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

PROGRAMOVÁNÍ ROBOTICKÝCH AKCÍ V ROZŠÍŘENÉ REALITĚ

ROBOT PROGRAMMING IN AUGMENTED REALITY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DAVID SABELA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL KAPINUS, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Sabela David, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Programování robotických akcí v rozšířené realitě**
Robot Programming in Augmented Reality
Kategorie: Uživatelská rozhraní
Zadání:

1. Prostudujte koncept rozšířené reality a její využití při návrhu uživatelských rozhraní. Seznamte se s platformami iOS, ARKit a ARFoundation.
2. Seznamte se s metodami využívanými k programování robotů, jako je například vizuální programování, programování pomocí demonstrace a podobně.
3. Vyberte vhodné metody a nástroje a navrhněte uživatelské rozhraní, které umožní pomocí rozšířené reality na platformě iPad programovat vybrané robotické úlohy a vizualizovat průběh jejich simulovaného vykonávání.
4. Navržené rozhraní implementujte.
5. Proveďte experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
6. Vytvořte stručný plakát nebo video prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2, 3 a rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kapinus Michal, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 21. května 2020

Abstrakt

Cílem práce bylo vytvořit aplikaci, která umožňuje prostřednictvím rozšířené reality programovat robotické akce. Aplikace je demonstrativního charakteru a je zde snaha o intuitivní ovládání a dobrou integraci rozšířené reality. Tato experimentální aplikace umožňuje pomocí vizuálních instrukcí, podmínek a spojů uživateli tvořit program pro robota a také jej otestovat pomocí vizualizace průchodu programem. Aplikace je implementována pomocí Unity3D a technologie AR Foundation. Bylo provedeno testování skupinou dobrovolníků. Zpětná vazba od účastníků testování byla pozitivní.

Abstract

The aim of this master's thesis was to develop an application, which would allow its users to program robotic actions with the help the augmented reality. The application is of demonstrative character and is made with the goal of intuitive handling and good integration of the augmented reality. This experimental application enables users to design a program for a robot using visual instructions, conditions and links and to test it by visualizing the passage through the program. The application is implemented with the use of Unity3D and the AR Foundation technology. The result was tested by a group of volunteers, whose feedback can be considered generally positive.

Klíčová slova

Rozšířená realita, programování robotů, Unity, ARKit, AR Foundation, HW pro AR, Markerless, Marker, iPad.

Keywords

Augmented reality, Robot Programming, Unity, ARKit, AR Foundation, HW for AR, Markerless, Marker, iPad.

Citace

SABELA, David. *Programování robotických akcí v rozšířené realitě*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Kapinus, Ph.D.

Programování robotických akcí v rozšířené realitě

Prohlášení

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

.....
David Sabela
3. června 2020

Poděkování

Děkuji panu Ing. Michalu Kapinusovi, Ph.D za odborné vedení, trpělivost a ochotu, kterou mi v průběhu práce věnoval.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Programování robotů se z velké části posunulo od nízkoúrovňového kódování k intuitivnějším metodám. Tento krok byl částečně podpořen touhou usnadnit práci operátorům méně či vůbec neznalých programování, kterým činily starší metody problémy. Dnes populární možností jak usnadnit a zlepšit práci operátorům je použití rozšířené reality.

Rozšířená realita se snaží usnadnit život uživatele tak, že přináší dodatečné virtuální informace do prostředí, v němž se uživatel vyskytuje, které zlepšují a zjednodušují rozhodování či jiné aspekty v daný čas. Pro programování robotů se využívá převážně vizuální rozšíření reality, a to prostřednictvím chytrých brýlí nebo mobilních zařízení. Chytré brýle jsou dražší variantou, ale jejich prostřednictvím má uživatel hlubší zážitek z rozšířené reality. Využití rozšířené reality pro programování robotů sebou přináší výhody, mezi nimi jsou možnost programování přímo na místě, kde se robot nachází a také možnost simulování akcí robota na místech reálného provádění těchto akcí. Tím odpadá nutnost použití simulátoru prostředí nebo tzv. programování naslepo.

Cílem mé práce je vytvoření aplikace pro iPad s využitím rozšířené reality, která umožňuje programování robotických akcí. K dosažení tohoto cíle jsou použity technologie Unity3D a AR Foundation.

Rozšířená realita je nyní na vzestupu, možnosti jejího využití jsou velké, a to zejména v oblasti průmyslu. Tato technologie mě zaujala vzhledem k jejímu širokému využití a budoucímu potenciálu, a také protože mám blízký vztah k uživatelskému rozhraní, kterému se aktivně věnuji v profesním životě.

Tato práce je rozdělena do čtyř hlavních částí. První část je věnována teorii, ve které jsou zmíněny některé metody vhodné k zvládnutí výše uvedeného cíle a ukázka již hotových řešení. V další části s názvem Návrh jsou popsány různé verze návrhu uživatelského rozhraní aplikace. Následuje část Implementace, zde je popsán způsob, jakým byl vytvořen prototyp mého řešení a část Testování, která nastiňuje průběh testování a shrnuje jejich výsledky.

Kapitola 2

Rozšířená realita

Kapitola obsahuje přiblížení historie rozšířené reality a vysvětlení co je rozšířená realita a jak se liší od reality virtuální. Kapitola se zaměřuje také na metody a technologie používané v rozšířené realitě.

2.1 Úvod do rozšířené reality

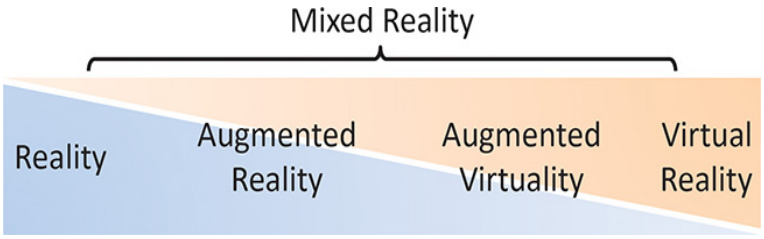
Už pojem rozšířená realita naznačuje, že se nějakým způsobem snaží nadstavit realitu o něco navíc [20]. Rozšířená realita doplňuje do reálného pohledu na svět v reálném čase počítačově vygenerované předměty, obrázky, atd. V dnešní době je možné díky technologiím pracovat s rozšířenou realitou interaktivně. Pokud nahlédneme do počátků rozšířené reality, dostaneme se do 50. let 20. století, kdy kameraman Morton Heilig přišel s myšlenkou kina, které dokáže diváka vtáhnout do děje všemi smysly, a to co možná nejefektivnějším způsobem. Svoji myšlenku pak v roce 1955 zrealizoval a nazval ji kino budoucnosti. V roce 1962 postavil prototyp svého vidění, jmenoval se Sensorama. V roce 1968 Ivan Sutherland vytvořil první systém rozšířené reality využívající optický průhledový displej. Další roky přinesly různé nápady a v roce 1997 vytvořil Ronald Azuma první široce uznávanou definici rozšířené reality: "kombinace reálného a virtuálního prostředí ve 3D, interaktivní v reálném čase".

Rozšířená realita se snaží usnadnit život uživatele tak, že přináší dodatečné virtuální informace do prostředí, v němž se uživatel vyskytuje, které zlepšují a zjednodušují rozhodování či jiné aspekty v daný čas. Rozšířená realita má smysl pro superponování virtuálních objektů, ale také nevizuálních. Potenciálně jde aplikovat na všechny smysly člověka. Pomocí vizuálních podnětů můžeme například u lidí se ztrátou sluchu substituovat zvuk.

Rozdíl AR od virtuální reality (VR) je, že VR člověka ponoří do kompletně virtuálního prostředí, které je uměle vytvořené a k jeho použití je zapotřebí speciálních zařízení, nejčastěji chytrých brýlí.

2.2 Metody a techniky

V rozšířená realitě se používají rozdílné motody počítačového vidění založené převážně na sledovnání obrazu [20]. Tyto metody se obvykle sestávají ze sledování, rekonstrukce a následného rozpoznávání. Rozšířená realita se zakládá na detekci zájmových oblastí na sledovaném videu. Exitují tři typy rozšířené reality, a to založené na visuálních značkách (marker-based), bez vizuálních značek (markerless) a lokalizační (location).



Obrázek 2.1: Kontinuum smíšené reality zachycuje všechny možné kombinace skutečného a virtuálního světa. Převzato z [20].

Lokalizační AR (location AR) spojuje rozšířenou realitu s konkrétním místem. Aplikace na této bázi mohou například uživateli říci, že se nachází na určitém místě a ukázat mu jakým směrem se má vydat nebo zobrazit maximální rychlost při jízdě v automobilu. Asi nejznámější příklad lokalizační AR aplikace je hra Pokemon Go.

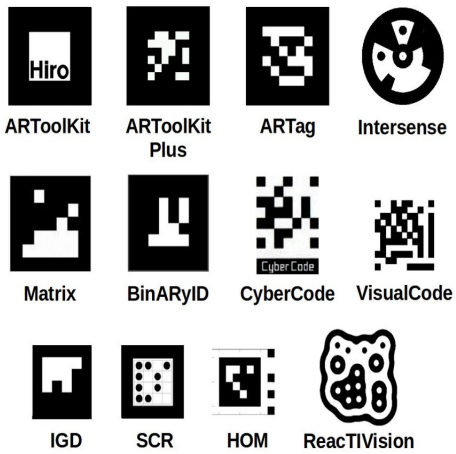
2.2.1 Marker based tracking

AR založené na vizuálních značkách (marker-based) ukotvuje digitální svět ke skutečnému světu prostřednictvím detekce vizuálních značek. Při marker-based AR, je nutné vědět, že uživatel ukazuje kamerou na konkrétní prvek. Proto musí zařízení nejprve rozpoznat, na který prvek se uživatel dívá. Toho lze dosáhnout umístěním výrazného obrázku nebo tvaru [20]. Tento obrázek bude rozpoznán a poté se na tuto pozici může umístit virtuální prvek. Nemusí se zobrazovat přímo na tuto konkrétní pozici, ale značka udává místo ve scéně a virtuální prvek je možné umístit po propočítání pozic jinam. Jako marker může být použit jakýkoliv objekt, ale musí být jednoznačný v rámci scény. Ne všechny objekty jsou stejně vhodné. Asi nejpoužívanější jsou černobílé markery.

Zaznamenávání černo-bílých markerů je velice oblíbené, nenáročné a i s horším video zařízením produkuje použitelné výsledky [1]. Popularita vychází převážně díky jednoduchosti [10]. Výhodou jednoduchých markerů je, že je software snadno rozpozná s velmi malými režijními náklady na zpracování a minimalizuje riziko, že aplikace nebude fungovat, například kvůli nekonzistentnímu okolnímu osvětlení nebo jiným okolním podmínkám. Populární design markeru je např. černý čtvercový obrys dané tloušťky na bílém pozadí s bílým 2D čárovým kódem uvnitř. Díky čárovému kódu může být jediná rodina markerů znovu použita k vyvolání mnoha různých virtuálních objektů změnou kódovaného vzoru. Například dětská kniha může mít na každé stránce vyskakovací AR s použitím stejného tvaru značky, ale čárový kód nasměruje aplikaci, aby v knize zobrazovala pouze objekty relevantní pro danou stránku. Dnes asi nejznámější marker je QR code, který se využívá například v reklamních kampaních.

2.2.2 Markerless based tracking

U markerless rozšířené reality uživatel skenuje a detekuje horizontální a vertikální plochy pomocí kamery tak, aby na ně bylo možné umístit virtuální prvky. Systém analyzuje prostředí, hledá korelace a rozdíly mezi pixely a pak definuje virtuální souřadnice do reálného prostředí. Markerless přístup obvykle vyžaduje lepší kvalitu obrazu a více výpočetních jednotek pro jeho zpracování [20].



Obrázek 2.2: Příklady markerů. Převzato z [7].

Pro hledaní shody obrázku s reálným světem se používají metody Dense matching a Sparse matching. Dense matching hledá shodu pro každý pixel v obrázku, kdežto Sparse matching hledá shodu pro malý počet vybraných zájmových bodů z obrázku.

Před několika lety se začaly výzkumné skupiny více zaměřovat na Sparse matching a to především z důvodu, že se sparse možiny zájmových bodů snadněji vyrábějí, protože je třeba vytvořit pouze digitální reprezentaci některých zájmových bodů, zatímco ostatní fyzické objekty mohou být zanedbány [22]. Dalším aspektem je, že jednotlivé zájmové body se zpracovávají samostatně, tím pádem nelze-li konkrétní bod zájmu nalézt kvůli okluzi nebo změnám osvětlení, samotný sledovací algoritmus není vážně narušen.

Dense matching má jednu zajímavou výhodu oproti Sparse matchingu a to, že může být lepší a robusnější volbou pro použití v obtížně zpracovatelných prostředí, například objekt se špatnou texturou, opakování struktury, reflexní povrch a jiné.

Postup metody obvykle začíná sledováním pomocí detekce. Pozice kamery je určována z odpovídajících zájmových bodů v každém snímku znovu, aniž by se spoléhala na informace o prioritách získané z předchozích snímků [3]. Zájmové body jsou reprezentovány deskriptory, které jsou datovými strukturami navrženými pro rychlé a spolehlivé párování. Deskriptory jsou vytvářeny pro zájmové body nalezené v novém obrazu kamery a jsou k nim přiřazeny v modelu. Typický proces sledování pomocí detekce sparse zájmových bodů sestává z pěti částí:

- Detekce zájmových bodů.
- Vytvoření deskriptorů.
- Párování deskriptorů.
- Perspektive-n-Point rozhodování o pozici kamery.
- Robust pose estimation.

Detekce zájmových bodů

Detekce zájmových bodů je důležitou oblastí výzkumu v oblasti zpracování obrazu a počítačového vidění. Zejména vyhledávání obrázků a kategorizace objektů se silně spoléhají na

detekci zájmových bodů, ze které se vypočítávají lokální deskriptory obrazu pro porovnávání obrázků [20].

Je otázkou, jaké body by měly být zvoleny jako zájmové. Shi a Thomas v roce 1994 se zabývali touto problematikou a pragmaticky stanovili, že zájmový bod by měl být především jednoduše identifikovatelný. V praxi to znamená, že okolí bodu by mělo být vizuálně odlišné, samotný bod by měl být dobře texturovaný a mít vysoký kontrast.

Výběr bodu zájmu by měl být opakovatelný, což znamená, že detekční algoritmus by měl vybrat stejné body zájmu nezávisle na parametrech pozorování, jako je úhel pohledu, osvětlení, rotace, zvětšení a jiné.

Jako detektory zájmových bodů se používají například metody jako Harris Corners, Difference of Gaussian (používána v SIFT[1]) nebo FAST.

Vhodné body zájmu budou obecně tvarovány jako kruhové kuličky nebo rohy. Harris detektor využívá auto-korelaci pro nalezení rohů, problém této metody je, že není invariantní proti změnám v měřítku, takže nefunguje dobře, když se kamera dívá na objekt z nepřímého směru pohledu. Řešením může být použití metody Difference of Gaussian (DOG), která pracuje v měřítkovém prostoru získaném z obrazové pyramidy.

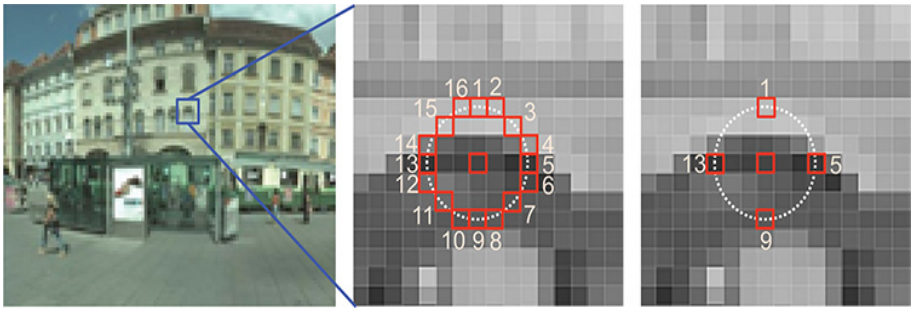
DOG v SIFT algoritmu pracuje tak, že se vytvoří scale-space o několika vrstvách, kde první vrstva odpovídá původnímu obrazu a každá následující je tvořena obrazem s vyšším měřítkem, jedná se o dolní propusť [13]. K tomuto účelu se používá funkce Gaussova a je tedy zkonstruována tzv. Gaussova pyramida viz pravý obrázek 2.3. Scale-space, ve které jsou hledány extrémy (kandidáti na zájmové body), je sestavena z rozdílných obrazů vzniklých rozdílem dvou sousedních obrazů v Gaussově pyramidě viz levý obrázek 2.3.

Obrázek 2.3: Levý obrázek znázorňuje Konstrukci měřítkově nezávislé formy obrazu v případě metody SIFT a pravý obrázek je Gaussova pyramida. Převzato z [13].

Další metodou je FAST detektor optimalizovaný pro rychlost a je velmi vhodný pro aplikace pro zpracování videa v reálném čase a zejména pro ty, které mají omezené výpočetní zdroje, jako je mobilní AR. FAST používá diskretizovaný kruh se středem na kandidátním bodě viz obr. 2.4. Bod je klasifikován jako roh, pokud existuje souvislý oblouk pixelů s dostatečným kontrastem ke středovému pixelu, který pokrývá až tři čtvrtiny kruhu.

Tato metoda má omezenou odolnost proti šumu a rozmazání pohybem. Existují variace FAST9, FAST10, FAST11 a FAST12, které jsou pojmenované podle délky oblouku v pixelech.

[1]SIFT - scale invariant feature transform



Obrázek 2.4: FAST vyhledává sekvenci sousedících pixelů v kruhu, které jsou buď světlejší nebo tmavší nežli střed. Převzato z [20].

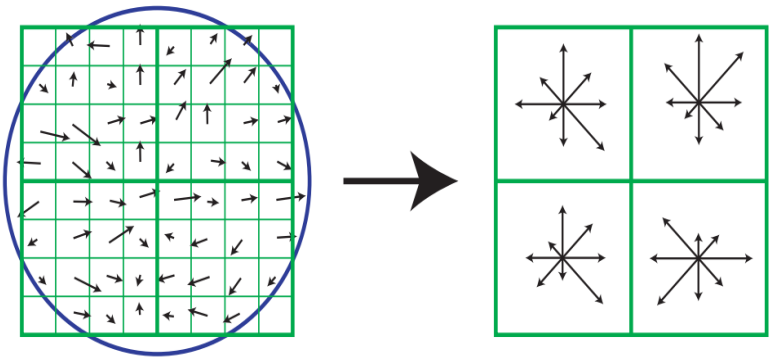
Vytvoření deskriptoru

Poté co byla nalezena množina kandidátů na zájmové body, následuje odstranění nestabilních bodů. Zbylým bodům se přesně definuje poloha a dominantní orientace podle histogramu orientací v jejich okolí [20].

Desktriptor popisuje okolí zájmových bodů a měl by být unikátní pro celý model a měl by zůstat stejný i po změně světla či úhlu pohledu. Existuje řada oblíbených deskriptorů jako je SIFT, SURF, BRIEF a další.

Deskriptor se sestavuje z gradientů v blízkém okolí významného bodu. Nejprve je použitím Gaussovy pyramidy vybrán nejbližší obraz k zajištění nezávislosti na měřítku. Okolí zájmového bodu je rozděleno na stejně velké čtvercové podoblasti a pro každou je sestaven histogram orientací. Tyto histogramy jsou poté natočeny podle určené dominantní orientace významného bodu, čímž je zajištěna nezávislost na rotaci. Rozčlenění okolí do několika oblastí má výhodu v tom, že výsledný deskriptor je odolný vůči malým posunům obrazu. Vzorky jsou váženy Gaussovým oknem, označeným kruhem na obrázku 2.3. Tyto vzorky se poté hromadí do histogramů orientace, které shrnují obsah přes podoblasti 4x4, jak je zobrazeno napravo na obr. 2.3 s délkou každé šipky odpovídající součtu gradientních velikostí blízko tohoto směru uvnitř regionu.

Deskriptor je ve standardním SIFT 128-rozměrný vektor vypočítaný na základě gradientů v okolí zájmového bodu. V okolí tohoto bodu jsou vypočtené gradienty a deskriptor je histogramem těchto gradientů [13].



Obrázek 2.5: Obrázek ukazuje pole deskriptoru 2x2 vypočítané ze sady vzorků 8x8. Převzato z[13].

Deskriptor matching

Každý zájmový bod je již popsán vektorem. Nyní se hledá největší shoda se zájmovým bodem ve sledovaném modelu. K tomu se používá běžná eukleidovská metrika [20]. Pro daný deskriptor z obrázku představuje nejlepší shodu deskriptor, který má v sledovaném modelu nejmenší vzdálenost. Výsledná shoda by měla být jedinečná. Pokud je tedy poměr nejmenší vzdálenosti a druhé nejmenší vzdálenosti větší než nějaký práh (obvykle nastavený na 80%), zájmový bod se zahodí.

Pokud je počet deskriptorů ve sledovaném modelu tak velký, že v dostupném čase nelze provést vyhledávání nejlepší shody, musí být použity heuristické struktury vyhledávání. Typickým přístupem jsou k-d stromy. Poté lze provést vyhledávání v logaritmickém čase. Pokud k-d strom není dostatečně účinný dá se použít metoda Spill forest.

Jakákoli přibližná struktura vyhledávání může vést k nesprávným výsledkům shody, což může vést k odlehlým hodnotám, které ovlivňují výpočet pozice. Proto je velmi důležité mít robustní a efektivní odstraňování odlehlých dat. Je užitečné použít kaskádu technik odstraňování, počínaje nejlevnější metodou a konče nejdražší technikou [13].

Perspective-n-Point Pose

PnP[2] je problém odhadu pozice kalibrované kamery vzhledem k sadě n 3D bodů v reálné scéně a jejich odpovídajícím 2D projekcím v obraze [3]. Řešení problému PnP mohou být iterativní nebo neiterativní metody [20]. Neiterativní metody jsou efektivní, ale jejich omezení je nestabilita v přítomnosti šumu, zejména když $n \leq 5$. P3P, který je nejmenší podmnožinou PnP má mnoho možných řešení a jeho stabilita je omezena. Pro jedinečné řešení je zapotřebí uvažovat více bodů. Čím více bodů bude použito, tím je řešení robustnější a schopnost rozlišování lepší [12].

Robust Pose Estimation

Obecně je žádoucí větší počet datových bodů, tím lze dosáhnout lepší numerické optimalizace [20]. Čím větší je sada vstupních dat, tím vyšší je pravděpodobnost výskytu odlehlých hodnot, což bude výsledek kontaminovat. Je lepší najít dobré inicializační data ze sady dat, i když obsahují silně odlehlé hodnoty, a pak se výsledky konvergují k získání přesného řešení. Cílem je vybrat all-inlier[3] podmnožinu z kontaminované sady datových bodů. K takto robustní inicializaci lze dospět pomocí konsensu náhodného vzorkování RANSAC.

Hlavní myšlenkou RANSAC je odhadnout modelové parametry $\mathbf{x}$ z náhodně vybrané podmnožiny datových bodů.

2.3 Technologie

V posledních letech rozšířená realita prosperuje a firmy se snaží rozšířit svoji nabídku služeb právě o AR. Pro kvalitní výsledek je zapotřebí dobře zvolená technologie na základě požadavků na aplikaci.

[2] PnP - Perspective-n-Point

[3] All-inlier - set bodů splňující množinu parametrů

2.3.1 Frameworky a SDKs

Existuje velké množství sad SDK[4] a vybrat tu správnou není lehké a v následujících letech se dá očekávat ještě narůst počtu těchto sad. Mezi velmi oblíbené patří ARKit, ARCore a Vuforia [5] [6].

AR SDK je klíčovým softwarovým nástrojem pro vývoj aplikací využívající AR [10]. Úlohou AR SDK je spojit digitální obsah a informace se skutečným světem. Sada SDK obsahuje množství funkcí pro vykreslování obsahu, rozpoznávání scény atd., je tedy nutné zvolit si vhodnou sadu pro tvorbu aplikace na základě požadavků.

Vuforia

Vuforia je bezplatná sada pro vývoj softwaru pro implementace mobilní rozšířené reality a byla spuštěna v roce 2010 [24]. Vuforia podporuje iOS, Android a Unity 3D. Platforma Vuforia umožňuje psát nativní aplikace na chytré telefony a tablety. Kromě základní funkce rozšířené reality Vuforia také poskytuje různé funkce, jako je rozpoznávání textu, Frame Markers, přehrávání videa a další. Aplikace lze vytvářet pomocí Unity, Android Studio, Xcode a Visual Studio. Přístup k Vuforia Engine lze také získat prostřednictvím Unity Package Manager. Vuforia SDK využívá technologii počítačového vidění k identifikaci a sledování obrazových cílů a 3D objektů v reálném čase. To umožňuje vývojářům při vývoji AR se orientovat v reálném prostředí a umisťovat do něj virtuální objekty, včetně 3D modelů či jiného obsahu. Virtuální objekty mohou být umisťovány přímo do reálného prostředí a překreslovat tak reálné elementy ve scéně.

Vuforia poskytuje API[7] v Javě, C++, Objective C++ a .NET. Prostřednictvím rozšíření herního enginu Unity je Vuforia schopna podporovat, jak nativní vývoj pro iOS a Android, tak vývoj AR aplikací a prototypů v Unity, které lze snadno přenášet přes obě platformy.

ARKit

V roce 2017 Apple vydal iOS 11 a následně spustil ARKit [23]. ARKit je jedinečný nástroj, který umožňuje společnostem a vývojářům navrhovat a vytvářet bezkonkurenční zážitky pro kompatibilní zařízení iPhone a iPad (kompatibilní zařízení iPhone a iPad musí být vybavena procesorem A9 nebo vyšším). ARKit SDK funguje stejným způsobem jako většina funkcí AR SDK tím, že umožňuje prolnutí digitálních informací a 3D objektů se skutečným světem, ale nabízí do značné míry jedinečnou dostupnost, pokud jde o počet existujících zařízení, která podporuje.

ARKit lze provozovat na jakémkoli zařízení vybaveném procesorem Apple A9, A10 nebo vyšší řady a využívá VIO (Visual Inertial Odometry) pro sledování okolního prostředí s bezproblémovou přesností. VIO umožňuje ARKitu kombinovat data Core Motion s daty kamerových senzorů a umožňuje vyvíjet aplikace, které dokážou detekovat horizontální roviny (podlahy, stoly) a vertikální roviny (stěny). To umožňuje ARKitu přesně porozumět dynamice a složení konkrétní scény a poskytuje možnost umisťovat 3D objekty a překrývat digitální informace kontextově relevantním způsobem.

Vývojáři a firmy mohou vytvářet aplikace pomocí ARKit a přidružených optimalizací prostřednictvím 3D strojů třetích stran, jako jsou Unity, Unreal Engine a SceneKit.

[4] Software Development kit - je sada nástrojů, které vývojáři používají k vytváření aplikací.
[5] Převzato z https://invisible.toys/best-augmented-reality-sdk/
[6] Převzato z https://dzone.com/articles/12-best-augmented-reality-sdks
[7] Application Programming Interface - označuje v informatice rozhraní pro programování aplikací

ARCore

ARCore[8] je proprietární rozšířená realita SDK společnosti Google. Podobně jako ARKit umožňuje firmám a vývojářům uvádět do provozu aplikace AR na kompatibilních chytrých telefonech a tabletech. Jednou z nejvýznamnějších vlastností ARCore je to, že také podporuje zařízení podporující iOS a poskytuje vývojářům jedinečný přístup k uživatelům napříč oběma platformami. ARCore má tři významné funkce, které umožňují vývojářům sloučit skutečný svět s virtuálním:

- umí odhadovat reálné světelné podmínky,
- je schopen detekovat velikost a lokalizaci vertikálních, horizontálních šikmých ploch,
- bere v úvahu pozici telefonu vzhledem k jeho okolí.

ARCore vychází z softwaru Tango, jenž byl vylepšením AR toolkitu [11]. Společnost Google chtěla Rozšířenou realitu více zpřístupnit lidem a proto vytvořila ARCore, který dokáže vytvořit obdobné výsledky, ale na rozdíl od Tanga k tomu nepotřebuje speciální senzory, ale vystačí si se softwarem.

AR SDKs a frameworky			
	Vuforia	ARCore	ARKit
Typ	SDK	Framework	Framework
Cena	zdarma, komerční	zdarma	zdarma
Podpora	A, iOS, W	A, (iOS)	iOS
Smart glass	ano	ano	ano
Unity3D	ano	ano	ano
Cloud recognition	ano	ne	ne

Tabulka 2.1: Základní souhrn vlastností (*A* - Android a *W* - Windows)[9]

2.3.2 Unity

Unity zůstává nejpopulárnější platformou pro AR, VR nebo MR, která se dnes používá a jedním z nejpokročilejších dostupných herních enginů [16]. Pomocí Unity mohou výrobci hardwaru třetích stran snadno vytvářet vlastní pluginy a sady SDK. Unity je kompletně zdarma pro uživatele či firmy, které mají zisk méně jak 100 000 dolarů ročně. Unity je určeno pro vytváření 3D nebo 2D her a podporuje tři programovací jazyky: C, UnityScript (JavaScript) a Boo. Unity3D je engine podporující vícero platforem s intuitivním prostředím pro vývoj aplikací. Je dostatečně jednoduchý pro začátečníky, ale také dostatečné výkonný pro profesionály [14]. Unity poskytuje kompletní 3D prostředí pro tvorbu menu, animací psaní skriptů a organizaci celého projektu. V Unity je také možné vytvářet objekty a materiály. Další silnou stránkou je Unity asset store, kde je spousta zajímavostí, které je možné stáhnou zdarma či za poplatek.

[8] Převzato z https://dzone.com/articles/12-best-augmented-reality-sdks
[9] Převzato z https://invisible.toys/best-augmented-reality-sdk/

2.4 HW pro rozšířenou realitu

Hlavními zařízeními pro rozšířenou realitu jsou displeje, vstupní zařízení (periferie), trekovací zařízení a počítače. Tato zařízení jsou většinou integrovány do jednoho kompletního zařízení [20]. Tato zařízení se pak dále rozdělují do skupin podle způsobu použití nebo jiných specifikací. Tato podkapitola je zaměřena především na techniky zobrazování AR a na některé zástupce nynějších zařízení pro AR.

Existuje mnoho způsobů, jak dosáhnout AR. Tato práce se zabývá pouze dvěma. První je nejběžnější a nejpřístupnější metoda pomocí ručního mobilního zařízení, jako je smartphone nebo tablet. Kamera zachycuje prostředí a počítačová grafika je vykreslena na obrazovce zařízení.

Druhá technika, používající přenosné inteligentní brýle AR, se pomalu objevuje v komerčních zařízeních, jako jsou Microsoft HoloLens a jiné. Jedná se o optický průhled skutečného světa s počítačovou grafikou zobrazenou na nositelném displeji.

Většina dnešního zájmu o AR se pohybuje právě směrem k chytrým brýlím s optickým sledováním. Tato sofistikovaná zařízení používají hloubkové senzory ke skenování a modelování prostředí a poté integrují počítačovou grafiku do reálného světa. Integrace nemusí být pouze viditelná, realitu je možné rozšířit o vibrace či zvuk [10].

2.4.1 Zobrazovací metody AR

V rozšířené realitě se používají tři hlavní typy displejů: ruční, náhlavní (HMD[10]) a prostorové displeje [20]. Pro AR jsou nejvíce užitečné HMD displeje pro pravý AR zážitek. V AR se používají dvě metody pro zobrazení AR na displeji a to Video-view-through a Optical-view-through. Existuje i jiná zobrazovací technika, která ale nepoužívá displej k zobrazení AR nýbrž projektor, který zobrazuje virtuální obsah přímo do reálného světa. Tento přístup je nazýván SAR - (spatial augmented reality) [2].

Video-view-through

Video-view-through je jednou z cenově dostupných technik pro poskytování AR zážitků [20]. Ve VST kamera zachycuje digitální video obrazu skutečného světa a přenáší jej do grafického procesoru v reálném čase. Grafický procesor pak kombinuje zdroj obrazových dat s obrázky generovanými počítačem (virtuální obsah) a zobrazuje je na obrazovce. Protože zpracování videa se provádí před zobrazením obsahu uživateli, je možné ovládat jas a kontrast skutečného světa i virtuálních prvků pro plynulý zážitek. Existují však nevýhody, které zahrnují nízké rozlišení reality (obrazovky neodpovídají rozlišení lidského oka) a omezené zorné pole. VSTs mohou používat smartphony jako v Samsung Gear VR, kde je telefon používán jako displej, který je umístěn pár centimetrů od očí uživatelů, což je úplně jiné, než u AR v ručních smartphonech [2].

Optical see-through (OST)

Optické průhledové displeje používají optické prvky, například polostříbrná zrdcadla, která jsou polopropustná a napůl reflexní, a tak mohou kombinovat skutečný svět a virtuální prvky [20]. Zrcadlo umožňuje průchod dostatečného množství světla ze skutečného světa, což umožňuje přímé vidění okolí. Počítačem generované obrazy se současně promítají do zrcadla prostřednictvím displeje umístěného nad hlavou nebo na boku, nejčastěji brýlí, což

[10] head mounted displays

vytváří vnímání kombinovaného světa. OST ukazují svět v reálném rozlišení bez paralaxy (způsobené posunutím polohy kamery ve vztahu k oku diváka). Použití je bezpečnější, protože uživatel vidí, i když dojde k výpadku napájení, což z něj činí ideální volbu pro vojenské a lékařské účely. Použití zrcadel a čoček však snižuje jas a kontrast virtuálního i reálného vnímání [2].

Spatial projection

Rozšířená realita je v tomto případě generovaná projektorem přímo do reálného světa [20]. Tato metoda je také jistou formou optické kombinace, ale nejsou tu nutné ani displej ani separátní optický combiner.

2.4.2 OST náhlavní displeje

Zařízení jdou rozdělit na více skupin [20]. V této práci jsou zařízení rozdělena na dvě skupiny. První skupinu zařízení lze považovat za úplná 3D AR zařízení. Tato zařízení mohou skutečně integrovat 3D grafiku, vkládat virtuální objekty do reálné scény, jako by to byly skutečné objekty. Příkladem jsou Hololens a Magic Leap a mohou se zde zařadit i Apple Glasses. Druhou skupinou jsou 2D zařízení. Tato zařízení překrývají 2D text a grafiku přes scénu, ale bez jejich plné integrace s reálným světem. Do této kategorie spadají brýle, jako jsou Google Glass nebo Epson Moverio, které jsou ekvivalentem heads up displejů [2].

Zařízení s podporou 3D

Jedním z nejvíce očekávaných brýlí byly Microsoft Hololens, které byly představeny roku 2015, jejichž stereoskopický 3D displej s vysokým rozlišením využívající technologii OST [2]. Jejich nástupcem byly brýle Hololens 2 viz obr. 2.6, které jsou velmi kvalitní, pohodlné avšak velmi drahé. Tato druhá verze vylepšila hlavní nedostatek jejich předchůdce a to rozšíření zorného pole. Hololens 2 jsou také lehčí a pohodlnější, ale kvůli ceně jsou vhodné spíše pro velké firmy .

Dalším zástupcem je Magic Leap One viz obr. 2.6. Tyto brýle vytvořila společnost Magic Leap, která začala v roce 2010 jako startup a slibovala revoluční technologie. Cena této firmy šplhala vzhůru, investovali do ni společnosti jako Alibaba či Google, nicméně Magic Leap One je obdoba brýlí od Microsoftu a neobsahuje nic převratného. Magic Leap One je lehčí než HoloLens 2 a stojí o něco méně .

Zařízení s podporou 2D

Epson Moverio BT-300 viz obr. 2.6, tento zástupce patří do skupiny 2D AR, tedy do zařízení, která zobrazují 2D elementy přes scénu, ale nijak nezasahují do skutečného světa.

Brýle Moverio BT-300 využívají technologii křemíkového digitálního displeje OLED od společnosti Epson [2]. OLED technologie přináší vylepšenou kvalitu obrazu a to například v zobrazování černé barvy, kdy je možné jednotlivé pixely vypnout. Moverio BT-300 jsou binokulární průhledné chytré brýle, které nabízí vysokou kvalitu obrazu. Díky displeji s vysokým rozlišením a vysokým kontrastem je rozšířená realita na dobré úrovni.

[12] Převzato z https://www.lifewire.com/microsoft-hololens-2-explained-4691204, https://jablickar.cz/facebook-hodla-do-roku-2025-predstavit-vlastni-chytre-bryle-chce-jimi-nahradit-iphone-a-dalsi-smartphony/magic-leap-one-lightwear/, https://www.epson.cz/products/see-through-mobile-viewer/moverio-bt-300

Obrázek 2.6: Chytré brýle na levo jsou zobrazeny Microsoft HoloLens 2 v prostřed Magic Leap One a vpravo Moverio BT-300[12]

2.4.3 Mobilní AR

Rozšířená realita u mobilních zařízení je realizována použitím kamery, která zachycuje snímky skutečného světa a poté je kombinuje s virtuálními elementy.

Jak je znázorněno na obr. 2.7 při spuštění aplikace AR na mobilním zařízení se jednoduše nasměruje kamera na cíl v reálném světě a aplikace rozpozná cíl a vykreslí 3D počítačovou grafiku zaregistrovanou podle polohy a orientace cíle [10].

Obrázek 2.7: Ukázka Handheld mobile AR převzato z [10].

Pro AR potřebují mobilní zařízení tyto specifikace:

- Baterii
- Grafický dotykový displej
- Zadní kameru
- CPU a GPU
- Pohybový senzor, GPS nebo jiný senzor pro geolokaci

Kapitola 3

Využití rozšířené reality dnes

Rozšířená realita se dnes používá v řadě odvětví a stále jich přibývá. Tato kapitola se zabývá některými zástupci různých odvětví.

3.1 Automobilový průmysl

Rozšířená realita má velký potenciál, a to zejména v průmyslu, kde se začíná více používat. Jedním z takovýchto průmyslových odvětí je automobilový. AR se v automobilovém průmyslu využívá například při výrobě automobilu nebo jako součást interního systému automobilu, jako je například parkovací asistence. Velký důraz na aplikování rozšířené reality je kladen v německé automobilce BMW.

3.1.1 BMW AR

Obrázek 3.1: Levý obrázek je parkovací asistent v automobilu uprostřed je znázorněno využití AR při montáži motoru a napravo aplikace BMW Individual 7[2].

V automobilce BMW se AR používá například při školení nových zaměstnanců, a to třeba pro montáž motoru. Díky AR jde postup výuky přizpůsobit danému jedinci. Úprava montážního postupu, například změna tloušťky šroubů a nastavení nového školicího programu pomocí softwaru s AR, je rychlá a snadná.

BMW rozšířenou realitu používá také k marketingu a to pomocí aplikací např. BMW Individual 7 Series Augmented Reality. Tato aplikace umožňuje uživateli si navrhnout BMW

[2]Převzato z http://creativeaudiodurban.co.za/our-services/parking-sensors/, https://iot-automotive.news/absolutely-real-virtual-and-augmented-reality-open-new-avenues-in-the-bmw-group-production-system/, https://www.bmw.com.mt/en/topics/offers-and-services/bmw-apps/virtual-and-augmented-reality.html

řady 7 přesně podle svých představ a potom si výsledek prohlédnout v jeho skutečné velikosti jako model v rozšířené realitě.

3.1.2 Ferrari Showroom App

Ferrari použila technologii rozšířené reality společnosti Metaio, aby umožnila potenciálním kupcům podívat se na Ferrari v showroomu, ale pomocí rozšířené reality může toto auto zobrazit v různých barvách. Pouhým nasměrováním iPhonu nebo iPadu na Ferrari v showroomu je možné změnit barvu na autě tak, aby uživatel zvolil vhodnou barvu, i když tato konkrétní barva není k dispozici pro fyzickou kontrolu v showroomu. Protože mnoho automobilových nadšenců chce vědět, co je uvnitř automobilu, aplikace Ferrari s rozšířenou realitou také umožňuje uživatelům zaměřit iPhone nebo iPad na auto a prohlížet si například vzhled motoru [23].

Obrázek 3.2: Ferrari A.R. Showroom App. Převzato z [23]

3.1.3 XARfix

XARfix má za úkol řešit problémy při výpadcích na výrobní lince. Systém pracuje s chytrými brýlemi, díky kterým je možné technika nasměrovat k problematickému sektoru. Systém následně identifikuje vadnou komponentu a data sdílí technikovi. Technik má k dispozici i vlastní data a pomocí nich může opravu závady uskutečnit a systém jej při opravě kontroluje. Díky tomuto řešení je možné kompetentní personál mít na jednom místě a ten pak zasílá data technikům na jakémkoliv místě na zemi. Díky holografickým brýlím mohou sledovat práci technika a zasílat mu potřebná data. Tento systém cílí na opravy, které nejdou řešit na dálku a čas dojezdu opraváře je příliš vzdálený [21].

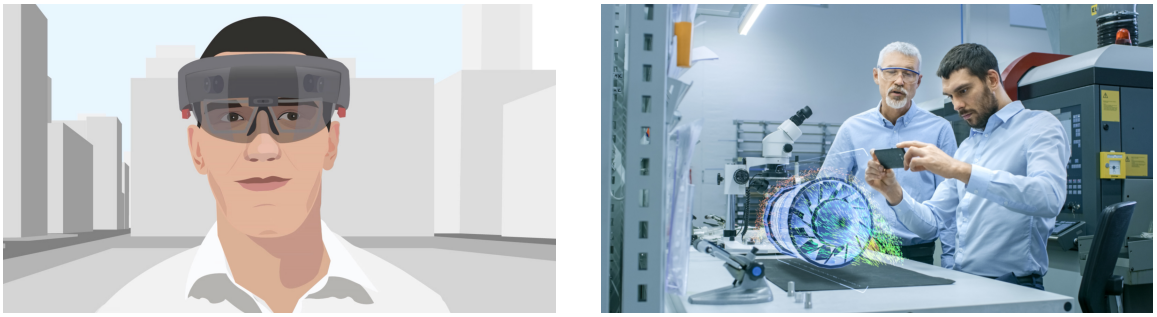
3.2 AR pro výuku

Člověk, který se učí poslechem a nevidí konkrétní problém na vlastní oči, je daleko méně schopen zpracovat nové informace. Tento problém řeší AR aplikace pro výuku.

3.2.1 XARguid

Společnosti se potýkají s nedostatkem kvalifikovaných pracovních sil [21]. Díky XARGuid je možné efektivně zaučit nového pracovníka prostřednictvím vizuální projekce na holografických brýlích a osobní přístup zkušeného technika, který vše vysvětluje.

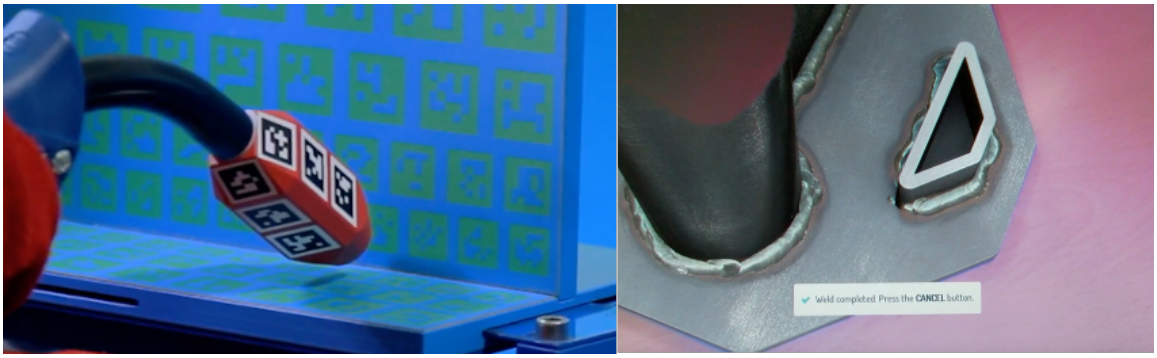
Vyškolený pracovník může vidět skutečné stroje, zařízení nebo produkty v reálném prostředí díky holografickým brýlím. Současně lze podle pokynů školitele promítnout do brýlí řadu dalších informací, pokynů, schémat a pracovních postupů. Online připojení je základem pro okamžitou interakci, korekci, vzájemnou diskusi nebo kontrolu správnosti a nezáleží na tom, kde je v tuto chvíli instruktor. Systémem je možné trénovat několik pracovníků současně.



Obrázek 3.3: XARguid, převzato z [21]

3.2.2 Soldamatic

Soldamatic IE je první svařovací tréninkové řešení využívající rozšířenou realitu na světě. Soldamatic IE používá AR pro trénování začátečníků i opravdových profesionálů. Všechny hardwarové komponenty Soldamatic IE jsou skutečné svařovací zařízení s implementovanými markery pro AR viz obrázek 3.4 a svářecí helma s holografickým sklem.



Obrázek 3.4: Obrázek vpravo: Soldamatic IE virtuální vizualizace objektu. Obrázek vlevo: AR za použití značek[4].

[4]Převzato z https://www.soldamatic.com/

3.3 AR pro architekturu

V současné době architektonické ateliéry často využívají virtuální procházky v namodelovaných domech. Tato skutečnost se projevila také na vývoji aplikací využívající AR.

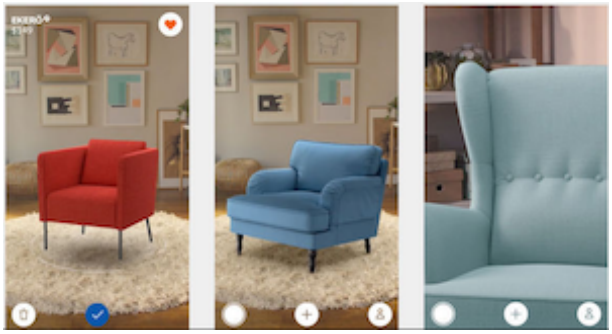
3.3.1 XARbuild

Tato aplikace umožnuje architektům nebo budoucím uživatelům se procházet po budově, která ještě není postavena a mohou budovu změnit do posledního detailu [21]. Aplikace využívá holografické brýle. S těmito brýlemi je možné začít stavět na zelené louce od země po nejvyšší patro. Projekce neomezeného počtu architektonických prvků jim umožní dokonalé umístění v reálném prostoru. Technologie podporuje vytváření stavebních projektů od samého začátku, nejen objekty hal, jejich vnitřní části, jako jsou stěny, dveře, okna, příčky, sanitární zařízení, větrání, ale také optimalizuje proces realizace menších projektů, jako je dokončení nebo rekonstrukce stávajících budov, skladů ... Díky tomuto řešení může architekt zamezit případným problémům při práci na projektu, protože si může celou budovu prohlédnout v reálném prostředí a v reálných dimenzích.

3.3.2 Ikea place

IKEA place[5]je postavena na ARCore a umožňuje prakticky „umisťovat“ výrobky IKEA do prostoru. Má také funkci „vizuálního vyhledávání“, to funguje tak, že uživatel nasměruje fotoaparát na jakýkoli nábytek, a telefon zjistí a vyhledá, který produkt IKEA se nejvíce podobá tomuto objektu.

Tato aplikace zahrnuje 3D modely v reálném měřítku od pohovek a křesel po podnožky a konferenční stolky. IKEA place uživateli poskytuje představu o velikosti, designu a funkčnosti nábytku v domácnosti, takže uživatel nemusí dlouze přemýšlet a rozhodovat se.



Obrázek 3.5: Ikea Place aplikace. Převzato z obchodu Google Play[5].

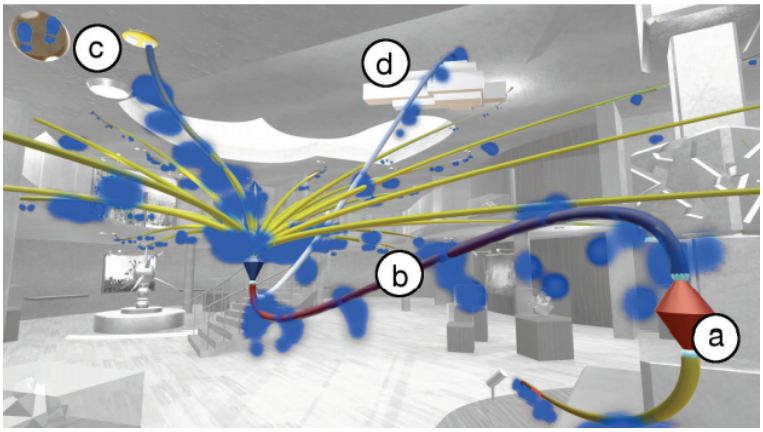
3.4 Průmyslový internet věcí (IIoT)

V roce 2017 odstartovalo ve velkém měřítku zavádění průmyslového internetu věcí do praktického života firem. Tento trend nadále pokračuje. Podniky se intenzivně zaměřují na příležitosti vedoucí k vyšší efektivitě, kvalitě, úsporám, produktivitě a inovacím, které jsou buď požadované legislativou, očekávané trhem nebo vynucené konkurencí.

[5]Převzato z https://play.google.com/store/apps/details?id=com.inter$_i$*kea.placehl* = *cs*

3.4.1 Ivy

Ivy je prostorově umístěný vizuální programovací nástroj využívající virtuální realitu. Ivy umožňuje uživatelům propojení inteligentních objektů, vkládání logických konstrukcí a vizualizací v reálném čase. Zobrazuje datové toky mezi senzory a akčními členy v reálném světě [5].



Obrázek 3.6: Uživatelé mohou (a) vytvářet programové konstrukce, (b) navazování logických odkazů, (c) vizualizace datových toků z dat senzoru v reálném světě a (d) nahrání dat do cloudu. Virtuální uzly portů (c) v tomto scénáři muzea fungují jako rozhraní k senzorům v reálném světě, jako jsou například údaje o stopách. Převzato z [5].

3.4.2 XARview

Průmyslový internet věcí (IIoT) vyžaduje řešení Plug and Produce. Tlak na produktivitu opět urychluje zavádění nových výrobních procesů [21]. V této situaci představuje XARview ideální nástroj pro jejich optimální a bezchybné uvedení do života.

Aplikace umožňuje vizualizovat konkrétní komponentu, objekt a lokalizovat jej v konkrétním reálném prostředí. Je primárně určen pro použití v průmyslu, podnikání a službách. Pomocí projekčních brýlí uživatel nejprve naskenuje skutečný prostor a poté může začít umisťovat 3D objekty pomocí holografické projekce, buď ze své vlastní databáze nebo z katalogu. Během práce systém zobrazuje ve svém zorném poli všechna relevantní data vázaná na objekt a parametry prostředí.

V průmyslovém prostředí přináší tento proces značné výhody, protože umožňuje navrhovat výrobní linky rychle, snadno a efektivně. Spojením pohledu na reálné prostředí s přihlédnutím ke všem daným podmínkám s 3D hologramem umístěných objektů se vytvoří vizuální projekce budoucího řešení. Díky komplexnímu pohledu a předdefinovaným přiřazením je možné předem vyloučit všechny kritické nebo rizikové faktory plánovaného výrobního procesu a změnit jeho parametry ještě před samotnou implementací.

Se systémem je možné pracovat ve více lidech zároveň, což umožnuje začlenění více odborníků z jiného odvětví a tím urychlit a zkvalitnit návrh.

3.5 Hry

Rozšířená realita umožňuje digitálním hrám dodat reálné prostředí, například Vuforia nabízí funkci Smart Terrain™ v sadě Vuforia SDK, která je průlomovou vizuální schopností umožňující novou úroveň ponoření do herních zážitků s rozšířenou realitou. Smart Terrain umožňuje rekonstruovat a rozšířit fyzické prostředí, vytvářet nové druhy herních a vizualizačních aplikací.

Obrázek 3.7: Obrázek vlevo: Vuforia Smart Terrain, převzato z [20]. Obrázek vpravo: Pokemon Go aplikace, převzato z obchodu Google Play[7].

Asi nejznámnějším zástupcem her využívajících rozšířenou realitu je hra Pokemon Go. Tato hra propojuje realný svět s herním prostředím a využívá k tomu GPS navigaci a kameru na telefonu. Tato hra se stala fenoménem. Pokemon Go vytvořila ve spolupráci s Pokémon Company firma Niantic [20].

Herní průmysl byl velkou hnací silou za vývojem grafických výpočetních jednotek a virtuální reality. AR hry mohou využívat vícero přístupů jak zkombinovat reálné data s virtuálními. Příkladem jsou hry využívající GPS takzvané lokalizační AR (Pokemon GO, Harry Potter Wizzards Unite a Zombies, Run!) nebo také markerless přístup s detekcí plochy (Jenga AR, Knightfall™ AR).

3.5.1 Zombies, Run!

Tato AR hra pro běžce umožňuje užvivateli hrát hrdinu ve svém vlastním příběhu o zombie. Takže, když běží ve vlastním bloku, snaží se předběhnout zombie ve hře a sbírat zásoby pro přežití.

Zombies, Run![8] používá GPS a sledování tempa a začleňuje hru do polohy, ve které se nachází hráč. Používá také vlastní seznam skladeb uživatele k přehrávání hudby na pozadí během joggingu, běhu nebo sprintu. Každý běh se stává misí, kde je uživatel hrdinou. Pohlcující zvukové drama uživatele postaví do středu vlastního příběhu o zombie.

Zombie, běž! funguje kdekoli a při jakékoli rychlosti je možné běhat v parku, běhat po pláži nebo chodit po stezce a funguje to i na běžících pásech.

[7] Převzato z https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongohl=cs
[8] Převzato z https://play.google.com/store/apps/details?id=com.sixtostart.zombiesrunclienthl=cs

3.5.2 Knightfall AR

Tato hra umožňuje prostřednictvím ARCore technologie detekovat plochu nejčastěji stůl a vytvořit na něm bitevní pole. Knightfall AR[9]je vytvořena společností Milkroom Entertainment a ve spolupráci s Milkroom Entertainment a Spectral Games. Prostřednictvím kamery na mobilním zařízení uživatel může umisťovat střely a chránit tak svůj hrad před vlnami nájezdníků, které se snaží hrad dobít. Hra obsahuje foto mód, který umožňuje uživateli si zvolit postavu, kterou může přemístit, změnit velikost, nastavit animaci a následně sdílet s přáteli.

Obrázek 3.8: Na levém obrázku je zobrazeno bitevní pole a na pravém je detekce plochy a umísťování bitevního pole ve hře Knightfall AR[9] .

3.5.3 The Machines

Hra využívající technologii ARKit s velmi povedeným grafickým zpracováním. Hra je schopna vykreslovat 1.2 milionu polygonů a využívá 4K rozlišení, také disponuje 3D zvukem, takže když se uživatel přiblíží souboji hlasitost se zvýší a pokud se pohybuje za vysokou horou, tak je zvuk ztlumen. Prostřednictvím mobilního zařízení je možné nejen přemisťovat jednotky, ale také mířit se speciálními zbraněmi. The Machines umožňuje hru vícero hráčů a je dostupná na Apple store.

Obrázek 3.9: Představení hry The Machines[10]Apple Keynote v roce 2017.

[9]Převzato z https://play.google.com/store/apps/details?id=com.aetn.games.android.history.knightfall
[10]Převzato z https://www.gamesradar.com/best-mobile-ar-games

Kapitola 4

Programování robotů

Tato kapitola je zaměřena na způsoby programování robotů. Programování robotů se z velké části posunulo od nízkoúrovňového kódování k intuitivnějším metodám. Tento krok byl částečně podpořen touhou usnadnit operátorům programování. Operátoři robotů nejsou vždy výrobci robotů a výrobci robotů nejsou vždy nejlepší lidé, kteří programují konkrétní úkol. Pro zkušeného člověka v daném zaměření by bylo mnohem snažší naprogramovat robota, který by pracoval na daném úkolu, nežli programátorovi, který se v tomto oboru neorientuje. Dříve tradiční způsoby programování robotů by operátorům méně či vůbec neznalých programování činily velké problémy.

4.1 Expertní programovaní

Expertní programovaní zahrnuje především metody programování robotů založené na simulaci.

4.1.1 Metoda offline programování (OLP)

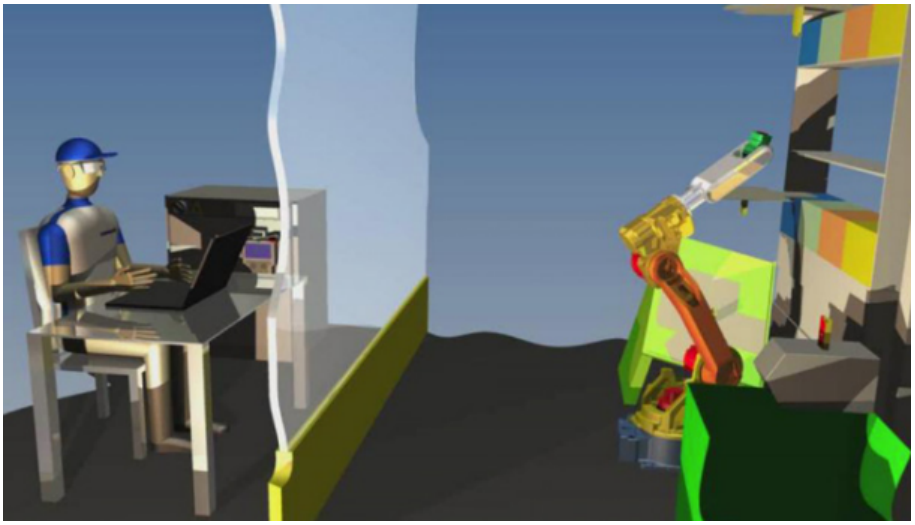
OLP je proces programování robota, jenž není plně automatizovaný, zahrnuje manuální tvorbu kódu robota a počítačový software simuluje reálný scénář robota [17]. Offline programování má výhody například v možnosti programování robota aniž by musela být výroba přerušena. Jak je možné vidět na obrázku 4.1, kde robot může pracovat mezi tím co programátor vytváří nový program. Až je program hotov je robot pozastaven po dobu nahrávání nového programu. Mezi další výhodu této metody je tvorba programu robota mimo prostředí výroby, tím je zajištěna vyšší míra bezpečnosti personálu. Je možné využití simulačních nástrojů, které otestují reálné chování robata a tím zefektivní proces programování.

Mezi nevýhody offline programování je možné zařadit například větší investici do softwaru a zaškolení pracovníků.

Mezi tři z nejběžnějších OLP balíčků patří Delmia od Dassault Systémes, RobCAD od Technomatix Technologies a Robotmaster od Jabez Technologies. Tyto balíčky poskytují modelovací a simulační nástroje, které jsou schopny grafické reprezentace robota a jeho doplňků, také generování programu a simulaci. Velké firmy jako například KUKA, ABB Robotics mají své vlastní OLP softwary.

V dnešní době jsou OLP balíčky mnohem více vylepšené po grafické stránce a to díky možnostem například importu CAD formátů [15]. CAD technologie přinesla vylepšené uži-

vatelské rozhraní a funkcionality ať už integrované moduly nebo samostatné řešení, vizualizaci a simulaci.



Obrázek 4.1: OLP způsob tvorby programu pro robota, kde programátor je v jiné místnosti než robot. Převzato z [17]

Programování robotů pomocí systému CAD bylo v posledních letech předmětem mnoha výzkumů. Počítačová grafická simulace robota a jeho pracovní buňky může být realizována pomocí různých modelů, jako jsou kostry a solid modely. Tyto modely a odpovídající algoritmy mohou být použity pro detekci kolizí a pro kinematické a dynamické chování robota.

4.2 Programování pomocí demonstrace (online)

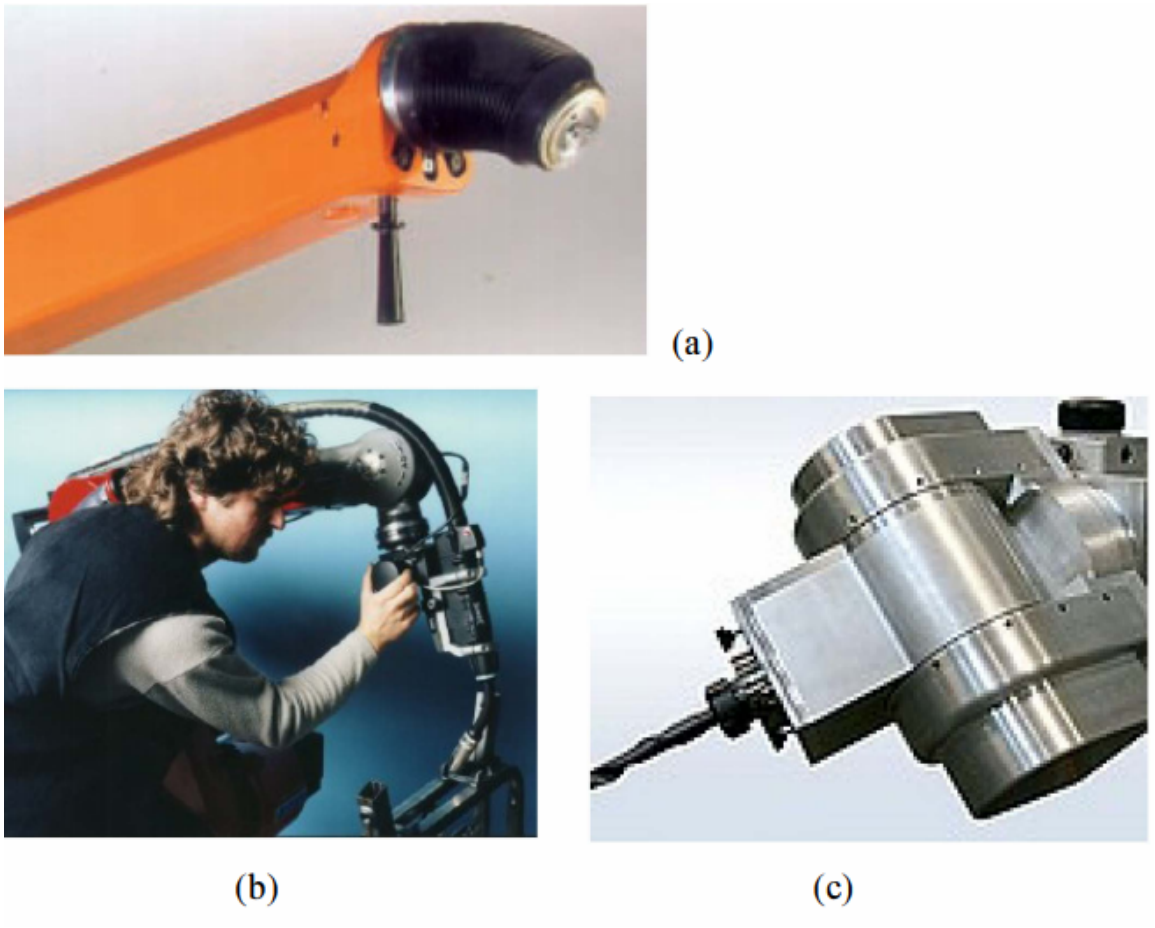
Online programování tradičně provádí kvalifikovaní operátoři robotů dané společnost vedením robota přes požadované cesty pomocí teach-pendantu nebo ručně lead-through metodou [19]. Online programování má relativně snadné použití v praxi po určitém zaškolení techniků. Online programování je jedna z častěji používaných metod programování robotů.

4.2.1 Lead Through

Metoda lead-through obvykle zahrnuje krokování s robotem po požadované cestě, zaznamenávání konkrétních bodů v ovladači robota a využití zaznamenaných bodů k vytvoření pohybových příkazů [4]. Většina průmyslových robotů je programována metodou Teach-andplayback, kde se programátor robotů musí pohybovat robotem k požadovanému cílovému bodu jeho TCP (středový bod nástroje robota) a poté zaznamená odpovídající společné hodnoty v paměti, které ovladač později přečte a použije pro přehrávání. Někdy je žádoucí, aby pohyb robota byl v souřadném systému nástroje, nikoli na skutečném nebo v souřadném systému robotické základny.

V mnoha případech se roboti ovládají pomocí joysticku nebo klávesnice, což není snadné ani intuitivní pokud operátor nemá dostatečné dovednosti a zkušenosti. Ve skutečnosti během výuky operátor často potřebuje změnit polohu těla kolem robota za účelem lepšího úhlu pohledu pro snadnější manévrování. To je pro operátora těžké se rychle přizpůsobit nové pozici a orientaci joysticku nebo klávesnice.

Tento problém se řeší tzv. vyučovací rukojetí, která je namontována na daném robotovi. Operátor ručně vede robota požadovanou cestou nebo po sobě jdoucími body pro definování cesty. Tato technika byla zvláště aplikována na průmyslové úkoly, jako je stříkání, nanášení tmelu, nanášení lepidla a svařování, kde je pohyb robotickým nástrojem podél hladké a souvislé cesty nebo řadou interpolovaných bodů na cestě.



Obrázek 4.2: Ukázka: (a) ovládací rukojeti na robotu určeného k malování, (b) ovládání robota rukojetí, (c) robotická ruka pro frézování. Převzato z [4].

4.2.2 Teach Pendant

Tato metoda programování se používá pro průmyslové úkoly typu pick&place, jako je bodové svařování a nakládka a vykládka stroje. Pohyby robota jsou ovládány pomocí Teach pendantu[1] nebo klávesnice. Programátor může určit pohyb a rychlost každé končetiny robota mezi dvěma body. Robotův cyklus je sled takových pohybů, které lze pozorovat během programování. V periodě přehrávání může programátor upravit pořadí tak, aby získal optimální čas a přesnost cyklu.

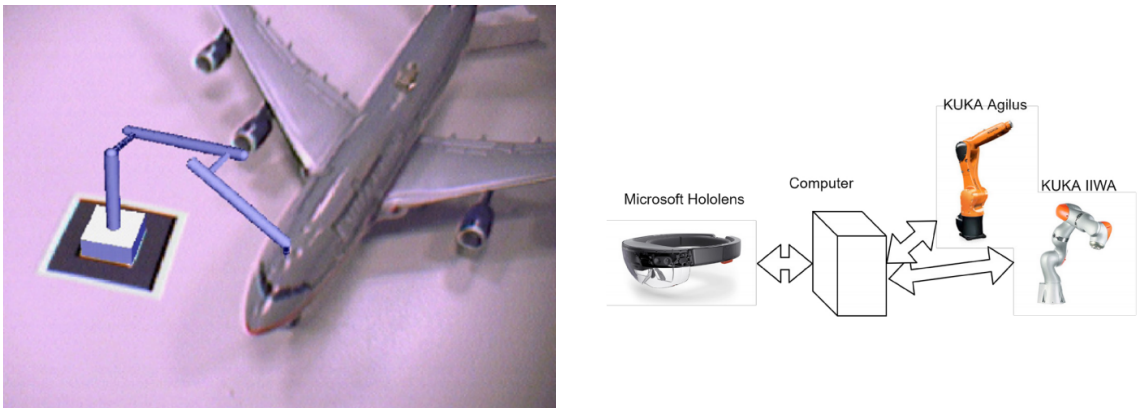
Teach pendant [9], obsahuje zobrazovací zařízení s grafickými zobrazovacími schopnostmi. Zobrazená data se zobrazují na grafickém displeji a zahrnují pohyb robota a naučené

[1]Převzato z http://engineertech.org/courses/industrial-robotics/?submit=viewvimeography$_{g}allery$ = 36$vimeography_{v}ideo$ = 134218994

body. Tato zobrazovací data jsou generována jednotkou pro generování zobrazovacích dat na základě programů robota uložených v jednotce pro ukládání dat vyrovnáváním dat z řídicí jednotky robota. Tato zobrazovaná data mohou také zahrnovat reprezentativní obrazy robotického nástroje spolu s jeho souřadnicemi v souřadném systému nástroje a naučené body v uživatelském nebo světovém souřadnicovém systému. Takto může být činnost robota snadno rozpoznatelná a robot může být učen snadno a přesně. Programátor nemusí být kvalifikovaný v robotickém úkolu, jako v případě průchodu. Při programování může být nutné, aby byl operátor velmi blízko k poháněnému robotu, což může způsobit potenciální bezpečnostní problémy.

4.3 Využití rozšířené reality

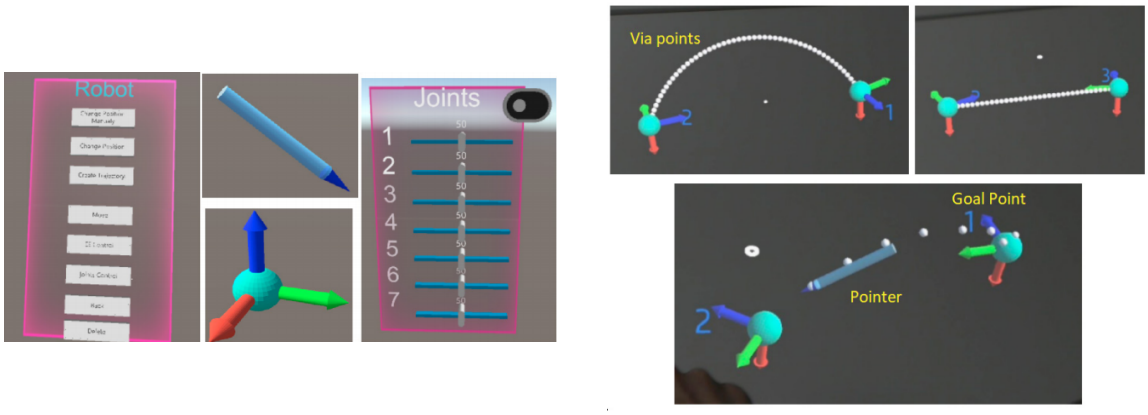
K programování robota je možné použít rozšířenou realitu, tj. interaktivní překrývání reálného prostředí virtuálními prostorovými informacemi. AR má výhody přímo dostupné grafické simulace v reálném výrobním prostředí a poskytuje efektivní a intuitivní komunikační kanál pro prostorové informace. Příkladem může být virtuální model mytí letounu viz obr. 4.3 vlevo, kde robot může být položen na zmenšený model letadla a pohybovat se po něm a generovat robotickou sekvenci, která může být později kalibrována a naprogramována pro skutečného mycího robota [19].



Obrázek 4.3: Obrázek vlevo zobrazuje miniaturu letadla s virtuálním robotem, převzato z [19]. Na obrázku vpravo je možné vidět možný systémový hardware pro AR v robotice. Převzato z [18].

Pro využívání rozšířené reality je více způsobů, jedním z nich je použití chytrých brýlí. Systémové požadavky pak mohou být například viz obr.4.3 vpravo. Pro vytvoření systému je možné využít Unity3D a Mixed Reality Toolkit-Unity pro podporu HoloLens.

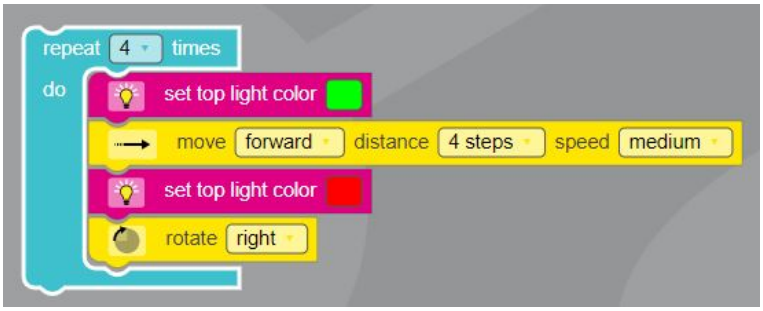
K interakci člověka s virtuálním robotem dochází přes brýle, gesta, virtuální nabídky a virtuální nástroje. Existují různé scénáře interakce pro programování robota na různé úkoly, například pro ukázání na bodové (PTP) operace, operátor určí startovací a cílovou polohu, vybere konfiguraci robota pro tyto dvě polohy. Po úkolu formování, je naplánována geometrická dráha pohybu. Trajektorie zohledňuje kinematický model robota a jeho omezení, jako limity kloubů, rychlost, a zrychlení. Poté se provede simulace a pokud operátor je s výsledkem simulace spokojen, výsledná trajektorie je odeslána skutečnému objektu [18].



Obrázek 4.4: Na obrázku vlevo je možné vidět virtuální nástroje: Hlavní menu, pointer, cílový bod, informační menu a na pravém je zobrazeno propojení cílových bodů obloukem, linkou a pomocí cesty stanovené uživatelem. Převzato z [18].

4.4 Využití vizuálního programování

Vizuální programování je typ programovacího jazyka, který umožňuje lidem popisovat procesy pomocí ilustrace. Tento způsob tvorby programu je pro běžného člověka jednodušší, protože vizuální reprezentace dává člověku větší smysl. Vizuální programování se často používá při učení programování a to právě díky výše zmíněným výhodám. Základní prvky programu jsou reprezentovány bloky a zarážkami viz obrázek 4.5, které říkají jak jdou jednotlivé bloky propojovat [8]. Příkladem může být rozhraní, které využívá Google Blockly[2],



Obrázek 4.5: Ukázka tvorby vizuálního programu pomocí bloků v aplikaci OzoBlokly.

nástroj k vytvoření uživatelského rozhraní pro konstrukci kódů založených na blocích.

Blockly je knihovna s otevřeným zdrojovým kódem, je flexibilní a podporuje velké množství funkcí pro různé aplikace. Používá se pro programování animovaných postav na obrazovce, vytváření skriptů příběhů, ovládání robotů a dokonce generování právních dokumentů. Ale Blockly není sám o sobě jazyk. Vývojáři, kteří používají Blockly, vytvářejí svůj vlastní jazyk. Když vývojáři vytvoří aplikaci pomocí Blockly, měli by pečlivě zvážit styl a rozhraní API.

Google Blockly používá aplikace OzoBlocky[3], která umožnuje programovat Ozobota viz obr. 4.5.

[2]Přavzato z https://developers.google.com/blockly
[3]Převzato z https://ozoblockly.com/

Kapitola 5

Návrh

Jak již bylo zmíněno v kapitole Teorie, rozšířená realita má velký potenciál v průmyslu. Je tomu tak, mimo jiné díky možnosti provádět daný úkon (zaškolení, programování, konstrukce a jiné) přímo na místě, kde se robot nachází a místa daných úkonů tak přesně označit na konkrétní reálné pozici. Tím odpadá nutnost simulace prostředí nebo tzv. programování naslepo. Jednou z dalších výhod může být způsob tvorby programu pomocí vizuálních objektů, což je pro běžného člověka jednodušší na pochopení.

Navrhovaná experimentální mobilní aplikace by měla splňovat především intuitivní ovládání a dobře zvládnutou integraci rozšířené reality, která uživateli práci usnadňuje a nikoliv znepříjemňuje. Při návrhu je nutné uvážit, jak a kdo danou aplikaci bude používat. Tato experimentální aplikace bude zaměřena na tvorbu programu pro robota pomocí rozšířené reality a za použití Video-view-through techniky. Tato technika využívá videokameru a grafický čip, který data z kamery zpracuje a začlení do nich virtuální objekty a výsledek následně vykreslí na displeji. Návrh aplikace je pro tablety od firmy Apple a tudíž aplikace bude funkční na zařízeních s operačním systémem iOS. Aplikace nebude mít výstupem data, ale umožní uživateli si danou konstrukci programu vyzkoušet, zda pracuje jak si představoval, pomoci simulace průchodu programem.

Kapitola popisuje návrh aplikace využívající rozšířenou realitu a volbu technologií pro návrh rozhraní aplikace.

5.1 Technologie

Volba technologií je zásadní pro vývoj, ale existují také technologie, které mohou zefektivnit tvorbu návrhů aplikace. Tato část stručně popisuje technologie pro návrh mobilních aplikací.

5.1.1 Adobe XD

Pro návrh byla nejdříve zvolena aplikace Adobe XD od společnosti Adobe. Tato aplikace používá vektorovou grafiku. Využívá se pro návrhy webových a mobilních aplikací a je funkční na MacOS i na Windows. Adobe XD umí vytvářet jednoduché interaktivní prototypy a obsahuje spoustu zajímavostí jako pluginy, tvorbu designu pomocí hlasu, responzivní změnu velikostí, znovupoužitelnost návrhu v jiných aplikacích od společnosti Adobe a další. Při výběru hrálo velkou roli také to, že po pár návodech je uživatel již schopen aplikaci používat.

5.1.2 Framer X

Je designový nástroj, který je velmi podobný Adobe XD, ale jeho schopnost vytvořený návrh importovat oskenováním QR kódu mobilním zařízením a následně jej otestovat přímo na zařízení, je pro uživatele velmi pozitivní a tudíž byla nakonec zvolena spolu s Origami studio, jako nástroj pro návrh aplikace v této práci.

5.1.3 Origami studio

Origami Studio[1] je bezplatný designový nástroj vytvořený společností Facebook a dostupný pro Mac. Umožňuje návrhářům rychle vytvářet a sdílet interaktivní rozhraní. Hlavním důvodem proč zvolit Origami studio je možnost připojení ke kameře, a tak prototyp otestovat. Framer X bohužel má tuto možnost značně omezenou.

5.1.4 Blender 2.8

Tvorba jednoduchých virtuálních objektů bude prováděna v Blenderu 2.8. Blender[2] je bezplatný 3D tvůrčí balíček. Podporuje modelování, animace, simulace, vykreslování, kompozici a sledování pohybu, dokonce i editaci videa a tvorbu her. Pokročilí uživatelé používají Blender API pro skriptování v Pythonu pro přizpůsobení aplikace a psaní specializovaných nástrojů.

5.2 Návrh aplikace

Tato experimentální aplikace by měla umožňovat uživateli vytvořit jednoduchý vizuální program v rozšířené realitě pro robota s robotickou paží. Uživatel bude moci umístit robota na plošný předmět, dále umístit tzv. akční body, které znázorňují místa kam se robotická paže bude přesouvat a vykonávat uživatelem zvolené akce. Tyto akce bude uživatel moci přiřazovat konkrétním bodům, a tak tvořit program. Průchod vytvořeným programem je následně možné nasimulovat a ověřit, zda je program správně navržen. Simulace je zde chápána, jako vizualizace průchodu programem v uživateli nastavené programové sekvenci, nikoliv jako simulace konkrétních instrukcí.

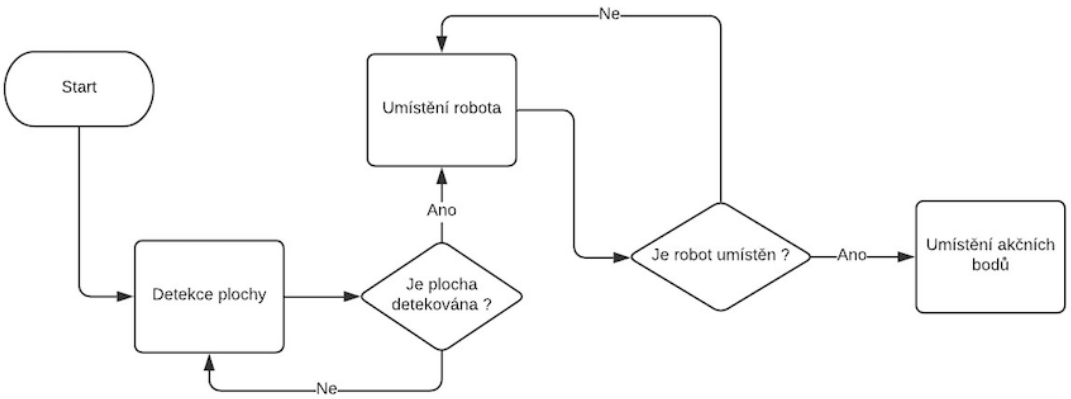
Aplikace se bude sestávat z těchto prvků:

- robot
- akční body
- instrukce
- podmínky
- menu

Robot se po spuštění aplikace umístí do scény na určité místo na detekované ploše, dále s ním již nebude možné pohybovat.

Aplikace je tvořena pro robota s robotickou paží, aby bylo možné určit robotovi místo, kam se má paže přesunout. Bude nutné nějakým způsobem určit tuto pozici. K tomuto účelu

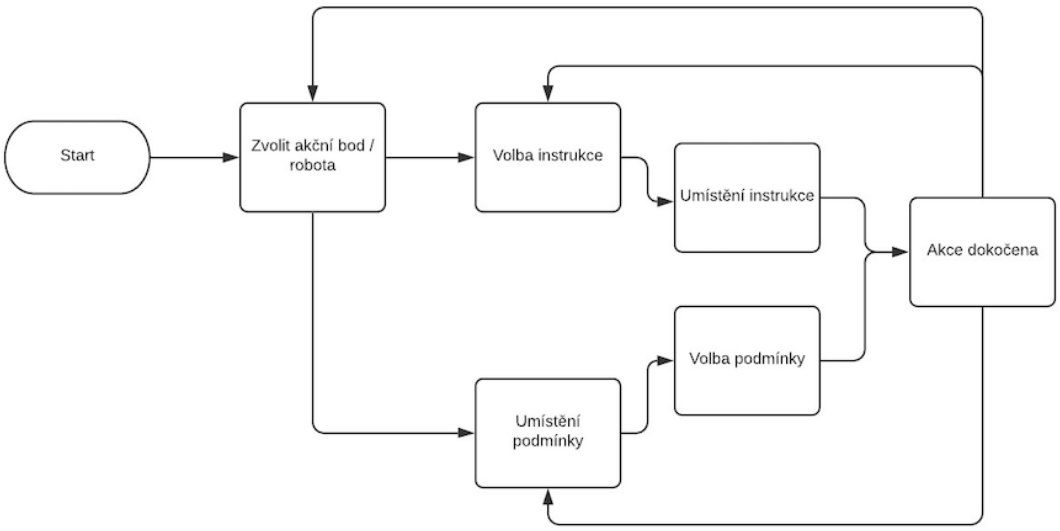
[1] Převzato z https://origami.design/documentation/
[2] Převzato z https://www.blender.org/about/



Obrázek 5.1: Flow diagram fáze tvorby základní scény.

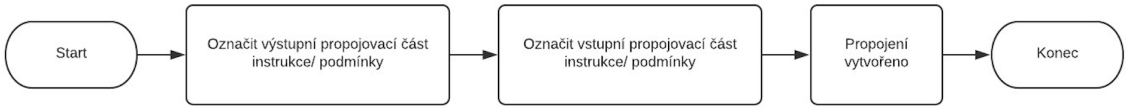
slouží tzv. akční body, což budou místa ve scéně, která umožňují uživateli přidávat instrukce, které se budou provádět na tomto místě ve scéně. Akční body bude možné přidávat během běhu aplikace a také jim měnit polohu. Průběh tvorby základní scény je znázorněn na obrázku 5.1.

Tvořený program se bude skládat z instrukcí, které jsou nejzásadnějším prvkem aplikace a určují co a kde se bude provádět. Instrukce budou vázané na prvek, ke kterému patří a to buď k robotovi nebo k akčnímu bodu. To stejné platí pro podmínky, které v aplikaci nebudou chybět. Podmínky budou fungovat způsobem, kdy ověří, zda operandy splňují podmínku danou operátorem a následně budou vracet pravda nebo nepravda. Menu v tomto případě bude obsahovat instrukce, podmínky a doplňkové elementy. Průběh tvorby instrukcí a podmínek je znázorněn na obrázku 5.2.



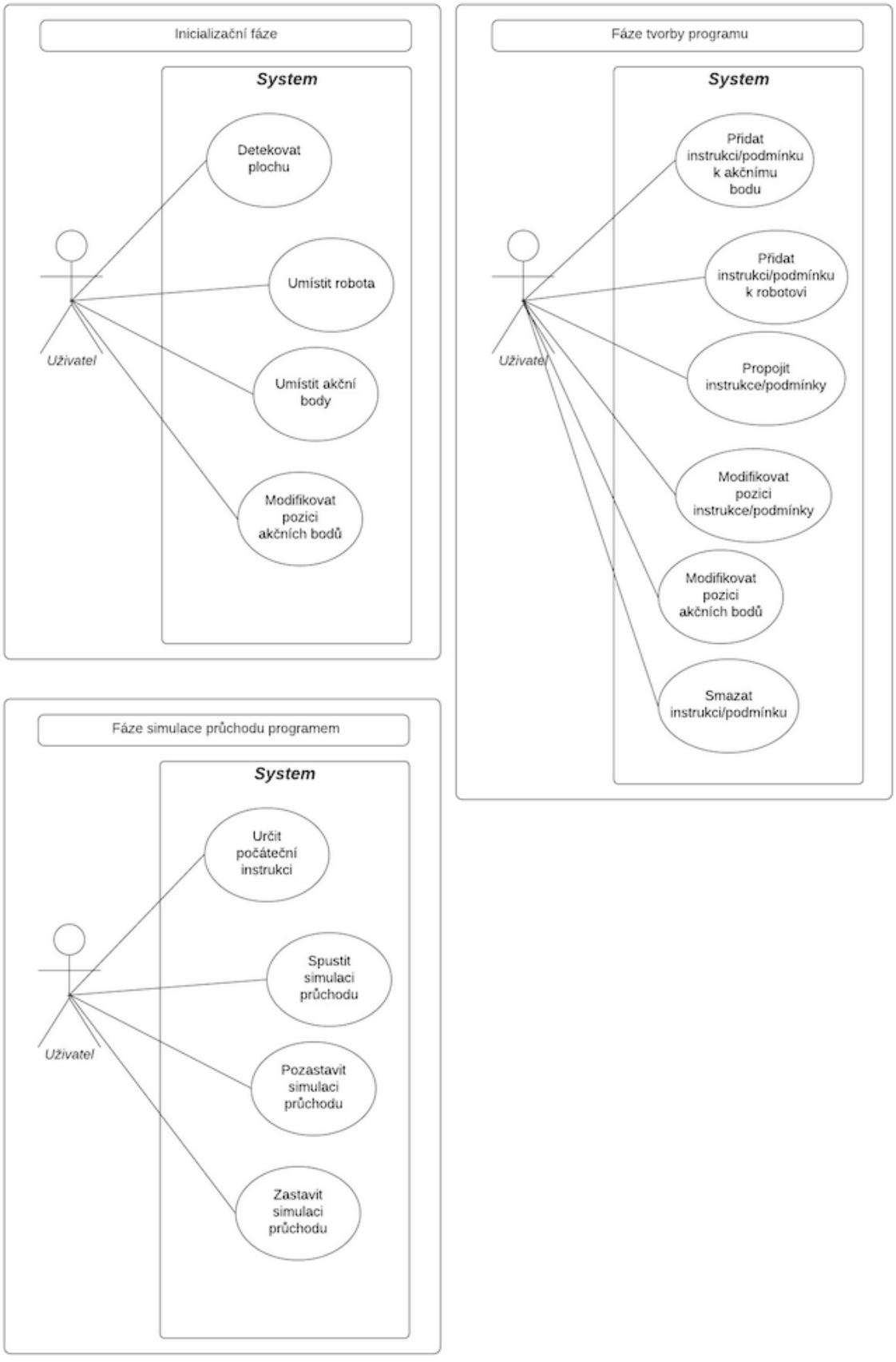
Obrázek 5.2: Flow diagram fáze tvorby programu.

Instrukce a podmínky na sebe musí navazovat, aby tvořily programovou sekvenci. Aplikace umožní tyto elementy propojovat. Průběh tvorby propojení instrukcí a podmínek je znázorněn na obrázku 5.3.



Obrázek 5.3: Flow diagram fáze tvorby propojení instrukcí a podmínek.

Na obrázku 5.4 jsou znázorněny pomocí usecase diagramu hlavní funkce, které aplikace bude umožňovat uživateli. Funkcionality jsou rozděleny do tří hlavních fází běhu aplikace.



Obrázek 5.4: Usecase diagramy, které zobrazují akce, které umožňuje aplikace.

5.2.1 Návrh podmínky

Podmínka pro své vyhodnocení potřebuje data. Bylo zapotřebí vymyslet, jak data do tvorby programu začlenit. Výsledkem jsou tzv. zdrojové instrukce. Tyto instrukce doplní do programu data (zdroje), s kterými mohou klasické instrukce a podmínky dále pracovat. Pod pojmem zdroj je možné si představit jakýkoliv předmět, s kterým robot bude pracovat. Příkladem může být umístění jedné zdrojové instrukce k akčnímu bodu. Zdrojová instrukce tak přidá akčnímu bodu fixní počet zdrojů. Počet zdrojů je nastaven na pět. Pokud bude uživatel chtít umístit více zdrojů je možné aplikovat cyklus. V případě, že zdrojů je potřeba menší počet, je nutné tak nastavit správně podmínku. Tato experimentální aplikace není tvořena pro konkrétního robota s konkrétním zaměřením, a proto jsou zdroje také nekonkrétní.

Podmínka bude pracovat se dvěma operandy, první operand bude typu jedné ze zdrojových instrukcí a druhý bude číselný udávající požadovaný počet daného zdroje. V podmínce je nabídka ze čtyř možných operátorů a to menší, větší, rovno a modulo. Tato množina je dostatečná, protože je pomocí ní možná jakákoliv další kombinace operátorů. Jak již bylo zmíněno podmínka potřebuje data, z tohoto důvodu nelze začínat program právě podmínkou. Podmínka bere v úvahu pouze zdroje, které patří k danému bodu, dá se říci, že zdroje jsou lokální proměnné pro daný bod ve scéně.

5.2.2 Návrh simulace průchodu programem

Simulace tak jako podmínka potřebuje pracovat se zdrojovými instrukcemi udávajícími počet zdrojů na počátku simulace. Zdroje se pojí ke konkrétnímu bodu ve scéně (robot, akční bod). Počet zdrojů mohou ovlivňovat některé instrukce. Například instrukce, které reprezentuje zvednutí zdroje odečte od tohoto místa jeden zdroj. Jiná instrukce jej zase může přidat nebo zdroj nijak neovlivnit. Průběh simulace průchodu programu bude probíhat po jednotlivých instrukcích, podmínky budou určovat následující instrukci na základě vyhodnocení operandů. Pro uživatelský komfort průběh simulace průchodu programem bude doprovázen i animací robotické paže, která se přesouvá na místa ve scéně podle toho, ke kterému akčnímu bodu se daná instrukce či podmínka vztahuje. Dále u každého akčního bodu i robota bude 3D panel, který uživateli zobrazí aktuální stav zdrojů na daném místě ve scéně.

5.3 Návrh GUI

Tato podkapitola obsahuje návrhy aplikace a úvod do problematiky funkčnosti aplikace.

5.3.1 Prvotní idea

Navrhnout AR aplikaci tak, aby byla přehledná a dobře použitelná, není lehké. Bylo zapotřebí vytvořit více návrhů, aby se co možná nejvíce zamezilo případným nedostatkům či problémům při implementaci nebo při používání aplikace.

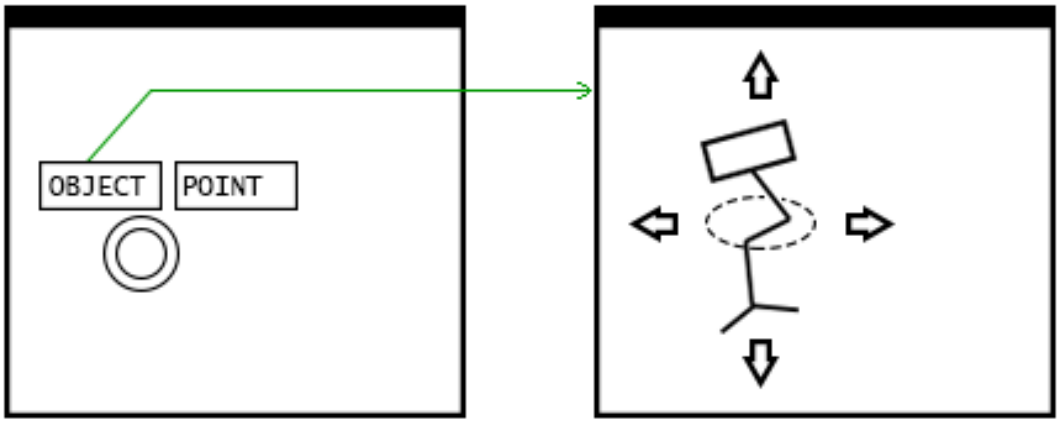
Prvotní návrhy takzvané drátové modely aplikace byly kresleny tužkou na papír. Pro dokumentaci jsou, ale přepracovány do digitální podoby.

Jedním z problémů návrhu AR aplikace je, jaké elementy se budou zobrazovat jako 2D prvky zakotvené na displeji a jaké prvky budou na určitých souřadnicích ve scéně. Možné varianty na zvážení jsou: a) vše umístěno u virtuálních prvků ve scéně, b) virtuální prvky

budou obsahovat jen to nejnutnější a ostatní bude řešeno pomocí menu a za c) vše řešeno přes menu a ve scéně budou pouze samotné prvky (robot, akční bod).

Varianta *a* je velice výhodná pro rozšířenou realitu, ale v tomto případě, kdy aplikace je určena na tablet a nikoli na chytré brýle, tak ovladatelnost by nebyla dostatečně komfortní. Varianta za *c* působí jako nedostatečně využitý potenciál rozšířené reality, tudíž nejlepší variantou je kompromis a tedy varianta *b*, která dostatečně využije potenciál AR, ale nezničí tím ovladatelnost aplikace.

Hlavním problémem, který nastává je, jakým způsobem umožnit uživateli samotnou tvorbu programu v rozšířené realitě.



Obrázek 5.5: Návrh části aplikace, kde uživatel umisťuje objekt nebo akční bod do scény a následně je umožněna modifikace polohy vloženého prvku.

Možná varianta je takzvané blokové řešení, které jednotlivé instrukce shlukuje do bloků. Těmto blokům je poté možno stanovit počet cyklů a vnořovat je do jiných bloků. Druhá varianta je grafové řešení, které umožňuje větší volnost a hlavně větvení programu. Jednotlivé varianty budou rozebrány podrobněji.

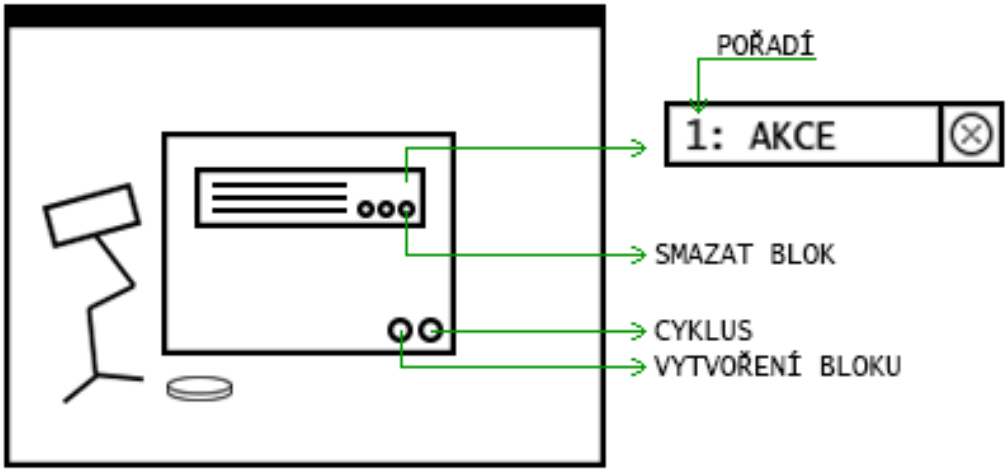
Sekvenční (bloková) varianta aplikace

Robot a každý akční bod bude mít vlastní panel, který bude obsahovat tlačítka pro přidání akce, rozdělení akcí na bloky a přidání/ubrání cyklu. Cyklus je aplikován na bloky a ve výchozím stavu má každý blok počet cyklů nastaven na jeden. Tento způsob umožňuje elegantní cyklení instrukcí v rámci bloku, ale není bohužel schopno takto elegantně řešit cyklus mezi instrukcemi z ostatních bloků. Celkové provázaní mezi panely by bylo možné pouze bez možnosti cyklů.

Toto řešení je svazující, protože nedovoluje uživateli tvořit komplexnější program, také propojení instrukcí z ostatních bloků by bylo komplikované a špatně čitelné. Absence podmíněných instrukcí je také velice svazující.

Varianta aplikace umožňující větvení

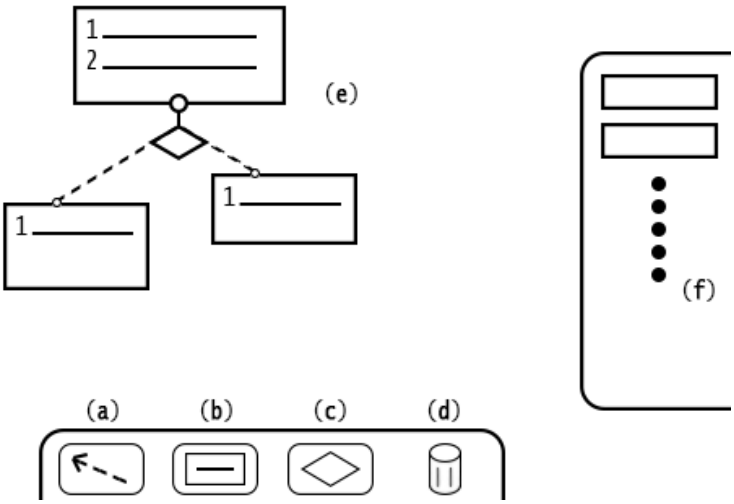
Předchozí varianta neuvažovala větvení programu. Robot při provádění určité instrukce může dojít do stavu, kdy je zapotřebí vyvolat různé instrukce na základě aktuální situace, aby uživatel mohl tuto situaci navrhnout pomocí rozšířené reality, je potřeba mít možnost přidání podmínky. Protože striktně blokové řešení není na větvení ideální, přichází na řadu



Obrázek 5.6: Sekvenční varianta, kde k robotovi přiléhá panel s bloky, které obsahují instrukce. Instrukce jsou v bloku seřazeny podle pořadí, které je globání vrámci celého programu vytvořeného uživatelem.

grafové řešení. Uživatel bude mít k dispozici panel nástrojů a jednotlivé bloky (uzly) bude propojovat pomocí křivek. Tato varianta je bezesporu o něco robustnější ovšem pořád není schopna tvořit cyklus v rámci více objektů, ale pouze v rámci jednoho bloku.

Po upravení této varianty, odstraněním bloků a separací instrukcí, bude možné propojovat jednotlivé instrukce a to i přes více objektů a také to umožní tvorbu cyklů bez omezení. Celkově tato varianta působí jako robustnější a vhodnější volba.

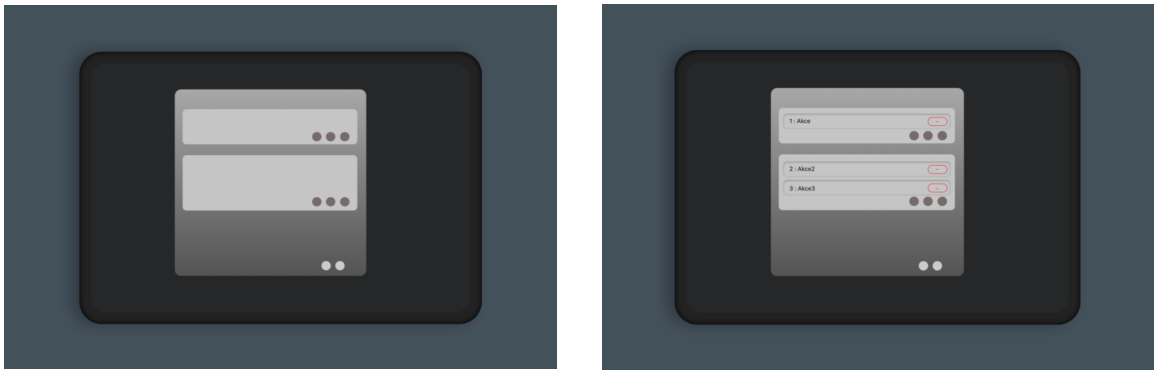


Obrázek 5.7: Návrh varianty s možností větvení. (a) Nástroj pro propojení bloků, (b) nástroj pro vytvoření bloku, (c) nástroj pro vytvoření podmínky, (d) koš, (e) grafová reprezentace programu, (f) seznam instrukcí.

Testování návrhů

Část aplikace jde, jak navrhnout, tak i otestovat, bohužel většina částí rozšířené reality se nesnadno modeluje a zatím je velice těžké nalézt nějaký nástroj na jejich návrh a testování. Pomocí aplikace Origami studio bylo možné nasimulovat první část aplikace, tedy hledání místa na tvorbu objektů pomocí kamery.

Pomocí aplikace Framer X byl vytvořen návrh rozhraní pro nastavování akcí k akčním bodům. Každý akční bod má vlastní panel s akcemi. Akce je možné slučovat do bloků, těmto blokům je možné nastavit počet opakování.



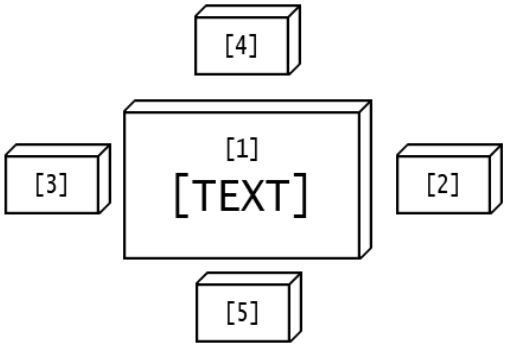
Obrázek 5.8: Tvorba návrhu panelu s bloky obsahující instrukce a pořadí v FramerX.

5.3.2 Finální návrh aplikace

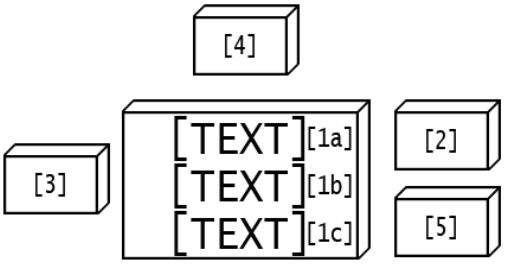
Po vyzkoušení několika různých návrhů bude aplikace pracovat s grafovou reprezentací uživatelem tvořeného programu a umožní tak větvení, cykly a tudíž bude pro uživatele více užitečná. Program bude tvořen pomocí objektů, nikoliv panelů, jenž byly popsány výše, které budou nazývané instrukce. Instrukce budou mít vícero typů, jejich vzhled bude ovlivněn několika faktory, a to zajištěním dostačující intuitivity, líbivého vzhledu a také funkčnosti viz obr. 6.7. Instrukce budou mít jeden vstupní a jeden výstupní bod. Kromě instrukcí se v aplikaci budou používat podmínky, které budou vycházet ze vzhledu klasických instrukcí, ale budou rozšířeny o jeden výstup, vrchní výstup bude kladná a spodní bude záporná větev programu viz obr. 6.7.

Instrukce a podmínky bude možné propojovat přes vstupní a výstupní body pomocí spojů viz obr. 5.11. Aplikace bude obsahovat jednoho robota, kterému uživatel tvoří program a také tzv. akční body. Robot i akční body mohou obsahovat instrukce a podmínky viz obr. 5.11. Pro vkládání podmínek a instrukcí bude vytvořeno drag&drop menu a uživatel bude moci tahem umisťovat instrukce do scény, ty se automaticky uchytí k danému akčnímu bodu, který byl uživatelem zvolen.

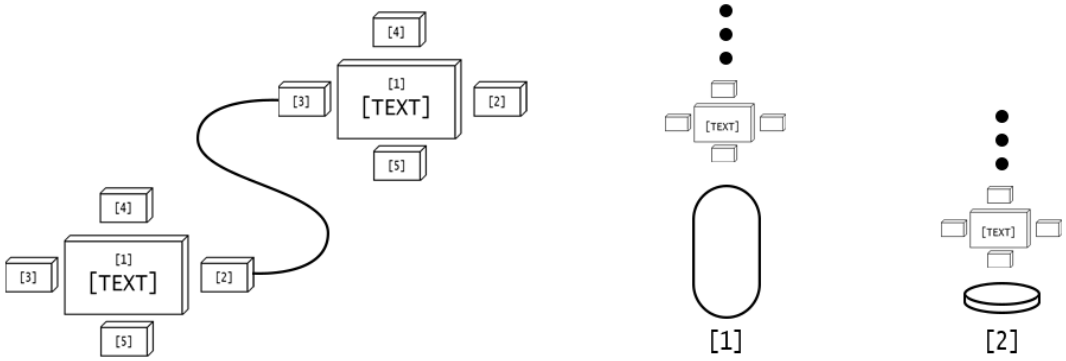
V momentě umístění robota již není možnost jej nikterak přemisťovat, to ovšem neplatí u akčních bodů, které je možné libovolně přemisťovat po ploše. Instrukce i podmínky se přemisťují s daným akčním bodem, aby uživatel měl stále přehled co k čemu patří.



Obrázek 5.9: Finální návrh instrukce. (1) Středová část instrukce obsahující název instrukce, (2) výstupní část instrukce, slouží pro propojení s ostatním instrukcemi, (3) vstupní část instrukce, také slouží pro propojování, (4) část pro odstranění instrukce, (5) část pro nastavení instrukce jako počáteční v rámci programu.



Obrázek 5.10: Finální návrh podmínky. (1a) Středová část podmínky obsahující levý operand, (1b) operátor podmínky, (1c) pravý operand podmínky, (2) kladný výstup, (3) vstupní část, (4) část pro odstranění podmínky, (5) záporný výstup.



Obrázek 5.11: Obrázek vlevo znázorňuje propojení instrukcí pomocí křivek, vstupních a výstupních prvků instrukce a obrázek vpravo znázorňuje robota i akční bod obsahující instrukce a podmínky.

Kapitola 6

Implementace

Kapitola implementace popisuje způsob vytvoření prototypu aplikace a zvolené technologie a přístupy.

Pro implementaci rozšířené reality byl zvolen AR Foundation balíček, který umožňuje práci s rozšířenou realitou a Unity 2018.1. Tyto balíčky umožňují multiplatformní vývoj. Pro použití je tedy nutné nainstalovat alespoň jeden z ARKit XR nebo ARCore XR pluginů. Vzhledem k tomu, že aplikace je cílena na platformu iOS, je nutné nainstalovat ARKit XR plugin.

6.1 Mapování scény

Aplikace využívá markerless metodu AR. Aby uživatel mohl umisťovat objekty do scény je nutné nejprve scénu zmapovat. Toto umožňuje ARKit, který je součástí AR foundation, tak že zmapuje reálnou scénu a vytvoří si digitální mapu. Digitální mapa je tvořena pomocí stereo vision kalkulace, která vyprodukuje sparse množinu bodů [6]. Díky ní poté ARKit může detekovat například plochu, objekty nebo jiné elementy. Aby objekty nelétaly ve vzduchu, aplikace používá detekci plochy s využitím AR Plane manageru, díky kterému je možné umístit objekt na nějaký plošný předmět.

Zaměření aplikace je na programování robotických akcí, tudíž plocha, na které se nejspíše bude pracovat bude podlaha či velký stůl. Díky digitální mapě reálné scény je možné se s mobilním zařízením pohybovat a virtuální objekty, které uživatel přidal zůstávají na daných místech ve scéně. Při pohybu si ARKit rozšiřuje a aktualizuje svou digitální mapu. Aby byla metoda úspěšná je zapotřebí statické prostředí. V momentě neúspěšné detekce je nutné restartovat celý koordinační systém.

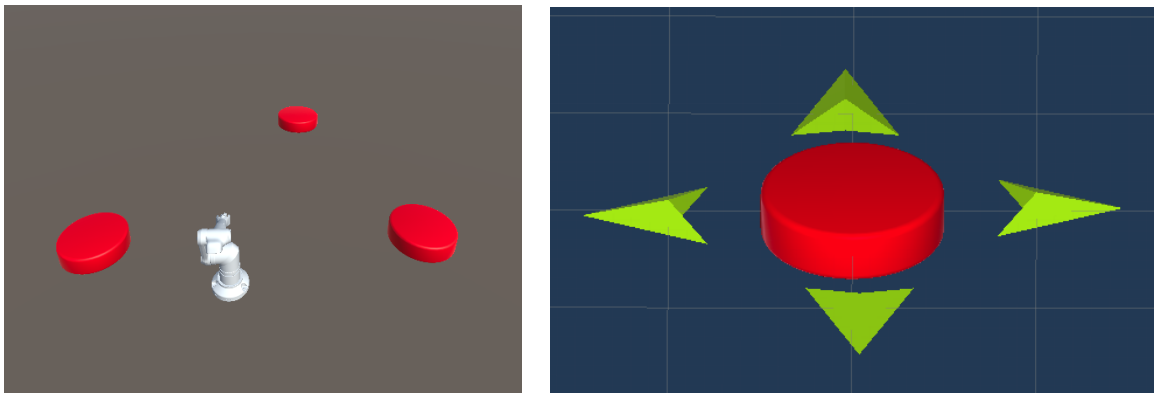
Unity i AR Foundation používají levotočivou souřadnou soustavu tudíž zde odpadl problém dvou rozdílných souřadných systémů, jako se děje například při použití chytrých brýlí Microsoft HoloLens, které používají levotočivou souřadnou soustavu.

Pohyb s virtuálními předměty umístěnými na detekované ploše je umožněn pomocí *ARRaycastManageru*. Díky tomu je možné reagovat na dotek na určité místo na detekované ploše, ale také to umožňuje zareagovat na dotek na virtuální objekt, který se nachází na této ploše.

6.2 Robot a akční bod

Robot a akční body jsou prvky, jejichž umístění je závislé na detekované ploše. Díky *ARFoudation* a třídě *ARRaycastManager*, která již byla zmíněna dříve, je možné tyto elementy umístit právě na tuto plochu viz obr. 6.1 *ARRaycastManager* umožňuje zaznamenávání doteku na akční bod i robota, díky tomu je možné akční body přemisťovat a pracovat s nimi. Aplikace umožňuje dva typy interakce s akčním bodem, a to krátký dotek, který vysune menu a poskytne tak možnost přidat akčnímu bodu instrukci či podmínku a nebo podržení prstu na akčním bodu, který umožní uživateli přemisťovat objektem po detekované ploše viz obr. 6.1. Toho je možné dosáhnout využitím takzvaných *touch* fází, které obsahuje Unity. Fáze *stationary* je aktivována v momentě, kdy se prst dotýká displeje, ale nijak svoji pozici nemění. Využitím *stationary* fáze a třídy *Time* je možné určit délku doteku, nutnou k vyvolání příslušné metody. Dalšími fázemi jsou *begun* a *ended*, které mimo jiné je možné využít pro drag&drop manipulaci a klasickou *onClick* operaci.

Unity umožňuje pozicování relativní ke scéně a také lokální pozicování, které je relativní k rodičovskému prvku. Vztah akčního bodu či robota k podmínkám a instrukcím je rodič a potomek. Díky tomuto vztahu a lokálnímu pozicování je možné v Unity snadno docílit toho, aby instrukce a podmínky reagovaly na změnu polohy akčního bodu.



Obrázek 6.1: Obrázek nalevo zobrazuje příklad možného rozestavení akčních bodů a robota ve scéně. Na obrázku vpravo je možné vidět zvolený akční bod, jehož postranní šipky znázorňují možnost změny polohy.

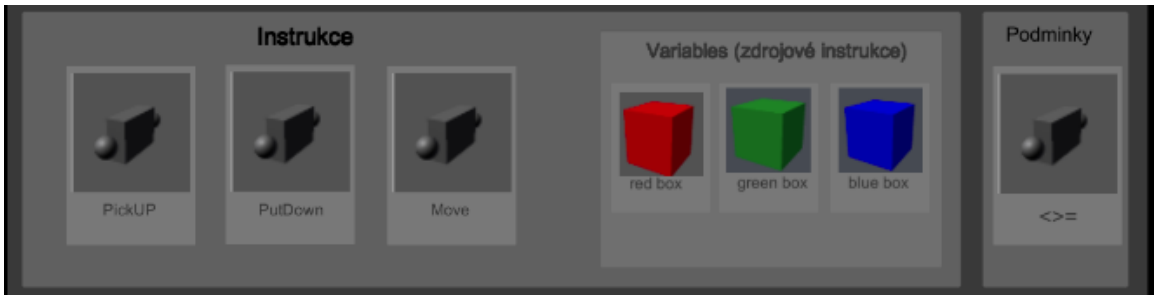
Aplikace obsahuje dvě rozdílné menu. První je závislé na objektu a druhé je staticky umístěné a slouží pro přidávání akčních bodů do scény, resetování celé scény aplikace a také k řízení simulace průchodu programem. Již zmíněné na objektu závislé menu nejsou staticky umístěné, to neznamená, že jsou ukotveny k nějakému objektu, ale jejich funkcionalita je ovlivněna konkrétním objektem. Na objektu vázané menu je vysouvací z dolní části obrazovky a slouží k přidávání instrukcí a podmínek.

6.2.1 Na objekt vázané menu

Tento typ menu pracuje stylem drag&drop. V Unity je toho možné docílit vytvořením dvou skriptů. A to skriptem pro zpracování události drag, který je nutné připojit k elementu, který bude přemisťován tahem po displeji, v tomto případě 2d obrázek. Elementy určené k přemisťování jsou umístěné v panelu, jako jednotlivé sloty, který má připojen skript pro zpracování události drop.

Při přetahování slotu z panelu musí menu vědět jakému objektu má daný element v tomto případě instrukci či podmínku přidat, z tohoto důvodu je menu pojmenováno, jako vázané. Jak již bylo zmíněno v sekci Robot a akční body, po doteku na akční bod se vysune menu, tomuto menu se nastaví aktuálně zvolený akční bod, jako cílový objekt.

Událost drop seskupí data potřebná pro vytvoření instancí či podmínek a následně data předá *GameManageru*, který již zmíněné instance vytvoří. To o jakou instanci se jedná je rozhodnuto na základě pojmenování slotu viz obr. 6.2, který je přetahován do scény.



Obrázek 6.2: Drag&drop menu obsahující instrukce, podmínky a zdrojové instrukce (lokální proměnné).

6.3 Instrukce

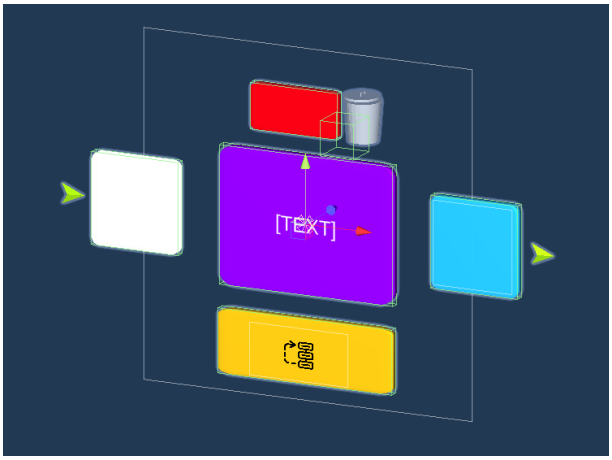
Instrukce, jak již bylo zmíněno, je potomek akčního objektu nebo robota. Instrukce je objekt, jehož instance je vytvořena pomocí menu a je umístěna přímo nad robota či akční bod. Robot má fixní polohu, ale akční bod svoji polohu měnit může, pokud se tak stane, tak všechny instrukce či podmínky se přemístí také, protože mají nastavenou lokální pozici, které je závislá na pozici rodiče.

Změna polohy samotné instrukce je možná ve směrech nahoru, dolů, doprava a doleva. Aplikace neumožňuje posun instrukcí ve směru dopředu a dozadu, protože dotykem na displej není možné nijak určit pozici na ose udávající vzdálenost od kamery. Pozicování v rámci této osy je řešeno automatickým umístěním dané instrukce nad akční bod či robota, kterému instrukce náleží. Pokud by uživatel trval na změně polohy instrukce i po této ose, je možné stiskem instrukce a pohybem tabletu instrukci přemístit, nicméně tento způsob v aplikaci není považován za standardní. Polohu uživatel mění tahem prstu po displeji, pro korektní pohyb v rámci scény je nutné použít funkci *ScreenToWorldPoint*, která umožňuje získat bod v reálné scéně převodem bodu obrazovky na stanovenou vzdálenost od roviny kamery. Polohu instrukce je možné měnit i jiným způsobem než-li pohybem prstu po displeji. Jako jinou variantu aplikace poskytuje pohyb pomocí joysticku. Ten byl vytvořen pomocí assetu *Joystick Pack*. K tomu, aby se instrukce pohybovala ji uživatel musí označit a poté ji nasměrovat pomocí joysticku kam potřebuje v rámci dvou směrů.

Pokud uživatel změní směr pohledu na scénu, změna polohy instrukcí a podmínek standardním způsobem, což je tah po displeji nebo použití joysticku, již není možná, protože se osy změnily v závislosti na směru pohybu uživatele ve scéně. V tomto případě, jak již bylo zmíněno, lze objekt přemístit tak, že uživatel stiskne místo, kde se nachází instrukce a pohybem tabletu přemístí danou instrukci. Problém změny směru pohledu by mohl jít vyřešit rotací celé scény v závislosti na rotaci kamery, která snímá reálnou scénu. Nicméně

pohyb instrukcí není nikterak zásadní v rámci této aplikace a tudíž tato problematika zde není řešena.

Instrukce je základním stavebním kamenem uživatelem tvořeného programu. Program musí mít počátek, proto každá instrukce má možnost se stát počátkem programu tím, že se uživatel dotkne žlutého tlačítka na instrukci viz obr. 6.3. Obdobně lze instrukci odstranit dotykem na červené tlačítko. Tyto operace jsou prováděny v *GameManageru* aplikace, a to z toho důvodu, že informace jako jsou počátek programu, jednotlivé instrukce a jiné, je vhodné spravovat z jednoho místa, aby byla zajištěna konzistence. Instrukce i podmínky



Obrázek 6.3: Unity prefab instrukce.

mají schopnost se naklonit přesně v kolmém směru ke kameře. Tato akce je vyvolána v případě pohledu středem zorného pole kamery na danou instrukci či podmínku. Aby uživatel věděl, kde je střed kamery, je na střed umístěn 2D ukazatel. Aplikace poskytuje naklonění ve dvou případech, a to při zvolení instrukce dotykem nebo na již zmíněném nasměrováním středu kamery na danou instrukci. V prvním případě se použije funkce *Quaternion.Lerp*, která provádí interpolaci mezi dvěma rotacemi a následně výsledek normalizuje. Porovnáním eulerových úhlů kamery a instrukce či podmínky je rotace ukončena nebo se v ní pokračuje. V druhém případě je využit podobný způsob, ale je tu navíc funkce *Quaternion.LookRotation*, která vytvoří rotaci podle zadaného směru vpřed a nahoru. Konkrétně odečtením pozice kamery od pozice instrukce udá potřebný parametr do této metody.

Jak již bylo zmíněno aplikace je schopna reagovat na pohled kamery uživatele, této schopnosti je využito pro zvukovou zpětnou vazbu prostřednictvím assetu *TextToSpeech* od společnosti Google. V případě pohledu kamery na danou instrukci či podmínku je uživateli řečeno na jakou instrukci či podmínku se právě dívá.

Instrukce mají různé funkce v rámci tvorby programu, těmto funkcím bude věnována pozornost v sekci Simulace průchodu programem. Instrukce a podmínky tvoří graf a to propojením pomocí křivek, tomuto tématu bude věnována sekce Propojení křivkami.

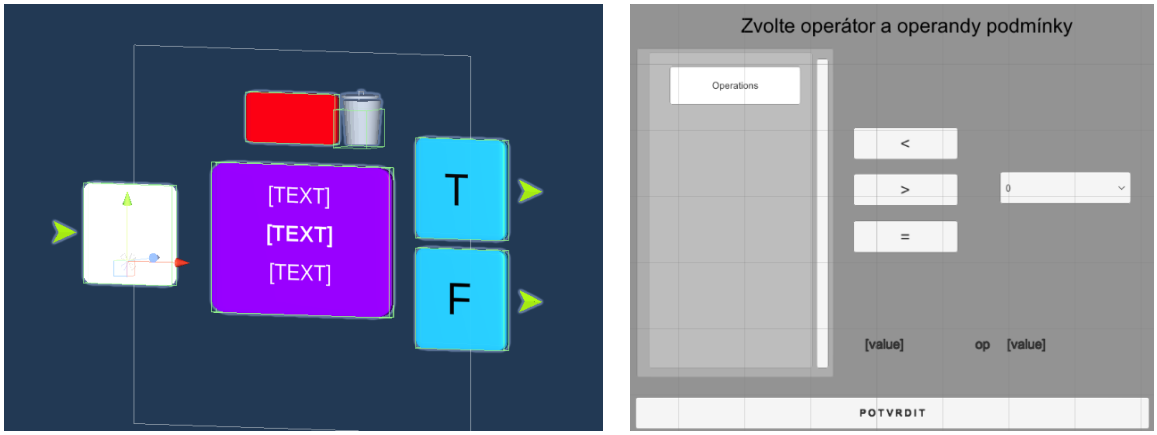
6.4 Podmínky

Jak již bylo zmíněno v návrhu aplikace, podmínky umožňují program větvit a kontrolovaně cyklit. Objekt podmínky je v této aplikaci vytvořen tak, aby umožnil rozdělit tok programu do dvou větví a to větve, kdy je podmínka splněna a do větve, kdy podmínka splněna není. Na rozdíl od instrukce podmínka není zvolena a poté umístěna do scény, ale nejdříve se

umístí a poté pomocí nabídky uživatel zvolí operandy a operátor podmínky, následně má podmínka vše nastaveno a může být vytvořena. Operandy jsou dvojího typu, levý operand je daný takzvanými zdrojovými instrukcemi, kterým bude věnován prostor v sekci simulace průchodu programem a pravý operand je číselný.

Vyhodnocení podmínky probíhá v průběhu Simulace průchodu programem a to tak, že podmínka, jejíchž cílový objekt je akční bod nebo robot, zkontroluje, zda konkrétní zdroj a počet daný pravým operandem odpovídá zvolenému operátoru. Zdroje jsou pouze lokální, takže podmínka bere v úvahu pouze to, co je obsaženo v daném cílovém bodě.

Podmínka by neměla dovolit vytvořit více možných toků programu něž-li dva a instrukce více něž jeden. Tomuto problému bude věnován prostor v sekci Propojení.



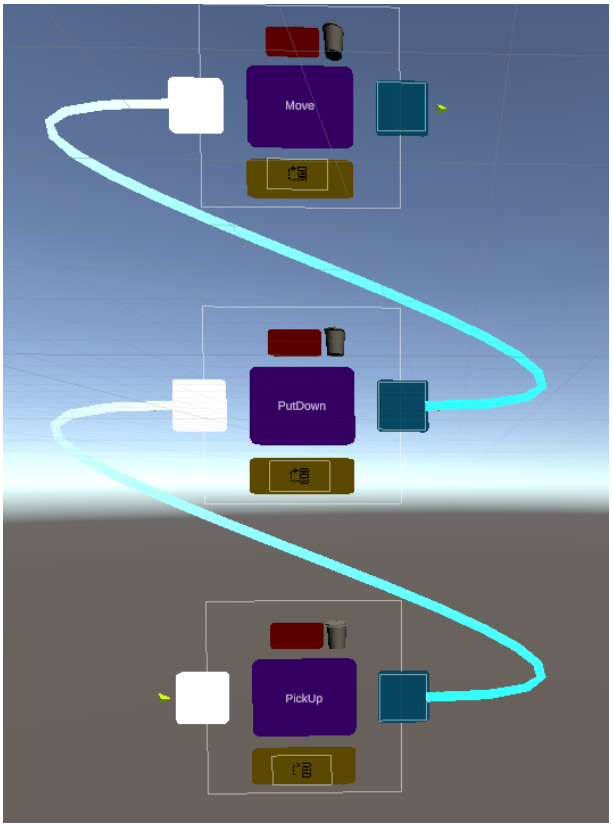
Obrázek 6.4: Obrázek vlevo je Unity prefab podmínky a obrázek vpravo zobrazuje modální okno pro vytvoření podmínky.

6.5 Propojení

Pro propojování instrukcí a podmínek je využit Unity asset *UIGraph*, který je sice 2D, ale dá se aplikovat i ve 3D. Podmínkou propojení je, aby obě strany byly tvořeny objektem typu *RectTransform*. Podmínky i instrukce mají prvky po stranách určené k propojování viz obr. 6.8. Tyto prvky jsou 3D a nejsou typu *RectTransform*. Řešením je umístit *RectTransform* přesně na místo objektu, tak aby je uživatel nebyl schopen zpozorovat a přidat skript *GraphNode*. Pro vytvoření propojení je potřeba znát oba koncové body. Zde se nabízí model Pozorovatel/Posluchač, kdy koncové body oznamují *GameManagerovi* změnu a ten se poté postará o samotné propojení pomocí statické metody *CreateConnection* obsažené v již zmíněném assetu.

Skripty obsažené *UIGraph* assetu bylo nutné mírně upravit, a to z důvodu příliš tlusté linky a také bylo nutné změnit průběh baziérovi křivky, která byla příliš dlouhá.

Propojení instrukcí či podmínek by nemělo být nedeterministické, tudíž pokud uživatel zvolí další propojení, tak propojení předešlé musí zaniknout. Je tedy nutné mít přístup ke všem propojením v aplikaci, což také umožňuje *UIGraph* asset. Díky funkcím *FindConnections* je možné vyhledat veškeré projení od daného bodu.



Obrázek 6.5: Propojení instrukcí pomocí křivek, které mají definované vstupní a výstupní body, prostřednictvím kterých je možné simulovat průchod programem.

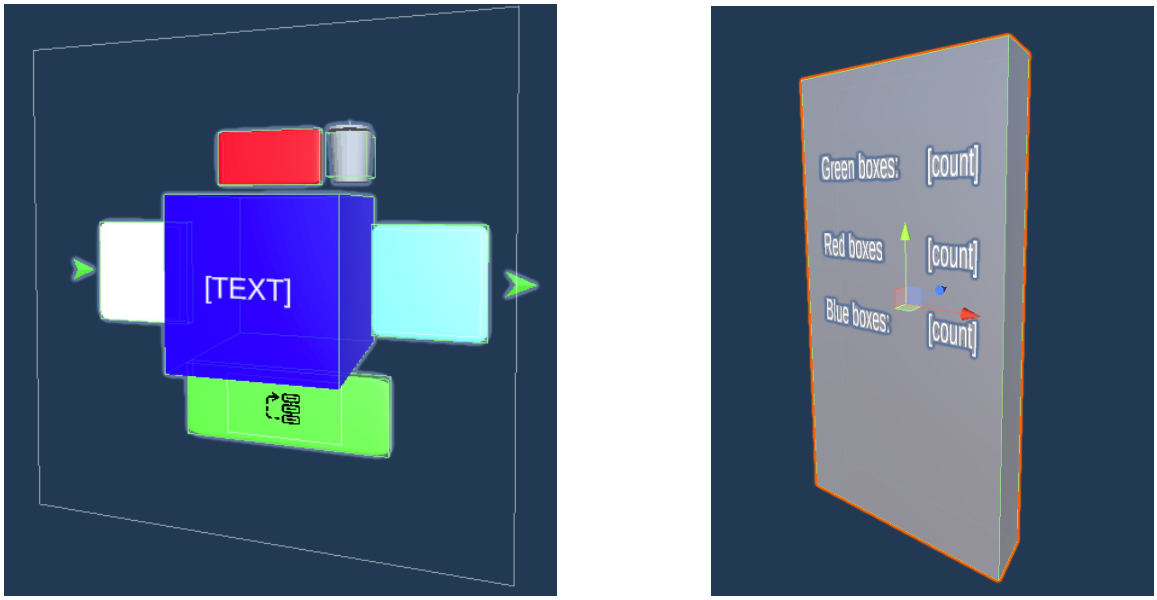
6.6 Simulace průchodu programem

Tato sekce je rozdělena na dvě části a to inicializační část a část běhu simulace průchodu programem. Veškeré předchozí elementy aplikace jsou využity při simulaci průchodu vytvořeného programu uživatelem. Tato experimentální aplikace nemá výstupem data, ale poskytuje zpětnou vazbu uživateli pomocí simulace průchodu programem na to, jaký program uživatel pro robota vytvořil a zda pracuje tak, jak si uživatel představoval.

6.6.1 Inicializační část

Ve velké části možných simulací průchodu je požadavek na práci s proměnnými. Z tohoto důvodu aplikace obsahuje dva druhy instrukcí, klasické, které byly popsány v sekci Instrukce a takzvané zdrojové instrukce viz obrázek vlevo 6.6, které byly přiblíženy v návrhu podmínky. Zdrojové instrukce se chovají identicky co se týče uživatelského rozhraní, ale obsahují odlišnou vnitřní logiku. Klasická instrukce může odebírat zdroj, přidat zdroj či zdroj nepoužít, ale zdroje v aplikaci již musí existovat. Není možné, aby sama nezdrojová instrukce si zdroj vyrobila. Zdrojové instrukce mají tu vlastnost, že přidají na konkrétní místo v aplikaci a ve scéně zdroj, konkrétně pět obecných položek např. (proložka, výrobek a jiné). Obecné z toho důvodu, že si uživatel pod tím může představit cokoliv. Aplikace obsahuje tři zdrojové instrukce pro jednoznačné odlišení mají nastavené barvy a i podmínky pracují s operandy nazvanými podle barev těchto instrukcí. Zdrojové instrukce nejsou pouze

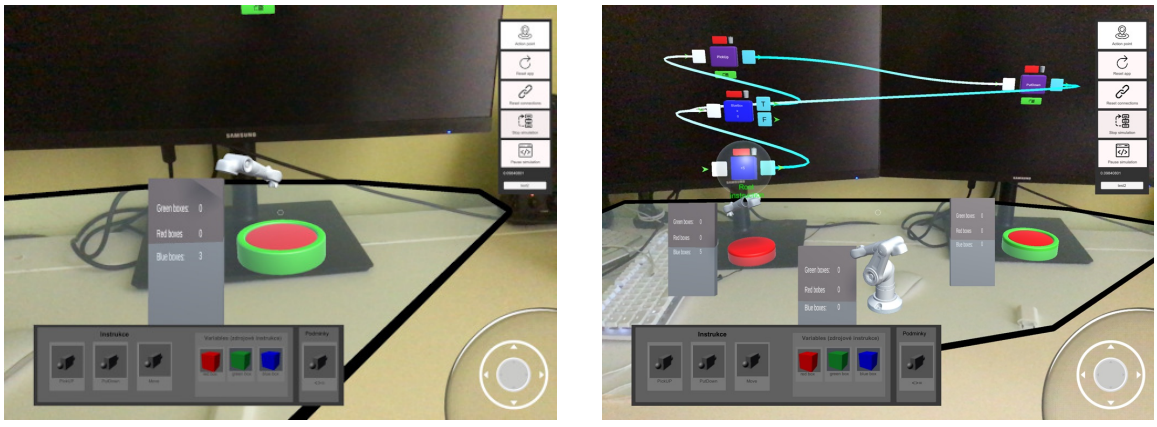
inicializační, jde je použít kdekoliv v rámci tvořeného programu uživatelem. Nutným požadavkem je zvolení počáteční instrukce, ta může být klasická nebo zdrojová, nikoliv však podmínka.



Obrázek 6.6: Obrázek vlevo je zdrojová instrukce a na obrázku vpravo je možné vidět panel jenž v průběhu simulace zobrazuje aktuální počet zdrojů na daném bodě ve scéně.

6.6.2 Běh simulace průchodu programem

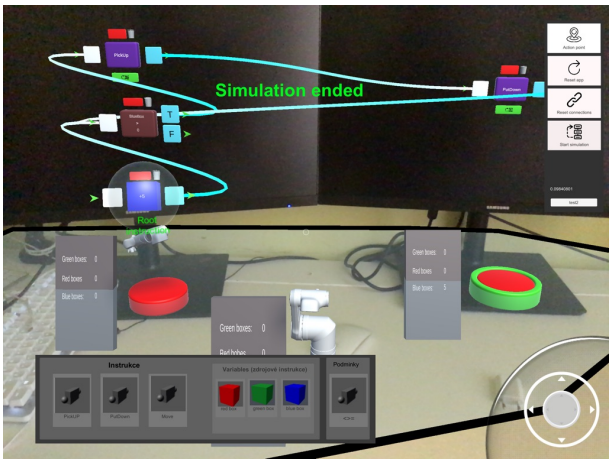
Jakmile je program sestaven je možné simulaci běhu programu spustit. Uživatel má možnost odstartovat běh tlačítkem na statickém menu, které dále umožňuje také běh pozastavit či zastavit úplně. Běh simulace je vytvořen pomocí takzvaných *coroutines*, což jsou neblo-



Obrázek 6.7: Obrázek vlevo zobrazuje robotickou paži nad akčním bodem a obrázek vpravo průběh simulace průchodu programem.

kující funkce, které jsou vhodné pro operace závislé na čase. Průběh simulace je uživateli

znázorňován zvýrazněním aktuálně zpracovávané instrukce či podmínky, kde se simulace na tři vteřiny pozastaví. K rozhodování, která instrukce či podmínka je následující krok, jsou použity funkce z assetu *UIGraph*, kde se v běhu simulace hledá následující krok pomocí propojení křivkami. Jak již bylo zmíněno instrukce i podmínky patří k určitému objektu ve



Obrázek 6.8: Konec simulace a zobrazení výsledného stavu zdrojů nad akčními body.

scéně (Robot, akční bod), to stejné platí o zdrojích v programu tvořeného uživatelem. Při hledání následující instrukce se zjistí jakému bodu instrukce náleží. Následuje také kontrola typu instrukce, na základě těchto dat se simulace adekvátně zachová.

Po spuštění se uživateli zobrazí u každého objektu 3D panel, který zobrazuje aktuální stav zdrojů na daném bodě ve scéně. Pro ještě lepší orientaci uživatele při běhu simulace průchodu programu je po spuštění vygenerován 3D objekt znázorňující paži robota, která se posouvá v rámci scény nad místa ve scéně (akční body), ve kterých se právě zpracovává instrukce. Posouvání paže robota je také implementováno pomocí *coroutines* tvořené již v běžící *coroutines*.

V případě nekonečných cyklů ať už úmyslných či nikoliv, je možné simulaci zastavit. Uživatel si může simulaci pozastavit a program přetvořit. Program je možné přetvářet i za běhu simulace, zde je ale nebezpečí možnosti úpravy právě simulované instrukce a běh tak ohrozit, z tohoto důvodu je implementována možnost pozastavení.

Kapitola 7

Testování

Tato kapitola je rozdělena do tří částí, první část je seznamovací a obsahuje přípravu pro testování, druhá část popisuje průběh testování jednotlivých úloh a třetí část zhodnocení výsledků.

Pro testování byly vytvořeny dvě testovací úlohy, které ověří funkčnost a obtížnost použití aplikace. První úloha je seznamovací, skládá se z několika dílčích úloh, kde si účastník testování vyzkouší ovládání rozhraní aplikace. Druhá úloha již bude konkrétní zadání pro možné reálné využití v praxi.

Účastníci byli roztříděni do dvou skupin podle toho, zda jsou technicky zaměřeni či nikoliv. Členům obou skupin byl předložen dotazník *System Usability Scale (SUS)* a jeho výsledky budou zhodnoceny ve třetí části kapitoly.

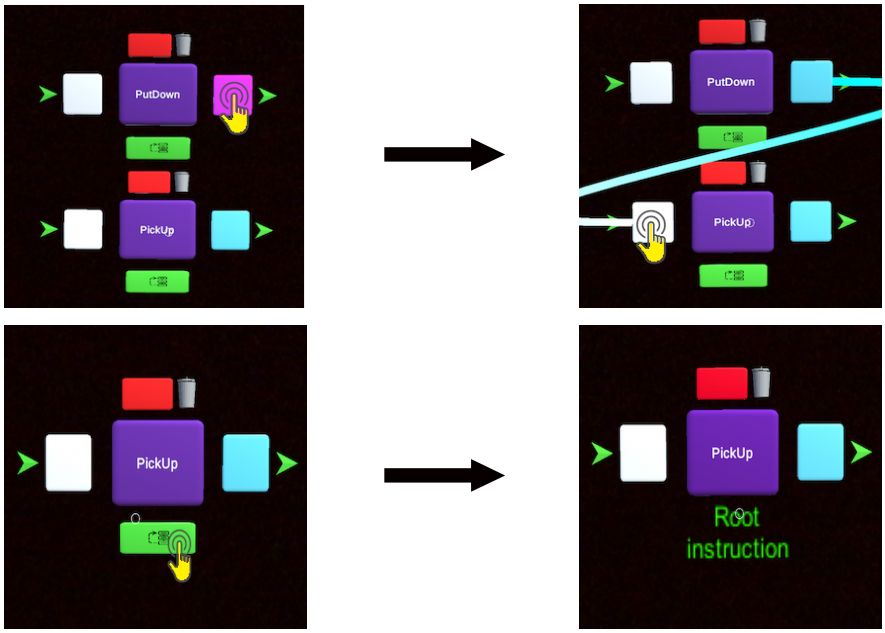
Veškeré úlohy připravené pro testování jsou lehčího charakteru, protože zde není účelem testovat programovací schopnosti účastníků, ale pochopitelnost a použitelnost aplikace. I když jsou obě úlohy lehčí, tak druhý úkol již pracuje s vícero proměnnými, a tak již není určen pro testovací skupinu číslo jedna, ale pouze pro skupinu druhou.

Účastník	**Pohlaví**	**věk**	**Vzdělání**	**Skupina**
A	Ž	30	VŠ	**skupina 1**
B	M	26	SŠ	
C	M	27	VŠ	**skupina 2** (IT vzdělání)
D	M	27	VŠ	
E	M	26	VŠ	
F	M	27	VŠ	

Tabulka 7.1: Základní údaje všech účastníků testování.

7.1 Seznamovací test

Každému účastníkovi testování byla aplikace nejdříve přiblížena vysvětlením na co se aplikace používá, co obsahuje a jaký je její výstup. Dále si účastníci pod vedením vyzkoušeli všechny úkony, které aplikace dokáže. Následně byl účastníkům představen první testovací úkol, který byl velice jednoduchý, protože jej musí být schopni vytvořit i členové první skupiny (bez IT zaměření). Na obrázcích viz 7.1 je možné vidět průběh seznamovacího testování některých aspektů aplikace, konkrétně test vytvoření spoje mezi instrukcemi a tím



Obrázek 7.1: Ukázka základních operací aplikace v rámci seznamovacího testu.

vytvoření programové sekvence. Obrázek 7.1 dole znázorňuje nastavení počáteční instrukce programu. Průběh seznamovacího testování byl prováděn v přesném pořadí, které je možné vidět níže. Účastník testování tak měl možnost si vyzkoušet konstrukci programu pod vedením a klást otázky, které ho při seznamovacím testování napadly.

Postup ukázky funkcionalit:

1. přiblížení aplikace účastníku testování,
2. detekce plochy a umístění robota,
3. vložení první instrukce,
4. modifikace polohy instrukce pomocí prstu a joysticku,
5. vytvoření akčního bodu,
6. přidání instrukce k akčnímu bodu,
7. propojení instrukcí,
8. modifikace polohy akčního bodu,
9. přidání zdrojové instrukce,
10. vytvoření první podmínky,
11. zvolení počáteční instrukce,
12. spuštění simulace průchodu programem.

7.2 Průběh testování

Ke každému úkolu byla účastníkům testování slovně i vizuálně představena možná konstrukce programu, kterou však účastník již musel být schopen vytvořit v aplikaci sám. Každému účastníkovi byla poskytnuta pomoc v případě, že si neví rady a případná intervence byla zaznamenána. Při testování byl měřen čas pro zjištění v jaké míře je aplikace pochopitelná a jaká skupina ji ovládá lépe či hůře.

7.2.1 Test - Základní

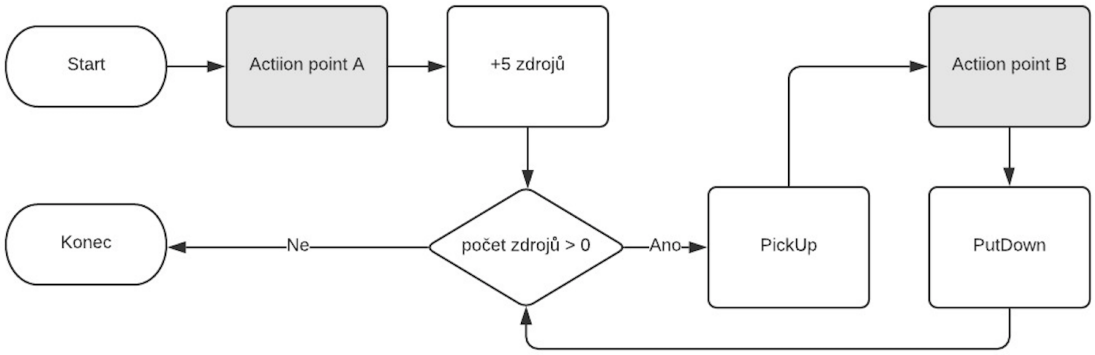
Tento test byl navrhnut, aby otestoval především intuitivnost a použitelnost aplikace na triviálním zadání. Tato experimentální aplikace není určena pro tvorbu komplexních a složitých vizuálních programů, ale měla by být schopna vytvořit základní konstrukce pro programování robota. Základní test je určen pro všechny účastníky testování a jeho obtížnost je také podle toho nastavena.

Zadání

Vytvořte vizuální reprezentaci programu pro robota, jenž bere předměty z místa A a přenáší je na místo B a po pěti přenesených kusech skončí.

Doporučená konstrukce

Na obrázku 7.2 je znázorněna možná konstrukce programu. Každý účastník testu měl k dispozici tento diagram a bylo testováno jakým způsobem se s tímto úkolem vypořádal, jak dlouho mu tvorba trvala a kolikrát mu byla poskytnuta pomoc.



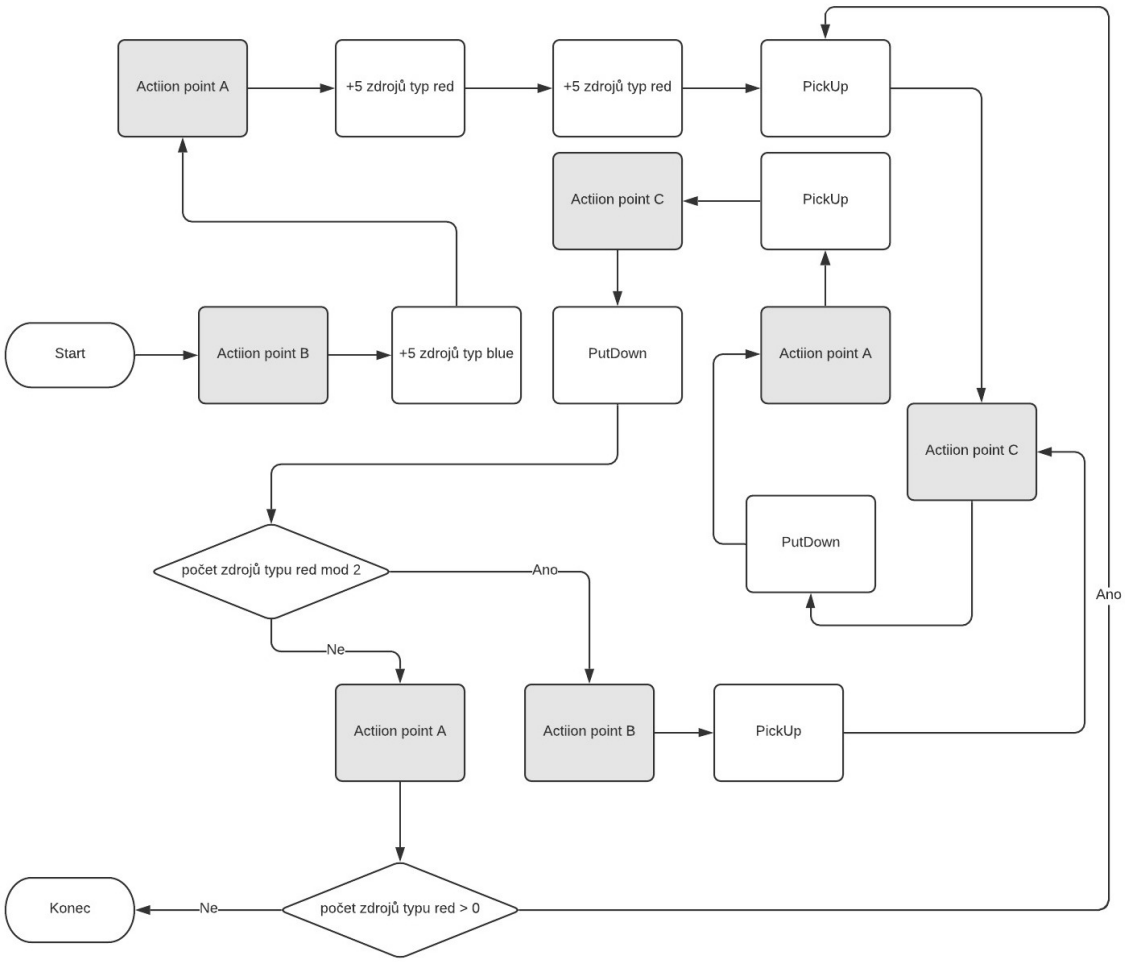
Obrázek 7.2: Diagram možného sestavení testovacího programu.

7.2.2 Test - Zásobovací robot

Vytvořte vizuální reprezentaci programu pro robota, který vyzvedává předměty z místa A a přenáší je na místo B, kde jej položí a na každý druhý předmět položí prokládací desku. Přenesených předmětů musí být 10 a proložka je i nad poslední dvojicí předmětů. Počet a umístění akčních bodů je libovolný.

Doporučená konstrukce

Nejprve je zapotřebí umístit takzvané zdrojové instrukce, které naplní dané body ve scéně určitými zdroji. Poté je již možné umisťovat podmínky, pro které jsou nutné tyto zdrojové instrukce. Na obrázku 7.3 je znázorněna možná konstrukce programu.



Obrázek 7.3: Diagram sestavení programu.

7.3 Výsledky

Účastníci testování vytvořili funkční řešení v solidním čase. Velmi pozitivní byla skupina, která nemá IT vysokoškolské vzdělání, ale určitou technickou zdatnost má. Subjekt B byl schopen vytvořit základní program s využitím vícero rad, ale bez použití pomocného diagramu. Následný těžší program zvládl také bez pomocného diagramu i takřka bez jakékoliv další pomoci. Subjekt A, který je ženského pohlaví si troufl pouze na lehčí ze dvou testů, ale tvorbu zvládl a aplikaci zhodnotil jako naučnou a zábavnou.

Skupina s IT vzděláním překvapivě dosahovala podobných či horších výsledků. Tento zajímavý výsledek může být ovlivněn mírou vysvětlení, která nebyla tak podrobná jako

u skupiny 1, což mohlo hrát roli. Účastníci testování ze skupiny 2 také měli problémy s jinou reprezentací programu a trvalo jim déle se s touto situací vyrovnat, protože byli zvyklí programovat klasickým způsobem.

Měření	**A**	**B**	**C**	**D**	**E**	**F**	**Průměr**
Použitelnost	87.5	82.5	82.5	82.5	85	62	**80.3**
Úkol č. 1 [min]	4.6	4.1	5.4	4.3	4.24	3.46	**4.35**
Počet intervencí	5	5	3	3	4	4	**4**
Úkol č. 2 [min]	-	5.7	7.2	6.3	6.5	6	**6.34**
Počet intervencí	-	1	2	2	3	2	**2**

Tabulka 7.2: Tabulka výsledků testování. Sloupce značí jednotlivé subjekty a řádky konkrétní test.

Bylo vypozorováno, že křivka učení je i v případě tvorby složitějšího programu velmi strmá, což je pozitivní zpráva. Na základě odpovědí z dotazníku SUS je možné usoudit, že aplikace má potenciál a je možné na ni v budoucnu navázat.

Po testování všichni zúčastnění zodpověděli dotazník, který obsahoval otázky, na které odpovídali známkováním od 1 do 5, kde 1 je nejhorší možné hodnocení a 5 nejlepší.

Otázky byly následující:

1. Je aplikace užitečná?
2. Je zvolený způsob tvorby programu vhodný?
3. Jsou navržené instrukce a podmínky vhodným způsobem?
4. Jsou vhodně navrženy lokální proměnné (zdrojové instrukce) pro tvorbu programu?
5. Je povedené grafické zpracování aplikace?

Účastník	**1**	**2**	**3**	**4**	**5**
A	5	5	4	4	5
B	5	4	4	5	5
C	4	4	5	5	5
D	4	4	4	4	5
E	4	5	4	4	5
F	5	4	4	3	5
Průměr	**4.5**	**4.3**	**4.2**	**4.2**	**5**

Tabulka 7.3: Tabulka odpovědí účastníků testování na dotazník uvedený výše. Sloupce značí číslo otázky a řádky jednotlivé účastníky testování. Hodnoty značí 1 - rozhodně ne, 5 - rozhodně ano

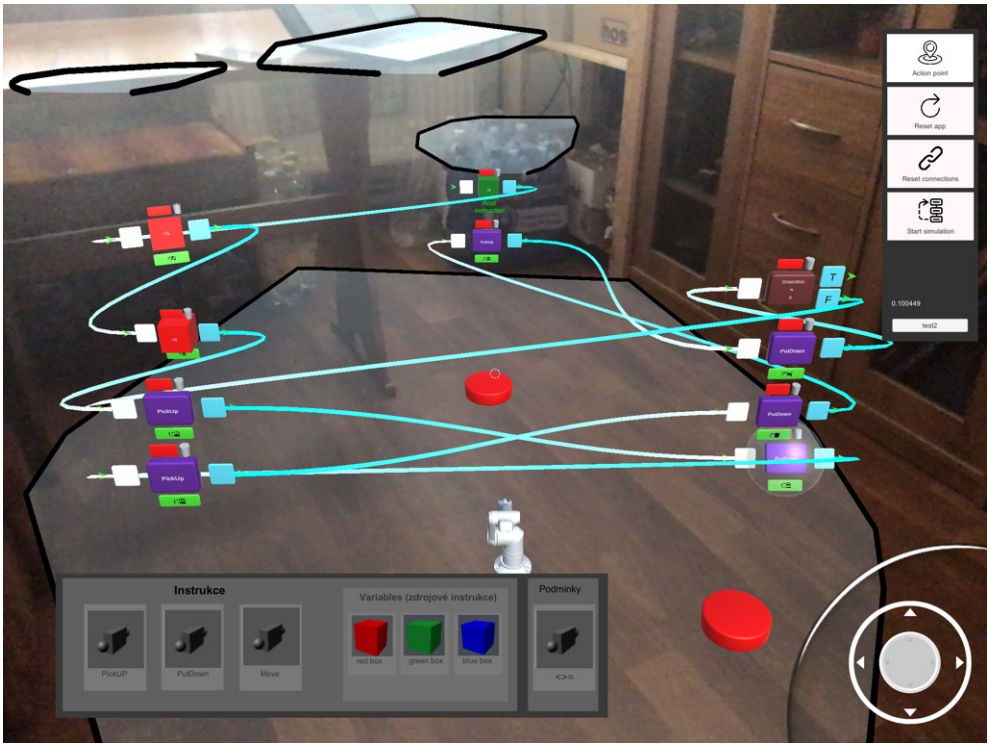
I zde byla aplikace hodnocena kladně, ačkoli řešení lokálních proměnných dělalo některým účastníkům potíže a na hodnocení se tento aspekt aplikace mírně projevil.

7.4 Záběry z testování

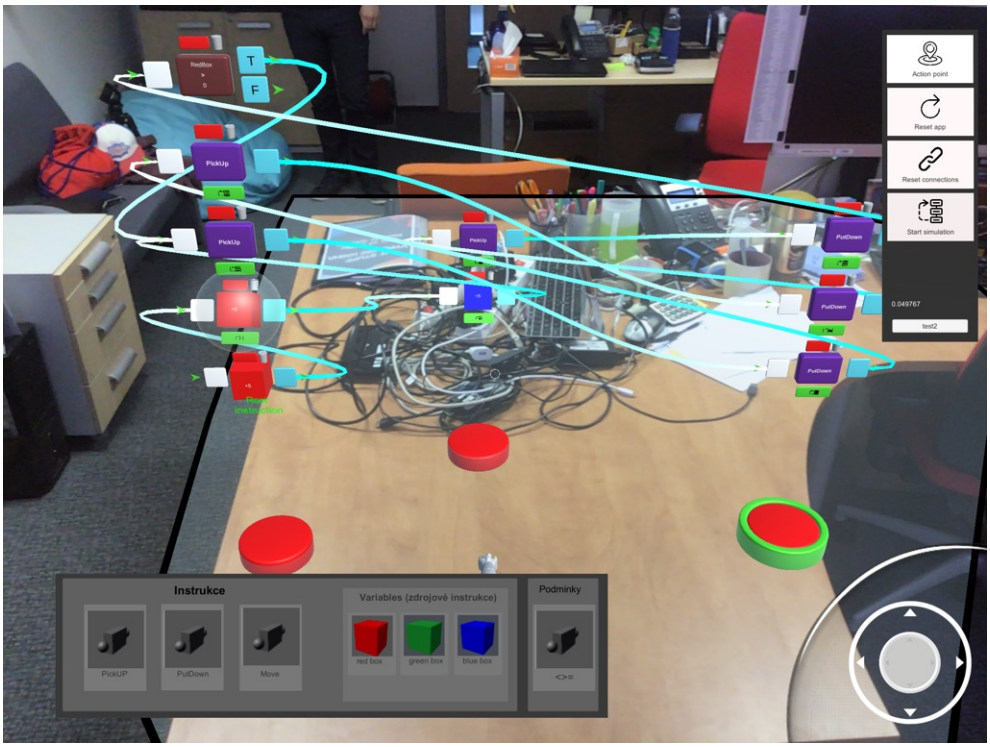
Průběh testování byl zaznamenáván a některé z pořízených záběrů zde budou k nahlédnutí. Testování probíhalo kvůli pandemii Covid-19 mimo laboratorní prostředí. Jednotlivý účastníci testování si volili místo, na kterém budou schopni aplikaci otestovat. Na obrázcích 7.4 a 7.5 je možné vidět některé účastníky testování. Testovací úkol číslo dva, který byl určen pro skupinu s IT vzděláním nakonec v nejlepším čase a s originálním řešení vytvořil subjekt A a řešení je možné vidět na obrázku 7.6.

Obrázek 7.4: vlevo: subjekt E, uprostřed: subjekt F, vpravo: subjekt C

Obrázek 7.5: Testování se subjektem D.



Obrázek 7.6: Výsledek testovacího úkolu č. 2 subjektu A.



Obrázek 7.7: Výsledek testovacího úkolu č. 2 subjektu F.

Kapitola 8

Závěr

Cílem mé práce bylo navrhnout a vytvořit vlastní řešení aplikace pro programování robotických akcí. Při studiu jsem se nejvíce zaměřil na technologii AR Foundation a Unity3D.

V rámci studia rozšířené reality jsem testoval její tvorbu v Unity a zjišťoval možné limity, které by mohly budoucí vývoj projektu ohrozit. AR Foundation má určitá omezení v image trackingu a v 3D object trackingu oproti standartnímu ARKitu nebo ARCoru, ale při testování jsem se s výraznou limitací nesetkal.

Vzhledem k aplikaci rozšířené reality do mobilních zařízení a nikoliv pro chytré brýle, bylo nutné na to přizpůsobit také uživatelské rozhraní. Při návrhu se ukázalo, že aplikace AR prvků, které jsou často využívané pro chytré brýle se jeví pro mobilní zařízení z pohledu uživatele jako nepraktické. Proto bylo zvoleno řešení nabídky ukotvené na displeji. Samotné programování robotických akcí je navrženo virtuálními prvky, které jsou umístěné na daných místech v reálné scéně. Poměr využití klasického menu a virtuálních prvků ve scéně se na základě testování ukázal jako vhodně zvolený.

Byla vytvořena experimentální aplikace, jenž prostřednictvím virtuálních prvků reprezentující instrukce a podmínky umožňuje tvořit vizuální program pro robota. Aplikace detekuje plochu v reálném světě a umožňuje usazení virtuální reprezentace robota na tuto plochu. Prostřednictvím aplikace je možné průchod programem simulovat a otestovat tak, zda program dělá to co si uživatel představoval.

Vizualizace instrukcí a podmínek obdržela od účastníků testování velmi dobré hodnocení, zatímco navržený způsob zdrojových instrukcí některým účastníkům dělal problémy. Účastníci testování z první skupiny (bez IT vzdělání) hodnotili aplikaci jako naučnou a zábavnou. Dokonce jeden účastník z této skupiny dosáhl nejlepšího času při tvorbě složitějšího programu. Členové druhé skupiny (s IT vzděláním) dosahovali podobných či horších výsledků. Dle zpětné vazby od účastníků je možné konstatovat, že aplikace je pozitivněji vnímána od účastníků první skupiny, protože členové druhé skupiny měli problém s jiným způsobem programování než byli zvyklí, což členům první skupiny nedělalo žádný problém a snadněji se tak v aplikaci orientovali.

Tato experimentální aplikace je spíše demonstrativního charakteru s využitím rozšířené reality a není napojena na žádný robotický systém. Možným vylepšením aplikace by mohlo být propojením s API, a tím konkretizování aplikace pro určitého robota s danou funkcionalitou. Aplikace má také nedostatek ve změně pohledu kamery ve 3D scéně, který by mohl být v budoucnu vyřešen. Dalším aspektem na zvážení je využití chytrých brýlí na místo tabletu.

Literatura

[1] Anuroop Katiyar, K. K. a Garg, C. Marker Based Augmented Reality. *I-Manager's Journal on Computer Science*. Nové Dillí, Indie: Krishi Sanskriti Publications. 2015, roč. 2, č. 3, s. 441–445. ISSN 23939907.

[2] Aukstakalnis, S. *Practical Augmented Reality*. Boston, Massachusetts, USA: Addison-Wesley Professional, 2016. ISBN 9780134094328.

[3] Bermudez, F. F., Ward, S., Diaz, C. S., Radkowski, R., Garrett, T. et al. *Comparison of Natural Feature Descriptors for Rigid-Object Tracking for Real-Time Augmented Reality*. Ames, State of Iowa, USA: Iowa State University, January 2014.

[4] Choi, S., Eakins, W., Rossano, G. a Fuhlbrigge, T. Lead-through robot teaching. In: *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. Piscataway, New Jersey, USA: IEEE, 2013, s. 1–4. ISBN 9781467362238.

[5] Ens, B., Anderson, F., Grossman, T., Annett, M., Irani, P. et al. Ivy: Exploring Spatially Situated Visual Programming for Authoring and Understanding Intelligent Environments. In: *Proceedings of the 43rd Graphics Interface Conference*. Waterloo, CAN: Canadian Human-Computer Communications Society, 2017, s. 156–162. GI '17. ISBN 9780994786821.

[6] Erin, P. *Creating Augmented and Virtual Realities*. Sebastopol, Kalifornie, USA: O'Reilly, 2019. ISBN 9781492044192.

[7] Garrido Jurado, S., Muñoz Salinas, R., Madrid Cuevas, F. a Marín Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*. Amsterdam, Nizozemsko: Elsevier. Červen 2014, roč. 47, s. 2280–2292.

[8] Jost, B., Ketterl, M., Budde, R. a Leimbach, T. Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One? In: *2014 IEEE International Symposium on Multimedia*. Piscataway, New Jersey, USA: IEEE, Dec 2014, s. 381–386. ISBN 978-1-4799-4311-1.

[9] Krause, K. W. *ROBOTIC SYSTEM WITH TEACH PENDANT*. U.S. Patent 6560513, May. 2003.

[10] Krystian Babilinski, J. L. *Augmented Reality for Developers*. Birmingham, UK: Packt Publishing, 2017. ISBN 9781787286436.

[11] Lanham, M. *Learn ARCore - Fundamentals of Google ARCore*. Birmingham, UK: Packt Publishing, 2018. ISBN 9781788830409.

[12] Li, S., Xu, C. a Xie, M. A Robust O(n) Solution to the Perspective-n-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Piscataway, New Jersey, USA: IEEE. 2012, roč. 34, č. 7, s. 1444–1450. ISSN 0162-8828.

[13] Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*. Berlín, Německo: Kluwer Academic Publishers. 2004, roč. 60, č. 2, s. 91–110. ISSN 0920-5691.

[14] Matt Smith, C. Q. *Unity 4.x cookbook : over 100 recipes to spice up your Unity skills*. Birmingham, UK: Packt Publishing, 2013. ISBN 978-1-84969-042-3.

[15] Mitsi, S., Bouzakis, K.-D., Mansour, G., Sagris, D. a Maliaris, G. Off-line programming of an industrial robot for manufacturing. *The International Journal of Advanced Manufacturing Technology*. New York City, New York, USA: Springer. Srpen 2005, roč. 26, s. 262–267.

[16] Murray, J. W. *C Game Programming Cookbook for Unity 3D*. Boca Raton, Florida, USA: CRC Press, 2014. ISBN 978-1-4665-8140-1.

[17] Neto, P. a Mendes, N. Direct off-line robot programming via a common CAD package. Amsterdam, Nizozemsko: Elsevier B.V. 2013, roč. 61, č. 8, s. 896–910. ISSN 0921-8890.

[18] Ostanin, M. a Klimchik, A. Interactive Robot Programing Using Mixed Reality. *IFAC PapersOnLine*. Amsterdam, Nizozemsko: Elsevier Ltd. 2018, roč. 51, č. 22, s. 50–55. ISSN 2405-8963.

[19] Pan, Z., Polden, J., Larkin, N., Van Duin, S. a Norrish, J. Recent progress on programming methods for industrial robots. *Robotics and Computer Integrated Manufacturing*. Amsterdam, Nizozemsko: Elsevier Ltd. 2012, roč. 28, č. 2, s. 87–94. ISSN 0736-5845.

[20] Schmalstieg, D. *Augmented reality : principles and practice*. Boston, Massachusetts, USA: Addison Wesley, 2016. ISBN 978-0-321-88357-5.

[21] S.R.O., X. *Xaratec* [online]. Xaratec, prosinec 2019 [cit. 2019-12-12]. Dostupné z: http://www.dorps.sk/en/portfolio_xaratec/.

[22] Stottinger, J., Hanbury, A., Sebe, N. a Gevers, T. Sparse Color Interest Points for Image Retrieval and Object Categorization. *IEEE Transactions on Image Processing*. Piscataway, New Jersey, USA: IEEE. 2012, roč. 21, č. 5, s. 2681–2692. ISSN 1057-7149.

[23] Wang, W. *Beginning ARKit for iPhone and iPad: Augmented Reality App Development for iOS*. New York City, New York, USA: Apress, 2018. ISBN 9781484241028.

[24] Xiao, C. a Lifeng, Z. Implementation of mobile augmented reality based on Vuforia and Rawajali. In: *2014 IEEE 5th International Conference on Software Engineering and Service Science*. Piscataway, New Jersey, USA: IEEE, 2014, s. 912–915. ISBN 9781479932788.

Příloha A

Obsah DVD

K diplomové práci je přiloženo DVD s touto adresářovou strukturou:

- dp.pdf - tento dokument,
- Demovideo.mp4 - prezentační video,
- README - návod pro spuštění,
- xsabel01-diplomova-prace.unitypackage - zdrojový kód aplikace,
- Scripts - složka se skripty.