VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUÁLNÍ PROGRAMOVÁNÍ ROBOTICKÝCH APLIKACÍ

VISUAL PROGRAMMING OF ROBOTIC TASKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID LING**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2019

Abstrakt

Tato bakalářská práce se zabývá rozšířením funkcionality systému ARTable, který je postavený na frameworku ROS. V rámci práce bylo rozšířeno uživatelské rozhraní tak, aby bylo možné vytvářet nové nebo upravovat stávající programy pro robota PR2 přímo na dotykovém stole. Veškerá rozšíření jsou implementována v jazyce Python za využití frameworků ROS a Qt. Rozšíření je plně funkční a propojené se zbytkem systému ARTable.

Abstract

The purpose of this thesis is to extend the functionality of ARTable system, which using Robot operating system (ROS). In this thesis, user interface was extended and now it is possible to create new or edit already created programs for PR2 robot using touch table. All extensions are implemented in Python using ROS and Qt frameworks. The extensions are fully functional and connected to the rest of the system.

Klíčová slova

Vizuální programování, ARTable, PR2, Robotický operační systém, Qt, uživatelské rozhraní, interakce člověka s robotem

Keywords

Visual programming, ARTable, PR2, Robot operating system, Qt, user interface, human–robot interaction

Citace

LING, David. *Vizuální programování robotických aplikací.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Kapinus,

Vizuální programování robotických aplikací

Prohlášení

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

.....
David Ling
14. května 2019

Poděkování

Na tomto místě bych rád poděkoval vedoucímu své práce panu Ing. Michalu Kapinusovi za odborné vedení, užitečné rady, připomínky a snahu pomoct s řešením problémů, které se v průběhu práce vyskytly. Dále bych rád poděkoval účastníkům uživatelského testování za jejich ochotu a čas.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá rozšířením projektu ARTable výzkumné skupiny Robo@Fit[1]. Systém ARTable je společné pracovní prostředí člověka a robota. Člověk a robot mezi sebou komunikují především skrze uživatelské rozhraní promítané na stůl s dotykovou deskou. Cílem tohoto prostředí je ověřovat základní principy spolupráce člověka s robotem a především to, jak by podobný systém byli schopni obsluhovat lidé bez IT vzdělání. Při návrhu a tvorbě grafického uživatelského rozhraní je kladen důraz na co nejjednodušší ovládání a co nejrychlejší a nejintuitivnější předávání instrukcí samotnému robotovi. Cílem by mělo být to, aby uživatel naučil robota vše, co po něm chce, a robot pak danou činnost sám nebo s dopomocí vykonával.

Systém ARTable v současnosti nepodporuje možnost vytvářet nové nebo editovat stávající programy přímo na stole. Lze je vytvářet nebo editovat pouze pomocí počítačového skriptu. Takto vytvořené programy lze pouze načíst a upravit jeho parametry na dotykovém stole. Pro vytvoření nového programu je nutná znalost programování.

Hlavním úkolem a cílem této práce je tedy rozšíření funkcionality uživatelského rozhraní, aby bylo umožněno lidem bez znalosti programování vytvářet nové programy a přímo je robota naučit a spustit.

Teoretická část této práce popisuje rozšířenou realitu, různé přístupy k uživatelským rozhraním, jejich návrhu a testování. Dále popisuje systém ARTable a frameworky ROS a QT, na kterých je postavený. V další kapitole je zmapován a popsán aktuální stav uživatelského rozhraní a jsou navrhnuty jeho úpravy. Čtvrtá kapitola popisuje samotnou implementaci a v páté kapitole je popsán postup při testování výsledné aplikace. Pátá kapitola také obsahuje shrnutí výsledků uživatelského testování a průzkumu.

[1] http://www.fit.vutbr.cz/research/groups/robo/.cs

Kapitola 2

Teoretická část

V této kapitole jsou popsány teoretické základy, ze kterých bylo vycházeno při návrhu a implementaci výsledného řešení. Obsahuje popis toho, co je to rozšířená realita a jak se v ní pracuje s uživatelskými rozhraními. Dále pak popisuje technologie použité při implementaci a ARTable jako platformu, pro kterou bylo řešení navrženo.

2.1 Rozšířená realita

První výskyt pojmu Rozšířená realita (anglicky Augmented reality, zkratkou AR) se datuje do 50. let minulého století a od té doby se tato technologie vyvinula do fáze, kdy je považována za jeden z dalších kroků v moderní budoucnosti. Rozšířená realita, jak ji definovali ve svém článku Julie Carmigniani a Borko Furht, je přímý či nepřímý pohled na okolí v reálném čase rozšířený o virtuální informace generované počítačem [3]. Rozšířená realita má za cíl ulehčit uživateli život pomocí virtuálních informací v jeho nebližším okolí. Dle Azumy [1] by měl systém rozšířené reality splňovat následující parametry:

- kombinuje reálné a virtuální objekty v reálném prostředí
- probíhá interaktivně v reálném čase
- zarovnává spolu reálné a virtuální objekty

Prostorová rozšířená realita

Prostorová rozšířená realita (anglicky spatial augmented reality, zkratkou SAR) je takovým typem rozšířené reality, která využívá k zobrazování videoprojektory a objekty reálného světa. Uživatel tak nemusí používat žádné speciální zařízení k ovládání aplikací v SAR. ARTable, se kterým se pracuje v této práci, je právě příkladem SAR.

Historie

Dle [11] je za toho, kdo jako první vytvořil něco, čemu by se v dnešní době dalo říkat virtuální nebo rozšířená realita, považován Ivan Sutherland, který v 60. letech minulého století vynalezl k hlavě připevněný displej (anglicky head-mounted display, zkratkou HMD). Poprvé se pojem „rozšířená realita“ vyskytl v roce 1990 v práci Caudella a Mitzella, ve které zkoumali, jak by se HMD mohly využívat v továrnách na výrobu letadel pro zefektivnění práce dělníků.

Jeden z prvních funkčních AR systémů vyrobil v roce 1994 L. B. Rosenberg. Systém nesl název Virtual Fixtures a pomáhal za pomocí virtuálních linií lepšímu ovládání vzdálené robotické ruky [9].

V roce 1997 provedl Azuma první průzkum o rozšířené realitě, ve kterém tento pojem jasně definoval [1].

V roce 2000 byla na Symposium of Wearable Computers představena první venkovní mobilní hra založená na rozšířené realitě.

Užívání rozšířené reality se začalo stávat více a více rozšířené po roce 2000 v návaznosti na to, jak se začaly vyvíjet mobilní telefony. V roce 2003 představili Wagner a Schamlstieg první ruční zařízení pro rozšířenou realitu. Dalo by se považovat za předchůdce dnešních chytrých telefonů. V roce 2009 se vznikem frameworku ARToolkit nastal velký rozvoj ve vývoji aplikací pro AR. Tento framework řeší dvě z hlavních otázek, se kterými se aplikace pro AR musí potýkat – sledování pohybu a interakci virtuálních objektů s reálnými.

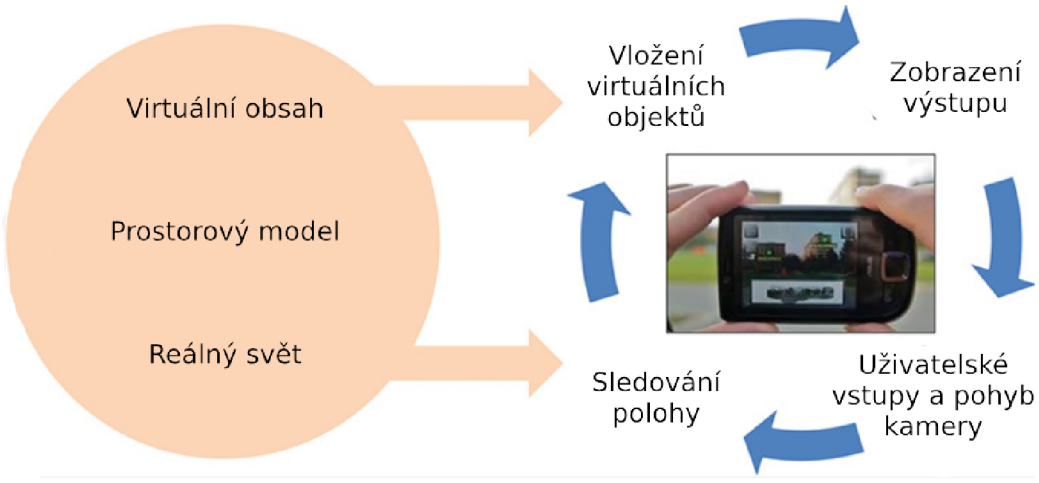
Dnes se AR využívá např. ve zdravotnictví, armádě, průmyslu i na osobních zařízeních v podobě her. V roce 2013 společnost Google spustila projekt Glass, který měl za cíl vyvinout brýle pro rozšířenou realitu. Tento projekt otevřel vývojářům cestu k vývoji aplikací pro takové zařízení a i přesto, že tento projekt zatím žádným velkým průlomem neskončil, brýle či jiná nositelná AR zařízení jsou pravděpodobně součástí naší budoucnosti.

Princip fungování rozšířené reality

V knize Augmented reality: Principles and practises [11] se píše, že rozšířená realita ke svému fungování potřebuje nejméně 3 komponenty a těmi jsou:

- **trackovací zařízení** - zpracovává data z kamery a dalších sensorů (dle zařízení)
- **registrační jednotka** - jednotka, která umisťuje virtuální objekty do reálného světa
- **zobrazovací zařízení** - prostředek pro zobrazení aktuálního stavu systému uživateli

Tyto tři komponenty spolu komunikují tak, jak je zobrazeno na obrázku 2.1. Aplikace rozšířené reality jsou většinou obsluhovány smyčkou, která nejprve zpracuje uživatelovy vstupy, zjistí polohu kamery a další data ze senzorů a poté umístí virtuální objekty do prostorového modelu, který pak pomocí zobrazovacího zařízení zobrazí uživateli.



Obrázek 2.1: Smyčka zpětných vazeb v aplikacích pracujících s rozšířenou realitou

Jedním z hlavních problémů, které musí AR systémy řešit, je to, kam přesně zobrazit virtuální objekty do uživatelova pohledu. To může být počítáno a zjišťováno několika způsoby.

Aplikace může využívat ke zmapování prostoru ***markery***, což jsou objekty (např. obrázky) s výraznými vizuálními prvky, tudíž jsou jednoduše rozpoznatelné. Typickým příkladem takového markeru může být například QR Code.

Jiné typy aplikací využívají k získávání informací o okolním prostoru ***GPS a jiné prostorové senzory***. Virtuální objekty jsou do uživatelova pohledu na základě jeho polohy, směru pohledu, rychlosti pohybu apod. Takové aplikace se často využívají například k navigaci.

Mezi nejnovější přístupy v AR patří také ***SLAM (Simultaneous Localization And Mapping)***, což je přístup, kdy AR využívá několik kamer s hloubkovými sensory k tomu, aby si vytvořila mapu okolí, o kterém nemá žádné informace a ve kterém pak sama pracuje [4][5]. Tento přístup se využívá i v projektech velkých technologických firem jako ARKit[1] od společnosti Apple nebo ARCore společnosti Google[2].

Existují také aplikace, které využívají detekci objektů k tomu, aby reálné objekty nahradily nebo rozšířily digitálním ekvivalentem. V jedné aplikaci lze využít několik přístupů najednou pro různé funkce.

Systémy rozšířené reality lze zobrazovat a ovládat různými způsoby. Mezi hlavní zařízení spojené s rozšířenou realitou jsou displeje, vstupní (ovládací) zařízení, počítače a trackovací zařízení.

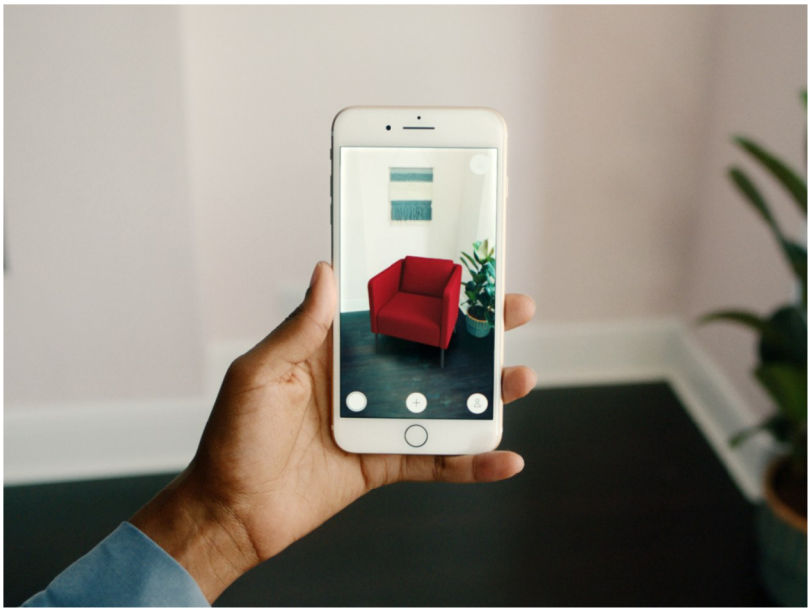
Displeje

Displeje připevněné k hlavě uživatele (HMD) jako například různé helmy nebo brýle umisťují jak reálné, tak virtuální prostředí přímo před uživatelův pohled. Výhodou HMD je to, že uživatel může pracovat bez omezení oběma rukama. Nevýhodou nebo problémem, který takové displeje a uživatelská rozhraní na nich zobrazována musí řešit, je to, aby virtuální objekty nezakrývaly velkou část uživatelova pohledu a nesnižovaly mu tak orientaci v reálném světě. Příklad takového displeje lze vidět na obrázku 2.2.

Obrázek 2.2: Brýle Microsoft Hololens pro rozšířenou nebo virtuální realitu

Nejrozšířenějším typem displejů jsou displeje, které uživatel drží v ruce a rozšířená realita je zobrazována skrze ně (viz obrázek 2.3). Typickými zástupci takových displejů jsou chytré mobilní telefony nebo tablety. Aplikací na mobil nebo tablet používající obrazovku jako výstup rozšířené reality je spousta. Pro ukázku můžou být zmíněny například aplikace pro zobrazování nábytku v prostoru nebo hry typu Pokemon GO, kde uživatel vidí herní postavy skrze kameru v telefonu v reálném prostředí.

[1] Apple ARKit – https://developer.apple.com/arkit/
[2] ARCore – https://developers.google.com/ar/



Obrázek 2.3: Ukázka využití AR v aplikaci obchodního řetězce IKEA. Uživatel si v aplikaci může vybrat kus nábytku a poté si pomocího fotoaparátu zobrazit, jak by vypadal v místě, na které uživatel fotoaparátem míří.

Prostorové displeje jsou takové plochy, na které lze promítnout nebo zobrazit virtuální realitu a zároveň nejsou nijak závislé na uživateli. Jsou umístěny v prostoru a používají se kupříkladu klasické displeje (většinou větších rozměrů), průhledné obrazovky, videoprojektory či hologramy. Mezi takové displeje patří i ARTable, na který je projektorem promítáno uživatelské rozhraní, které lze ovládat pomocí dotykové desky položené na tomto stole.

Vstupní zařízení

Jako vstupní zařízení může být použito buď stejné zařízení, na které se zobrazuje i výstup (typicky používáno na chytrých telefonech, kdy uživatel vidí rozšířenou realitu skrze kameru a na displeji s ní může pracovat) nebo může být použito jiné speciální zařízení. Jako takové můžou být zmíněny rukavice, náramky či různé další ovladače, které uživatel drží v ruce. Ovládání vstupu musí být zvoleno v závislosti na použití dané aplikace. Pokud se jedná o aplikací využívanou v situacích, ve kterých jsou potřeba volné ruce, jsou vstupní zařízení držená v ruce pro danou aplikaci nevhodná a musí se zvolit jiný přístup ovládání.

Kamery a snímací zařízení

AR pro interakci virtuálních objektů s okolními potřebuje snímat okolí. To se provádí pomocí kamer a senzorů jako např senzorů GPS, akcelerometrů, kompasů atd. Každá z těchto technologií má jinou přesnost, citlivost a je jinak náročná na zpracování. Proto je důležité při vývoji AR systému zvážit vhodnost dané technologie pro vyvíjený systém.

Využití rozšířené reality

V dnešní době hlavně díky rychlému vývoji výpočetních jednotek lze využívat relativně složité AR systémy prakticky kdekoliv (mobil, tablet, notebook, atd.). V této sekci bude uvedeno několik typických příkladů AR systémů z různých odvětví.

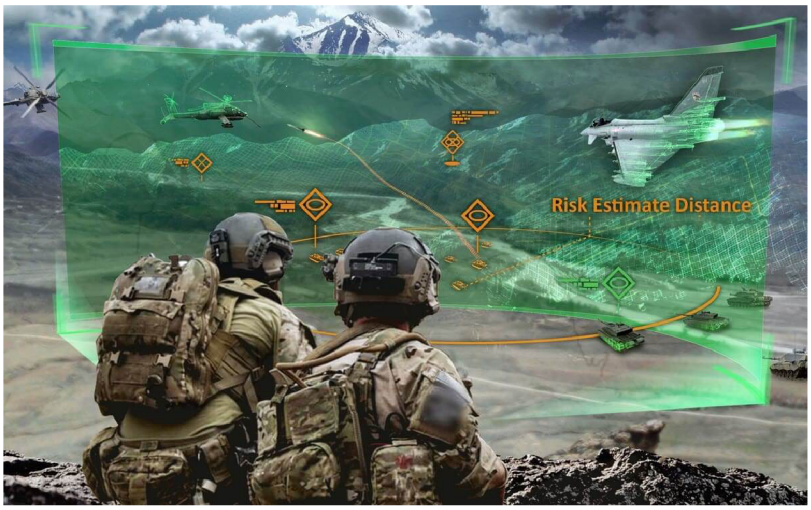
Jak bylo popsáno již v kapitole o historii AR systému, jedny z prvních aplikací byly orientovány na použití v průmyslových továrnách pro lepší efektivitu práce. AR aplikace jsou dále využívány v oblastech jako je vojenství, medicína, vzdělávání, navigace či v herním průmyslu.

V armádách jsou AR aplikace využívány pro zobrazování dodatečných informací např. pro piloty, kteří tak mají informace o rychlosti letadla, výšce atd. přímo před očima a nemusí tak sklánět hlavu při pohledu na palubní desku.

V medicíně se AR využívá jako prostředek pro trénink a výuku provádění operací v kontrolovaném prostředí, kde si nový chirurg může operaci vyzkoušet nebo jako pomoc při reálných operacích, kde AR aplikace na základě rentgenových a dalších sensorů zobrazuje data o pacientově stavu přímo do pohledu doktora. AR může taktéž pomáhat při vedení řezu tělem tak, aby nedošlo k nechtěnému poranění.

Aplikace pro navigaci užívají AR pro jednodušší přesun z místa A do místa B. Na základě kamery a GPS sensorů může uživatel vidět zvolenou cestu přímo ve svém reálném pohledu.

V neposlední řadě budou uvedeny pro příklad aplikace pro vzdělávání či turistiku. Existuje spousta aplikací, které umožňují za pomocí kamery chytrého telefonu ukazovat dodatečné informace o místě nebo objektu, na který je kamera namířena. Takto se uživatel může dozvědět více informací o daném místě nebo se podívat, jak vypadalo před 50 či 100 lety. Takové aplikace jsou využívány i v muzeích a galeriích, kde jsou zobrazovány informace o vystavených exponátech.



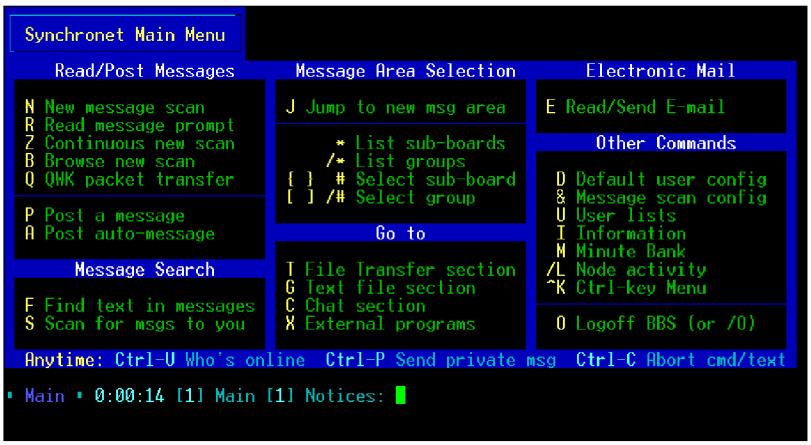
Obrázek 2.4: Využití AR v armádním prostředí. Vojákům s brýlemi pro rozšířenou realitu jsou do zorného pole zobrazovány informace o terénu, vzdálenostech, nepřátelích nebo taktické informace.

2.2 Uživatelská rozhraní

Uživatelské rozhraní je souhrn způsobů, jakými lidé (uživatelé) ovlivňují chování strojů, zařízení, počítačových programů či komplexních systémů. V knize User interface design and evaluation je uživatelské rozhraní definováno jako součást počítačového systému, se kterým uživatel pracuje proto, aby prostřednictvím určitých úkonů dosáhl požadovaných cílů [13]. Uživatelské rozhraní má za cíl zpracovat vstupy od uživatele, kterými daný systém ovládá a prezentovat výsledky chování systému zpět uživateli.

Mezi hlavní typy uživatelských rozhraní patří:

- **textové uživatelské rozhraní** – nazýváno také jako příkazová řádka. Program je ovládán zadáváním textových příkazů
- **grafické uživatelské rozhraní** – uživatel ovládá software pomocí interaktivních grafických prvků
- **hlasová rozhraní** – uživatel ovládá software mluveným slovem (např. aplikace Siri na zařízeních od firmy Apple)
- **rozhraní založené na gestech** – uživatel ovládá software gesty, která jsou snímána kamerou a převáděna na příkazy



Obrázek 2.5: Ukázka textového uživatelského rozhraní. Konkrétně terminálového serverového systému Synchronet

Grafická uživatelská rozhraní

Jak již bylo zmíněno výše, grafické uživatelské rozhraní (anglicky Graphical user interface, zkratkou GUI), je takové uživatelské rozhraní, které uživatel ovládá interakcí s grafickými prvky zobrazovanými většinou na displeji.

Tyto prvky můžeme dělit na:

- **vstupní prvky** - prvky, díky kterým uživatel vkládá data do aplikace. Můžou být buď textové (textové pole) nebo výběrové (výběrový seznam, zaškrtávací pole, slider)
- **výstupní (informační) prvky** - prvky, díky kterým je uživatel informován o stavu aplikace

- **akční prvky** - prvky, díky kterým uživatel vyvolá nějakou akci. Většinou jsou to různé typy tlačítek po jejichž stisknutí nebo kliknutí na ně je provedena určitá akce (odeslání formuláře, přechod na jinou stránku, ...)

Obrázek 2.6: Ukázka GUI v systému Windows. Konkrétně nábídky Start ve Windows 8

Existuje několik různých způsobů, jak zobrazovat a ovládat GUI. Nejběžnějším způsobem je zobrazení na nějaký displej (mobilní telefon, tablet, počítač). Jiným přístupem, který využívá i systém ARTable je zobrazovat GUI pomocí projektoru na plochu, skrze kterou lze pak GUI ovládat. Takový projektor je možné umístit:

1. Pod nebo za promítanou plochu a promítat tak GUI zespod nebo zezadu plochy.
2. Před nebo nad promítanou plochu a promítat tak GUI přímo.

Ovládání je u počítačů uskutečňováno většinou pomocí myši a klávesnice, u dotykových displejů lze GUI ovládat dotyky. Toto jsou asi nejběžnější přístupy především díky jejich jednoduchosti a rychlosti. Méně často se můžeme setkat i s jinými způsoby ovládání jako jsou různé ovladače (herní konzole), naklánění zařízení (mobily), snímaní pohybu očí atd.

Hmotná uživatelská rozhraní

Hmotné uživatelské rozhraní (anglicky Tangible user interface, zkratkou TUI) je takové rozhraní, které propojuje fyzické objekty reálného světa a digitální data [12]. Fyzické objekty slouží v TUI jak k ovládání vstupu a definování akcí, tak k prezentaci výstupu a výsledkům provedených akcí.

Jednoduchým příkladem TUI je klasická počítačová myš. Pohyb fyzickým objektem je převáděn na digitální informaci v podobě pohybu kurzoru po obrazovce.

Rozšířená realita, která využívá TUI, je nazývána jako hmotná rozšířená realita (anglicky Tangible augmented reality, zkratkou TAR). Taková realita poskytuje 3D virtuální objekty kdekoliv v reálném prostředí, zatímco v ten samý čas dovoluje uživateli interagovat s tímto virtuálním obsahem stejně jako s reálnými objekty [2]. Jako příklad takové reality může působit i systém ARTable, se kterým se pracuje v této práci.

Obrázek 2.7: Hudební systém Reactable využívající TUI. Umisťováním a propojováním jednotlivých kostek na stůl je ovládán syntetizátor, tvořeny nové zvuky nebo zvukové efekty.

2.3 Návrh a testování uživatelských rozhraní

Tato kapitola popisuje to, jaké fáze obsahuje proces návrhu uživatelského rozhraní. Je zaměřena na tzv. uživatelsky-orientovaný návrh. Dále kapitola popisuje, jaké jsou možnosti při získávání zpětné vazby od budoucích uživatelů vyvíjeného uživatelského rozhraní.

Uživatelsky-orientovaný návrh

Uživatelsky-orientovaný návrh (anglicky User centered design, zkratkou UCD) je dle [13] přístup k návrhu a vývoji software, do kterého jsou zapojeni i samotní budoucí uživatelé. Tento přístup má za cíl zvýšit použitelnost daného produktu. Použitelnost je definována jako rozsah, ve kterém produkt nebo služba mohou být využity k dosažení specifických cílů s efektivitou, účinností a zadostiučiněním ve specifickém kontextu užití [6].
4 hlavní principy UCD popsané v knize [13] jsou:

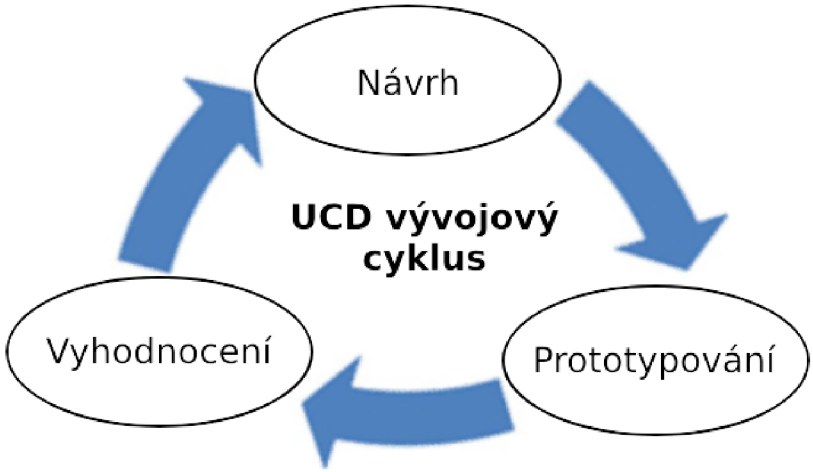
- aktivní zapojení uživatelů
- správné vymezení rolí mezi uživateli a systémem
- iterativní přístup k návrhu řešení
- mezidisciplinární týmy návrhářů

Mezi 4 základní techniky používané při uživatelsky orientovaném návrhu patří:

- specifikace a porozumění kontextu užití
- specifikace požadavků uživatelů
- tvorba návrhů řešení (prototypy)

- porovnávání navržených řešení se specifikovanými požadavky uživatelů

UCD využívá iterativní návrh UI ve kterém se střídají 3 fáze a to návrh, prototypování a vyhodnocování pomocí uživatelského testování (viz. obrázek 2.8).



Obrázek 2.8: UCD vývojový cyklus složený z návrhu, prototypování a vyhodnocování

Prototypování

Vytváření a testování prototypů je dobrý nástroj k tomu, jak získat v časných fázích vývoje zpětnou vazbu od uživatelů. Prototyp je neúplná verze výsledného produktu, která se od výsledku může velice lišit a obvykle modeluje jen jeho určitou část. Prototypy můžeme rozdělit na dva druhy:

- **Fyzické prototypy** - mezi takové prototypy patří takové prototypy, které nejsou realizovány žádným programem ani softwarem, ale jsou vytvořeny fyzicky například z papíru.
- **Softwarové prototypy** - prototypy vytvořené jako program nebo vytvořené pomocí prototypovacích nástrojů. Mezi takové nástroje patří například nástroje pro tvorbu mock-upů. Mock-up je prototyp, který umožňuje testování designu a někdy také základní funkcionality.

Uživatelské testování

Uživatelské testování se v UCD používá pro získávání zpětné vazby od uživatelů k zatím dosaženým výsledkům. Tato zpětná vazba pak ovlivňuje další návrh.

V raných fázích vývoje software je smyslem uživatelského testování ověřování uživatelských požadavků, odhad použitelnosti produktu nebo jeho části. Pro to se nejčastěji využívají fyzické prototypy či mock-upy. Díky těmto metodám lze za nízkou cenu získat kvalitní hodnocení.

V pozdějších fázích vývoje se testování využívá převážně k ověření toho, nakolik produkt splňuje požadavky uživatelů. V těchto fázích už je výsledný produkt hotov nebo téměř hotov a výsledky uživatelských testování se spíše než do aktuálního vývoje zařadí až do pozdějších verzí softwaru.

Mezi nejpoužívanější způsoby testování a získávání zpětné vazby patří:

Pozorování uživatelů při reálné práci s produktem. Pozorování může být přímé (uživatel je pozorován jinou osobou) či nepřímé (natáčení na záznam a pozdější analýza). Tento typ vyhodnocování lze dobře použít pro specifikaci požadavků. Pomáhá zjistit, jaké části jsou pro uživatele srozumitelné a co mu naopak dělá problém.

Rozhovory a dotazníky slouží ke zjišťování toho, co si uživatel o softwaru myslí. Může sice dosahovat skvělých výsledků při práci s programem, ale z nějakého důvodu ho nerad používá, a to lze zjistit kladením otázek. Strukturovaný rozhovor má předem danou sadu otázek, které jsou položeny každému uživateli. Oproti tomu flexibilní rozhovor má definovanou pouze skladbu témat, ale žádnou pevnou strukturu. Je tak volnější a lze získat širší okruh informací, ale je náročnější na analýzu.

2.4 ARTable

ARTable je systém vytvořen výzkumnou skupinou Robo@FIT na Fakultě informačních technologií Vysokého učení technického v Brně. ARTable je pracovní prostředí, ve kterém by měl být člověk schopen efektivně spolupracovat s robotem. Prostředí se skládá ze samotného robota PR2, dotykového stolu, projektoru a kinectů. Knihovny pro práci s ARTable jsou k nalezení na serveru Github.com[3].

Robot PR2

PR2 (PR = personal robot, česky osobní robot) je robot vyvinutý především pro výzkumné účely. Cílem robota je spolupráce s člověkem v společném prostředí. Robot se skládá z teleskopického těla, dvou paží a hlavy. Výšku robota lze upravit mezi 133 a 164 cm. Ruce obsahují kameru a dlaně jsou vybaveny akcelerometrem a sensorem tlaku. Hlava je vybavena kamerami a kinectem, který je v sytému ARTable používán pro detekci a rozpoznávání objektů. Robot umí provádět následující instrukce:

- podat ze stolu – najde, uchopí a zvedne objekt z dotykového stolu
- podat z podavače – uchopí a zvedne objekt z podavače umístěného na jedné ze stran stolu
- položit na stůl – položí dříve zvednutý objekt na určené místo
- umístit objekt do krabice – umístí objekt do krabice na základě podmínky
- nanést lepidlo – simulace nanášení lepidla na objekt

Pro každou z těchto instrukcí musí být zadán určitý počet parametrů jako typ objektu nebo místo na stole.

Robot umí provádět i další instrukce, které nepotřebují zadání dalších parametrů. Mezi tyto instrukce patří instrukce:

- připrav se – nastaví ramena do základní pozice
- počkej na uživatele – program je pozastaven, dokud uživatel nepřijde ke stolu
- počkej, než uživatel skončí – čeká na to, než uživatel dokončí svou práci s objektem na stole

[3]Github repozitář projektu ARTable - https://github.com/robofit/arcor

Další komponenty systému ARTable

Středem rozestavení je klasický pracovní stůl o rozměrech 150 x 70 cm, na kterém je položena dotyková kapacitní fólie s 4 mm speciální plastovou vrstvou, která je do jisté míry vodivá [4]. Stůl je vybaven také červeným tlačítkem, kterým lze kdykoliv zastavit veškerou práci robota. Nad stolem je konstrukce, která drží projektor Acer P6600, jehož primární funkcí je promítat uživatelské rozhraní na pracovní stůl. Na této konstrukci jsou po stranách zavěšené také dva reproduktory a 3 Kinecty, dva pro detekci objektů a kalibraci a jeden pro snímání uživatele. Všechny komponenty jsou společnou sítí spojené s centrálním počítačem, který je umístěn pod pracovním stolem.

Obrázek 2.9: Ukázka systému ARTable

2.5 Robot operating system

Robot operating system (ROS) je framework pro programování robotických aplikací s otevřeným zdrojovým kódem (open-source). Poskytuje funkce typické pro operační systém jako hardwarovou abstrakci, zasílání zpráv mezi procesy nebo implementaci základních funkcionalit. Dále obsahuje knihovny a nástroje pro vývoj, překlad a běh aplikací na různých platformách.

Podporovanými platformami jsou momentálně pouze systémy založené na Unixu (Unix-based systems). Primárně je ROS vyvíjen pro Operační systém Ubuntu a MAX OS X, ale značně velká komunita, která se okolo ROS vytvořila, udržuje i podporu pro další linxové distribuce (Fedora, Gentoo či Arch Linux).

Systém je vyvíjen firmou Willow Garage a aktuální verze nese název Melodic Moreniea. Systém ARTable je implementován pro verzi ROS z roku 2014 Indigo Igloo.

[4] Stránka firmy Fortes http://www.fortes.cz/portfolio/artable-na-vut-fit-2/

Architektura ROS je založená na uzlech a peer-to-peer komunikaci mezi jednotlivými uzly pomocí zpráv nebo služeb. Zdrojové soubory jsou shlukovány do balíčků.

Balíčky

Balíček je základní jednotkou ROS pro organizaci zdrojového kódu, překlad i spouštění [8]. ROS si klade za cíl nebýt velkým monolitickým systémem, ale co nejlehčím ve svém jádru, na které lze navázat velké množství nezávislých komponent, které lze znovupoužít v různých systémech. Tyto komponenty implementovány právě pomocí balíčků. Každý balíček obsahuje ve svém kořenovém adresáři soubor package.xml, který obsahuje informace o balíčku jako název, verze či autoři a závislostech na jiné balíčky

Uzly

Uzel je v ROS proces, který provádí výpočty [8]. Uzly jsou propojené dohromady do grafu a komunikují mezi sebou pomocí tematických zpráv nebo služeb. Každý uzel má na starost jednu funkcionalitu. V systému ARTable existuje například uzel pro lokalizaci objektů, ovládání paží, kalibraci, přijímání vstupů ze stolu atd. Mezi hlavní výhody architektury uzlů patří:

- chyba ovlivní pouze běh daného uzlu, nemusí ovlivnit celý systém
- složitost kódu je menší díky takovému členění
- implementační detaily jsou schovány za rozhraní skrze které uzly komunikuji
- implementace uzlu může být jednoduše nahrazena jinou implementací (i v jiném programovacím jazyce)

Zprávy

Zpráva je silně typovaná datová struktura složená z atributů [8]. Podporovány jsou základní typy (int, float, boolean, atd.) stejně jako jejich pole. Zpráva může obsahovat další vnořené struktury. Každý balíček, který pracuje se zprávami obsahuje adresář msg, ve které jsou soubory s příponou .msg obsahující informace o struktuře jednotlivých zpráv. Tyto struktury pak můžou být implementovány jak v jazyce Python tak v C++.

ROS podporuje zasílání zpráv protokoly TCP/IP i UDP. Přenos protokolem TCP/IP je výchozím nastavením ROS systému a podpora tohoto přístupu je vyžadována po klientských knihovnách. Data jsou v tomto případě posílána přes trvalé spojení. Přenos zpráv protokolem UDP, kdy jsou zprávy děleny do UDP packetů, je nyní podporováno pouze knihovnou pro jazyk C++.

Témata

Témata jsou pojmenované sběrnice, přes které si uzly posílají zprávy [8]. Každé téma je silně typované typem zprávy, která je k tématu publikována a uzly smějí z daného tématu přijímat pouze zprávy tohoto jednoho typu. Uzly generující data publikují zprávy k danému tématu a uzly, které mají zájem o tato data, se přihlašují k odběru zpráv k danému tématu. Tato vazba je typu M:N, což znamená, že téma může mít vícero uzlů publikující zprávy i uzlů tyto zprávy přijímající. ROS také poskytuje rozhraní pro sběr statistik k tématům. Dokáže sbírat statistiky o počtu zpráv, časovém intervalu mezi jednotlivými zprávami, počtu zahozených zpráv nebo velikosti provozu v bajtech.

Služby

Služby (anglicky Services) jsou vhodným prostředkem pro volání vzdálených procedur (anglicky remote procedure call, zkratkou RPC) [8]. Každá služba je definována dvojicí typu zpráv. Jeden typ pro dotaz a druhý pro odpověď. Uzel nabízí službu pod určitým názvem a klient (jiný uzel) může tuto službu volat a čekat na její odpověď. Služby jsou podobně jako zprávy definovány ve speciálních souborem s příponou .srv.

2.6 Framework Qt

Qt je framework pro tvorbu uživatelských rozhraní napsaný v jazyce C++ [7]. Existuje i implementace pro jazyk Python PyQt. Používá preprocesor MOC (Meta-Object Compiler), který před samotným překladem přepíše kód napsaný v rozšířeném C++ pro knihovnu Qt do standardního kódu jazyka C++.

Framework používá vlastní kompilátor pro překlad aplikací qmake a je dodávaný společně s vlastním vývojovým prostředím Qt Creator, který nabízí pohodlnou manipulaci s projekty v programovacím jazyce C++ a deklarativním jazyce QML. V prostředí QtCreator lze jednoduše vytvářet a skládat uživatelské rozhraní pomocí widgetů. Qt Creator se používá také pro překlad, spouštění a ladění aplikací napsaných v jazyce C++.

Kapitola 3

Návrh uživatelského rozhraní

Zatímco předchozí kapitoly se zabývaly teoretickou stránkou a úvodem do problematiky, tato kapitola se bude zabývat především definicí problémů, které má táto práce za cíl řešit a návrhu řešení těchto problému. V této kapitole bude popsán současný stav uživatelského rozhraní a bude navrženo jeho rozšíření pro požadovanou funkcionalitu.

3.1 Použité technologie

ROS umožňuje implementaci jak v jazyce Python, tak v jazyce C++. Celý systém ARTable je naimplementován pomocí programovacího jazyku Python, tudíž jsem se rozhodl v této práci pokračovat a všechna rozšíření provádět taktéž pomocí tohoto jazyka.

Uživatelské rozhraní je v systému ARTable naprogramováno pomocí frameworku Qt. Tento framework zabezpečuje propojení mezi front–endem a aplikační logikou jednotlivých ROS uzlů.

3.2 Současný stav uživatelského rozhraní

Systém ARTable v současné době již obsahuje uživatelské rozhraní promítané na stůl s dotykovou deskou. Jako *největší problém stávajícího uživatelského rozhraní* vidím to, že umožňuje pouze spouštění jednotlivých programů, které musí být dopředu naprogramované skriptem v programovacím jazyce Python[1], který uloží strukturu programu do databáze, ze které je pak tento program načten a pomocí uživatelského rozhraní může být naučen a spuštěn. Bez znalosti programování je tedy nemožné pomocí uživatelského rozhraní měnit strukturu programu, přidávat nové instrukce či bloky nebo vytvořit nový program. Jedinou dostupnou možností je zkopírovat již stávající program a naučit robota provádět jednotlivé instrukce jinak (např. zvednout jiný objekt).

Jelikož cílem společného pracovního prostředí robota s člověkem je to, aby bylo lehce ovladatelné lidmi bez IT vzdělání a znalosti programování, je potřeba aktuální rozhraní o tyto funkce rozšířit.

```
prog = Program()
prog.header.id = 1
prog.header.name = "Trenink - oblast"
```

[1] Oficiální webová stránka programovacího jazyka Python - https://www.python.org/

```
pb = ProgramBlock()
pb.id = 1
pb.name = "Zvedni ze stolu a poloz"
pb.on_success = 1
pb.on_failure = 0
prog.blocks.append(pb)

pb.items.append(polygon_item(1))
pb.items.append(place_item(2, ref_id=[1], on_success=3, on_failure=0))
pb.items.append(item(3, "GetReady", on_success=4, on_failure=0))
pb.items.append(wait_item(4, ref_id=[2], on_success=1, on_failure=0))

art.store_program(prog)
art.program_set_ro(prog.header.id)
```

Výpis 3.1: Ukázka skriptu v jazyce Python, který vytvoří jednoduchý program pro ARTable a uloží jej do databáze

Jednou z možností, jak nahradit původní skripty pro tvorbu programů by mohla být externí aplikace (např. webová), skrze kterou by bylo možné programy vytvářet a ukládat do databáze, ze které by je pak bylo možné přečíst a naučit. Nevýhody tohoto přístupu jsou dle mého:

- Nutnost používat druhé prostředí pro tvorbu programu (webový prohlížeč, mobilní aplikaci, ...).
- Není možné při vytváření programu pracovat s aktuálním rozložením objektů na stole a program je potřeba vždy po vytvoření ještě naučit.

Proto jsem se rozhodl pro rozšíření aktuálního rozhraní na dotykovém stole, což přináší výhody jako:

- Využití společného prostředí pro tvorbu programů, učení i spouštění.
- Učení instrukci přímo při jejich vkládání do programu, tudíž urychlení procesu tvorby programu.

3.3 Popis uživatelského rozhraní a návrh jeho úprav

Uživatelské rozhraní je tvořeno jedním hlavním blokem, který lze po stole libovolně přesouvat a poté obrysy objektů (jako jsou různé dřevěné kostky nebo krabice), které jsou detekovány na stole.

V následujících odstavcích budou popsány jednotlivé části rozhraní a navrženy úpravy jednotlivých částí.

Výpis programů

Výpis programů je první, co uživatel vidí po spuštění aplikace. Tento výpis je načten z databáze, kterou obsluhuje ROS balíček `art_db`. Pod tímto výpisem jsou tlačítka, která umožňují uživateli jednotlivé programy spouštět (jsou-li naučené), vytvořit jejich kopii nebo spustit jejich vizualizaci v brýlích Microsoft Hololens.

Tento výpis neobsahuje možnost přidávat či odebírat programy, což bude hlavní úpravou této části aplikace. Přidání nového programu vytvoří program s jedním prázdným blokem a uloží jej do databáze, takže bude moct být editován i po opětovném spuštění. Odebrání programu daný program odebere z databáze. Bude tak muset být znovu vytvořen nebo nahrán do databáze původním způsobem pomocí skriptu v jazyce Python. Obě tyto nové funkce budou realizovány rozšířením lišty pod výpisem o tlačítka přidat a odebrat.

Obrázek 3.1: Původní verze výpisu programů. Rozšířena bude spodní ovládací lišta.

Výpis bloků programu

Výpis bloků programu se zobrazí po kliknutí na editaci programu. Blok je opět barevně označen dle toho, jestli jsou všechny instrukce v něm naučeny nebo ne. Podobně jako u celých programů zde zcela chybí možnost přidat nebo odebrat blok. Třetí funkcionalitou, o kterou bude tato část rozšířena, je možnost měnit pořadí bloků. Program je po spuštění vykonáván lineárně po jednotlivých blocích. Každá manipulace s bloky (přidání, odebrání, přesunutí) je ihned uložena do databáze. V této části bude stejně jako v předchozí rozšířena ovládací lišta o tlačítka přidání a odebrání bloku a šipky budou sloužit také k přesouvání bloků.

Obrázek 3.2: Původní verze výpisu bloků programu. Rozšířena bude opět spodní ovládací lišta.

Výpis instrukcí v bloku

Výpis instrukcí v bloku se zobrazí po kliknutí na editaci bloku. Po vytvoření nového programu je tento výpis prázdný.

V systému ARTable každá instrukce může skončit buď úspěšně či neúspěšně a každá instrukce má nastavený skok na instrukci, která se má provést při úspěchu a skok na instrukci, která se má provést při neúspěchu. Tak lze dosáhnout nelinearity programu (cyklů, podmínek, apod.).

V současné podobě tento výpis slouží k zobrazení instrukcí v bloku a učení jednotlivých instrukcí. Instrukce, která je nahrána z databáze má definovaný pouze svůj typ. Každý typ instrukce potřebuje poté specifikaci (naučení) jednotlivých parametrů.

Opět neexistuje funkcionalita pro přidávání a odebírání instrukcí a momentálně není ani možné měnit strukturu programu (nastavovat jednotlivým instrukcím skoky po úspěšném či neúspěšném ukončení).

Obrázek 3.3: Původní verze výpisu instrukcí v bloku. Rozšířena bude spodní ovládací lišta přidání možnosti vyvolat u objektů kontextové menu, které vloží naučenou instrukci do programu.

Největší motivací rozšíření uživatelského rozhraní je umožnění uživateli co nejjednodušeji a nejrychleji vytvářet a editovat strukturu programu. I v této části aplikace bude rozšířena spodní lišta o tlačítka přidání a odebrání instrukce. *Přidávat instrukce* bude možné dvěma způsoby:

Po kliku na již zmíněné tlačítko přidat instrukci ve spodní liště se zobrazí výběr všech instrukcí, ze kterých uživatel vybere instrukci, která se uloží do programu jako neuložená. Uživatel poté musí systému nadefinovat její parametry (např. objekt, který se má zvednout, místo, kam se má položit atd). Ideální by ovšem bylo, kdyby se instrukce do programu vložila přímo naučená a bylo možno program ihned spustit.

Tudíž pro instrukce, které využívají nějaké objekty, jsem navrhl druhý způsob jejich možného vkládání do programu. Při kliknutí na obrys objektu se zobrazí kontextové menu (v závislosti na typu objektu) a když uživatel vybere instrukci z tohoto menu, vloží se do programu již naučená a připravená ke spuštění.

Kapitola 4

Implementace

Celá práce je implementována pro platformu ARTable s využitím frameworků Robot Operating System a Qt pro jazyk Python. Jako databázový systém pro uložení programů je použitý systém MongoDB a pro překlad ROS balíčků systém catkin využívající CMake.

4.1 Analýza existujících ROS balíků

Systém ARTable je složen z více než 20 ROS balíků. V této práci budou rozšířeny převážně balíky `art_projected_gui` a `art_helpers`. Na následujících řádcích popíšu jejich úlohy v systému a také to, jak byly oba balíky rozšířeny.

Balík `art_projected_gui`

V balíku `art_projected_gui` je naimplementováno uživatelské rozhraní pomocí knihovny Qt. Toto uživatelské rozhraní je upraveno pro projekci na dotykový stůl místo klasického zobrazení na monitor počítače, na kterém běží.

Třída `UICore` obstarává vytvoření scény (instance třídy `QGraphicsScene`) pro vkládání jednotlivých prvků rozhraní a vytvoření okna (instance třídy `QGraphicsView`) pro spuštění uživatelského rozhraní na monitoru počítače, ne na stole. Na to je potřeba při vývoji aplikace myslet a případně funkcionalitu upravit pro obě rozhraní (monitor, dotykový stůl), protože například přijímání vstupu od uživatele se výrazně liší (klik myši vs. dotek stolu). Třída `UICore` se taktéž stará o zobrazování obrysů detekovaných objektů na stole.

Třída `UICoreRos` obsahuje převážnou část aplikační logiky celého rozhraní. Třída využívá konečný automat a přidává věci spojené s robotickým operačním systémem. Po spuštění se nejprve provede kalibrace projektoru a poté dotykového stolu stisknutím 18 bodů, které se zobrazí na stole. Pokud je systém zkalibrován, zobrazí se výpis programů a čeká se na uživatelovu akci. Třída na základě aktualního stavu systému, který je uložen v instanci třídy `InterfaceStateManager`, zobrazuje uživateli jednotlivá okna a hlášky informující uživatele o tom, jaký je stav systému a jaká akce se od něj právě očekává. Jednotlivé ovládací prvky jsou reprezentovány jako instance podtříd třídy `Item`. Pokud má být daný prvek vložen do scény, je vytvořena instance této třídy, nastavena poloha ovládacího prvku a zaregistrovány callback metody, které jsou implementovány ve třídě `UICoreRos` a jsou volány poté, co uživatel provede nějakou akci (výběr programu, editace, apod.).

Balík `art_helpers`

Balík `art_helpers` obsahuje implementaci tříd, které obsahují funkce pro komunikaci mezi aplikační logikou obsaženou v balíku `art_projected_gui` a jinými částmi systému.

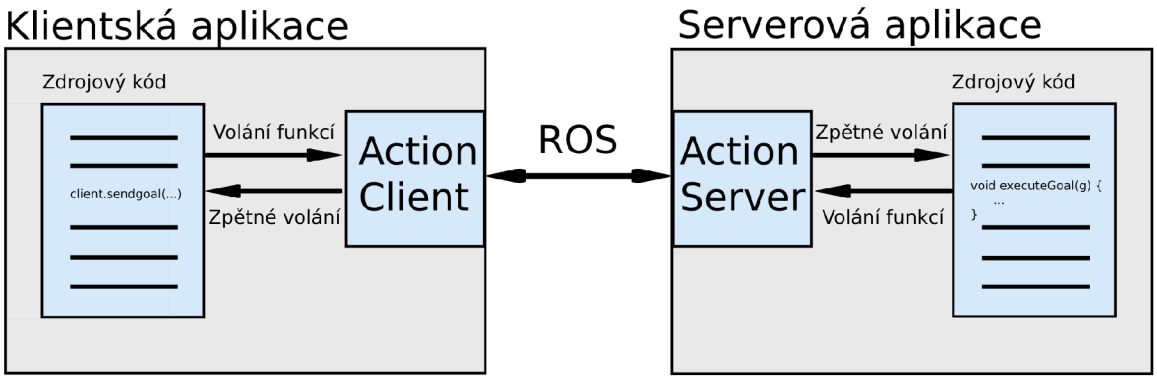
Třída `ProgramHelper` obsahuje funkce pro načítání a ukládání struktury programu do databáze a pro práci s touto strukturou. Důležitou funkcí je funkce `load`, která pro dané id programu načte z databáze jeho strukturu a uloží ji do proměnné `cache`. Ta obsahuje pole bloků, z nich každý blok je pole instrukcí. Ke každému bloku a instrukci jsou uloženy informace o daném objektu (ID, typ, ...).

Třída `InstructionHelper` implementuje metody, které načítají dostupné instrukce z konfiguračních souborů. V této třídě byla naimplementována nová metoda `get_instruction_msgs`, která pro daný typ instrukce vrátí její objektovou reprezentaci z balíku `art_utils`. Tento objekt lze pak vložit do struktury programu a uložit do databáze pomocí funkce `store_program` dostupné v balíku `art_db`.

Rozhraní `actionlib`

V této sekci bude popsáno rozhraní `actionlib`, které sysém ARTable používá pro komunikaci mezi uživatelským rozhraním a robotem PR2 především pro učení jednotlivých instrukcí. Informace jsou převzaty z oficiální dokumentace ROS [10].

Komunikace skrze toto rozhraní je založena na architektuře klient-server. `ActionClient` a `ActionServer` komunikují pomocí protokolu `ROS Action Protocol`. Pro komunikaci je nutné definovat `action`, která se skládá ze tří částí: `Goal`, `Feedback`, `Result`. Tyto akce jsou definovány v souborech s příponou `.action`. `ActionClient` tedy zasílá zpravy skrze rozhraní `actionlib` a implementuje callback metody, které jsou volány po přijetí odpovědi od `ActionServer`. Ten naopak implementuje callback funkce po přijmutí zprávy skrze rozhraní `actionlib` a skrze toto rozhraní posílá zpět odpovědi. Celý proces lze vidět na obrázku 4.1.



Obrázek 4.1: Komunikace v rozhraní `actionlib`

V systému ARTable je server pro zpracovávání dotazů týkajících se učení jednotlivých instrukcí naimplementován v balíku `art_brain` a dotazy jsou posílány ze třídy `UICoreRos`, která poté na základě odpovědí na tyto dotazy upravuje uživatelské rozhraní. Tato komunikace probíhá pomocí akce nadefinované v souboru `LearningRequest.action`. Ve výpise 4.1 lze vidět obsah tohoto souboru.
Request je reprezentován celočíselnou hodnotou request, která může nabývat hodnot od 0 do 3 v závislosti na tom, v jaké fázi učení se robot nachází.

Response je tvořena pravdivostní hodnotou success, která udává to, jestli byl dotaz úspěšný a případně nějakou zprávou informující o chybě.
Feedback je reprezentován opět celočíselnou hodnotou progress, která vrací úplnost zpracování v procentech.

```
uint16 request

uint16 GET_READY=0 # prepare robot for learning current item
uint16 GET_READY_WITHOUT_ROBOT=1 # prepare robot for learning current item
uint16 EXECUTE_ITEM=2 # test programmed item
uint16 DONE=3 # learning
---
bool success
string message
---
uint16 progress # percents
```

Výpis 4.1: Definice akce pro rozhraní `actionlib` v souboru LearningRequest.action.

Pro naučení instrukce jsou třeba dva dotazy na server.

První dotaz má nastavený parametr request na hodnotu 0 (`GET_READY`). Server po přijmutí takového dotazu spustí učící funkci aktuální instrukce. Ta má na starost jak změnu uživatelského rozhraní, tak uložení zadaných parametrů. Kontextové menu, která se zobrazují po kliknutí na objekt, obsahují instrukce, které pro své naučení vyžadují specifikaci objektu. U těchto instrukcí je po spuštění učení volána funkce `object_selected`, která uloží do instrukce typ objektu, u kterého bylo kontextové menu vyvoláno. Speciálním případem je instrukce položit na stůl. Kontextové menu je vyvoláno kliknutím na stůl. Pro naučení instrukce je zapotřebí specifikovat objekt a polohu, kam má být položen. Objekt je specifikován tak, že instrukce referencuje nejbližší instrukci, která zvedá nebo podává nějaký objekt (instrukce `Pick from feeder` nebo `Pick from polygon`). Pozice je do instrukce vložena voláním funkce `place_pose_changed`, které jsou předány souřdanice, na kterých bylo vyvoláno kontextové menu.

Druhý dotaz má nastavený parametr request na hodnotu 3 (`DONE`). Tento dotaz ukončuje proces učení.

4.2 Rozšíření balíku `art_projected_gui`

Balík `art_projected_gui` byl rozšířenou o funkce upravující strukturu aktuálního programu a ukládající tuto pozměněnou verzi zpět do databáze.

Rozšíření třídy `UICoreRos`

Třida `UICoreRos` musela být rozšířena o callback funkce pro tlačítka v jednotlivých ovládacích prvcích. Většina těchto funkcí funguje tak, že po jejím zavolání je volána příslušná funkce ze třídy `ProgramHelper`, která upraví aktuálně načtený program a v případě úspěchu je změna v programu zobrazena v uživatelském rozhraní. V případě neúspěchu je vypsaná chybová hláška pomocí notifikační lišty systému ARTable, kterou lze vidět na obrázku 4.2.

Obrázek 4.2: Notifikační lišta systému ARTable, která se zobrazuje ve spodní části stolu a předává uživateli informace o stavu systému

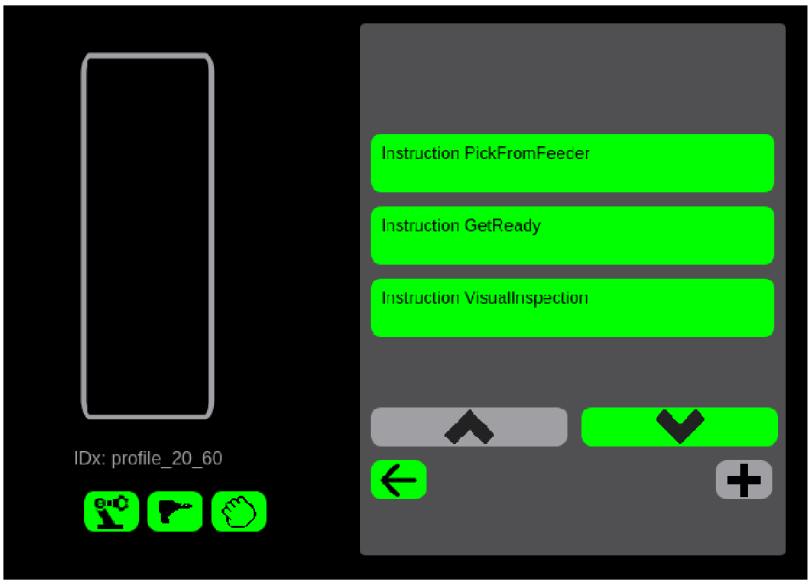
Rozšíření tříd `ProgramListItem` a `ProgramItem`

Třídy `ProgramListItem` a `ProgramItem` reprezentují okna aplikace popsána v kapitole 3 (výpis programů, bloků a seznamu instrukcí). Jejich instance jsou vytvářeny v metodách třídy `UICoreROS` po adekvátních akcích uživatele. Rozšíření jednotlivých oken byla popsána v kapitole 3. Na obrázcích níže je ukázáno to, jak jednotlivá okna vypadají po tom, co byla nová funkcionalita implementována.

Obrázek 4.3: Výpis programů s přidanými tlačítky pro vytvoření a smazání programu

Obrázek 4.4: Výpis bloků programu s přidanými tlačítky pro vytvoření a smazání programu a možností přesouvat bloky

Obrázek 4.5: Výpis instrukcí v bloků rozšířený o možnost přidání, odebrání instrukce, nastavení skoků pro instrukce a vizualizace průběhu programu



Obrázek 4.6: Výběr nové instrukce: Seznam všech instrukcí nebo kontextové menu u objektu

Třída `SelectInstructionItem`

Třída `SelectInstructionItem` je stejně jako ostatní třídy reprezentující jednotlivé prvky vkládané do scény zděděná ze třídy `Item`, která dědí ze třídy `QGraphicsItem` knihovny Qt. Ve třídě `Item` jsou naimplementovány funkce pro pozicování jednotlivých objektů nebo pro volání callback funkcí po kliknutí na daný prvek. Třída `SelectInstructionItem` zobrazí po inicializaci buď seznam všech instrukcí nebo kontextové menu, jak lze vidět na obrázku 4.6. Po výběru dané instrukce je zavolána callback funkce ze třídy `UICoreRos`, která se postará o vložení nové instrukce do programu (metoda `add_item` třídy `ProgramHelper`). V případě kontextového menu je poté provedeno taktéž naučení instrukce.

Učení instrukce probíhá pomocí ROS rozhraní `actionlib` a učících funkcí jednotlivých instrukcí z balíku `art_instructions`.

4.3 Rozšíření balíku `art_helpers`

Balík `art_helpers` byl rozšířenou o funkce upravující strukturu aktuálního programu a ukládající tuto pozměněnou verzi zpět do databáze.

Rozšíření třídy `ProgramHelper`

Třída `ProgramHelper` neobsahovala funkce pro jakékoliv úpravy struktury aktuálně načteného programu. Proto byla v této práci rozšířena o funkce `add_item`, `delete_item`, `add_block`, `delete_block`, `move_block`, `move_item` a další, které umožňují měnit strukturu programu v proměnné `cache`. Všechny funkce upraví již načtenou strukturu programu a poté uloží program do databáze, ze které jej znova načtou pomocí funkce `load`, aby byl v aktualizované podobě přístupný aplikační logice. Z funkce `load` musely být odstraněny některé kontroly, protože tato funkce kontrolovala to, jestli struktura programu nebyla změněna.

Kapitola 5

Testování

Testování a ladění programů probíhalo ve třech krocích:

- Testování pomocí debuggovacího okna na stolním počítači.
- Testování v laboratoři na dotykovém stole.
- Uživatelské testování pomocí flexibilního rozhovoru.

5.1 Testování na stolním počítači

Pro urychlení vývoje je v systméu ARTable naimplementována sada metod, umožňující vývoj na stolním počítači bez robota či dotykového stolu. Debuggovací okno se zobrazí pomocí instance třídy `QGraphicsView` a pro ovládání rozhraní slouží myš. Aby bylo možné celý systém spustit na klasickém počítači, je potřeba spuštění ROS uzlů, které simulují činnost robota. Tyto uzly jsou již v systému ARTable naimplementovány. Tento přístup má jednu velkou nevýhodu, kterou je to, že prostředí, ve kterém aplikace běží, se výrazně liší od prostředí reálného. Uzly, které simulují činnost robota, málokdy havarují, a tak je těžké odhalit veškeré problémy, které mohou nastat. Taktéž interakce mezi uživatelem a aplikací se výrazně liší. Na počítači uživatel kliká klasickou myší a tyto akce zpracovává okno, které je zobrazeno. Toto debuggovací okno ale při spuštění na stole vůbec neexistuje, tudíž je nutné zpracovávat uživatelovy interakce se systémem úplně jiným způsobem.

5.2 Testování v laboratoři

Jak bylo naznačeno v předchozí sekci, druhá fáze testování odhalila několik problémů, které se v první fázi vůbec nevyskytly. Nejzávažnější problém, který se v této fázi vyskytl, bylo to, že pro instrukci „položit na stůl“ byla naimplementována možnost vložení této instrukce do programu taktéž pomocí kontextového menu. To bylo vyvoláno klikem na jakékoliv místo na stole. Při vývoji na stolním počítači jsem počítal se zobrazením okna, na jehož kliknutí jsem zaregistroval callback funkci. Po spuštění v laboratoři se samozřejmě toto kontextové menu nezobrazovalo, jelikož se nezobrazovalo žádné okno, na které by šlo kliknout. Bylo potřeba zvolit jiný přístup a původní zachovat.

5.3 Uživatelské testování

Po otestování funkčnosti na dotykovém stole společně s robotem PR2 bylo provedeno uživatelské testování s celkem čtyřmi uživateli. Jejich demografické rozložení lze vidět v tabulce 5.1.

Účastník	Pohlaví	Věk	Vzdělání	Zkušenosti s roboty
A	Ž	50	vysokoškolské (humanitní)	žádné
B	M	23	středoškolské (technické)	pracoval na pracovišti s roboty
C	M	41	vysokoškolské (technické)	žádné
D	Ž	17	základní	žádné

Tabulka 5.1: Demografické rozložení účastníků testování

Testovací protokol

Testování s každým účastníkem probíhalo dle předem vytvořeného protokolu ve čtyřech krocích:

Představení – Každý účastník byl nejprve seznámen se systémem ARTable. Především tím, jaké instrukce robot PR2 podporuje a z čeho jsou složeny programy pro něj.

Trénink – Každému účastníkovi byla vysvětlena práce s uživatelským rozhraním, práce s bloky a instrukcemi. Každý uživatel si vyzkoušel naučit robota program, který byl v systému již vytvořen, ale nebyl naučen. Tento program byl podobný hlavnímu úkolu a uživatelé tak mohli porovnat původní a nový stav uživatelského rozhraní a postupu učení jednotlivých instrukcí.

Hlavní úkol – Hlavním úkolem pro každého účastníka bylo sestavit program a nastavit všechny jeho parametry tak, aby fungoval následovně:

1. Robot zvedne vybraný objekt z dotykového stolu.
2. Robot provede vizuální inspekci.
3. Na základě úspěchu či neúspěchu provedené inspekce vloží objekt do jedné z krabic na stole.
4. Robot vrátí ruce do původní polohy a poté opakuje činnost.

Tento program byl zvolen proto, že obsahuje jak vkládání instrukcí, tak nastavení skoku instrukce (úspěch či neúspěch vizuální inspekce), tak vložení instrukce mimo kontextové menu (návrat do původní polohy).

Zpětná vazba – při plnění úkolu byl jednotlivým uživatelům měřen čas a každý uživatel byl po dokončení úkolu podroben krátkému flexibilnímu rozhovoru, ve kterém byl tázán nejprve na věci, které se na základě pozorování zdály pro danou osobu problematické. Poté byl každý účastník vyzván ke shrnutí toho, jak se mu s aplikací pracovalo.

Shrnutí výsledků testování

Tabulka 5.2 ukazuje to, kolik času jednotliví účastníci potřebovali ke splnění zadaného úkolu. Za splněný úkol se považovalo to, když byly všechny instrukce vloženy do programu, naučené a nastaveny správně všechny skoky mezi instrukcemi.

Účastník	Čas na dokončení úkolu
A	8 minut a 20 sekund
B	2 minuty a 30 sekund
C	4 minuty a 40 sekund
D	7 minut a 50 sekund

Tabulka 5.2: Čas potřebný ke splnění úkolu

Jak lze vidět v tabulce 5.2, všechny osoby dokázaly úkol dokončit, ale rozdíl mezi nejrychlejším a nejpomalejším časem byl více než trojnásobný. Tento rozdíl přikládám tomu, že člověk s nejrychlejším čase již měl nějaké zkušenosti s roboty i s vizuálním programováním, takže učení bylo rychlejší než u osoby ve středním věku s nulovými zkušenostmi s podobnými technologiemi. Založení nového programu a otevření seznamu bloků nedělalo uživatelům problém. Uživatelé A a D strávili větší čas v seznamu bloků, protože bud zkoušeli přidávat a odebírat bloky (osoba A), nebo zkoušeli klikat na obrysy objektů jako osoba D, což ale k překvapení nevedlo k otevření kontextového menu, protože to se otevírá až v editaci konkrétního bloku. Po chvilce přemýšlení nebo menší radě se všem podařilo otevřít prázdný seznam instrukcí prvního bloku.

V seznamu instrukcí většina osob správně vložila instrukci zvednutí objektu přímo z kontextového menu konkrétního objektu. Instrukci vizuální inspekce již neměli všichni spojenou přímo s objektem a uživatelé A a C ji hledali v seznamu všech instrukcí, což trvalo o něco déle než opětovné vyvolání kontextového menu. I pro instrukci vložení do krabice volili uživatelé možnost kontextového menu, ale osoba A poté přesunula krabici na druhé místo a spustila editaci té samé instrukce, čímž přepsala původní parametry. Až poté si podala druhou krabici a vložila druhou instrukci. Instrukci `Get ready` pro navrácení do původní polohy našli všichni účastníci v seznamu všech instrukcí.

Poslední částí bylo nastavení správných skoků mezi jednotlivými instrukcemi. Účastník B, který měl nejlepší čas, vkládal instrukce ve správném pořadí, a jelikož již měl i zkušenosti s programováním, nedělalo mu problém nastavit pouze skoky pro instrukci vizuální inspekce a skok po položení objektu do krabice. Ostatní měli s nastavením skoků menší či větší problémy. Největším problémem se zdálo býti to, že instrukce se nevešly do výpisu všechny najednou, a tudíž pro osoby bylo těžké představit si průběh programu. Osoba D nevkládala do programu instrukce podle toho, jak měly jít po sobě, a tak měla víc práce s přeskládáním jednotlivých instrukcí a horší orientaci podle ID instrukcí, které se přiřazují podle pořadí, v jakém byly do programu vloženy. Dalším problémem bylo občas to, že uživatelé omylem přesunuli instrukci, kterou měli označenou, i když se chtěli pouze posunout v seznamu dolů. Museli pak instrukci vrátit na původní místo, odznačit ji a teprve poté se posunout níže.

Celkově se uživatelé relativně rychle naučili základní práci se systémem a je poměrně vysoký předpoklad, že po delším užíváním by se jejich práce s ním stala ještě efektivnější. Osoby A, C i D uvedly, že se ve chvílích, kdy nevěděli jak dál postupovat, báli udělat nějakou akci, protože se obávali, že nebude správná a systém tak „nerozbili“. V těchto chvílích by se jim po ruce hodila zkušenější osoba, která by je ujistila ve správnosti dalšího kroku. Dále všichni uživatelé uvedli, že aspoň jednou se omylem dotkli plochy tak, že vyvolali nechtěnou akci, kterou pak museli vracet.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo rozšířit systém ARTable o možnost vytvářet nové nebo editovat stávající programy na dotykovém stole a nahradit tak skripty, které vytvářejí programy doposud. Návrhu a implementaci předcházelo studium systému ARTable (především jeho částí souvisejících s uživatelským rozhraním) a frameworků Qt a ROS. Výsledné řešení obsahuje požadovaná rozšíření.

Rozšíření bylo vyvinuto jak pro dotykový stůl systému ARTable, tak pro debuggovací okno pro účely vývoje na stolním počítači. Řešení bylo po odladění otestováno na malé skupině uživatelů, kteří se s aplikací setkali poprvé. Během zhruba 30–45 minut se všichni naučili základní práci se systémem a poté byli schopni vytvořit jednoduchý program složený z několika instrukcí. Oproti původnímu stavu účastníci testování oceňovali především to, že můžou instrukce přidávat do programu již naučené (nemusí tedy znovu vybírat objekty k instrukcím).

Do budoucna by se dala dlé mého zlepšit především vizualizace toku programu. Momentálně jsou všechny instrukce řazeny pod sebe i když pořadí ve výpisu nemusí kvůli skokům jednotlivých instrukcí vůbec odpovídat. Lepší řešení by bylo například zobrazování instrukcí ve stromu, ze kterého by bylo skoky po úspěchu a neúspěchu lépe viditelné.

Literatura

[1] Azuma, R. T.: A Survey of Augmented Reality. [Online; navštíveno 03. 01. 2019].
URL https://www.mitpressjournals.org/doi/pdfplus/10.1162/pres.1997.6.4.355

[2] Billinghurst Mark, P. I., Hirokazu Kato: Tangible augmented reality. SIGGRAPH, 2008, doi:10.1145/1508044.1508051.

[3] Carmigniani, B., J.; Furht: Augmented reality: An Overview. [Online; navštíveno 03. 01. 2019].
URL http://pire.fiu.edu/publications/Augmented.pdf

[4] Durrant-Whyte Hugh, B. T.: Simultaneous localization and mapping: part I. June 2006: s. 99–110, ISSN 1070-9932, doi:10.1109/MRA.2006.1638022.

[5] Durrant-Whyte Hugh, B. T.: Simultaneous localization and mapping: part II. Aug 2006: s. 108–117, ISSN 1070-9932, doi:10.1109/MRA.2006.1678144.

[6] ISO Central Secretary: Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. Standard ISO 9241-11:2018, International Organization for Standardization, 2018.

[7] Martin, C.: *Grafická uživatelská rozhraní v Qt a C++*. Computer Press, 2013, ISBN 978-80-251-4124-3.

[8] Morgan., Q.; Ken., C.; Gerke; aj.: ROS: an open-source Robot Operating System. 2009.

[9] Rosenberg, L. B.: Virtual fixtures: Perceptual tools for telerobotic manipulation. Sep 1993: s. 76–82, doi:10.1109/VRAIS.1993.380795.

[10] Saito, I.: Actionlib documentation. [Online; navštíveno 03. 05. 2019].
URL http://wiki.ros.org/actionlib

[11] Schmalstieg, D.; Höllerer, T.: *Augmented reality: Principles and practice*. IEEE, 2017, ISBN 978-1-5090-6647-6, doi:10.1109/VR.2017.7892358.

[12] Shaer, O.; Hornecker, E.: *Tangible User Interfaces: Past, Present and Future Directions*. Now Foundations and Trends, 2010, ISBN 9781601983282.

[13] Stone, D.; Woodroffe, C. J.: *User interface design and evaluation*. Boston: Morgan Kaufmann Publishers, 2005, ISBN 0-12-088436-4.

Přílohy

Příloha A

Obsah přiloženého paměťového média

- *zdrojove_kody.zip* – soubor se zdrojovými kódy balíků `art_helpers`, `art_projected_gui` a `art_instructions`. Upravené metody a třídy jsou řádně okomentovány.
- *bakalarska_prace.zip* – soubor se zdrojovými soubory k této bakalářské práci
- *bakalarska_prace.pdf* – soubor obsahující tuto bakalářskou práci
- *plakat.pdf* – plakát shrnující téma a výsledek bakalářské práce
- *README.md* – soubor obsahující základní informace a popis obsahu paměťového média