblasti interakce, počínaje pouhým pohybem v prostoru až po manipulaci s objekty, házení, střílení, apod. A nejedná se jen o potřebu interakce směrem od uživatele k aplikaci, ale často také o zpětnou vazbu od aplikace k uživateli, která nemusí být jen vizuální (lze například využít vibrací ovládacího zařízení).

Pro přehlednost je níže uvedeno několik typických případů užití (tj. seznám kroků, který definuje interakci mezi uživatelem a systémem - aplikací):

Cave painting - chceme virtuálně malovat po stěnách CAVE [7]

1. označení místa, kde má tah započít
2. vedení tahu
3. ukončení tahu

Architektura - chceme chodit prostorem (městem, budovou, apod.) a manipulovat s objekty (otevírání/zavírání dveří, přesun objektu, přidání a umístění nového objektu, apod.)

1. pohyb prostorem na požadované místo (včetně otáčení se)
2. označení objektu se kterým chceme manipulovat (nebo přidání nového objektu)
3. provedení manipulace, tj. například:
   - posun
   - změna velikosti
   - rotace
   - otevření/zavření dveří
   - apod.
4. ukončení manipulace objektu (odznačení objektu)

Hry - chceme chodit virtuálním světem a moci provádět akce jako manipulace s objekty, střelba na objekty, aj.

1. pohyb prostorem (včetně otáčení se)
2. určení cíle
3. provedení manipulace, tj. například:
   - označení cílového objektu (pro manipulaci)
   - zamíření nějakým směrem (pro střelbu apod.)
4. provedení akce
   - manipulace (přesun, rotace, aj.)
   - střelba, hod, fouknutí, aj.
5. ukončení akce

Umění a design, vizuální simulace, GIS - chceme si pouze "prohlížet" objekt/prostor, tj. pohybovat se okolo objektu (skrze simulaci, prostor) nebo pohybovat objektem

1. pohyb prostorem (včetně otáčení se)
2. označení objektu
3. manipulace s objektem
    - posun
    - rotace

2.3.3 Standardní používané události

Z výše zmíněných případů užití vyplývá, že vždy potřebujeme nějakou standardní množinu událostí, jimiž bychom mohli virtuální realitu ovládat. Tyto události jsou:

- pohyb prostorem (vpřed, vzad, vlevo, vpravo, nahoru, dolů)
- otočení v prostoru (doleva, doprava, nahoru, dolů, "valení" doprava, "valení" doleva)
- výběr objektu (označení, odznačení, způsob jak na objekt "ukázat")
- pohyb objektem v prostoru
- otočení objektem v prostoru
- změna velikosti objektu
- speciální akce (vystřel, hoď, otevři, zavři, foukni, lítej, vlož/vymaž objekt, apod.)

2.4 Možnosti komunikace

2.4.1 Přenosové protokoly

Rodina TCP/IP obsahuje sadu protokolů pro komunikaci v počítačové síti. Architektura TCP/IP je rozdělena do čtyř vrstev a pro tuto práci nás zajímá vrstva transportní. Ta je implementována v koncových zařízeních a poskytuje služby pomocí protokolů TCP a UDP.

TCP [18]

Transmission Control Protocol vytváří virtuální okruh mezi koncovými aplikacemi, tedy garantuje spolehlivé doručování dat adresátovi beze ztráty a ve správném pořadí. Je to služba se spojením - má fázi navázání spojení, přenos dat a ukončení spojení. Poskytuje současný obousměrný přenos dat.

UDP [19]

User Datagram Protocol poskytuje "nespolehlivou"transportní službu - nemá fázi navázání a ukončení spojení a posílá rovnou data. Nezaručuje doručení všech datagramů, může se stát že některý nedorazí, dorazí víckrát nebo se změní pořadí doručení. Je bezestavový a oproti protokolu TCP je rychlejší právě z důvodu nekontrolování doručení.

2.4.2 Jazyky

Pro zápis informací, které chceme předávat při komunikaci, můžeme použít jakýkoliv serializovatelný jazyk. Níže je seznam několika nejpoužívanějších z nich a jejich specifikace:

XML [12]

Extensible Markup Language je velmi rozšířený obecný značkovací jazyk, který se používá pro serializaci dat a nese strukturované informace. Tyto informace obsahují jak obsah, tak indikaci, jakou roli tento obsah hraje. XML je tzv. lidsky srozumitelný přehledný jazyk, avšak díky vysoké strukturovanosti (na níž závisí jeho efektivita) je také poměrně náročný z hlediska rozsahu dat potřebných při jeho zápisu.

JSON [9]

JavaScript Object Notation je odlehčený formát pro výměnu dat, nezávislý na počítačové platformě. Data mohou být organizována v polích nebo agregována v objektech, které lze do sebe vnořovat. Vstupem je libovolná datová struktura (číslo, řetězec, boolean, objekt nebo z nich složené pole). Výstupem je vždy řetězec. JSON má jednoduchý způsob uložení dat, je lehce lidsky čitelný i zapisovatelný a velikost přenášených dat je menší než u XML.

BSON [1]

Binary JSON je datový formát pro binární serializaci datových struktur podobných formátu JSON. BSON stejně jako JSON podporuje vnoření objektů a polí. BSON obsahuje rozšíření o reprezentaci datových typů Date a BinData.

2.5 Podobné projekty

Obdobnou problematiku, jakou se zabývá tato práce, řeší i některé další projekty jako jsou:

2.5.1 VRPN

VRPN [14] je na zařízení nezávislý a síťově transparentní systém pro přístup k periferním vstupním zařízením v aplikacích virtuální reality. Poskytuje jednotná rozhranní pro vstupní zařízení několika daných typů (analog, button, dial, force device, sound, text, tracker). Tzn. všechna zařízení určitého typu pak vypadají a chovají se stejně, např. všechna zařízení typu tracker vypadají, že jsou typu vrpn_tracker. Pro každý typ existuje jeden nebo více serverů a klientská strana umožňující čtení hodnot ze zařízení a kontrolu jejich operací. Pokud jedno zařízení poskytuje více typů výstupních informací tak u něj dochází k exportu více rozhranní najednou a klientská strana s tím zachází jako s několika různými zařízeními jednotlivých typů.

2.5.2 trackd™

trackd™ [4] je malá tzv. daemon aplikace, která bere informace z množiny pohyb sledujících (motion-tracking) a jiných vstupních zařízení a činí tyto informace dostupné pro ostatní aplikace. Aplikace využívající trackd nepotřebují vědět, který typ vstupního zařízení je použit, protože získávají data ze zařízení skrze trackdAPI, které nezávisle na typu vstupního zařízení zajišťuje přístup k datům v nezměněném formátu.

Kapitola 3

Návrh řešení

Tato kapitola popisuje navrhovaný komunikační protokol pro systémy virtuální reality. Jsou zde informace o vstupních a výstupních událostech a způsobu jak je adaptovat. Dále následuje samotný popis protokolu a komunikace. Nakonec je popsána architektura systému, způsob konfigurace systému a struktura zpráv pro komunikaci.

3.1 Specifikace vstupních/výstupních událostí

Jak již bylo zmíněno v předchozích kapitolách, z různých vstupních ovládacích zařízení mohou přicházet rozličná (vstupní) data, stejně tak jako různé aplikace virtuální reality mohou pro své ovládání požadovat různé typy dat (výstupních). Proto bylo nutné stanovit základní množinu datových typů, které zahrnují všechny tyto možnosti, a určit způsob jak s nimi dále pracovat.

3.1.1 Definice typů událostí

Tento protokol uvažuje 8 datových typů pro vstupní a výstupní události:

- 6DOF
  - informace o 3D poloze v prostoru a o otočení kolem tří os (x, y, z)
  - jednotka: [m,m,m,°,°,°] - tzn. pro polohu metry, pro otočení stupně

- 3DOF
  - informace o 3D poloze v prostoru
  - jednotka: [m,m,m] - metry
- 3DOF_R
  - informace o otočení kolem tří os (x, y, z)
  - jednotka: [°,°,°] - stupně
- 2DOF
  - informace o 2D poloze
  - jednotka: [m,m] - metry
- 1DOF
  - informace o 1D poloze/vzdálenosti na jedné ose
  - jednotka: [m] - metry
- 1DOF_R
  - informace o otočení kolem jedné osy
  - jednotka: [°] - stupně
- range
  - hodnota z rozsahu <-1, 1>
  - jednotka: bezrozměrné
- boolean
  - hodnota pravda/nepravda
  - jednotka: bezrozměrné

3.1.2 Konverze událostí

Je dána množina výchozích konverzních funkcí, které lze používat při adaptaci dat. Tato množina je dále rozšiřitelná a způsob připojení vlastní funkce do této množiny je popsán v manuálu API softwarové knihovny implementující tento protokol. Při tvoření konfiguračního souboru (jenž je popsán dále v textu) je třeba používat právě funkce z této množiny.

Všechny konverzní funkce musí splňovat tento předpis K: $\{E^N\}$ -> E, kde K je konverze a E je událost. Neboli vždy musí být výsledkem konverze jedna událost, která může být složena z různého množství vstupních událostí.

Níže je seznam všech výchozích konverzních funkcí (přesnější definice funkcí a možnosti jejich vstupních parametrů jsou popsány v manuálu API softwarové knihovny):

- pass - pouze beze změny převede vstupní hodnotu na výstupní stejného typu
- make_1DOF - ze vstupních hodnot různých typů vytvoří událost typu 1DOF
- make_1DOF_x - ze vstupních hodnot vytvoří událost typu 1DOF ve směru osy x
- make_1DOF_y - ze vstupních hodnot vytvoří událost typu 1DOF ve směru osy y
- make_1DOF_z - ze vstupních hodnot vytvoří událost typu 1DOF ve směru osy z
- make_2DOF - ze vstupních hodnot různých typů vytvoří událost typu 2DOF
- make_3DOF - ze vstupních hodnot různých typů vytvoří událost typu 3DOF
- make_1DOF_R - ze vstupních hodnot různých typů vytvoří událost typu 1DOF_R
- make_3DOF_R - ze vstupních hodnot různých typů vytvoří událost typu 3DOF_R
- make_range - ze vstupních hodnot různých typů vytvoří událost typu range
- make_boolean - ze vstupních hodnot různých typů vytvoří událost typu boolean
- vector_size - ze vstupních hodnot (2DOF nebo 3DOF) vypočte velikost vektoru, kterou vrátí jako typ 1DOF
- add - vytvoří součet vstupních hodnot (všechny vstupní hodnoty musí být stejného typu)
- subtract - odečte od sebe vstupní hodnoty (vždy 2 vstupní hodnoty stejného typu)
- multiple - vynásobí vzájemně vstupní hodnoty
- sin - vytvoří sinus vstupní hodnoty - výstupem vždy hodnota typu range
- cos - vytvoří cosinus vstupní hodnoty - výstupem vždy hotnota typu range
- negation - vytvoří negaci dané vstupní hodnoty

3.2 Návrh protokolu pro komunikaci se systémy VR

Tento protokol částečně vychází z komunikačního protokolu navrženého Vladimírem Paločkem [2]. Jedná se o protokol pro komunikaci mezi ovládacími vstupními zařízeními a aplikacemi virtuální reality, přičemž komunikace by měla být možná při kombinaci jakéhokoliv zařízení a jakékoliv aplikace. Cílem tohoto protokolu je, aby co nejméně často docházelo k situaci, že ovládání nějaké aplikace za použití některého ze zařízení nebude možné. Na rozdíl od Paločkova protokolu, který řeší jen komunikaci směrem od vstupního zařízení k aplikaci, zvažuje tento protokol i možnost zpětné vazby z aplikace na zařízení (vibrace, LED diody, zvuk).

3.2.1 Součásti a jejich role

Protokol je založen na komunikaci dvou stran:

Vstupní (ovládací) zařízení

Vstupní zařízení (případně více zařízení najednou) je zařízení, pomocí nějž uživatel propaguje svou interakci s virtuální realitou (a v některých případech na něj může dostat zpětnou vazbu). Takových zařízení je velká škála a mohou být velmi odlišná (viz. 2.2 Dostupná zařízení). Každé zařízení má vždy množinu datových typů, které je schopné generovat a také může mít množinu datových typů, které je schopné přijmout.

Aplikace

Aplikace je jakákoliv aplikace virtuální reality, která pracuje s datovými vstupy ze vstupního zařízení. Má vždy množinu požadavků na události (a jejich datové typy), které chce přijímat a může odesílat zpětnou vazbu na vstupní zařízení.

Protože ne vždy pasuje množina datových typů generovaná (a přijímaná) vstupním zařízením přesně na množinu datových typů požadovaných (a odesílaných) aplikací, je třeba použít jakéhosi prostředníka, který adaptuje vstupní data zařízení na potřeby aplikace a naopak. Tento prostředník je ve skutečnosti rozdělen na dvě části - rozhraní aplikace a rozhraní vstupních zařízení.

Rozhraní vstupních zařízení

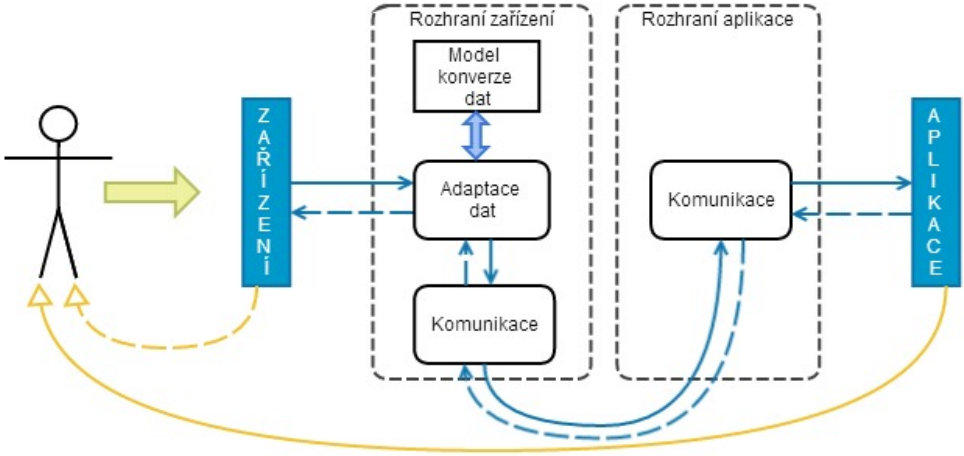
Rozhraní vstupních zařízení je implementováno daemonem zařízení, spravujícím připojená vstupní zařízení. Vytváří se zde model konverze dat ze vstupního zařízení na data požadovaná aplikací a také model konverze dat posílaných aplikací na data pro vstupní zařízení (pokud to aplikace požaduje a zařízení umožňuje). Rozhraní podle vytvořeného modelu konverze dat adaptuje příchozí data ze vstupních zařízení na data požadovaná aplikací a ty pak odesílá rozhraní aplikace. Případně přijímá data z rozhraní aplikace, která konvertuje a posílá do vstupního zařízení.

Rozhraní aplikace

Rozhraní aplikace je implementováno aplikací. Posílá na rozhraní vstupních zařízení požadavky aplikace a dostává od rozhraní vstupních zařízení adaptovaná data a předává je aplikaci. Případně dostává od aplikace data (zpětné vazby) a posílá je na rozhraní vstupních zařízení.

Schéma protokolu

Schéma (Obr. 3.1) znázorňuje proces ovládání VR aplikace. Zelená šipka značí uživatelovy akce, modré šipky výměnu dat a žluté šipky znázorňují zpětnou vazbu k uživateli. Čárkované šipky jsou pro případ možnosti zpětné vazby na vstupní zařízení.



Obrázek 3.1: Schéma komunikačního protokolu

3.2.2 Komunikace

Komunikace má daný konkrétní průběh, aby nejdříve došlo k potřebným nastavením, poté k posílání dat a nakonec i k ukončení komunikace. Nejprve se uskuteční fáze zvaná “Pozdrav”, kde dojde k výměně základních informací o potřebách aplikace. Na základě toho dojde k nastavení rozhraní zařízení podle potřeb aplikace a následuje fáze “Výměna dat”, kdy mezi zařízeními a aplikací proudí čistá, adaptovaná data. Nakonec dochází k “Ukončení spojení”, kdy dochází k přerušení přenosu dat a uzavření komunikace.

Pozdrav

Při pozdravu dochází k první komunikaci mezi rozhraním vstupních zařízení a rozhraním aplikace. Aplikace k rozhraní zařízení zašle požadavky na události (a jejich datový typ), které si přeje přijímat. Rozhraní zařízení tyto požadavky vyhodnotí a odpoví aplikaci, které z jejích požadavků může poskytnout.

Výměna dat

V této fázi dochází k výměně dat mezi vstupními zařízeními a aplikací. Data, která přicházejí ze vstupních ovládacích zařízení jsou konvertována na data požadovaná aplikací. Následně jsou odeslána na rozhraní aplikace a předána aplikaci. Data putující z aplikace jsou odeslána na rozhraní zařízení, kde jsou konvertována na data pro vstupní zařízení a předána daemonu zařízení.

Ukončení spojení

Ukončení spojení mezi vstupními zařízeními a aplikací ze strany aplikace probíhá tak, že aplikace vyšle požadavek o ukončení spojení na rozhraní vstupních zařízení. Poté dostane odpověď o ukončení spojení a dochází k odpojení aplikace. Ze strany vstupních zařízení probíhá ukončení spojení tak, že je aplikaci zaslána informace o ukončení spojení a to je následně ukončeno.

3.3 Architektura systému

Architektura systému, který využívá výše zmíněný protokol, je postavena na dvojici server-klient, mezi nimiž probíhá komunikace. Na straně klienta je aplikace (respektive aplikační rozhraní) a na straně serveru je tzv. daemon zařízení (respektive rozhraní vstupních zařízení), který extrahuje data příchozí ze vstupních ovládacích zařízení. Na serveru se také provádí adaptace těchto dat na data potřebná pro aplikaci. Adaptovaná data jsou poté posílána klientovi. Může dojít i k situaci, kdy přijde ze strany klienta (aplikace) požadavek na zpětnou vazbu pro vstupní zařízení (ta se může projevovat vibracemi zařízení apod.) - také v tomto případě se data na serveru adaptují a jsou daemonem předána zařízení. Požadované adaptace jsou blíže specifikovány pomocí konfiguračních souborů, jejichž popis je uveden v následující podkapitole.

Vlastnosti serveru

- dostává extrahovaná data od daemona zařízení
- ví, jaká data chce klient (aplikace)
- má přístup ke konfiguračnímu souboru, jímž jsou popsány adaptace dat
- na základě popsaných adaptací konvertuje data a odesílá je klientovi (pouze ta žádaná)
- případně přijímá data od klienta (zpětnou vazbu), ta adaptuje a předává na daemona zařízení

Vlastnosti klienta

- ví, jaká chce data (posílá na ně serveru požadavek)
- ví, jaká data mu server může poskytnout
- přijímá data ze serveru a dále je předává aplikaci
- v případě potřeby odesílá data na server (zpětná vazba)

3.3.1 Konfigurace

Konfigurační soubor se zpracovává na straně serveru a slouží ke specifikaci adaptace dat. Tj. pokud přijdou nějaká data z ovládacího zařízení, server ví, jak a jestli je má dále zpracovat. Konfigurace popisují proces adaptace od událostí přicházejících ze zařízení až po události výsledné. Určují, jaká konverzní funkce se má použít pro kterou adaptaci, jaké do této adaptace vstupují události a jaká událost z ní vystupuje.

3.3.1.1 Jazyk zápisu konfigurací

Konfigurace jsou zapsány v textovém souboru a skládají se z několika popisných prvků:

- typ dat (požadovaný typ výstupní události, např. 1DOF, 2DOF, range, aj.)
- název výstupní události
- název funkce, která má být použita pro konverzi dat
- seznam vstupních proměnných událostí, které mají být adaptovány
- volitelně alternativní hodnoty pro jednotlivé vstupní události - ty mají využití především ze začátku, než jsou potřebné proměnné iniciovány

Tyto jednotlivé prvky jsou proloženy speciálními znaky podle konkrétních pravidel zápisu (uvedeno níže), aby bylo možno popis adaptací na serveru vyhodnotit.

Lze použít dva způsoby zápisu:

1) Zápis celé adaptace

typDat názevUdálosti = názevFunkce($proměnná$_1$[alternativníHodnota], $proměnná$_2$, ..., $proměnná$_N$);

2) Definice proměnné (v případě, že do dané proměnné nepřiřazujeme žádnou vstupní událost z ovladače či jinou proměnnou, musíme takto definovat její typ)

typDat názevProměnné;

Jak je tedy vidět výše, pravidla zápisu jsou následující:

- mezi názvem výstupní události a názvem konverzní funkce musí být symbol “rovná se”
- seznam vstupních proměnných událostí dané adaptace musí být uzavřen mezi kulatými závorkami
- před název každé vstupní události (proměnné) vždy musíme napsat symbol dolaru
- alternativní hodnoty buď můžeme zcela vynechat, nebo je zapisujeme do hranatých závorek za danou proměnnou, která má být alternována
- zakončení zápisu každé adaptace či definice proměnné je určeno středníkem

Důležité je, že počet, typ a pořadí proměnných, které vstupují do funkce musí odpovídat použité konverzní funkci - může to být buď některá z výchozích konverzních funkcí, nebo některá vlastní definovaná.

Do konfiguračního souboru také můžeme vkládat komentáře, které nebudou systémem nijak vyhodnocovány. Pro vložení komentáře je třeba před tento komentář zapsat symbol procenta.

% toto je komentář

Řádek s komentářem nijak neukončujeme - text, který je zapsaný za symbolem "%"je až do konce daného řádku je považován za komentář. Celý konfigurační soubor musí být vždy ukončen symbolem zvaným hashtag.

3.3.1.2 Ukázka zápisu konfigurací

V rámci konfiguračního souboru můžeme popsat jednotlivé adaptace. Všechny vstupní proměnné události každé adaptace, musí být definovány buď jako výstupní událost jiné adaptace, nebo přímou definicí proměnné.

Příklad níže znázorňuje jednoduchou specifikaci jedné adaptace a jejích vstupních proměnných:

```
% zde tvoříme událost s názvem move2D, která je typu 2DOF a vstupují do ní 2 proměnné
% které mají definovány alternativní hodnoty, přičemž konverze bude provedena funkcí
% make_2DOF

2DOF move2D = make_2DOF($move2D_dx[0],$move2D_dy[0]);

% zde říkáme, že proměnná s názvem move2D_dx je typu 1DOF a přebírá hodnotu ze
% vstupního ovladače zarizeni1.tlacitko5 (pomocí funkce pass)

1DOF move2D_dx = pass($zarizeni1.tlacitko5);

% zde dochází pouze k deklaraci proměnné move2D_dy typu 1DOF, do které nebudou
% tímpádem přicházet žádné hodnoty a proto bude v adaptaci vždy nahrazována svou
% alternativní hodnotou

1DOF move2D_dy;

% označení konce konfiguračního souboru

#
```

Jedné proměnné můžeme přiřadit i více možných vstupů ze zařízení a tím umožníme vytvoření jedné stejné události různými vstupy z ovládacích zařízení. V takovém případě bychom do konfiguračního souboru zapsali více příkazů přiřazujících hodnotu ze vstupního ovladače dané proměnné:

```
% v tomto případě bude proměnné move2D_dx typu 1DOF přiřazena příchozí hodnota
% ze vstupního ovladače zarizeni1.tlacitko5 i ze vstupního ovladače zarizeni1.tlacitko6

1DOF move2D_dx = pass($zarizeni1.tlacitko5);

1DOF move2D_dx = pass($zarizeni1.tlacitko6);
```

3.3.2 Komunikace - struktura zpráv

Při komunikaci server-klient je možné použít jakýkoliv serializovatelný jazyk jako je XML, JSON, apod., jak již bylo uvedeno v analýze. Každý z těchto jazyků má své výhody i nevýhody. V tomto návrhu je použit jazyk JSON, protože je jednoduše lidsky srozumitelný a zároveň poměrně stručný svým zápisem, což je vhodné pro posílání komunikačním kanálem.

Během jednotlivých fází komunikace přenášíme tři druhy zpráv:

1) Hello message

Hello message je prvním krokem komunikace mezi serverem a klientem - je použita při fázi "Pozdrav". Tento typ zprávy přichází od klienta (aplikace) ihned po připojení na server a sděluje, jaké události aplikace požaduje zasílat. Odpovědí od serveru je zpráva stejného typu obsahující seznam událostí, které server klientovi (aplikaci) může poskytnout.

Zpráva typu Hello má formát:

```
{
    "_type":“hello”;
    "h_events": [
    [“název události1”,”typ události1”],
    [“název události2”,”typ události2”]
    ....
    ]
}
```

2) Event message

Event message je typ zprávy, která se používá při komunikaci v průběhu fáze "Výměna dat". Je to zpráva přenášející právě uskutečněnou událost. Tato zpráva může být posílána od serveru ke klientovi a v případě zpětné vazby může být zasílána i od klienta (aplikace) k serveru. V obou případech nese informaci o názvu události, jejím typu a její hodnotě.

Zpráva typu Event má formát:

```
{
    "_type":“event”;
    "e_name":”název události”
    "e_type":”typ události”
    "e_value": [hodnota1, hodnota 2, ..., hodnota n]
}
```

3) Goodbye message

Goodbye message je zpráva, která se používá ve fázi "Ukončení spojení"mezi klientem a serverem. Může být odeslána oběma stranami. V případě zaslání zprávy serverem na klienta je tímto klient informován, že bude odpojen. Pokud jde o odeslání zprávy klientem na server jedná se o žádost o odpojení - v takovém případě obdrží klient od serveru ještě potvrzení ve formě stejné zprávy.

Zpráva typu Goodbye má formát:

```
{
    "_type":"goodbye";
}
```

Kapitola 4

Realizace

Tato kapitola obsahuje popis implementace softwarové knihovny pro komunikaci se systémy virtuální reality. Popis je rozdělen do sekcí podle funkčních částí knihovny, kterýmiž jsou server, klient a komunikace. Na konci kapitoly je ukázka použití této knihovny.

4.1 Popis implementace

Všechny části knihovny jsou napsány v jazyce C++, konkrétně ve verzi C++11. V implementaci je použito objektově orientované programování, což umožňuje daleko přehlednější tvorbu kódu a jeho rozdělení do celků podle funkčnosti. Implementovaná knihovna je použitelná pod systémy Linux a Windows.

Knihovna slouží jako rozhraní pro aplikace virtuální reality a připojení ovládacích zařízení. Tvoří vpodstatě prostředníka mezi vstupy ze zařízení a aplikací a umožňuje tak zpracování vstupních dat na abstraktnější rovině. Může být zahrnuta do jakékoliv aplikace C/C++, která požaduje ovládání ze vzdálených zařízení.

Celá knihovna je uzavřena ve jmenném prostoru s názvem "adaptations". Hlavní částí knihovny je serverová strana, která poskytuje rozhraní pro implementaci daemona abstrahujícího data ze vstupních zařízení. Serverová strana zpracovává veškerá daemonem abstrahovaná vstupní data ze zařízení i události přicházející od aplikace a také řeší komunikaci s klientem. Klientská strana poskytuje rozhraní pro aplikace a slouží pouze k příjmu a odesílání událostí, tedy ke komunikaci se serverem.

4.1.1 Server

Server je základní jednotkou celé knihovny. Kromě toho, že komunikuje s klientskou stranou, poskytuje především aplikační rozhraní pro daemona zařízení a obsahuje všechny třídy sloužící ke zpracování dat abstrahovaných ze vstupních zařízení (i dat příchozích od aplikace jako zpětná vazba).

Nejdůležitější třídy využívané na straně serveru jsou:

server (API)

Rozhraní serverové strany, třída "server", nabízí množinu funkcí pro práci s příchozími daty ze zařízení a komunikaci s klientskou stranou. Funkce "run" a "stop" umožňují správu TCP připojení (spuštění a zastavení TCP serveru). Funkce "addFeedbackEvent" zajišťuje přidání definice události do seznamu zpětných vazeb, tedy událostí, které přijímají vstupní zařízení. Rozhraní serverové strany umožňuje také přidat odkaz na vlastní konverzní funkci do takto rozšiřitelné množiny konverzních funkcí a to použitím funkce "addMyFunction". Samozřejmostí je funkce "sendEvent" sloužící pro předání události abstrahované daemonem k dalšímu zpracování.

Rozhraní také obsahuje abstraktní metodu zvanou "gotEvent", která je zavolána vždy při příchodu události z klientské strany. Tuto metodu si daemon zařízení doimplementuje podle svých potřeb.

tcpServer

Třída "tcpServer" slouží k vytvoření a obsluze síťového připojení pomocí protokolu TCP. Probíhá zde komunikace s klientem a zpracovávájí se zde zprávy, které od klienta přicházejí. K jejich zpracování se využívají metody z třídy "jsonFunc".

parser

Třída "parser" slouží ke zpracování informací z konfiguračního souboru. Při inicializaci serverové strany parser načítá konfigurační soubor a vytváří podle něj model adaptací dat, který se poté využívá pro zpracování vstupních dat ze zařízení i aplikace.

adaptFunc

Třída "adaptFunc" obsahuje výchozí množinu konverzních funkcí používaných pro adaptace dat. Při zpracovávání událostí jsou volány funkce z této třídy.

jsonFunc

Třída “jsonFunc” slouží k serializaci a deserializaci zpráv posílaných mezi klientem a serverem.

4.1.2 Klient

Klientská strana poskytuje aplikační rozhraní pro aplikace virtuální reality. Také zaštiťuje síťovou komunikaci se serverem.

Nejdůležitější třídy využívané na straně klienta jsou:

client (API)

Rozhraní klientské strany, třída “client”, poskytuje správu komunikace se serverovou stranou - vytvoření a ukončení síťového připojení funkcemi “run” a “stop”. Funkce “addReqEvent” umožňuje přidání definice události do seznamu požadavků na příchozí události. Odeslání tohoto seznamu požadavků na server zajišťuje funkce “sendHello”. Rozhraní obsahuje i funkci “sendEvent” pro odeslání události zpětné vazby na server.

Rozhraní také obsahuje abstraktní metodu "gotEvent", která se volá při příchodu události ze strany serveru a kterou si koncová aplikace doimplementuje podle svých potřeb.

tcpClient

Třída “tcpClient” slouží k vytvoření a obsluze síťového připojení pomocí protokolu TCP. Probíhá zde komunikace se serverem a jsou zde zpracovávány zprávy, které od serveru přicházejí. K jejich zpracování jsou využity metody z třídy “jsonFunc”.

4.1.3 Komunikace

Při implementaci komunikační složky této knihovny ve třídách “tcpServer” a “tcpClient” byly použity dvě externí knihovny. Přímo pro práci se síťovou komunikací je zapojena knihovna PracticalSocket - metody této knihovny jsou použity pro vytváření síťového připojení a následující komunikaci mezi serverem a klientem. Další použitou knihovnou je knihovna SuperEasyJSON, která poskytuje metody pro práci s JSON a umožňuje jeho serializaci a deserializaci. JSON je při komunikaci používán jako nosič přeposílaných dat mezi serverem a klientem.

4.2 Ukázka použití knihovny

Zde je ukázka použití knihovny na straně serveru i klienta spolu se stručným návodem.

Na straně serveru i klienta je k použití knihovny potřeba zahrnout odpovídající hlavičkový soubor - pro server je to soubor "server.h"a pro klienta je to soubor "client.h". Dále je v obou případech třeba vytvořit novou třídu dědící od odpovídajícího rodiče (tj. adaptations::server nebo adaptations::client) jak je vidět v ukázkách níže. Nezbytné je implementovat požadovaným způsobem virtuální metodu "gotEvent", která se volá při přijetí nové události. Dalším krokem je vytvoření objektu nově vytvořené třídy a jeho spuštění funkcí "run" v novém vlákně.

Dále můžeme nastavit na straně serveru přijímané zpětné vazby a na straně klienta požadované události. Poté zahájíme fázi "Pozdrav" odesláním zprávy Hello message z klientské strany, která předá serveru seznam požadovaných událostí. Po provedení fáze "Pozdrav" již lze odesílat uskutečněné události.

4.2.1 Použití na straně serveru

```
include "server.h"

class serverAPP : public adaptations::server {
public:
    serverAPP(){}

    //implementace virtualni metody volane pri prichodu udalosti od klienta
    virtual void gotEvent(adaptations::pdataType e) override {
         std::cout << "New event from client!" << e << std::endl;
    }
};

int main(int /* argc */, char ** /* argv */) {
    //vytvorime instanci serveru
    serverAPP server;

    //spustime server (TCP komunikaci) v novem vlakne
    std::thread t([&](){server.run();});

    //muzeme nastavit, jake chceme prijimat zpetne vazby
    server.addFeedbackEvent("vibrate",adaptations::event_types_t::event_1dof);
    server.addFeedbackEvent("blink",adaptations::event_types_t::event_bool);
```

```
    //ukazka simulace udalosti ze zarizeni - stejnym zpusobem muzeme odeslat
    //udalost extrahovanou daemonem zarizeni
    server.sendEvent(adaptations::makeEvent_1DOF("zariz1.ovlad4", 0.5f));

    t.join();
}
```

4.2.2 Použití na straně klienta

```
#include "client.h"

class clientAPP: public adaptations::client {
public:
    clientAPP(const std::string& address):client(address) {}

    //implementace virtualni metody volane pri prichodu udalosti od serveru
    virtual void gotEvent(adaptations::pdataType e) override {
          std::cout << "New event from server!" << e << std::endl;
    }
};

int main(int /* argc */, char ** /* argv */) {
    //vytvorime instanci klienta - parametr je adresa serveru
    clientAPP client("127.0.0.1");

    //spustime klienta (TCP komunikaci) v novem vlakne
    std::thread t([&](){client.run();});

    //nastavime, jake chceme prijimat udalosti
    client.addReqEvent("move2D", adaptations::event_types_t::event_2dof);
    client.addReqEvent("move3D", adaptations::event_types_t::event_3dof);
    client.addReqEvent("move_fw", adaptations::event_types_t::event_1dof);

    //posleme Hello message, ktera preda serveru pozadavek na zadane udalosti
    client.sendHello();

    //muzeme poslat na server zpetnou vazbu
    client.sendEvent(adaptations::makeEvent_1DOF("zpetna_vazba",12.0f));

    t.join();
}
```

Kapitola 5

Testování

5.1 Popis testování

Cílem testování bylo ověřit dvě věci - funkčnost implementované softwarové knihovny a vliv protokolu na velikost latence při komunikaci. Celé testování probíhalo na platformě Linux, ačkoliv implementovaná knihovna je použitelná i pod systémem Windows. Pro účely testování byla použita jednoduchá testovací aplikace a daemon abstrahující vstupní zařízení.

5.1.1 Testování funkčnosti

V první části testování jsme zkoušeli funkčnost se třemi různými vstupními zařízeními:

- Microsoft Natural® Ergonomic Keyboard 4000
- Logitech USB Optical Mouse
- Gamepad Elecom

Použitá konfigurace pro tento test byla:

```
1DOF move_fw = make_1DOF($up[0], $down[0], $fw_param[1]);
1DOF move_fw = make_1DOF($fw_dof[3.14], $up_axis);
boolean up = pass($zariz1.button289);
boolean up = pass($zariz1.button418);
```

```
boolean down = pass($zariz1.button290);

boolean down = pass($zariz1.button419);

range up_axis = pass($zariz1.axis1);

1DOF fw_dof;

1DOF fw_param;

1DOF_R rotate = make_1DOF_R($rotate_axis);

1DOF_R rotate = make_1DOF_R($zariz1.button158[0], $zariz1.button159[0], $rot_step[10]);

range rotate_axis = pass($zariz1.axis5);

1DOF_R rot_step;

boolean reset = pass($zariz1.button293);

boolean reset = pass($zariz1.button172);

#
```

Cílová aplikace požadovala zasílání 3 událostí:

- move_fw - pro pohyb vpřed/vzad
- rotate - pro rotaci
- reset - pro reset aplikace

Jak vyplývá z výše uvedné konfigurace, vstupy pro události move_fw a rotate byly změneny pomocí definovaných konverzí. Pro všechna zařízení proběhl test úspěšně.

Také bylo vyzkoušeno zaslání události z aplikace do zařízení. Konfigurace této události je následující:

```
1DOF vibrate = make_1DOF($vibrate_fwd[0], $vibrate_bck[0], $vibrate_param[3.6]);

%killed je událost odeslaná aplikací jako zpětná vazba

boolean vibrate_fwd = pass($killed);
```

```
boolean vibrate_bck;
1DOF vibrate_param;
#
```

Přijímaná zpětná vazba na serveru pro zařízení byla událost vibrate - v tomto případě také došlo ke změně typu dat mezi událostí odeslanou aplikací a událostí přijímanou. Předání zpětné vazby z aplikace proběhlo úspěšně.

5.1.2 Měření latence

Testování za účelem změření latence také probíhalo pod systémem Linux a to jak při spuštění serveru i klienta na localhostu, tak při spuštění klienta na jiné lokaci a běhu přes síť.

Bylo otestováno poslání události ze serveru na klienta a zpět - klient obdržel událost a automaticky odeslal zpětnou vazbu. Byl měřen čas (výsledek je v tabulce 5.1), který uběhne mezi abstrahováním vstupní události ze zařízení a přijetím zpětné vazby serverem. (Mezi tím dochází ke zpracování vstupní události ze zařízení, odeslání zpracované události na klienta, odeslání zpětné vazby z klienta na server, přijetí a zpracování této zpětné vazby.)

Takto byly otestovány dvě adaptace - jedna triviální, kde dochází pouze k předání proměnné funkcí pass a jedna komplexnější, kde dochází k více vnořeným manipulacím se vstupními proměnnými - událost move_ping:

```
3DOF move_ping = make_3DOF($move_fw, $fw_dof, $fw_dof);
1DOF move_fw = make_1DOF($up[0], $down[0], $fw_param[1]);
1DOF move_fw = make_1DOF($fw_dof[3.14], $up_axis);
boolean up = pass($zariz1.button289);
boolean up = pass($zariz1.button418);
boolean down = pass($zariz1.button290);
boolean down = pass($zariz1.button419);
range up_axis = pass($zariz1.axis1);
```

```
1DOF fw_dof;
1DOF fw_param;
#
```

5.2 Výsledky testování

Testováním jsme ověřili, že knihovna funguje správně - všechny příchozí události ze všech tří vyzkoušených vstupních zařízení způsobily očekávanou reakci. Stejně tak proběhla podle předpokladu i zpětná vazba od aplikace.

Měření času proběhlo pro triviální i komplexní adaptaci a při komunikaci na localhostu i při komunikaci po síti. Výsledné zprůměrované naměřené hodnoty latence jsou následující:

	triviální, localhost	komplexní, localhost	triviální, po síti	komplexní, po síti
latence	541 μs	573 μs	579 μs	549 μs

Tabulka 5.1: Naměřené latence

Kapitola 6

Závěr

Na základě rešerše jsem specifkovala typy vstupních událostí, které se používají v aplikacích virtuální reality. Navrhla jsem protokol pro obousměrnou komunikaci aplikace virtuální reality s daemonem abstrahujícím vstupní zařízení. Následně jsem navrhla rozšiřitelný systém adaptací mezi různými typy vstupních událostí a implementovala jsem softwarovou knihovnu, která využívá navržený protokol a adaptace. Knihovnu jsem realizovala v jazyce C++ a funguje jak pod systémem Linux tak pod systémem Windows. Dále jsem implementovala demonstrační aplikaci využívající tuto knihovnu pro obsluhu vstupů.

Ve své diplomové práci jsem otestovala komunikaci mezi daemonem a testovací aplikací, k čemuž jsem použila 3 různá vstupní zařízení. Při testu aplikace vyžadovala vstupy 3 různých typů a u 2 z nich byl změněn typ pomocí definovaných konverzí. Otestovala jsem také odeslání události z aplikace do zařízení - u této události byl také změněn typ pomocí konverzí.

Změřila jsem latenci posílání událostí pomocí daného protokolu. Výsledné časy jsou velmi nízké - mezi odesláním události a přijetím odpovědi uběhlo průměrně cca 600μs. To ukazuje, že vliv protokolu na velikost latence v komunikaci je minimální.

Tato práce přinesla možnost zlepšení komunikace mezi vstupními zařízeními a aplikacemi virtuální reality. Výsledkem je, že komunikace využívající navržený protokol může probíhat na daleko abstraktnější úrovni, než tomu je při získávání neadaptovaných dat přímo z daemona.

Protože v mém řešení je potřeba adaptace událostí specifikovat manuálně pomocí konfiguračního souboru, jako možnost pokračování této práce se nabízí začlenění určité automatiky do tohoto procesu.

Literatura

[1] *BSON* [online]. Dostupné z: <http://bsonspec.org/>.

[2] BC. PALOčKO, V. Generalized interactive techniquesfor Collaborative VR System. diploma thesis, České vysoké učení technické v Praze, Praha, 2011. <https://dip.felk.cvut.cz/browse/pdfcache/palocvla_2011dipl.pdf>.

[3] CMU, G. *Virtual reality and interaction* [online]. Dostupné z: <http://graphics.cs.cmu.edu/nsp/course/15-462/Spring04/slides/23_vr.pdf>.

[4] CORPORATION, M. *trackd* [online]. Dostupné z: <http://www.mechdyne.com/trackd.aspx>.

[5] CVMT. Interaction for VR. web. <http://www.cvmt.dk/education/teaching/e07/cvg9/VR/Interaction_for_VR.pdf>.

[6] DAMBROT, S. M. *The doctor will see you now* [online]. Dostupné z: <http://www.columbia.edu/cu/21stC/issue-1.4/doctor.html>.

[7] Daniel F. Keefe Daniel Acevedo Feliz Tomer Moscovich David H. Laidlaw Joseph J. LaViola Jr. CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. Technical report, Brown University Providence. <http://vis.cs.brown.edu/docs/pdf/Keefe-2001-CPF.pdf>.

[8] GREGORY BARATOFF, S. B. *Tracking Devices* [online]. Dostupné z: <http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html>.

[9] JSON group. *Introducing JSON* [online]. Dostupné z: <http://www.json.org/>.

[10] KJELDSKOV, J. Interaction in virtual reality. Technical report, CVMT. <http://www.powershow.com/view1/17b166-ZDc1Z/Interaction_in_Virtual_Reality_Jesper_Kjeldskov_CVMT_powerpoint_ppt_presentation>.

[11] LATOSCHIK, M. E. *Realtime 3D Computer Graphics: Virtual Reality Input Devices* [online]. 2005. Dostupné z: <http://www.techfak.uni-bielefeld.de/ags/wbski/lehre/digiSA/WS0607/3DVRCG/Vorlesung/3a.RT3DCGVR-Input-Devices.pdf>.

[12] Norman Walsh. *A Technical Introduction to XML* [online]. 2008. Dostupné z: <http://www.xml.com/pub/a/98/10/guide0.html>.

[13] PROF. JUNG, B. Virtuelle Realität: Interaction in VR - Manipulation. Technical report, TU Freiberg, 2006. <http://www.informatik.tu-freiberg.de/lehre/pflicht/VR/

ws06/VR12_Manipulation.pdf>.

[14] Russell M. Taylor II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, Aron T. Helser. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. Technical report, University of North Carolina at Chapel Hill, 2001. <http://www.cs.unc.edu/Research/vrpn/>.

[15] Scott K. Isabelle, Robert H. Gilkey, Robert V. Kenyon, George Valentino, John M. Flach, Curtis H. Spenny, and Timothy R. Anderson. Defense applications of the CAVE. Technical report. <http://www.cs.uic.edu/~kenyon/Conferences/GILKY/CAVE_DOD.html>.

[16] UPC, L. *VIRTUAL REALITY: Input devices. Technologies for the direct interaction.* [online]. 2012. Dostupné z: <http://www.lsi.upc.edu>.

[17] VISBOX. *Immersive 3D Applications* [online]. Dostupné z: <http://www.visbox.com/boxApps.html>.

[18] Wikipedia contributors. *Transmission Control Protocol* [online]. 2014. Dostupné z: <http://en.wikipedia.org/wiki/Transmission_Control_Protocol>.

[19] Wikipedia contributors. *User Datagram Protocol* [online]. 2014. Dostupné z: <http://en.wikipedia.org/wiki/User_Datagram_Protocol>.

Příloha A

Seznam použitých zkratek

3D trojdimenzionální

API Application Programming Interface

BSON Binary JSON

D-PAD Directional pad

DOF Degree of freedom

HMD Head-mounted display

JSON JavaScript Object Notation

L/P levý/pravý

LED Light-Emitting Diode

POV Point-Of-View

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

VR virtuální realita

VRPN Virtual Reality Peripheral Network

XML Extensible Markup Language

Příloha B

Obsah přiloženého CD

- text
  - partyjul_2014dipl.pdf
  - partyjul_2014dipl.zip (LaTex source)
- src
  - doc
    - LICENSE.txt
    - README.txt
    - UserGuide.pdf
    - ProgrammerGuide.pdf
    - doxygen_output
  - lib
    - adaptLib.lib
  - zdrojové soubory