VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

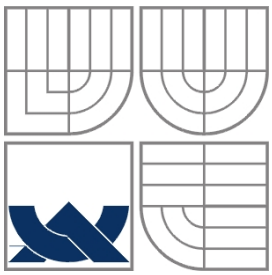
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WIIMOTE JAKO VSTUPNÍ ZAŘÍZENÍ

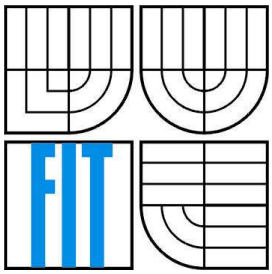
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ONDŘEJ VRANA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WIIMOTE JAKO VSTUPNÍ ZAŘÍZENÍ

WIIMOTE AS INPUT DEVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ VRANA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JOZEF MLÍCH

BRNO 2009

Abstrakt

Tato práce se zabývá možnostmi využití Wii remote k ovládání počítače místo klasických vstupních zařízení. Klade důraz na využití gest rukou jako novou alternativu zadávání příkazů počítači. Zabývá se algoritmem „dynamické borcení času“ na rozpoznávání signálů a technik k sloužících k mezi-procesové komunikaci DCOP a D-BUS. Součástí práce je návrh a implementace aplikace s využitím WiiMote.

Abstract

This thesis is concerned with possibilities of PC control by Wii remote controller compared to classical input devices. It focus on utilization of hand gestures as an alternative way to set orders to PC. Algorithm “ Dynamic time warping” for signals recognition and inter-process communication technics DCOP and D-BUS are presented. Other part of the thesis is dedicated to proposal and implementation of application using WiiMote.

Klíčová slova

WiiMote, dynamické borcení času, rozpoznávání gest, mezi-procesová komunikace, DCOP, D-BUS, X-Windows sytém.

Keywords

WiiMote, Dynamic time warping, motion analysis, inter-process communication, DCOP, D-BUS, X-Windows system.

Citace

Vrana Ondřej: WiiMote jako vstupní zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2009

WiiMote jako vstupní zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jozefa Mlícha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Vrana
18.května 2009

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce Ing. Jozefu Mlíchovi za vedení mé práce a cenné rady. Dále bych chtěl poděkovat panu Ing. Aleši Lánikovi za pomoc při natáčení demonstračního videa.

Obsah

Seznam obrázků

1 Úvod

Uživatelské rozhraní a způsob interakce uživatele s počítačem v dnešní době prochází velkým vývojem. Po dlouhou dobu od nástupu personálních počítačů na přelomu 80. a 90. let byly možnosti komunikace velmi omezené na použití klávesnice a myši, popřípadě na zařízení určená pro specifické aplikace jako jsou joystick pro hraní her a tablety pro CAD programy. Rostoucí výkon a hardwarová výbava personálních počítačů umožňují použití sofistikovanějších a přirozenějších způsobů komunikace s uživatelem, kterými jsou například ovládání pomocí hlasu a gest.

Cílem této práce bylo vytvořit rozhraní pro komunikaci s personálním počítačem pomocí herního ovladače ke konzole Nintendo Wii. K této komunikaci bylo využito jak samotných tlačítek ovladače, tak gest rukou provedených s ovladačem. Popsáno v 6. kapitole.

Tento herní ovladač, dále označovaný jako WiiMote, je velmi důmyslným zařízením, které obsahuje infračervenou kameru a akcelerometry. Konkrétní vlastnosti WiiMote jsou popisovány ve 3. kapitole. Pro rozpoznávání gest rukou jsou využity akcelerometry, které zaznamenávají informace o přetížení vzhledem k x, y a z ose. Takto získaná data jsou analyzována a porovnána pomocí algoritmu dynamického borcení času. Součástí této kapitoly jsou i způsoby komunikace a získávání dat z periferních zařízeni.

Algoritmus dynamického borcení času je určen především pro rozpoznávání řeči, lze ho však použít pro porovnání specifických průběhů přetížení při pohybu ovladače – tedy pro rozpoznávání gest rukou. Tímto algoritmem se podrobně zabývá kapitola pátá.

Komunikací s operačním systémem a emulace vstupu z klávesnice se zabývá 2. kapitola. Zde je podrobně popsána struktura X-Window systému a komunikace pomocí knihovny Xlib.

V další kapitole se věnuji komunikaci mezi procesy, která je zprostředkovaná DCOP a D-BUS technikami. Zde jsou rozebrány jejich principy a adresace procesů.

Hlavním cílem práce byl návrh a implementace programu, který využívá výše zmíněný algoritmus a komunikační techniky. Problematika programování aplikace a vyhodnocení výsledků jsou obsahem šesté a sedmé kapitoly.

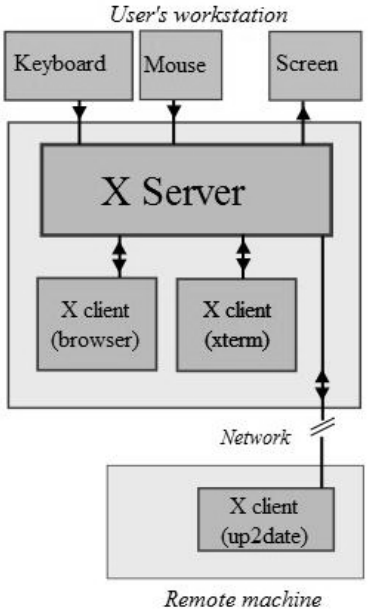
2 X-Window system

Systém X Window (též X11 nebo jen X) je síťově transparentní okenní systém. Díky X, mohou různé aplikace běžet současně v okně, generovat textový a grafický výstup na obrazovku. X, poskytuje prvky pro generování mnoha stylů textu a dvourozměrné grafiky (jako jsou body, linie či polygony) do hierarchické struktury obdélníkových oken. Každé okno přitom může být „virtuální plocha“ a sama pak obsahovat další podřízená okna s libovolnou hloubkou. Okna pak jsou překrývána jako stoh papírů na stole.

X systém byl vyvíjen jako součást projektu Athena v Laboratory for Computer Science na MIT v roce 1983. Následně pak byly v rychlém sledu vyvinuty následující verze od X1 až po X11. Komunikační protokol X sytému již zůstává neměnný od verze X11 Release 1 v roce 1987. Různé úpravy a dodatky se dělají pomocí rozšíření (extensions).

2.1 Klient – Server Model

X je založeno na modelu klient – server (viz obrázek 2.1). Server běží na počítači, ke kterému je připojen grafický výstup, klávesnice, myš a komunikuje s různými klientskými programy. Server zprostředkovává komunikaci mezi uživatelem a klientským programem, přijímá požadavky na grafický výstup od klientského programu a zobrazí je uživateli. Přijímá uživatelovi požadavky(myš, klávesnice) a přeposílá je klientskému programu. V X Windows systému běží vždy server na uživatelově počítači, zatímco klientský program může běžet na jiném, a to i s jiným operačním systémem.



Obrázek 2.1: Model X-Window systému (převzato [9])

2.2 Správce oken (Windows Manager)

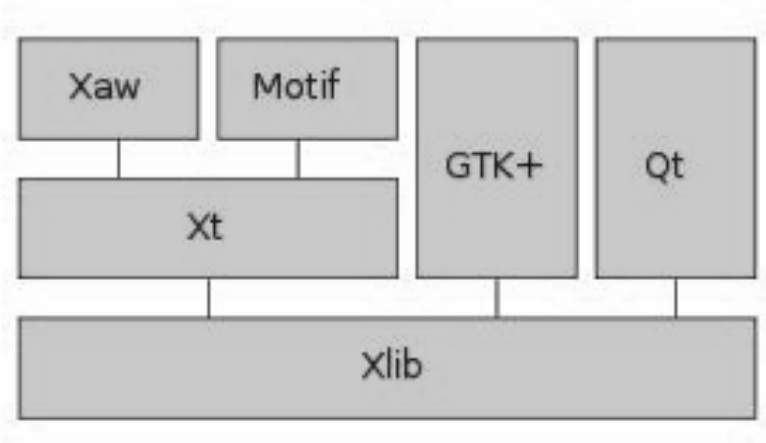
Správce oken je důležitou komponentou v X Window systému. Stará se o správu top-level oken aplikací. Zajišťuje změnu velikosti okna a pozici na ploše. Stará se o zanořování oken vzájemně pod sebou, vykreslování okrajů a titulků okna. Kromě základních dovedností nabízí uživateli podporu v grafickém prostředí. Jedná se o různé menu pro spuštění aplikací, čí panelu pro přidělení fokusu oknu.

Správce oken je samostatná komponenta. Architektura X systému je navržena totiž tak, aby se dosáhlo spolupráce více komponent, a nebylo vše implementováno v jednom celku. Je tady možnost změnit správce oken, a docílit jiného grafického prostředí.

2.3 Xlib

Jedná se o knihovnu, která zapouzdřuje X Window protokol a zprostředkovává komunikaci klientských programů s X serverem. Je napsaná jazyce v C a poskytuje jen základní (low-level) příkazy - základní kameny pro tvorbu grafického prostředí. Implementuje příkazy pro vykreslení základních grafických objektů jako bod, linie, čtverec, kružnici.

Proto vznikly nástavby (toolkity), pro tvorbu uživatelského rozhraní. Tyto nástavby poskytují již stavební prvky jako je okno, roletové menu čí tlačítko. Daný prvek pak stejně rozloží na příkazy knihovních funkcí Xlibu. Následující obrázek zobrazuje strukturu grafických toolkitů (viz obrázek 2.2). Použití přímo Xlib knihovny bez použití nástavby, je značně nevhodné. Může se s výhodou použit při tvorbě speciálních programů (čtečka pro slepce,odchytávání zpráv).



Obrázek 2.2: Struktura komunikace s X-Window sytémem (převzato [8])

3 WiiMote

Na konci roku 2006 Nintendo vydalo pátou domácí herní konzolí Nintendo Wii. Předchozími verzemi, jako byla Gamecube, na trhu selhaly v konkurenci s výkonnějšími herními konzolemi od firem Microsoft a Sony. I nová verze nenaznačovala výrazný růst výkonu, aby se mohla srovnávat s konkurencí. Avšak o rok později se stala jednou z nejprodávanější s 20 miliony prodaných kusů po celém světě. Její úspěch spočíval v inovační technologii a interaktivní komunikaci s herní konzolí, který umožňoval nový herní ovladač Wii remote (viz. obrázek 3.1). V rámci této kapitoly jsou informace a ilustrační obrázky čerpány z [3].

3.1 Popis

Herní ovladač Nintendo Wii remote, zkráceně WiiMote, je ruční zařízení podobající se televiznímu ovladači. Kromě tlačítek obsahuje navíc akcelerometry ve 3 osách, IR kameru s vysokým rozlišením, reproduktor, vibrační motor a bezdrátové Bluetooth spojení. Využití těchto technologií ve WiiMote z něj dělá důmyslné a jedinečné zařízení.

Obrázek 3.1 Wii remote zařízení (převzato [2])

Infračervená kamera – tracker

Součástí každého WiiMote je infračervený senzor od firmy Pikart Imaging. Součástí senzoru je více-objektový sledovací nástroj, který poskytuje vysoké rozlišení a rychlost sledování až čtyř světelných objektů současně. Parametry senzoru nebyly publikovány, ale odhaduje se rozlišení 1024×768 při vzorkovací frekvenci 100 Hz a zorném úhlu 45° na horizontální ose. Objektový

„sledovač“ integrovaný v hardwaru minimalizuje množství dat posílané bezdrátovým spojením a umožňuje implementovat v aplikaci jednoduché sledování objektů založené na IR kameře.

Vibrační motor

Malý vibrační motor slouží jako hmatová zpětná vazba. Je podobný tomu jaký se používal v mobilních telefonech. Motor umožňuje pouze dva stavy (zapnuto/vypnuto). Dalo by se ale použít šířkové modulace signálu ke změně intenzity vibrace.

Akcelometry

Lineární Akcelometr ADXL330, od firmy Analog Devices, pracuje ve 3 osách s rozsahem až +/-3 g. Produkuje datový výstup na 8b pro jednu osu při vzorkovací frekvenci 100 Hz.

Bluetooth

Bezdrátové Bluetooth spojení zajišťuje Broadcom 2042 chip. Využití tohoto standardu umožňuje téměř 100% komptabilitu s ostatními Bluetooth zařízeními a připojení k běžnému PC.

Vytváření komunikační vrstvy pro Bluetooth umožňuje na Linuxu BlueZ [19], která je přímo součástí jádra a značně využívána. Tato knihovna implementuje flexibilní a efektivní protokol komunikace na nízké vrstvě. Umožňuje vytvořit soket pro připojení periferních zařízení, zjistit jeho jméno, popřípadě načítat a zapisovat data do registrů zařízení přes Bluetooth. O významu dat které příjmá nebo odesílá ale nemá informace, protože jsou ve specifickém formátu daného zařízení (WiiMote).

Komunikace na této vrstvě je velmi obtížná. Je nutné mít údaje o struktuře řídících registrů a významu konkrétní bitů, popřípadě formátu dat která může přijímat například reproduktor. Tyto informace, týkající se WiiMote, nebyli publikovány a byly proto získány pomocí revizního inženýrství [20]. Zatím se nepodařilo zjistit význam všech registrů WiiMote.

Pro práci se zařízením lze využít API[1], která umožňují komunikaci na vyšší vrstvě. Na Linuxu je nejčastěji využívá knihovna cwiid [18], popřípadě na Windows knihovna WiiYourSelf [21]. Tyto knihovny již poskytují kolekci funkcí pro jednoduchý přístup ke všem komponentám WiiMote.

[1] *Application programming interface*

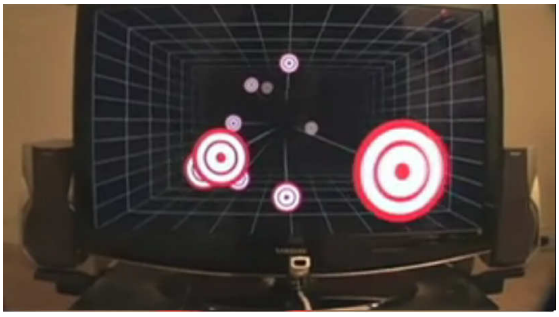
3.2 Uživatelské rozhrání s WiiMote

Pro představu jaké jsou možnosti využití WiiMote nejen pro hraní her a to především z hlediska uživatelského rozhraní bude popisováno v této podkapitole. Aplikace zmíněné v této kapitole, nejsou nikterak přelomové. Co je však na nich obdivuhodné, že jsou až obdivuhodně jednoduché a přístupné po cenové stránce, neboť vyjma WiiMotu se náklady na pořízení IR pera či brýlí pohybují kolem 10$. WiiMote tady otvírá nový směr, který byl dlouho uzavřen pro praktické využití.

3.2.1 Sledování hlavy
(Head Tracking for Desktop VR Display)

Pro lepší pochopení principu této aplikace je třeba si obrazovku představit jako okno do jiného pokoje. Můžeme měnit pozici pozorování před oknem a uvidíme předměty z jiného úhlu či vzdálenosti. V této aplikaci se u WiiMote využívá infračervené kamery a speciálních brýlí, které jsou opatřeny dvěmi infračervenými diodami po stranách (viz. obrázek 3.2). WiiMote, umístěný pod obrazovkou (viz obrázek 3.3), zjistí relativní polohu pohledu (pozorovatele) vůči obrazovce a upraví se scéna s polohou kamery. Výsledný vjem pro pozorovatele dává reálnější pohled na 3D scény.

Obrázek 3.2: Brýle s IR diody



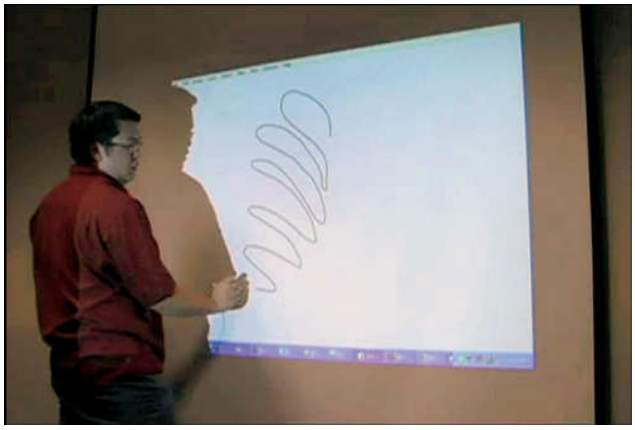
Obrázek 3.3: Pohled na virtuální 3D scénu

3.2.2 Více-dotykový displej

(Low-Cost Multi-point Interactive Whiteboards)

Přeložený celý název zní „Nízkonákladová vícedotyková interaktivní plocha“. Podobně jako předchozí aplikace (3.2.1) využívá infračervenou kameru a jako zdroj infračerveného světla využívá pero (viz. obrázek 3.5) s IR diodou. Použití je stejné jako u dotykových obrazovek. Místo stylusu použijeme pero s IR diodou. Pokud budeme chtít psát , zmáčkneme navíc na peru tlačítko, které nám označí daný bod na ploše. IR kamerou se sejme bod, interpretuje stejně jako na dotykovém monitoru (viz obrázek 3.4).

Před použitím aplikace se musí nejdřív provést kalibrace rohů displeje (poloha bodů na IR kameře), aby se získaly potřebné informace pro transformaci bodů s IR kamery do rozlišení obrazovky.



Obrázek 3.4: Dotyková plocha ovládána IR perem

Obrázek 3.5: Pero s IR diodou

4 Mezi-procesová komunikace

V této kapitole se bude zabývat protokoly a nástroji pro komunikaci mezi procesy (IPC - Inter-Process Communication) a vzdáleným voláním procedur (RPC - Remote Procedure Calling).

4.1 DCOP - Desktop COmmunications Protocol

DCOP je jednoduchý IPC/RPC komunikační mechanizmus postavený na schránkách(soketech) a to buď unixových nebo TCP/IP. Byl využit protokol ICE (Inter Client Exchange), který mimo to je standardní součástí X Window systému od verze X11R6. Je závislý na knihovnách Qt, je výpočetně nenáročný a určený výhradně pro aplikace KDE. V této kapitole bylo čerpáno z [1] a [11].

Model DCOP komunikace je jednoduchý. Každá aplikace používající DCOP se stává klientem. Ke komunikaci používají prostředníka, DCOP server, který se stará o správné adresování a doručení zprávy.

DCOP umožňuje dva druhy posílání zpráv. Zprávy typu „pošli a zapomeň“ (tedy neblokující) a druhé opačné, které zablokují program a čeká se, že se vrátí nějaká data. Posilaná data se vždy serializují pomocí operátoru QDataStream, dostupné v každé Qt třídě. Ve skutečnosti je to tak jednoduché, že není problém napsat kód pro seřazení ručně. Také jsou k dispozici IDL[2] kompiléry (dcopidl a dcopidl2cpp), které vygenerují kostru za Vás.

Identifikace objektu se provádí pomocí tří základních parametrů: identifikátor aplikace, vzdálený objekt a název funkce.

Pro práci s komunikační mechanizmem můžeme použít programy i mimo prostředí KDE.

kdcop - KDE DCOP prohlížeč

Jedná se o jednoduchou grafickou aplikaci, která zobrazuje všechny DCOP klienty spuštěné na daném PC ve stromové struktuře, včetně jejich rozhraní, které podporují. Umožňuje i posílání zpráv klientům.

dcop –Jedná se o obdobnou aplikaci s konzolovým rozhraním (vhodné pro skripty).

Rozhraní aplikace:

dcop [options] [application [function [arg1] [arg2] ...]]

[2] Intermediate density lipoprotein

4.2 D-BUS

Stejně jako DCOP i D-BUS je komunikační mechanizmus založený na principu IPC/RPC, vyvinut primárně pro Linux, aby nahradil konkurenční IPC řešení s jednotným protokolem. Byl navrhnut tak, aby umožňoval komunikaci jak mezi uživatelskými aplikacemi, tak s procesy na systémové úrovni. Komunikace většinou probíhá přes centrální prvek server, zvaný „bus“(sběrnice), ale je podporována i přímá interakce mezi aplikacemi. Nakonec byla importována i na operační systém MS Windows. V této kapitole bylo čerpáno z [13] a [15].

D-BUS je tvořena dvěmi sběrnicemi: system bus a session bus. Session bus je určena pro komunikaci v rámci desktopového sezení jednoho uživatele. System bus je určen pro aplikace v rámci desktopového sezení s operačním systémem, tedy kernelem. D-BUS byl navržen tak, aby měl co nejmenší latenci, režijní náklady a snadno se používal.

Pro komunikaci aplikací D-BUS nabízí dvě možnosti. První možností je obdobná jako u DCOP a jedná se a klasické volání procedur(RPC) s možností blokování nebo možností pošli a zapomeň. Druhá možnost nabízí zaregistrovat u objektu signál (událost). Princip je opačný jak u funkcí. Neboť při registrování signálu nastavíme i proceduru, která se zavolá při jejím výskytu v klientském programu.

K adresování se využívá tří prvků:

- název služby
- název objektu
- název rozhraní

Název služby - jako jednoznačný identifikátor se používá internetová adresa organizace, která definovala rozhraní, ale je zapsána pozpátku. Například služby D-BUS jsou definovány od „`freedesktop.org`“ a tak název služby zní „`org.freedesktop.DBus`“.

Název objektu (Object paths) - pro danou službu tvoří objekty hierarchickou strukturu. Proto se název objektu zadává jako cesta k němu (např. „ `/pub/something/“).`

Název rozhraní – obvykle bývá stejný jako název služby, avšak místo teček se píší lomítka. Tento identifikátor totiž definuje seznam metod a signálů, které aplikace exportuje.

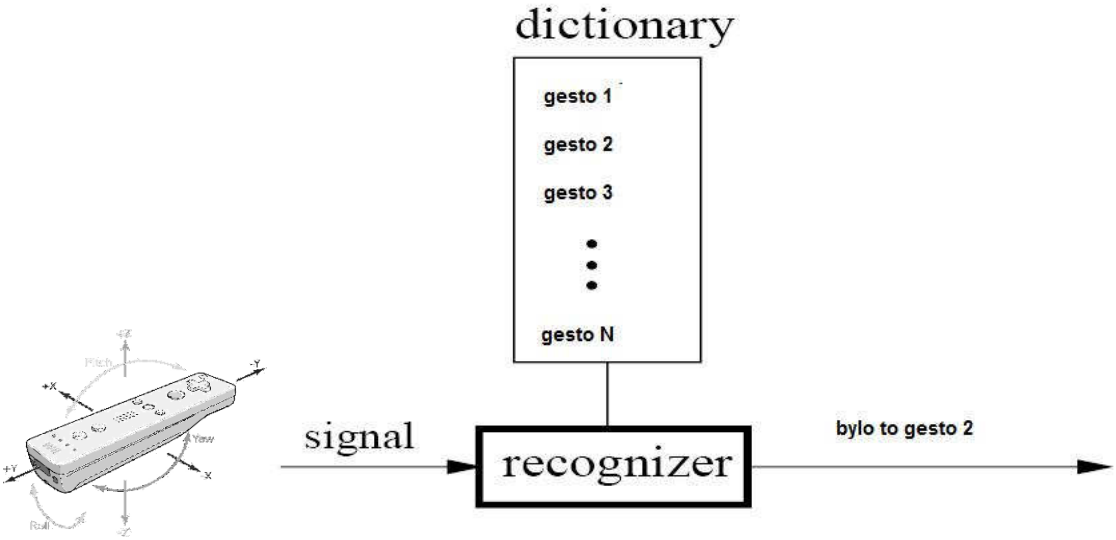
Pro práci s D-BUS sběrnicí můžeme použít programy:

dbus-monitor – jednoduchý konzolový program, pro monitorování zpráv na příslušné sběrnici ,rozhraním či objektu.

qdbusview – Jedná se o jednoduchou grafickou aplikaci obdobnou kdcop pro zobrazení dostupných služeb přes D-BUS.

5 Rozpoznávání signálů

Kapitole se zabývá metodami a algoritmy pro rozpoznávání řeči. Je to proto, že jejich princip rozpoznávání můžeme použít i s jinými daty (signály), než je řeč. Snahou bude z dat získaných z WiiMote akcelometrů rozpoznat, jaké se udělalo gesto. V této kapitole jsem použil obrázky a čerpal informace převážně z přednášek pana Doc. Dr. Ing. Černockého z předmětu ZRE [12].



Obrázek 5.1: Princip rozpoznávání gest

Ve slovníku budeme mít uloženy gesta neboli referenční matice R .. R_N, které budeme chtít rozpoznávat. Na vstup rozpoznávače nám přijde testovací matice s navzorkovanýma hodnotami O. Cílem bude určit, ke které referenční matici patří test. Otázkou ale zůstává, jak porovnávat matice?

5.1 Lineární srovnání

Lineární srovnání je definována vzorcem 5.1. Sčítá vzdálenost jednotlivých vektorů přes celou matici [12]. Matice tedy musí mít stejný počet vzorků. Problém ale nastává, že lidé nikdy neřeknou tutéž řeč se stejným časováním, tak i gesta budou od sebe odlišná. Proto je tento algoritmus nevhodný.

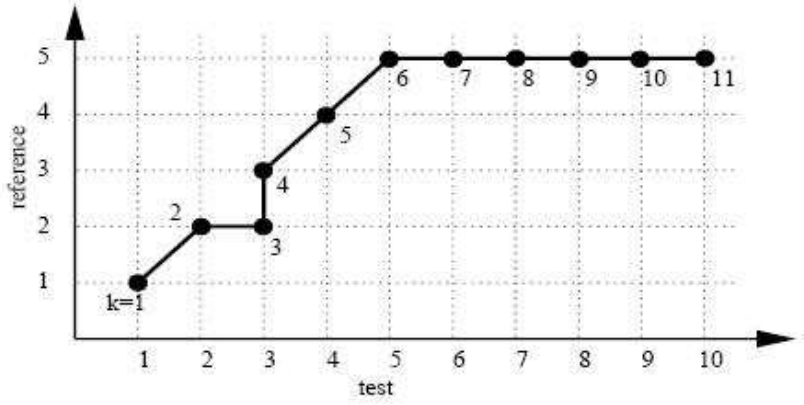
$$D(O,R) = \sum_{i=1}^{N} d\left[o(i), r(i)\right] \tag{5.1}$$

Testovací matice ***O*** je porovnávána s referenční maticí ***R***. Délka obou matic je stejné velikosti ***N***. Postupně porovnáváme vektory reference ***r(i)*** a testu ***o(i)*** se stejným indexem ***i***. Výsledek porovnávací funkce ***d*** sčítáme. Celkovým součtem dostaneme výslednou vzdálenost ***D*** matic ***O*** a ***R***.

5.2 DTW – dynamické borcení času

Dynamické borcení času je algoritmus pro porovnávání dvou podobných sekvencí, které se mohou rozcházet v čase nebo rychlosti. Obecně se dá říci, že metoda se snaží najít nejlepší způsob, jak porovnat jednotlivé vektory reference a testu. Výsledkem je pak stejně jako u lineárního porovnání, akumulovaná vzdálenost.

Při porovnání matic tímto algoritmem nám vznikne jako vedlejší výsledek „optimální cesta“, kterou lze vykreslit grafem (viz obrázek 5.2). Podle optimální cesty pak můžeme zjistit, které jednotlivé vektory se porovnávaly. Také hodně vypovídá o podobnosti samotných signálů. Podle sklonu křivky se dá vyčíst časové posunutí čí rychlejší průběh jednoho ze signálu.

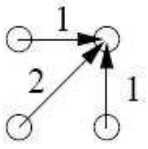


Obrázek 5.2: DTW - Graf optimální cesty

Na ose X jsou vyneseny vektory testovacího signálu, zatímco na ose Y vektory referenčního signálu. Průnik pak značí největší shodu.

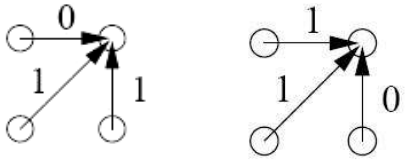
Dále je u DTW algoritmu potřeba definovat váhové funkce. Ty nám totiž předepisují, jakým způsobem lokálně porovnávat vektory a udělit jim penalizaci při posuvu na cestě. Šipky značí posuv na další vektor (směr po ose podle grafu 5.2) a hodnota u ní penalizaci.

1) Symetrická



Obrázek 5.3: Symetrická váhovací funkce

2) Asymetrická



Obrázek 5.4: Asymetrická váhovací funkce

Váhovacích funkcí existuje velké množství i těch, které nezkoumají jen nejbližší okolí a zabývají se vzdálenějšími vektory. Ovšem nejpoužívanější funkcí je symetrická, popřípadě

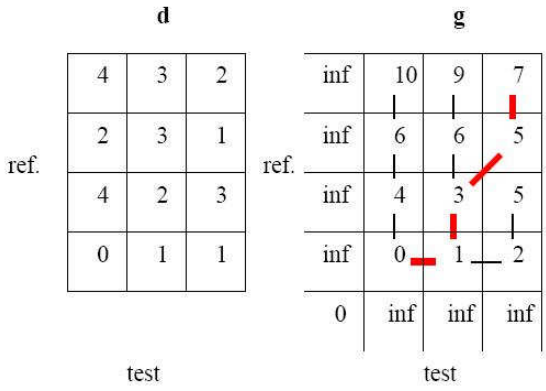
asymetrická. U symetrické je typické, že penalizuje nejvíc cestu, která je nejrychlejší. Načež u asymetrických není vůbec penalizován posuv v jedné ose.

Postup při porovnávání algoritmem DTW je následující.

- Vytvoří se matici **d** o rozměrech reference × testu (viz obrázek 5.5) a zapíši se do ní vzdálenosti testovacích a referenčních vektorů porovnaných každý s každým.
- Vytvoří se matici **g** pro výpočet akumulované vzdálenosti větší však o jeden sloupec a řádek než matice **d**. Vyplní se celý první řádek a první sloupe na hodnotu ∞, kromě počátku ([0,0]). Ten se inicializuje na hodnotu 0.
- Pak se postupně prochází matice **d** a počítá se akumulovaná vzdálenost (viz rovnice 5.2).

$$g(m,n) = \forall \min_{predchudce} [g(predchudce) + d(m.n) * w(k)] \qquad (5.2)$$

Váhovací funkce se značí ***w(k)***. Akumulovaná vzdálenost v předchozím bodě cesty je ***g(předchůdce)***. Vzdálenost dvou porovnávaných vektorů ***m*** a ***n*** je ***d(m,n)***.Tato vzdálenost je uložena v matici ***d***. Výsledek rovnice ***g(m,n)*** je akumulovaná vzdálenost od začátku cesty až do průsečíků vektorů ***m*** a ***n***.



Obrázek 5.5: Porovnání matic pomocí DTW algoritmu

V matici ***d*** jsou uloženy vzdáleností vektorů porovnaných každý s každým. V matici ***g*** jsou zapsány akumulované vzdáleností s využitím symetrické váhovací funkce. Červené čáry značí optimální cestu.

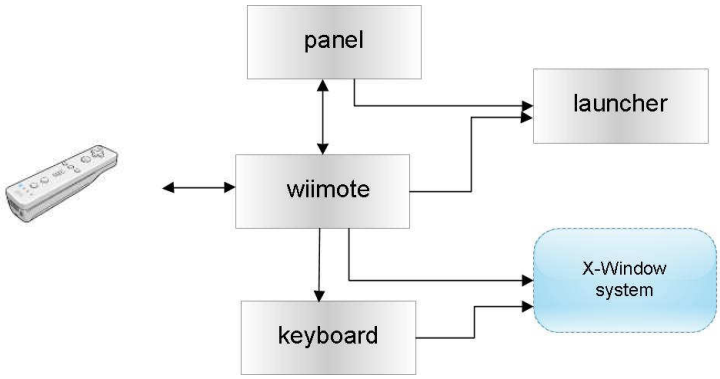
6 Návrh a implementace aplikace

Cílem je naprogramovat aplikaci, která by umožnila vzdálené ovládání počítače pomocí Wii remote zařízení. Využívala by nejen tlačítka, ale i akcelerometry součástí WiiMote. Pomocí jich by totiž rozeznala pohyb ruky a mohla by se využít k dalšímu způsobu ovládání PC. Aplikace by měla na základě spuštěného programu, interpretovat události WiiMote na zprávy pro program.

Součásti aplikace by měla být virtuální klávesnice, která by umožňovala psát text jen s WiiMote a modul pro snadné spouštění jiných programů. Aplikace by měla být bez problému přeložitelná na operačním systému Linux a běžet jako démon služba připravena na asynchronní požadavky od uživatele.

Celý systém je rozdělen do čtyř programů (viz obrázek 6.1) a to: wiimote, panel, keyboard, launcher. Každý z nich vykonává určitou specifickou funkci popsanou níže.

6.1 Schéma programů

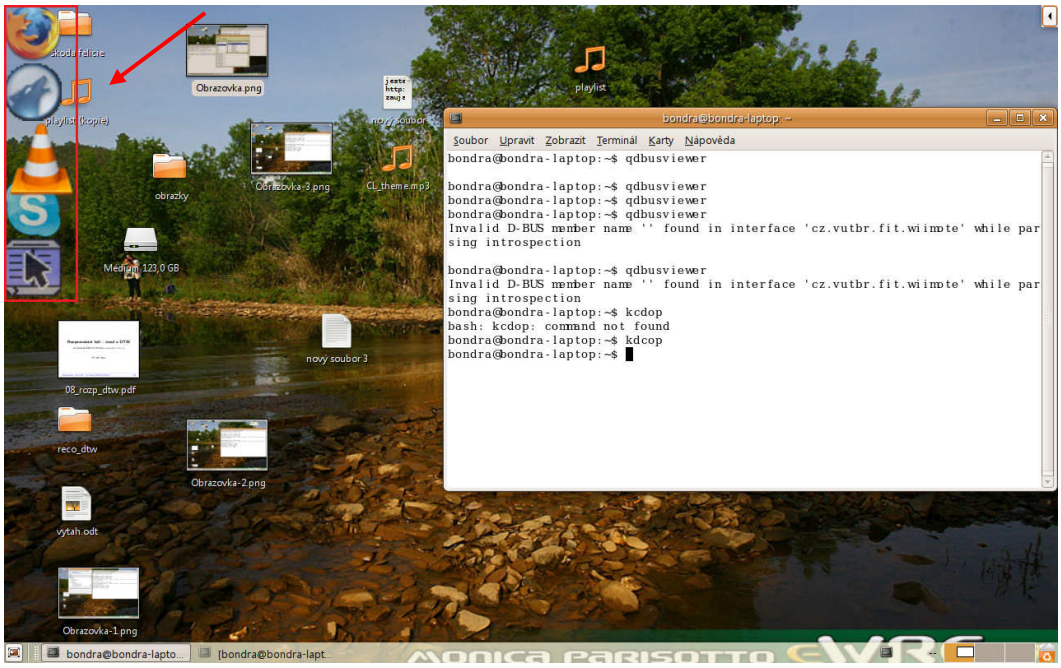


Obrázek 6.1: Schéma komunikace mezi programy

Na tomto schématu (6.1) je zobrazena komunikace mezi jednotlivými prvky celého systému. Programy mezi sebou komunikují přes sběrnici D-BUS. Příloze E najdeme všechny adresy objektů a popis funkcí, které programy exportují. Se systémem X-Windows komunikují přes knihovnu Xlib. Příloze C najdeme podrobnější schéma propojení jednotlivých programů.

6.2 Panel

Tento program vytváří na levé straně plochy panel s ikonami programů. Obsahuje dvě záložky, kdy první umožňuje spouštění programů. Druhá záložka obsahuje seznam programů, z které je možné si vybrat program k ovládání přes WiiMote (viz obrázek 6.2).



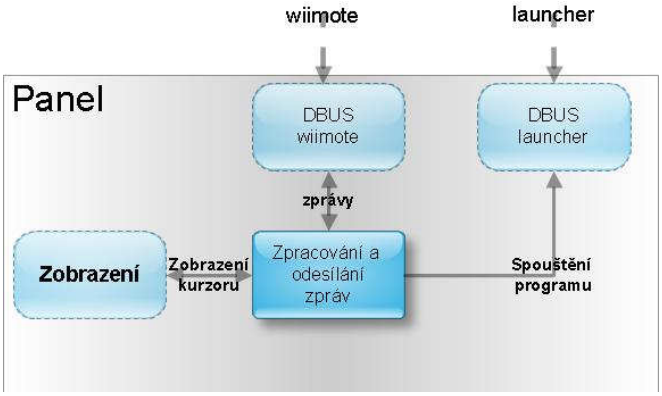
Obrázek 6.2: Program panel zobrazen na ploše

Přepínání mezi záložkami se děje pomocí tlačítek „**1“** a „**2“** na WiiMote. Panel je standardně schovaný a zobrazuje se či skrývá tlačítkem „**Home**“. Protože je určený na ovládání z dálky, lze zvětšit nebo zmenšit rozměr ikon na panelu tlačítky „+“ , respektive „-“. K výběru aplikací pak se používají tlačítka „**up**“, „**down**“ a pro potvrzení „**A**“. Popis jednotlivých tlačítek najdeme v příloze B Popis WiiMote.

Program panel je naprogramován v jazyce C++ s využitím knihoven Qt. S ostatními programy wiimote a launcher komunikuje pomocí mezi-procesové sběrnice D-Bus. S prvním jmenovaným komunikuje oboustranně, kdy od něj dostává události o Wii remote. Nazpět mu pak odesílá zprávy o vybraném programu, který chce uživatel ovládat. Druhému programu odesílá požadavky na spuštění programů.

Panel přijímá jeden nepovinný parametr a to “*ManyColor*“. Tento parametr určuje, jakým způsobem se budou alokovat barvy (velikost barevné hloubky) a vykreslovat ikony. Při použití parametru bude pozadí okna panelu průhledné. V opačné situaci bude vyplněno bílou barvou. To ale není jediná podmínka pro funkčnost průhlednosti. Ve správci oken (na Gnome) musí být nastaveny

grafické efekty minimálně na střední hodnotu. Jinak bude průhlednost u panelu X-Window systémem ignorována. Toto nastavení aplikace nelze po spuštění měnit.



Obrázek 6.3: Schéma programu panel

Na tomto schématu (6.3) je rozkreslen program panel do bloků (tříd) a znázorněna jejich vzájemná komunikace. Po spuštění programu se nejdříve načtou data z XML souborů a inicializují všechny bloky. Následně se předá řízení programu Qt pro zpracovávání zpráv.

Blok Zobrazení je implementován pomocí tříd ***TIcon.*** Tato třída tvoří základní grafický prvek panelu a každá zobrazená ikona na panelu tvoří jednu instanci této třídy. ***TIcon*** dědí z *QGraphicsItem* a zapouzdřuje v sobě animaci a chování ikon na reakce kurzoru. Dále exportuje rozhraní přes něž informuje jak ostatní ikony tak ***TPanel*** o změně kurzoru ikony jako reakci na událost kliknutí myší.

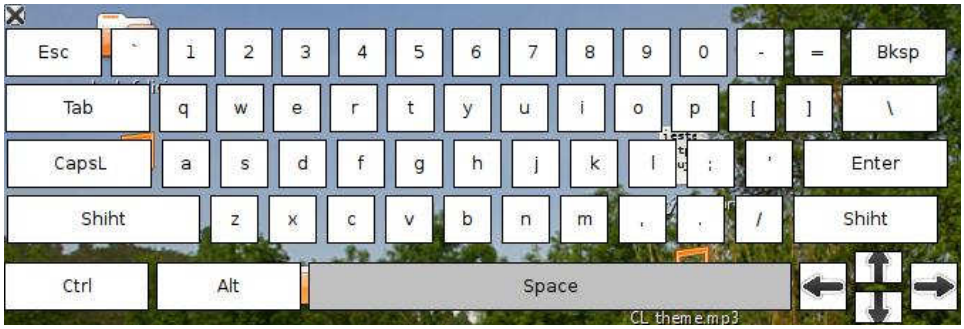
O mezi-procesovou komunikaci na sběrnici D-BUS se starají dvě třídy. První třída ***TDbus*** dědící z QDBusAbstractAdaptor, vytváří na sběrnici D-BUS objekt *„/panel"* a zprostředkovává předávání zpráv rodičovské třídě (***TPanel***) a naopak. Touto třídou komunikuje s programem wiimote.

Druhá třída ***TDataDbus*** je implementována i v ostatních programech a zprostředkovává připojení k zadanému objektu na sběrnici D-BUS. Umožňuje mu následně posílání zpráv, či zaregistrování požadavku na příjem signálů. V tomto případě zasílá zprávy programu launcher na spuštění zadaného programu.

Poslední modulem je třída ***TPanel***. Tato třída dědí z QWidget a vytváří okno pro zobrazení panelu. Je přizpůsobena tak, že okno nepřijímá editační fokus a pokud je zobrazeno, je vždy vykresleno nad všemi ostatními okny na ploše. Dále zpracovává zprávy z D-BUS a podle požadavků řídí kurzor ikon a předává informace o kurzoru třídám ***TIcon***, jenž pak tvoří animaci.

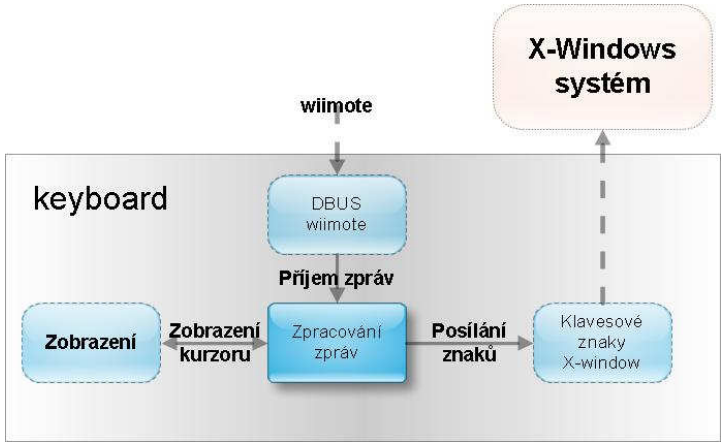
6.3 Keyboard

Jedná se o jednoduchý program napsaný v jazyce C++ za podpory knihoven Qt a Xtst. Program vytváří na ploše virtuální klávesnici (viz. obrázek 6.4), kterou můžeme ovládat jak myší tak přes sběrnici D-BUS. Aplikace vytvoří okno jenž je bez editačního fokusu a může tak zasílat X-Window systému zprávy o stisknutí klávesy. Události tak budou směrovány jinému oknu (požadovanému), než které je na popředí.



Obrázek 6.4: Vzhled programu keyboard

Aplikace má jeden nepovinný parametr *"ManyColor"*, který má stejnou funkci jako u předchozího programu (6.2**)**. K dálkovému ovládání je přístupný kurzor, pomocí něj vybíráme jednotlivá písmena. Jinak výběrem myši. Popis D-BUS rozhraní tohoto programu je uveden příloze E. Lze si nadefinovat způsob ovládání kurzoru podle vlastního mínění.



Obrázek 6.5: Schéma programu keyboard

Program je implementován podle tohoto schématu (6.5) a je tedy tvořen čtyřmi třídami. ***LabelItem*** třída je základní grafický prvek virtuální klávesnice, kdy každá klávesa tvoří jednu instanci této třídy. Stará se o správné zobrazení popisku a kurzoru klávesy.

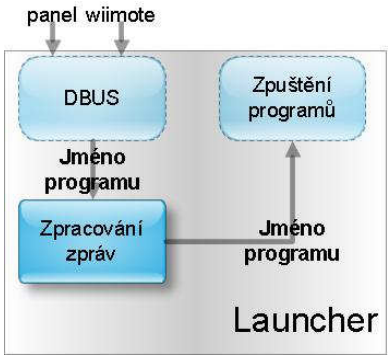
Další třídou je ***WinKeyboard***. Tato třída vytváří okno bez editačního fokusu pro zobrazení klávesnice. Je implementována tak, aby zobrazená klávesnice byla vždy na popředí. Definuje obsluhu zpráv z D-BUS sběrnice, tedy pohyb kurzoru a změny velikosti klávesnice. Třída dědí z QWidget.

Velmi důležitou je následující třída ***TKeyboardEvent.*** Tato třída obstarává spojení s X-Window systémem. Komunikuje s ním prostřednictvím Xlib a Xtst knihovny. Xtst je knihovna, která rozšiřuje množinu funkcí Xlib pro zasílání znaků stisknutých kláves. Navíc si zaznamenává stisknutí modifikačních kláves (shift, ctrl, alt), aby je po ukončení programu vrátila do polohy „nestisknuto“. X-Window by je po ukončení spojení sám nevrátil do původní polohy a psaní na hardwarové klávesnici by značně zkomplikoval.

Poslední třídou je ***TDbus*** pro mezi-procesovou komunikaci. O této třídě byla již zmínka v předchozím programu (6.2). Vytváří však jiný objekt na D-BUS sběrnici jménem */keyboard*. Více o rozhraní tohoto programu na D-BUS je v příloze E.

6.4 Launcher

Jedná se o jednoduchý konzolový program (démona) napsaný v jazyce C++ a knihoven Qt. Aplikace naslouchá na D-BUS sběrnici a podle požadavků spouští nové procesy. Důvodem proč je samostatnou komponentou a není implementován v potřebných aplikacích je více. Hlavní problém nastával u programu wiimote (6.5), neboť při vytváření nového procesu přerušoval spojení s Wii remote .



Obrázek 6.6: Schéma programu launcher

Jedná se o jednoduchý program, který je tvořen dvěma třídami. Vytváření objektu */Launcher* na sběrnici D-BUS zajišťuje střída ***TDbus***. Jedná se o stejnou třídu jako v předchozích programech (6.2).

Druhou třídou je ***TLauncher***, která zpracovává zprávy z D-Bus a zároveň implementuje samotné spouštění nových procesů pomocí funkcí fork a execvp (popřípadě system) . Tento program je závislý na implementaci těchto funkcí na daném operačním systému (nemusí být součástí na všech distribucích Linuxu) .

6.5 Wiimote

Wiimote je program napsaný v jazyce C++ s využitím knihoven Qt, cwiid a Xtst. Program se stará o připojení k WiiMote, učení gest, zadávání příkazů a jejich samotné vykonání. Po spuštění aplikace vytvoří ikonu v oblasti zvané system tray (nebo notification area). Přes ní pak uživatel komunikuje a umožňuje spouštět následující akce.

První možností je učení gest, které probíhá velice intuitivně. Na WiiMote se stiskne tlačítko „**A**“ a po celou bodu jeho držení se zaznamenává signál (gesto). Délka nahrávání je omezena na 1500 vzorků. To při vzorkovací frekvenci přibližně 80 Hz dává až 20 vteřin. Po nahrání gesta lze ihned otestovat jeho správnost. Pokud se gesto rozpozná, zobrazí se o tom informace u ikony programu.

Obrázek 6.7: Záznam nového gesta

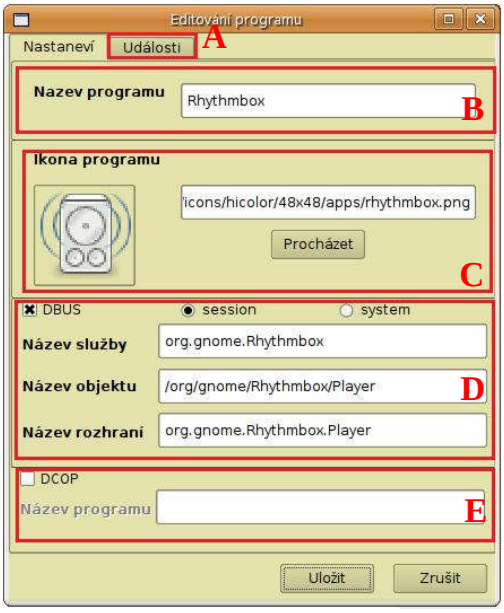
A - Editační pole Název. Důležité jej vyplnit, neboť se následně používá k adresování gest.

B - Editační pole Popis. Souží k podrobnějšímu popisu gesta a následné snadnější rekonstrukci gesta.

C – V tomto místě aplikace informuje o své činnosti.

D - Panel pro zobrazování délky gesta. Respektive kolik schází k dosažení maximální délky.

Další možností je nastavení prostředí pro konkrétní program, který chceme ovládat. Toto nastavení je trochu komplikovanější, pokud má být komunikováno s programem přes DCOP či D-BUS. K tomu je zapotřebí buď použít dokumentaci k programu nebo využít jeden programů qdbusview, či kdcop.



A – Záložka se seznamem příkazů

B – Název programu

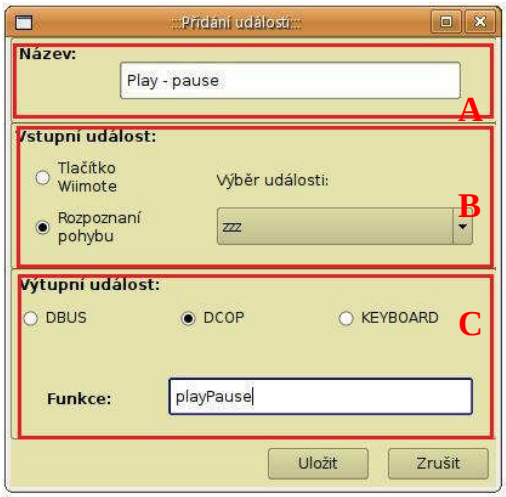
C – Cesta k ikoně, která se následně zobrazuje i na panelu.

D – Nastavení adresy objektu na sběrnici D-BUS.

E – Nastavení adresy DCOP objektu.

Obrázek 6.8: Přidávání nového programu k ovládání

Dalším krokem po vytvoření programu je následné přidaní příkazů, podle kterých se budou interpretovat stisknutí tlačítek, či rozpoznání gest na příkazy pro programy.



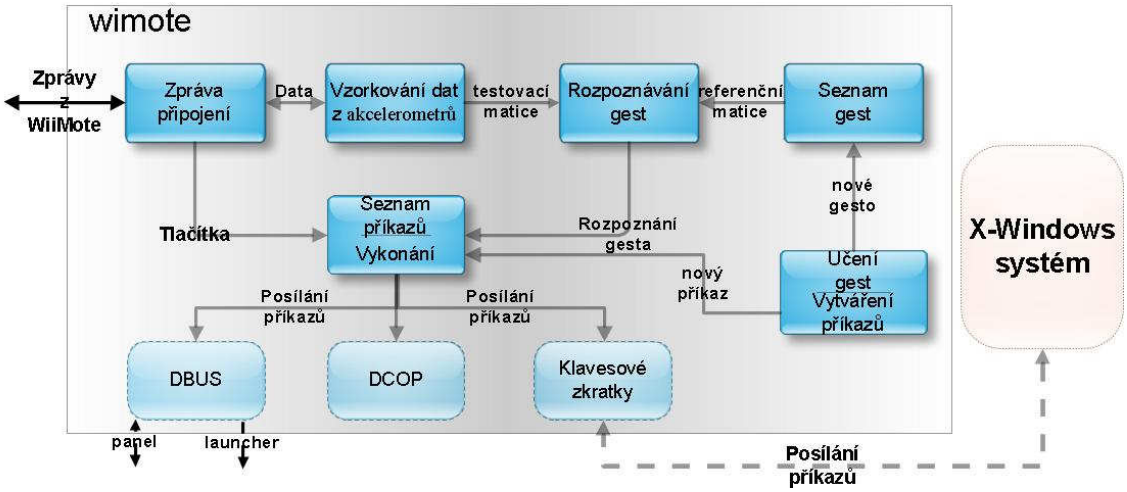
A – Editační pole „Název“, které by mělo vystihovat funkci příkazu

B – Zde se vybírá vstupní událost neboli na jaký podmět bude reagováno.

C – Zde se zadává funkce, která se má být provedena, jestliže se vyskytl podnět **B**.

Obrázek 6.9: Přidávání nového příkazu

Na následujícím obrázku je schéma programu wiimote, která popisuje vnitřní komunikaci.



Obrázek 6.10: Schéma programu wiimote

Zprávu připojení Wii remote má na starost třída ***Wii***. Pro získaní dat z WiiMote využívá knihovnu cwiid [18], která umožňuje pomocí jednoduchého API napsaného v C s ním komunikovat Tato třída dědí QThread, takže snadno vytvoří nové vlákno po dobu vyhledání WiiMotu. Jinak by došlo k zablokování programu na několik vteřin po dobu vyhledávání. Třída implementuje funkce pro samostatné vyhledávání WiiMote v časových intervalech 5s. Dále implementuje funkce pro ovládání diod a vibračního motoru. Tato třída umožňuje připojit pouze jedno WiiMote současně.

Záznam gesta se ukládá do třídy ***TVzor*** a zapouzdřuje v sobě všechny data. Stará se taktéž o načítání a ukládání dat z textového souboru. Průběhy gest jsou znázorněny na grafech v příloze A.

Meziprocesovou komunikaci přes D-BUS se stará třída ***TDataDbus***. Tato třída vytváří spojení k zadanému objektu (viz obrázku 6.8) na sběrnici D-BUS. Umožňuje tak snadné vyvolání jeho funkce, čí příjímání jeho signálů. Používá se i ke komunikaci s programy launcher a panel.

Program implementuje i druhý zmíněný IPC mechanizmus a to DCOP. Tuto službu implementuje třída ***TDcop***, jež obstarává zasílání zpráv objektům. Dělá to tak, že spouští program *dcop* (viz kapitole 4.1), který se za ní postará o doručení zprávy. Vytvoření spojení přímo v programu by byly značně náročné, neboť by bylo potřeba importovat potřebné knihovny z KDE, což by s sebou přinášelo další závislost a problémy při kompilaci. Takto pokud nebude DCOP na OS podporován, nebude to činit žádné problémy.

Zobrazení ikony v systémové časti „system tray“ obstarává třída ***SystemIcon***. Její hlavní funkcí je informovat uživatele o událostech v aplikaci a umožňovat komunikaci s programem, který běží jako démon. Podle vybraného programu k ovládání zobrazuje jeho ikonu zadávanou přes dialog na obrázku 6.8. Pro správný běh aplikace je zapotřebí, aby OS podporoval oblast „system tray“.

Data zadávána přes zmíněný dialog (6.8) zapouzdřuje třída ***TProgram***. Ta komunikuje s třídami ***TDcop*** a ***TDbus*** a stará se o jejich správnou inicializaci a obsahuje i seznam příkazů pro daný program.

Dialogové okno (6.7) pro záznam nového gesta je implementován v třídě ***TZaznam***. Má vlastní časovač a podle vzorkovací periody doluje data z akcelerometrů přes třídu ***Wii***. Pro zpětnou vazbu k uživateli, aby si mohl vyzkoušel zaznamenané gesto, musí navíc komunikovat s třídou ***TDtw***. Pokud dojde k rozpoznání zaznamenaného gesta, pošle zprávu třídě ***SystemIcon***, jež zobrazí tuto informaci.

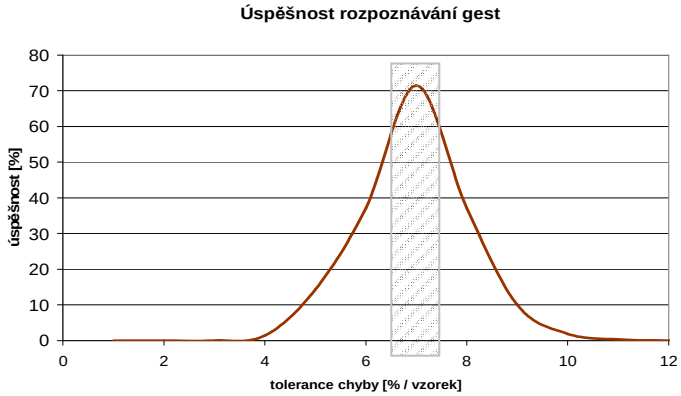
Interpretaci příkazů má na starost třída ***TControl***. To zahrnuje posílání příslušných povelů programu panel (6.2), vybírání gest k porovnání, kontrolovat tlačítka WiiMote a rozdávat povely k vykonání příslušných příkazů. Výběr gest k porovnání se děje na základě vybraného programu a jemu přirazených gest k ovládání.

Rozpoznávání gest pomocí DTW algoritmu (dynamické borcení času) je implementováno ve třídě **TDtw**. Ta je řízena třídou ***TControl***, která ji předává gesta (referenční matice) a aktuální navzorkovaný signál z akcelerometrů. Data z akcelerometrů se nejdřív upraví dolní propustí [16] a poté jsou uložena do testovací matice, která jsou součástí třídy. Testovací matice má stejnou velikost jako nejdelší možné gesto. Podle referenční matice se vezme i přibližně stejná oblast z testovací matice a porovnají se. Algoritmus při porovnání kontroluje akumulovanou vzdálenost a pokud je překročena ještě před porovnáním celého gesta, automaticky prohlásí za „nerozpoznáno“ a ukončí algoritmus. Dále implementuje funkci pro výpočet změn (derivaci) signálů za časový okamžik. Myšlenka je taková, že pokud je signál z akcelerometrů neměnný (konstantní), nedochází k žádnému gestu a nemusí být nadbytečně porovnávány matice. Tím je posléze sníženo vytížení CPU.

7 Výsledky testování

Procesem testování bylo snahou zjistit, které jsou nejlepší parametry pro rozpoznávací modul. Bylo potřeba se zaměřit, jak na úspěšnost rozpoznání gest, tak na optimalizaci zátěže procesoru.

Nejvíce ovlivňující rozpoznávání je parametr, který určuje maximální možnou vzdálenost dvou porovnávaných signálů (gest). Výpočet této hodnoty algoritmus provádí způsobem, že se vezme délka (počet vzorků) gesta a vynásobí ji procentuální chybou, kterou se může dopustit při porovnání jednoho vzorku (viz graf 7.1), aby mohl ještě prohlásit za gesto „rozpoznáno“.

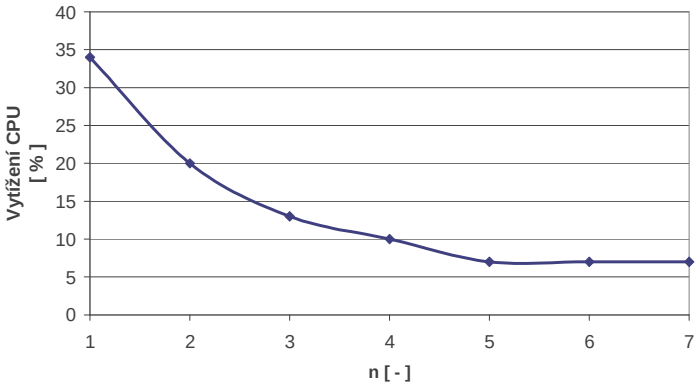


Obrázek 7.1: Graf úspěšnosti rozpoznání gest závislý na nastavené toleranci chyby

Pokud se chyba pohybovala v rozmezí 0 – 3%, bylo takřka nemožné gesta rozpoznat. Nastavený práh chyby byl tak nízký, že provedené gesto modul nezaregistroval. Na horní hranici tohoto intervalu se už se dařilo rozpoznat jednoduché gesta (pohyb do stran), ale i tak s velmi nízkou úspěšností. Již při 5% se podařilo rozpoznat všechna gesta, tedy vyjma třepání (shaking). Úspěšnost rozpoznávání gest se na této chybě pohybovala kolem 50%. Nejlepších výsledků rozpoznávání bylo dosaženo při chybě 6%. Úspěšnost rozpoznávání se dosahovalo až k 70%. Problémy opakovaně přinášelo třepání (úspěšnost kolem 60%), neboť u tohoto gesta snadno dojde ke zrychlení. Zrychlení způsobuje rozpoznávacímu modulu problémy obecně. Následně úspěšnost jen klesá, neboť nastavená chyba je již tak velká, že dochází k rozpoznání špatného gesta.

Dalším krokem bylo nalézt optimální periodu, s níž se budou porovnávat gesta. V předchozím testu se porovnávání bylo prováděno se stejnou periodou jako vzorkovací. Ale není to nutná podmínka. Porovnávání může být prováděno i na nižší frekvenci a je tak snížena zátěž CPU i při zachování úspěšnosti rozpoznávání. Na následujícím grafu (7.2) je zobrazena charakteristika výše uváděné závislosti.

Na ose *Y* je vyneseno maximální zatížení CPU programem wiimote při rozpoznávání gest, v závislosti na periodě porovnávání. Na ose *X* je vynesen n-násobek vzorkovací periody. V tomto případě je vzorkovací perioda T_{vz} = 13 ms.



Obrázek 7.2: Graf zatížení CPU v závislosti na periodě porovnávání

Zatížení CPU s každým násobkem periody klesá až ke hranici 76 ms (n=6). Zde už rozdíl s následujícím násobkem periody není tak patrný a projevuje se více režie programu. Zatížení CPU vynesené v grafu je maximální. Průměrné zatížení v praxi je menší jak jedna třetina ve srovnání s maximem, uvedeném v grafu (7.2). Měřeno bylo provedeno programem *top* na Ubuntu 8.04 (Hardy), Intel Pentium Duo T2130 - 1,86GHz, 2 GB RAM.

Vliv na úspěšnost rozpoznávání gest se začíná negativně projevovat na periodě větší jak 90 ms (n>6). Je to dáno tím, že není zaregistrován začátek gesta a poté algoritmus nedokáže najít optimální cestu s malou chybou. Úspěšnost rozpoznání bude velmi náhodná.

Z testovaných hodnot pak byl nastaven porovnávací modul na chybu 6 % s periodou porovnání pětkrát větší, než perioda vzorkování.

8 Závěr

Cílem této práce bylo prostudovat vstupní metody v prostředí X-Window systému, možné komunikace mezi procesy a získávání dat z periferních zařízení, se zaměřením na WiiMote. Tato práce dále pojednává o vývoji aplikace k ovládání počítače pomocí WiiMote. K zadávání příkazů byla využila jak tlačítka WiiMote, tak gesta rukou provedená s ovladačem. Tyto úkoly se podařilo splnit a implementovat navrhovanou aplikaci.

V rámci zkoumání problematiky emulování vstupu z klávesnice byl v 2. kapitole popsán model X-Window systému a komunikace s nim pomocí Xlib knihovny. O možnostech meziprocesové komunikace v tomto prostředí pojednává čtvrtá kapitola. Jsou zde podrobně probrány dvě hlavní techniky DCOP a D-BUS, které jsou v dnešní době ve velké míře používány v aplikacích v Linuxu.

Třetí kapitola je věnována popisu Wii remote a způsobu získávání dat z periferních zařízení. Součástí této kapitoly je i popis možnosti, jak využit WiiMote pro tvorbu uživatelského rozhraní.

Pro rozpoznávání gest provedených rukou s ovladačem byl využit algoritmus dynamické borcení času, který je podrobněji rozepsán v kapitole 5. Zpracování signálu.

V následující šesté kapitole je popsán návrh a implementace aplikace. Ta umožňuje ovládat počítač zařízením WiiMote. Obsahuje virtuální klávesnici umožňující zadávání textu. Dále byla vytvořena pracovní lišta pro snadné spouštění nových procesů a k výběru ovládaného programu. Aplikace umožňuje uživateli zadávat – „učit“ - gesta a k nim přiřazovat příkazy.

Úspěšnost rozpoznávání gest dosahovala hranice 70%. Problematická gesta byla ta, která byla oproti naučeným gestům provedena rychleji. Výsledek rozpoznávání také záležel na způsobu učení gest. V tomto vidím další možnost vývoje aplikace. Bylo by možné vytvořit referenční gesto jako průměr z několika provedení daného gesta. Pro správné rozpoznávání gest by mohl být použit sofistikovanější algoritmus například HMM algoritmus, který je však výpočetně náročnější.

V rámci práce bylo vytvořeno demonstrační video, které je přiloženo na CD.

Literatura

[1] HONEYFORD, Martyn. *Connect KDE applications using DCOP* [online]. 2004 [cit. 2009-05-06]. Dostupný z WWW: <https://www.ibm.com/developerworks/linux/library/l-dcop/>.

[2] Nintendo. *Wii.com* [online]., [cit. 2009-05-06]. Dostupný z WWW: <http://wii.com/>.

[3] CHUNG LEE, Johnny . Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing* [online]. 2008, vol. 7, no. 3 [cit. 2009-05-09], s. 39-45. Dostupný z WWW: <http://www2.computer.org/portal/web/csdl/doi/10.1109/MPRV.2008.53>.

[4] SCHEIFLER, Robert W. , GETTYS, James. *XWindow system : the complete reference to Xlib, X Protocol*. 3. vyd. USA, Buligton : Digital Press, 1992. 1000 s. ISBN 1-55558-088-2.

[5] TINKL, Lukáš. *DCOP : povídáme si s KDE* [online]. 2002 [cit. 2009-05-06]. Dostupný z WWW: <http://www.linuxzone.cz/index.phtml?ids=2&idc=363>.

[6] X.Org Foundation. *X.Org Wiki* [online].[cit. 2009-05-04]. Dostupný z WWW: <http://www.x.org/wiki/>.

[7] *X Window System protocols and architecture* [online]. Wikipedia , 8 May 2009 [cit. 2009-05-03]. Dostupný z WWW: <http://en.wikipedia.org/wiki/X_Window_System_protocols_and_architecture>.

[8] *Xlib* [online]. Wikipedia., 9 March 2009 [cit. 2009-04-20]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Xlib>.

[9] *X Window System protocols and architecture* [online]. Wikipedia., 8 May 2009 [cit. 2009-04-20]. Dostupný z WWW: <http://en.wikipedia.org/wiki/X_protocol>.

[10] CHAPMAN, Matt. *X Window Managers for X : Introduction* [online]. 1995-2009 [cit. 2009-05-05]. Dostupný z WWW: <http://xwinman.org/intro.php>.

[11] BROWN, Preston, ETTRICH , Matthias, MEINE, Hans. *DCOP : Desktop COmmunications Protocol* [online]. 1999 , May 12, 2003 [cit. 2009-05-06]. Dostupný z WWW: <http://developer.kde.org/documentation/other/dcop.html>.

[12] ČERNOCKÝ, Jan. Rozpoznávání řeči : Úvod a DTW. *UPGM FIT VUT Brno* [online]. [cit. 2009-04-10]. Dostupný z WWW: <http://www.fit.vutbr.cz/study/courses/ZRE/public/>.

[13] MACH, Petr , *D-BUS* [online]. 2006 [cit. 2009-05-10]. Dostupný z WWW: <http://python.wraith.cz/tech-dbus-koncept.php>.

[14] DREDGE, Stuart. *WiiWii : Enough already with the Wii controller hype, says analyst* [online]. 2006 [cit. 2009-05-12]. Dostupný z WWW: <http://www.wiiwii.tv/2006/10/>.

[15] Nokia Corporation. *Qt 4.5 : Introduction to D-Bus* [online]. c2009 [cit. 2009-05-10]. Dostupný z WWW: <http://doc.trolltech.com/4.5/intro-to-dbus.html>.

[16] JAN, J. *Číslicová filtrace : analýza a restaurace signálů*. VUT v Brně : VUTIUM, 2002. ISBN 80-214-1558.

[17] *Wiili.org : Wii Linux* [online], 7 December 2007 [cit. 2009-03-02]. Dostupný z WWW: <http://www.wiili.org/index.php/Main_Page>.

[18] Trac Open Source Project. *CWiid* [online]. 2007 , 1/11/2009 [cit. 2009-02-12]. Dostupný z WWW: <http://abstrakraft.org/cwiid/>.

[19] BlueZ Project. *BlueZ : Official Linux Bluetooth protocol stack* [online]. c2000-2009 [cit. 2009-05-10]. Dostupný z WWW: <http://www.bluez.org/>

[20] *Wiili.org : Wii Linux*: WiiMote [online], 7 December 2007 [cit. 2009-03-02]. Dostupný z WWW: < http://www.wiili.org/index.php/Wiimote >.

[21] *WiiYourself* [online]. 2007 [cit. 2009-05-17]. Dostupný z WWW: <http://wiiyourself.gl.tter.org/>

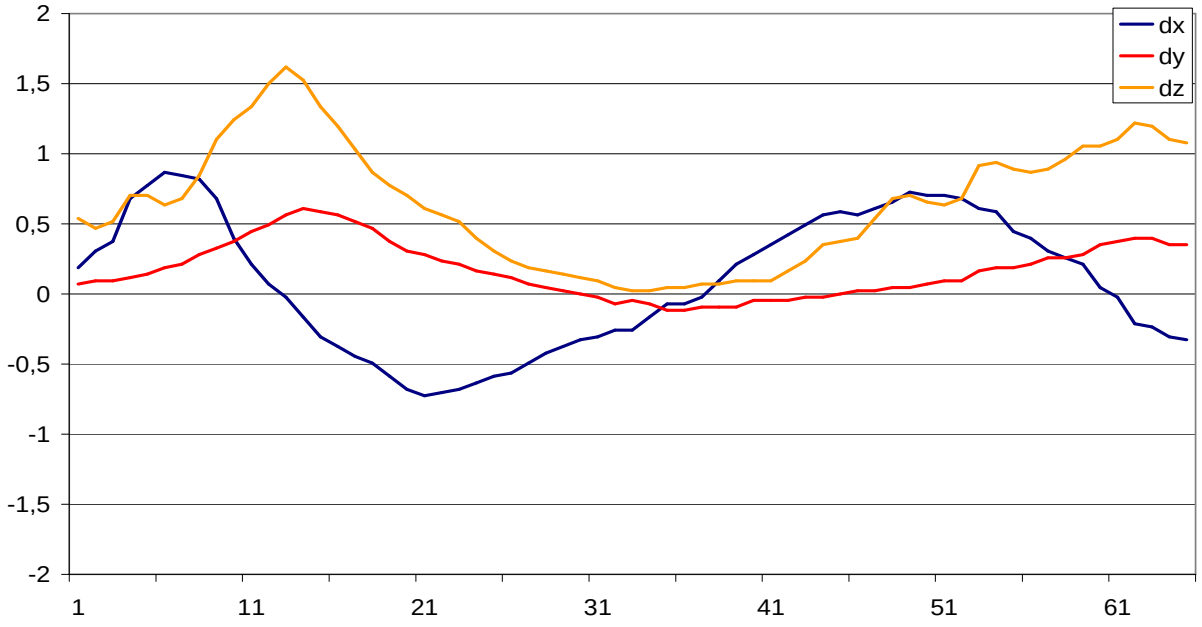
Příloha

Seznam příloh

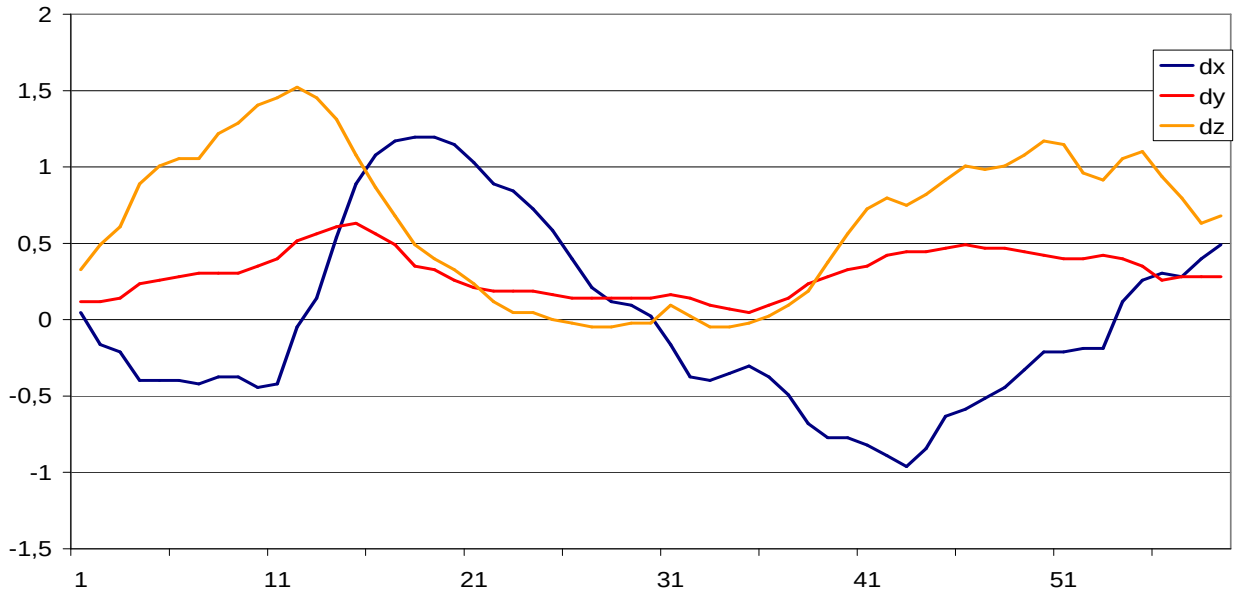
Příloha A: Grafy průběhů vybraných gest:

V této příloze je zobrazen průběh signálů z akcelometrů pro vybraná gesta. Zajímavé je porovnat průběh gesta s jejím opačným typem (vytvořených pozpátku).

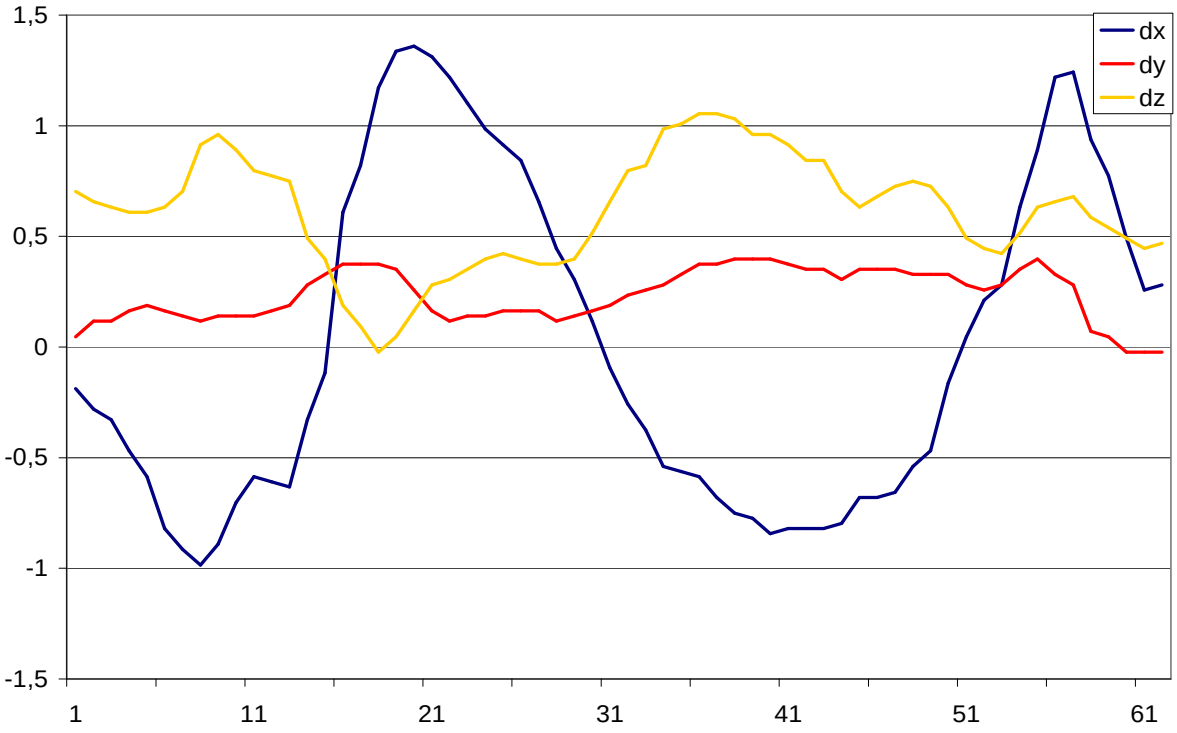
Na ose *Y* je vyneseno přetížení g (g = 9,83 m s^{-2}). Na ose *X* jsou jednotlivé vzorky gesta.



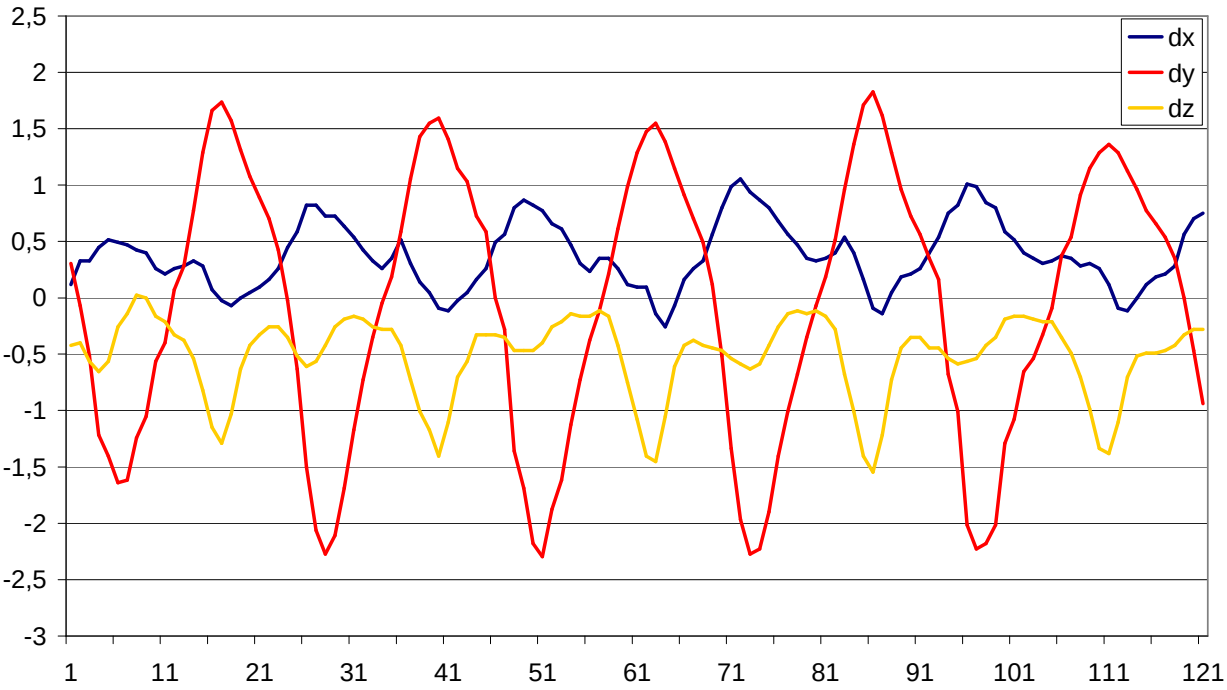
Obrázek A.1: Průběh gest - kruh po směru hodinových ručiček



Obrázek A.2:Průběh gesta - kruh v protisměru hodinový ručiček

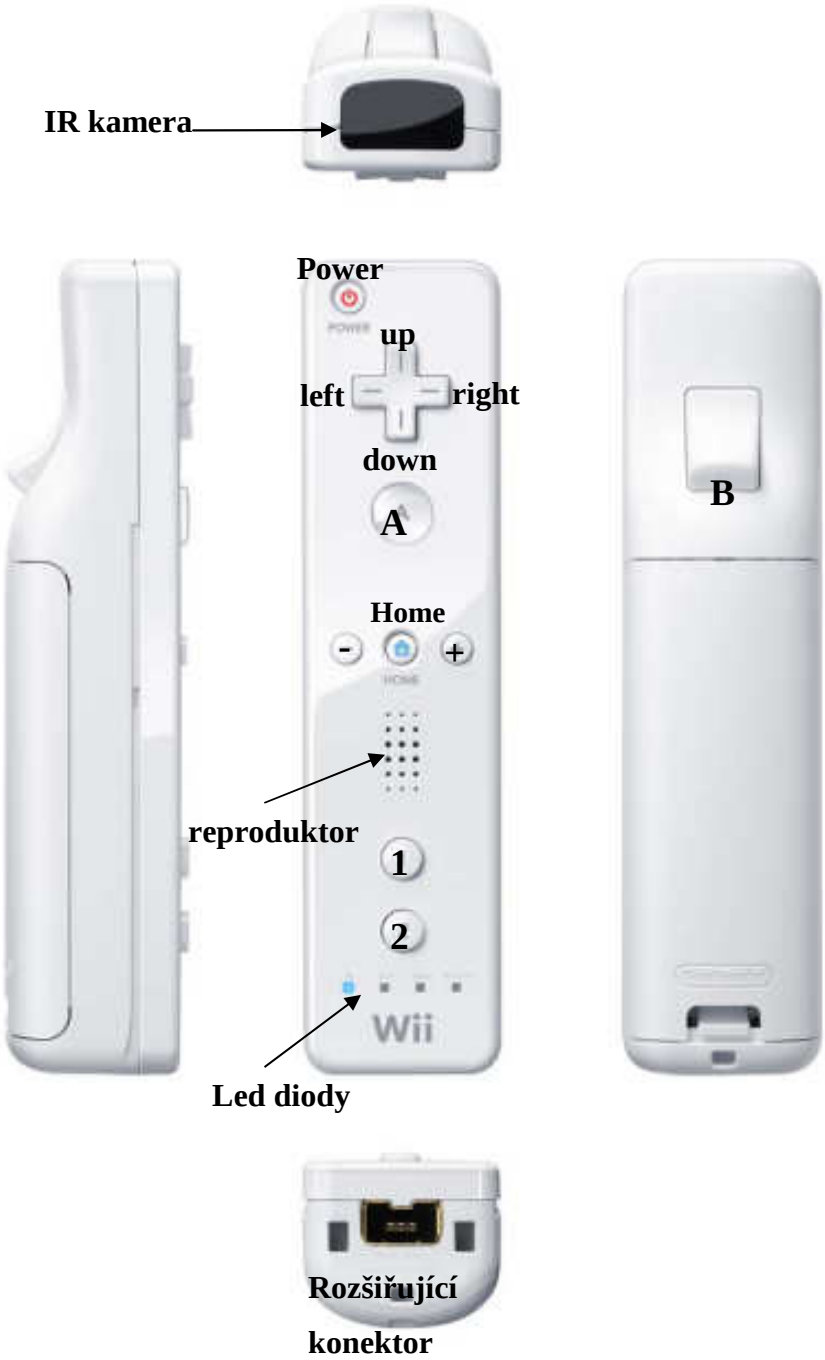


Obrázek A.3: Průběh gesta - písmeno "Z"



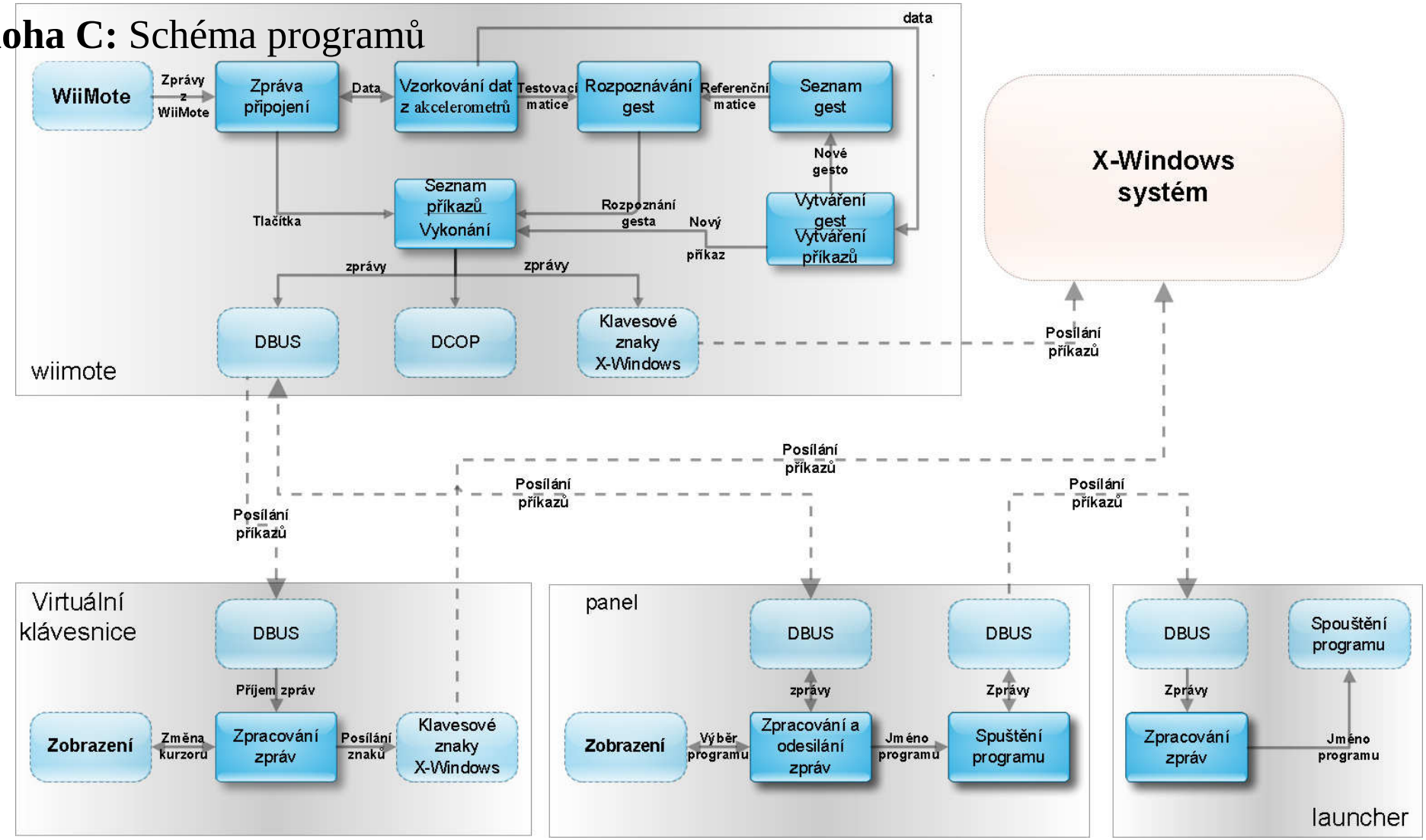
Obrázek A.4: Průběh gesta – „třepání“ (shaking)

Příloha B: Popis WiiMote



Obrázek B.1: Popis komponent WiiMote (převzato [14])

Příloha C: Schéma programů



Obrázek C.1: Schéma komunikace programů

Příloha D: Obsah přiloženého CD

K práci je přiloženo CD, které obsahuje tyto soubory.

.

|------- bp	- zdrojové soubory textu bakalářské práce
|------- help	- Uživatelská nápověda
|-------src	
| |------- keyboard	- zdrojové soubory programu ***keyboard***
| |------- launcher	- zdrojové soubory programu ***launcher***
| |------- panel	- zdrojové soubory programu ***panel***
| |------- readme.pdf	- instrukce pro překlad programů
| ‘------- wiimote	- zdrojové soubory programu ***wiimote***
|------- video	- demonstrační video
|	
‘------- xvrana01-bp.pdf	- písemná zprávu ve formátu PDF

Příloha E: Popis D-BUS rozhraní programů

V této příloze jsou popsány adresy objektů a význam jednotlivých funkcí a signálů na sběrnici D-BUS, které programy implementují.

Adresa objektů na D-BUS sběrnici:

	panel	keyboard	launcher
název služby	*cz.vutbr.fit.wiimote*	*cz.vutbr.fit.keyboard*	*cz.vutbr.fit.launcher*
název objektu	*/panel*	*/keyboard*	*/Launcher*
název rozhraní	*cz.vutbr.fit.wiimote*	*cz.vutbr.fit.keyboard*	*cz.vutbr.fit.launcher*

Program panel:

Tento program definuje tyto služby:

- *quit* – Funkce pro ukončení programu.
- *update* – Funkce pro aktualizaci panelu, pokud došlo ke změně zobrazovaných ikon.
- *showorhide* – Funkce pro zobrazení či skrytí panelu.
- *wiimoteevent* – Funkce předávající data o události od Wii remote.
- *enterIcon* – Signál, který informuje ostatní aplikace o výběru programu k ovládání.

Program keyboard:

Tento program definuje tyto služby

- *up , down, left, right* – Funkce pro pohyb kurzoru.
- *enter* – Výběr požadovaného písmene.
- *showorhide* – Funkce pro skrytí, respektive zobrazení klávesnice.
- *resizeplus. resizeminus* – Funkce pro změnu velikosti klávesnice.
- *quit* – Funkce pro ukončení programu.

Program launcher:

Tento program definuje tyto služby:

- *start_up* – Funkce s jedním parametrem jména programu ke spuštění.
- *start_upargv* - Funkce s dvěmi parametry a to jménem programu a seznamem argumentů ke spuštění.
- *quit* – Funkce pro ukončení programu.