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

PROMÍTANÁ ROZŠÍŘENÁ REALITA PRO ROBOTICKÉ PRACOVIŠTĚ

SPATIAL AUGMENTED REALITY FOR ROBOTIC WORKPLACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR KRUPICA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZDENĚK MATERNA, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

140497

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Krupica Petr**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Promítaná rozšířená realita pro robotické pracoviště**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2022/23

Zadání:

1. Proveďte rešerši využití projekce pro robotická pracoviště.
2. Seznamte se se systémem ARCOR2 umožňujícím programování robotů v rozšířené realitě a možnostmi jeho uživatelského rozhraní AREditor.
3. Navrhněte promítané uživatelské rozhraní, které bude vhodně doplňovat AREditor.
4. Navržené řešení implementujte.
5. Proveďte uživatelské testování.
6. Zdrojové kódy a dokumentaci publikujte na GitHubu.
7. Vytvořte video prezentující vaši práci, její cíle a výsledky.

Literatura:

- MATERNA Zdeněk, et al. Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation. In: *RO-MAN 2018 - 27th IEEE International Symposium on Robot and Human Interactive Communication*. NanJing: Institute of Electrical and Electronics Engineers, 2018, s. 80-87. ISBN 978-1-5386-7980-7.
- Mengoni, Maura, et al. "Spatial Augmented Reality: An application for human work in smart manufacturing environment." *Procedia Manufacturing* 17 (2018): 476-483.
- Avalle, Giancarlo, et al. "An augmented reality system to support fault visualization in industrial robotic tasks." *IEEE Access* 7 (2019): 132343-132359.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3.

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

Vedoucí práce:	**Materna Zdeněk, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Tato práce se zabývá vývojem promítaného uživatelského rozhraní a kalibrací systému složeného z kamery a projektoru. Uživatelské rozhraní doplňuje aplikaci AREditor při programování kolaborativních robotů v systému ARCOR2. Toto rozhraní umožňuje uživateli přesnější vkládání objektů do scény díky promítání obrysu objektů přímo na pracovní plochu. Pro zajištění přesnosti projekce byla použita kalibrační metoda typu SLSCalib. Touto metodou byla zjištěna pozice a rotace projektoru vůči kameře a projekční matice, díky které je následně vypočítána pozice virtuálních objektů v projekční rovině. Uživatelské rozhraní bylo implementováno v herním enginu Unity. Výsledná aplikaci zvyšuje efektivitu práce s aplikací AREditor a zpříjemňuje uživatelskou zkušenost při správě robotického pracoviště.

Abstract

This work focuses on the development of a projected user interface and the calibration of a system composed of a camera and a projector. The user interface complements the AREditor application for programming collaborative robots in the ARCOR2 system. This interface allows the user to place objects in the scene more accurately by projecting the outlines of objects directly onto the work surface. To ensure the accuracy of projection, a SLSCalib calibration method was used. This method determined the position and rotation of the projector relative to the camera and the projection matrix, which is later used to calculate the position of virtual objects in the projection plane. The user interface was implemented in the Unity game engine. The resulting application improves the efficiency of working with the AREditor application and enhances the user experience in managing the robotic workspace.

Klíčová slova

rozšířená realita, unity, projekce, kalibrace projektoru, kinect, testování

Keywords

augmentet reality, unity, projection, projector calibration, kinect, testing

Citace

KRUPICA, Petr. *Promítaná rozšířená realita pro robotické pracoviště*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zdeněk Materna, Ph.D.

Promítaná rozšířená realita pro robotické pracoviště

Prohlášení

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

.....
Petr Krupica
8. května 2023

Poděkování

Chtěl bych poděkovat vedoucímu práce, Ing. Zdeňkovi Maternovi Ph.D., za čas věnovaný pravidelným konzultacím a ochotu kdykoliv poradit a nasměrovat správným směrem

Obsah

1 Úvod **3**

2 Rozbor problematiky **4**
2.1 Kolaborativní robot . . . 4
2.2 Rozšířená realita . . . 4
2.3 Promítané uživatelské rozhraní . . . 5
2.4 ARCOR2 . . . 6
2.4.1 Architektura . . . 6
2.4.2 Serverová část . . . 7
2.4.3 AREditor . . . 8
2.4.4 Komunikace . . . 11
2.5 Kalibrace . . . 12
2.5.1 Kalibrační data . . . 14
2.5.2 Kalibrační metody . . . 15

3 Návrh **19**
3.1 Cíl . . . 19
3.2 UI . . . 19
3.3 Aplikace . . . 22
3.4 Kalibrace . . . 23

4 Implementace **24**
4.1 Kalibrace . . . 24
4.2 Komunikace . . . 25
4.3 Implementace návrhu UI . . . 26
4.4 Architektura . . . 28
4.4.1 Scéna . . . 28
4.4.2 Aplikační logika . . . 29
4.4.3 Herní objekty . . . 31

5 Testování **34**
5.1 Poznámky . . . 34
5.2 Vyhodnocení a plány do budoucna . . . 36

6 Závěr **38**

Literatura **39**

Seznam obrázků

2.1 Architektura systému Arcor2. Převzato z [8]. 7
2.2 Hlavní obrazovka AREditoru. Převzato z [1]. 8
2.3 Kolizní objekty. Převzato z [1]. 9
2.4 Modifikace akčních objektů. Převzato z [1]. 9
2.5 Přidávání akce. Převzato z [1]. 10
2.6 Vizualizace přechodu mezi akcemi. 11
2.7 Pinhole camera model. Převzato z [9]. 12
2.8 Radiální zkreslení. Převzato z [9]. 13
2.9 Tangenciální zkreslení. Převzato z [9]. 14
2.10 Příčina tangenciálního zkreslení. Převzato z [9]. 14
2.11 Kalibrační parametry. Převzato z [9]. 14
2.12 Převod do souřadnicového systému kamery. Převzato z [9]. 15
2.13 Geometrické vyjádření SLS. Převzato z [18]. 17
2.14 a) Šachovnice použita v ProcamClib and SLSCalib ; b) A 3D kalibrační objekt použit v 3DCalib metodě. Převzato z [18]. 18

3.1 Návrh vytváření scény 20
3.2 Návrh rotace objektu 20
3.3 Návrh vkládání akčních bodů 21
3.4 Návrh vkládání akcí 21
3.5 Návrh provádění akce 22
3.6 Návrh upozornění 22

4.1 Pracoviště připravené ke kalibraci. 24
4.2 Ovládací prvky 26
4.3 Reprezentace scény. 26
4.4 Reprezentace projektu. 27
4.5 Nastavení barev a modelu. 31
4.6 Úprava rotace KinectAzure. 32
4.7 Vizuální reprezentace akčních objektů. 33

5.1 Testování aplikace. 34
5.2 Osvícená šachovnice. 35
5.3 Osvícení kalibračního vzoru znemožňující kalibraci. 35

Kapitola 1

Úvod

V současné době se rozšířená realita stává stále populárnější v oblasti průmyslu a robotiky. Tato technologie umožňuje vytvářet virtuální prvky, které jsou následně zobrazeny v reálném prostředí.

Jednou z aplikací rozšířené reality v průmyslu je systém ARCOR2, který slouží pro správu robotického pracoviště. Systém lze ovládat pomoci mobilní aplikace s názvem AREditor. Díky této aplikaci se dá skrze mobilní zařízení manipulovat s virtuální scénou v reálném prostředí.

Tato bakalářská práce se zaměřuje na vývoj promítaného uživatelského rozhraní, které doplňuje aplikaci AREditor při práci se systémem ARCOR2[1]. Cílem práce je vytvořit promítané uživatelské rozhraní, které usnadní práci s robotickým pracovištěm pomocí prvků rozšířené reality. U aplikace AREditor byly vypozorovány nedostatky v oblasti orientace v prostoru a s vnímáním hloubky, dané 2D zobrazením pomocí tabletu. Tyto nedostatky budou doplněny promítáním pozice virtuálních objektů scény přímo na povrch reálného pracoviště.

V této práci jsou nejprve vysvětleny klíčové pojmy, jako je kolaborativní robot, rozšířená realita a systém ARCOR2, včetně jeho součástí. Následně je proveden rozbor problematiky kalibrace systému strukturovaného světla, který se skládá z projektoru a kamery, včetně významu jednotlivých kalibračních parametrů a jejich vlivu na výsledky kalibrace. Práce také porovnává různé metody získání těchto parametrů. Hlavní částí práce je popis návrhu a implementace promítaného uživatelského rozhraní, které bude použito při správě robotického pracoviště. Součástí práce je také návrh a implementace kalibrace systému, která je nezbytná pro správné fungování promítaného rozhraní. V poslední části práce jsou prezentovány výsledky vývoje, včetně uživatelského testování, a jsou zde uvedena doporučení pro budoucí vývoj tohoto uživatelského rozhraní.

[1] https://github.com/robofit/arcor2

Kapitola 2

Rozbor problematiky

Tato kapitola má čtenáře přiblížit problematice kolaborativních robotů, rozšířené reality a promítaných uživatelských rozhraní. Dále se věnuje systému ARCOR2, včetně aplikace AREditor, která je s ním spojena. Hlavním tématem této kapitoly jsou základní principy kalibrace kamery a projektoru, které jsou klíčové pro realizaci tohoto systému.

2.1 Kolaborativní robot

Kolaborativní robot (cobot) je speciální typ průmyslového robota, který je navržen tak, aby mohl bezpečně a efektivně spolupracovat s člověkem v továrnách, skladech a jiných pracovištích. Tito roboti vykazují vysokou a konstantní přesnost, jsou schopni pracovat nepřetržitě a snadno se adaptují do stávající výroby. Zavedení kolaborativních robotů do provozu může značně zvýšit efektivitu jakékoliv firmy. Automatizace procesů neznamenají pouze lepší využití lidského potenciálu, ale také eliminaci lidských chyb při montážních úkonech, spolehlivost při plánování dlouhodobých projektů a podstatné snížení celkových nákladů.

Narozdíl od klasických průmyslových robotů, kteří jsou zpravidla izolováni od lidí a pracují ve speciálních uzavřených prostorách, jsou kolaborativní roboti typicky vybaveni snímači a funkcemi umožňující detekci přítomnosti a pohybu lidí na pracovišti, které společně s přizpůsobeným chováním, rychlostí a silou minimalizují riziko úrazu na pracovišti. Tyto úpravy umožňují pracovat bezpečně v přítomnosti lidí bez nutnosti používání speciálních ochraných bariér nebo omezení přístupu.

V posledních letech došlo k velkému nárůstu zájmu o tuto technoligii. V současnosti se kolaborativní roboti využívají v mnoha různých průmyslovích odvětvích, jako jsou například automobilový průmysl, zpracování potravin, farmaceutický průmysl a mnoho dalších. Tito roboti jsou využíváni pro úkoly, jako je svařování, šroubování, lepení, umísťování předmětů nebo odměřování přesných měr například při míchání barev.

Tato sekce čerpá informace z článků [6, 7].

2.2 Rozšířená realita

Rozšířená realita (Augmented Reality - AR) je technologie, která umožňuje spojení mezi digitálním a fyzickým světem. Jedná se o technologii, která umožňuje uživatelům vidět digitální prvky, které byly vložené do reálného prostředí. Využívá se k tomu technologie zpracování obrazu a počítačového vidění pro rozpoznání prvků v reálném světě a technologie

pro zobrazení digitálních prvků. Existuje několik způsobů, jak lze digitální prvky zobrazovat v prostředí fyzického světa. Mezi nejpoužívanější způsoby patří zobrazení na obrazovce mobilního zařízení, projekce na reálné objekty, holografické zobrazení a head-up displeje [14].

Rozšířená realita má široké využití v mnoha odvětvích včetně průmyslu, zdravotnictví, vzdělávání, marketingu a zábavy. V průmyslu se využívá rozšířená realita například pro návrh a vizualizaci produktu a řízení výrobních procesů. V zdravotnictví se rozšířená realita využívá pro výuku lékařů, simulaci chirurgických zákroků a diagnostiku. [14]

Existují dva hlavní způsoby rozpoznání prostředí rozšířené reality. První způsob je použití takzvaných kotevních bodů. Tento přístup spojuje digitální prvky s reálnými markery, které jsou snímány kamerou a zpracovány programem, který počítá, jaké prvky se mají zobrazit. Další způsob nevyužívá kotevní body. V tomto případě se může využít rozpoznávání informací o poloce a orientaci kamery v reálném světě nebo rozpoznání reálných objektů pomocí algoritmů počítačového vidění.

Tato sekce čerpá informace z článku [14].

Promítaná rozšířená realita

Projection-based AR se používá tehdy, pokud jsou digitální prvky promítány přímo na reálné objekty pomoci projektoru. Výhodou tohoto přístupu je, že uživatel nepociťuje únavu v souvislosti s rozšířenou realitou, neboť nepotřebuje nosit žádné další zařízení. Další výhodou je možnost více uživatelů zobrazovat rozšířenou realitu současně.

Existují dva typy promítené rozšířené reality. Prvním je statická. V tomto případě je projekční plocha statická a stejně tak i promítaný obsah. Lze využít animace a digitální efekty, ale nelze zde dosáhnout změn digitálního obsahu či projekční plochy na základě nějaké události. V případě dynamické projekce může být projekční plocha pohyblivá a rozšířená realita může reagovat na její změny.

Promítaná rozšířená realita má využití například v oblasti reklamy a marketingu, vzdělávání, zábavy, průmyslu a designu.

Tato sekce čerpá informace z článku [3].

2.3 Promítané uživatelské rozhraní

Jde o formu uživatelského rozhraní, která využívá projektory k promítání informací na různé povrchy, jako jsou stěny, stoly nebo třeba lidské tělo.

Výhodou promítaného uživatelského rozhraní je, že poskytuje uživateli velkou pracovní plochu, která se snadno přizpůsobí prostředí, ve kterém se nachází. Umožňuje uživatelům interagovat s digitálními informacemi v reálném světě.

V kontextu promítaných uživatelských rozhraní se tradiční vstupní rozhraní, jako jsou klávesnice a myš, ukázaly jako nevhodné pro efektivní interakci [15]. Z tohoto důvodu jsou v tomto kontextu preferovány jiné metody uživatelského vstupu. Stávající systémy využívají metody ovládání gesty rukou, dotyková plocha nebo mobilní zařízení [12], [13]. Tyto metody umožňují uživateli intuitivní a přirozený způsob interakce se systémem. Mezi výhody těchto nových metod patří také větší pracovní prostor a větší flexibilita.

Důležitou výhodou promítaného uživatelského rozhraní je, že umožňuje sdílení informací mezi více uživateli, což může být užitečné při spolupráci na projektech nebo prezentování informací v týmu.

Nevýhodou promítaného uživatelského rozhraní může být omezená viditelnost v jasném denním světle nebo na nevhodném typu povrchu, jako jsou například lesklé nebo prosvítající

materiály. Dalším omezením je omezená schopnost zobrazit obsah kdekoliv v prostoru. Namísto toho může být zobrazen pouze na pevném povrchu, který není zastíněný jiným objektem.

Promítané uživatelské rozhraní má široké využití v mnoha odvětvích. V průmyslu se promítané uživatelské rozhraní mohou využít pro vizualizaci průmyslových procesů. V oblasti vzdělávání se promítané uživatelské rozhraní mohou použít pro zlepšení výuky a vytvoření interaktivních vzdělávacích materiálů. Ve zdravotnictví se promítané uživatelské rozhraní mohou uplatnit pro vizualizaci lékařských dat a informací, jako jsou například rentgenové snímky [15].

2.4 ARCOR2

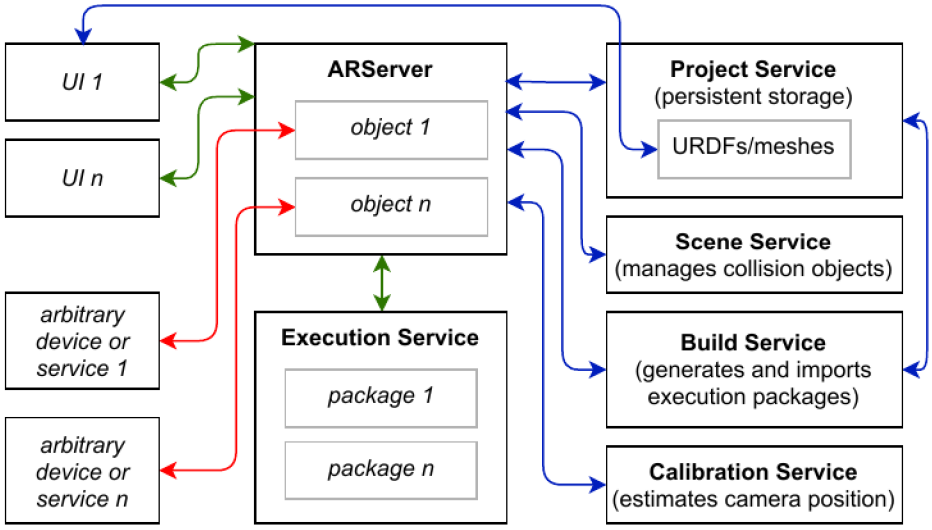
Arcor2 je systém, který umožňuje snadnou správu robotického pracoviště pomocí rozšířené reality, zaměřený na potřeby malých a středních podniků s vysokou variabilitou výroby. Tento framework umožňuje integraci jakéhokoliv typu robota, zařízení nebo služby. Modulární architektura systému umožňuje snadnou integraci nové funkcionality. Většina kolaborativních robotů využívaných většími firmami je obvykle naprogramována pouze jednou a poté plní svou jednu konkrétní funkci po dobu několika let. Menší společnosti jsou obvykle flexibilnější v oblasti produkce, což umožňuje využití robotů pro více úkolů a tím využít jejich plného potenciálu. Vzhledem k tomu by bylo výhodné umožnit koncovým uživatelům, kteří jsou odborníci v daném oboru, jednoduše programovat roboty [11].

2.4.1 Architektura

Systém je rozdělen na na klientskou a serverovou část. Klientskou část představuje aplikace pro OS Android vytvořená v Unity3D (AREditor) viz 2.4.3. Serverovou část představuje soubor nezávislých služeb založených na Pythonu viz 2.4.2. Momentálně se používá jedna implementace uživatelského rozhraní, aplikace pro tablet poskytující plnou funkcionalitu (AREditor). Záměrem je umožnit zapojení dalších, doplňkových rozhraní, které by poskytovaly pouze některé aspekty funkcionality. Příkladem takového rozhraní může být RGB LED páska indikující stav systému nebo rozhraní založené na detekci rukou. Proto musí být server schopen pracovat s více připojenými rozhraními i pro práci jednoho uživatele.

Jednotlivé rozhraní jsou připojena k serveru pomocí Websocketů, což umožňuje obousměrnou komunikaci, která je více popsána v kapitole 2.4.4. Server udržuje stav systému. Klienti s ním mohou manipulovat pomocí sady RPC a přihlásit se k odběru oznámení o změnách pomoci websocketu. Předpokládá se, že každé pracoviště běží vlastní instanci serveru a server proto udržuje pouze jednu relaci pro všechny uživatele. To znamená, že pokud jeden uživatel otevře projekt, stejný projekt se zobrazí i ostatním připojeným uživatelům. Aby mohlo systém využívat více uživatelů najednou, existuje mechanismus zamykání, který brání více uživatelům manipulovat se stejným prvkem.

Server také slouží jako proxy mezi kódem v jazyku Python a prostředím rozšířené reality, které je nezávislé na programovacím jazyce [11, 8].



Obrázek 2.1: Architektura systému Arcor2. Převzato z [8].

2.4.2 Serverová část

Server udržuje stav systému a aktuální relaci, dynamicky načítá pluginy a analyzuje dostupné akce jednotlivých objektů a jejich parametrů. Výsledkem této analýzy jsou JSON metadata, která jsou spolu s dalšími informacemi o stavu systému k dispozici uživatelským rozhraním. [2]

Serverová část se skládá z několika nezávislých služeb:

ARServer

ARServer je hlavní službou systému. Slouží jako centrální bod pro uživatelská rozhraní a zprostředkovává komunikaci s ostatními službami. Uživatelské rozhraní posílá na ARServer požadavky. ARServer tyto požadavky zpracuje a následně odpoví.

Project Service

Služba poskytuje perzistentní úložiště pro data relevantní pro pracoviště, včetně scén, projektů, typů objektů, modelů atd.

Scene Service

Služba, která je používána v případech, kdy je implementace založena na ROS (Robot Operating System). Služba má na starosti správu kolizních objektů.

Build Service

Služba vytváří pro specifický projekt izolovaný spustitelný balíček. Logika může být definována v prostředí rozšířené reality nebo může být poskytnuta v odděleném souboru. Při generování logiky se JSON reprezentace nejprve sestaví do formy abstraktního syntaktického stromu a následně se zkompiluje do kódu jazyka Python. Tato služba také generuje soubory s přídavnými třídami, jako například třídy, které usnadňují práci s akčními body.

Execution Service

Služba spravuje balíčky, které jsou sestaveny pomocí služby sestavení (Build Service). Její nejvýznamnější funkcionalitou je spouštění balíčků a během tohoto procesu služba streamuje informace o aktuálně prováděných akcích a jejich parametrech. Spuštění může být také pozastaveno a později obnoveno podle potřeby.

Calibration Service

Služba poskytuje metodu pro odhad polohy kamery na základě detekce ArUco markerů a metodu pro nastavení polohy robota pomocí RGBD kamery.

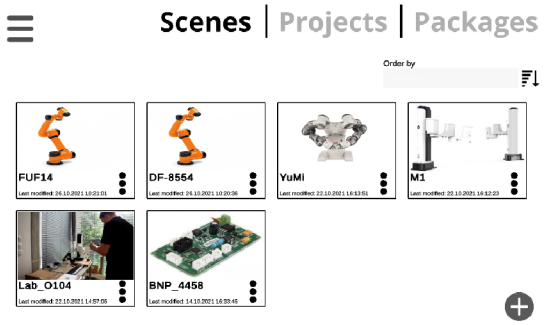
2.4.3 AREditor

AREditor je aplikace pro mobilní zařízení s podporou rozšířené reality. Uživatel může skrz tuto aplikaci interagovat s robotickým prostředím. Jejím cílem je umožnit koncovým uživatelům bez znalosti robotiky a programování snadno nastavit, popisovat, programovat a upravovat pracoviště pomocí rozšířené reality. Programování v AREditoru probíhá ve formě grafických prvků, které jsou zobrazovány přímo v prostředí a usnadňují vnímání prostorových vztahů. Aplikace umožňuje uživatelům popisovat robotické pracoviště, vytvářet program spojováním akcí a následné spuštění programu. [1]

Aplikace se skládá ze dvou částí. Hlavní obrazovka a editor.

Hlavní obrazovka

Pokud není otevřený žádný projekt, scéna, nebo balíček, zobrazí se uživateli hlavní obrazovka. Ta obsahuje seznam vytvořených scén, projektů a balíčků, s nimiž může uživatel vykonávat několik operací. Mezi tyto operaci patří přidání mezi oblíbené, duplikování, přejmenování, změna náhledového obrázku, smazání nebo otevření. Uživatel může mezi jednotlivými scénami a projekty libovolně přepínat. [1]



Obrázek 2.2: Hlavní obrazovka AREditoru. Převzato z [1].

Editor

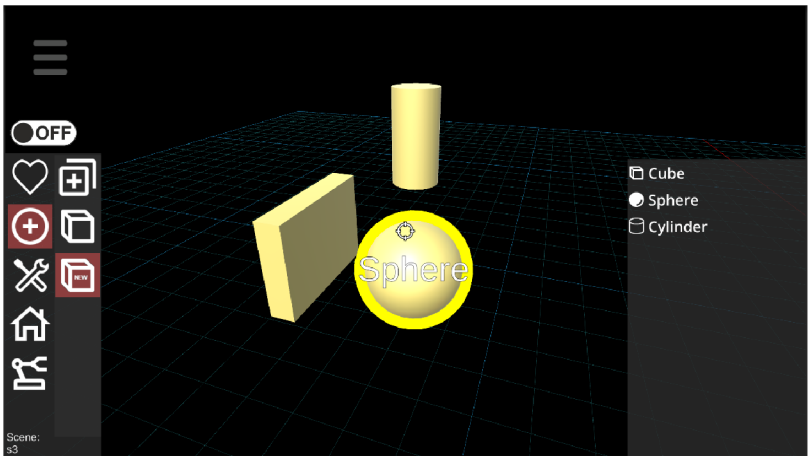
Editor je součástí aplikace, která se stará o jakoukoli 3D vizualizaci a interakci. Existuje Editor scény pro tvorbu a úpravu scén a Editor projektu pro tvorbu a úpravu projektů.

Nejprve je potřeba provést kalibraci rozšířené reality. Toho se docílí snímáním kalibrační značky kamerou.

Scéna

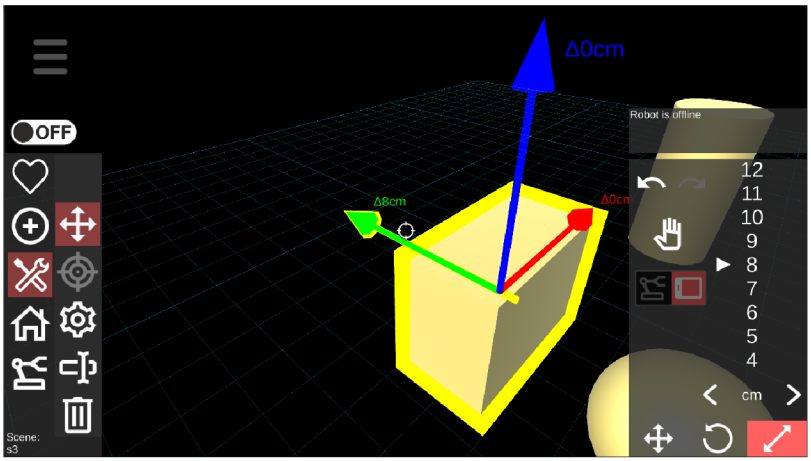
Při přípravě nového projektu je prvně potřeba definovat scénu. Scéna se skládá z akčních objektů, které lze přidávat, mazat, nebo modifikovat. Akční objekt popisuje nějaký skutečný díl, zařízení nebo robota v pracovním prostoru. Tyto objekty mohou mít definované akce, pozici, kolizní model, URDF[1] model, atd. Akční objekty představují fyzické objekty, jako jsou třeba krabice, robot atd.

Objekty jsou zobrazeny jako 3D modely. Fyzické akční objekty je před použitím nutno definovat na straně serveru spolu s jejich modelem a dostupnými akcemi. Virtuální kolizní objekty jsou tří typů: kostka, koule a válec. Tyto objekty se nemusí definovat na ARServeru a mohou být přidány kdykoliv. [1]



Obrázek 2.3: Kolizní objekty. Převzato z [1].

Pokud je objekt definován na ARServeru, můžeme ho přidat do scény. Každý objekt může být dále upraven pomocí nástrojů pro transformaci, jako jsou posun, rotace a škálování.



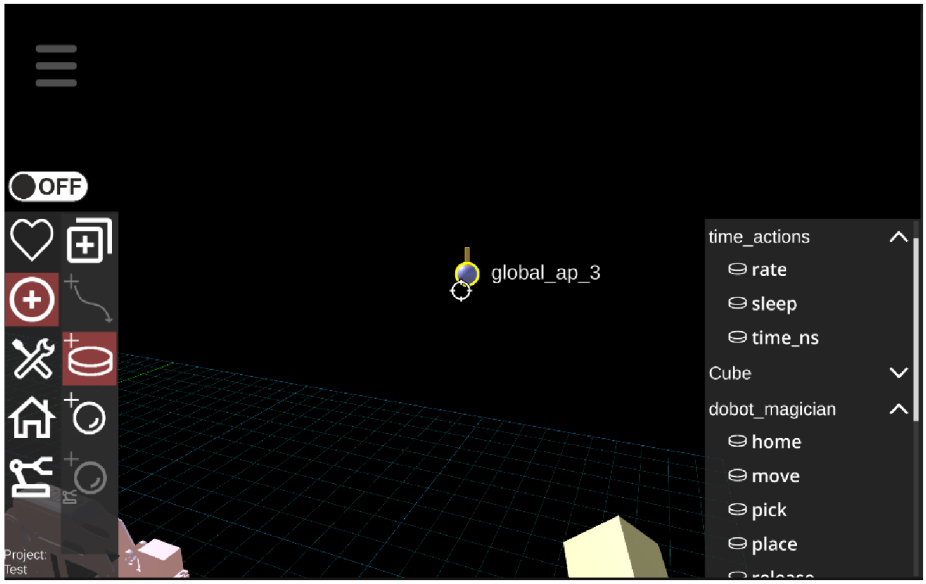
Obrázek 2.4: Modifikace akčních objektů. Převzato z [1].

[1]Unified Robot Description Format viz http://wiki.ros.org/urdf

Projekt

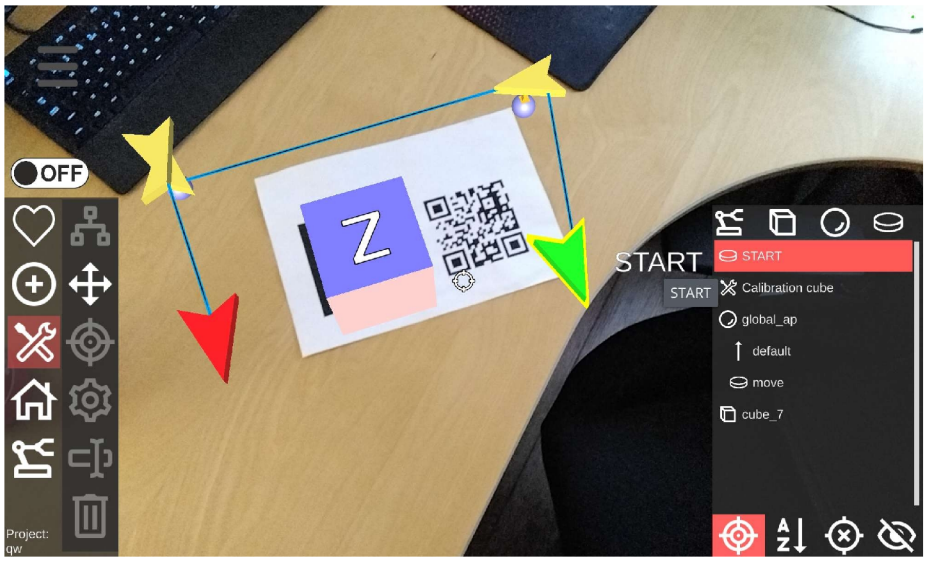
Po definování scény je možné vytvořit projekt. V projektu definujeme chování jednotlivých akčních objektů. Prvním krokem při vytváření projektu je definování akčních bodů. Tyto body definují důležitá místa v prostoru, kde má nastat nějaká akce. Příkladem takové akce může být místo, kde má být zvednut předmět. Akční bod jako takový nemá rotaci. Jde o konteiner, kterému se přiřazuje libovolné množství orientací. Orientace označuje směr, v jakém robot přistaví svůj efektor na danou pozici a ve scéně se označuje malou šipkou poblíž akčního bodu. Orientace lze přidávat buď za použití efektoru robota, nebo manuálně. Akční body mohou být definovány jako globální, nebo jako závislé na pozici jiného akčního bodu.

K akčním bodům lze následně přiřadit akce. Akce může být složen z několika atomických instrukcí robota (jako například akce "Pick"může být složena z několika instrukcí pohybu a instrukcí pozice efektoru). K přidání akce musí být nejprve vybrán akční bod. Každá akce se váže právě na jeden akční bod a více akcí se smí vázat na stejný akční bod. Po stisknutí tlačítka přídání akce se otevře nabídka s všemi dostupnými akcemi.



Obrázek 2.5: Přidávání akce. Převzato z [1].

Tok programu je definován sekvencí propojených akcí. Propojení slouží jako vizualizace přechodu mezi jednotlivými akcemi. Propojení je reprezentováno modrou čarou mezi dvěma akcemi. Směr toku programu je naznačen orientací šipek, které reprezentují akce. Začátek programu je označen zelenou šipkou a konec červenou šipkou. Program může být spuštěn přímo z editoru projektu nebo zabalit do samostatného balíčku a spuštěn později.



Obrázek 2.6: Vizualizace přechodu mezi akcemi.

2.4.4 Komunikace

Systém ARCOR2 je rozdělen do několika modulů, z nichž každý zajišťuje určitou funkcionalitu. Tyto moduly spolu komunikují pomocí REST API. Každý modul má definované vlastní rozhraní API, které popisuje, jaké požadavky jsou k dispozici a jaké odpovědi mohou být vráceny.

Pro komunikaci s uživatelským rozhraním se využívají websockety.

Websocket

Systém ARCOR2 používá pro komunikaci mezi ARServerem a AREditorem komunikační protokol websocket. Tento protokol umožňuje komunikaci typu full-duplex mezi klientem a serverem prostřednictvím jednoho TCP spojení. To znamená, že po navázání spojení je možné posílat data v obou směrech bez nutnosti navazování nového spojení pro každou zprávu.

V Systému Arcor2 je ARServer serverem, který poskytuje rozhraní pro ovládání robota a správu pracoviště. Uživatelské rozhraní, jako například AREditor slouží jako klient, který se k ARServeru připojí přes websocket. Po připojení začíná ARServer posílat klientům události a vykonává jejich požadavky na základě jejich žádostí.

Komunikační protokol je založen na událostech a Remote Procedure Calls (RPC). Události jsou zprávy, které informují klienta o změně stavu v ARServeru, jako například o změně polohy robota. RPC jsou zprávy, které slouží k vykonání konkrétní akce na straně serveru, například k pohybu robota na zadanou pozici.

Zprávy jsou přenášeny v podobě objektů typu JSON a jsou definovány pomocí datových tříd. ARServer umožňuje generování definice pro všechny modely používané v komunikačním protokolu pomoci OpenAPI. Klienti poté mohou importovat potřebné datové třídy z balíčku `arcor2_arserver_data` a používat je pro generování zpráv v komunikačním protokolu. [2]

REST API

Pro komunikaci mezi jednotlivými komponenty serverové části, mezi kterýma není potřeba obousměrné komunikace, se používá REST API. Jendá se o typ webové API, který umožňuje komunikaci mezi klientem a serverem pomocí standardních HTTP požadavků. REST API poskytuje funkce pro vytváření, čtení, aktualizaci a mazání v rámci určitého zdroje. REST API umožňuje používat různé formáty pro reprezentaci zdrojů, například JSON, XML nebo HTML. [5]

2.5 Kalibrace

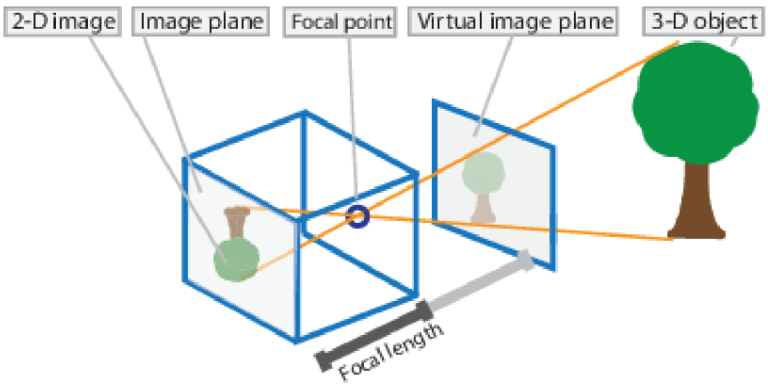
Kalibrace představuje klíčový krok při implementaci rozšířené reality. Kalibrace je důležitým faktorem pro dosažení přesného zobrazení virtuálních objektů v reálném světě. Jedná se o techniku odhadu charakteristik kamery, která umožňuje získat všechny potřebné informace, jako jsou parametry a koeficienty nutné k určení přesného vztahu mezi 3D bodem v reálném světě a jeho odpovídající 2D projekcí na snímku získaném kalibravanou kamerou. [9, 10]

Kamerový model

Kamerový model je matematický popis chování kamery. Slouží k popisu vztahu mezi reálným objektem v prostoru a jeho projekcí do obrazové roviny kamery. Tento model se používá k přepočtu 3D prostorových souřadnic objektů na jejich 2D projekce v obrazu kamery. V rozšířené realitě je kamerový model zásadním prvkem pro správnou integraci virtuálních objektů do reálného prostoru. [10]

Pinhole Camera Model

Jedná se o základní matematický model, který popisuje, jak obraz vzniká při fotografování pomocí dírkové kamery. Tento model předpokládá, že světlo prochází malou dírkou v kameře a promítá inverzní obraz pozorované scény.



Obrázek 2.7: Pinhole camera model. Převzato z [9].

Kamerový model je definován několika parametry. Následující sekce popisuje význam jednotlivých parametrů kamerového modelu.

Ohnisková vzdálenost v pixelech (f_x, f_y)

Ohnisková vzdálenost v pixelech f_x, f_y je poměr ohniskové vzdálenosti a velikosti pixelu p_x, p_y.

Ohnisková vzdálenost určuje vzdálenost mezi ohniskem kamery a senzorem. Tento parametr ovlivňuje velikost projekce bodů ve světě senzoru a tedy i velikost obrazu, který je kamera schopna zaznamenat. Ohnisková vzdálenost se obvykle udává v milimetrech a označuje se jako F. [9]

Poloha optické osy (c_x, c_y)

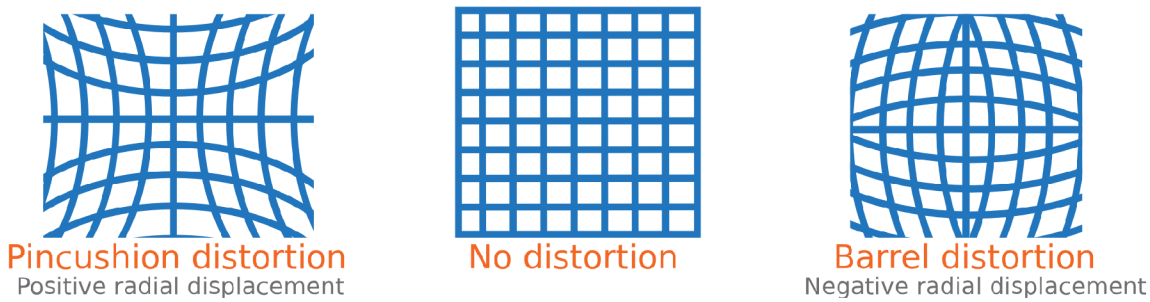
Poloha optické osy se definuje jako pozice středu obrazu vůči senzoru kamery. Optická osa prochází ohniskem objektivu a je kolmá k rovině senzoru. Kolem této osy se formuje obraz. Obvykle se poloha optické osy určuje pomocí kalibračních objektů s přesně definovanou geometrií. Tento parametr se obvykle udává v pixelech a značí se jako c_x pro horizontální polohu a c_y pro vertikální polohu. [17]

Koeficient zkosení

Koeficient zkosení je geometrická vlastnost kamery, která popisuje sklon obrazu vůči senzoru. Jedná se o úhel mezi optickou osou kamery a rovinou obrazu, který může být odlišný od 90 stupňů v případě, že obrazový senzor není umístěn přesně kolmo k optické ose kamery. [17]

Koeficienty radiálního zkreslení (k_1, k_2, k_3)

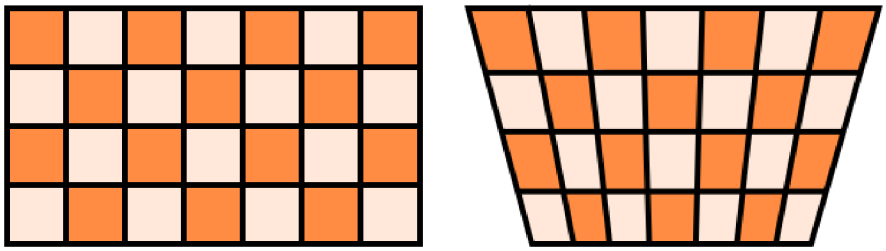
Radiální zkreslení je druh optického zkreslení, které se projevuje jako zkreslení tvarů a velikostí objektů, zejména u okrajů obrazu. Radiální zkreslení nastává tehdy, když se světelné paprsky více ohýbají blíže ke krajům čočky než v jejím optickém středu. Čím menší je čočka, tím větší je zkreslení. Radiální zkreslení se obvykle popisuje pomocí tří koeficientů: k_1, k_2 a k_3. [9, 10]



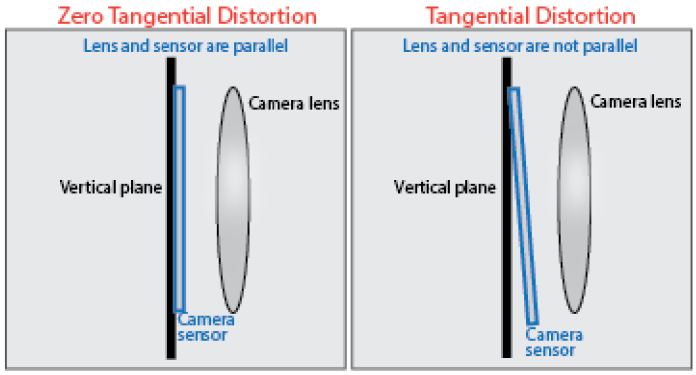
Obrázek 2.8: Radiální zkreslení. Převzato z [9].

Koeficienty tangenciálního zkreslení (p_1, p_2)

Tangenciální zkreslení je typ optického zkreslení, které se projevuje jako zkreslení tvarů a velikostí objektů, zejména v blízkosti okrajů obrazu. Tento typ zkreslení vzniká kvůli nepřesnému umístění čočky vůči snímači obrazu, což vede k nesouhlasnosti mezi rovinami obrazu a čočky. Tangenciální zkreslení se obvykle popisuje pomocí dvou koeficientů: p_1 a p_2.



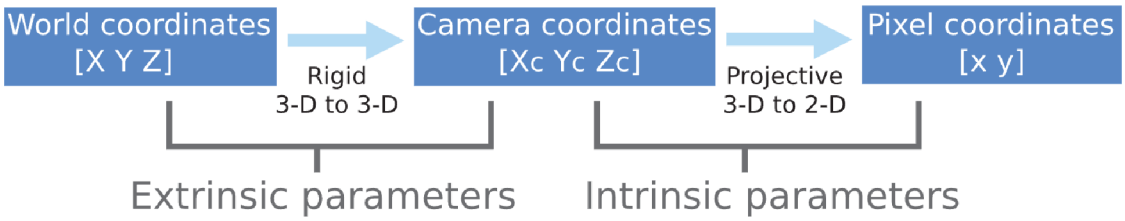
Obrázek 2.9: Tangenciální zkreslení. Převzato z [9].



Obrázek 2.10: Příčina tangenciálního zkreslení. Převzato z [9].

2.5.1 Kalibrační data

V rámci kalibrace kamery se využívají kalibrační parametry, které se dělí na vnitřní a vnější. Vnitřní parametry popisují vlastnosti kamery, jako je ohnisková vzdálenost, poloha optické osy a koeficient zkosení. Vnější parametry popisují polohu a rotaci kamery v prostoru. Tyto parametry se společně využívají k transformaci bodů v prostoru na souřadnice pixelů v obraze kamery. [9]



Obrázek 2.11: Kalibrační parametry. Převzato z [9].

Vnitřní parametry (intrinsic nebo internal parameters)

Mezi vnitřní parametry patří transformační matice K a koeficienty zkreslení. Transformační matice K je matice, která kombinuje důležité parametry kamery a slouží k transformaci bodů ze světových souřadnic do souřadnic obrazu kamery. Tato matice je obvykle definována jako:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Kde f_x a f_y jsou ohniskové vzdálenosti v horizontálním a vertikálním směru, c_x a c_y jsou polohy optické osy v rámci senzoru, s je koeficient zkosení a poslední řádek se používá pro normalizaci.

[9, 4]

Vnější parametry (extrinsic nebo external parameters)

Vnější parametry kamery popisují její pozici a orientaci v rámci souřadnicového systému světa. Tyto parametry se využívají pro transformaci mezi souřadnicemi světa a souřadnicemi kamery.

Vnější parametry se obvykle popisují pomocí rotace a translace kamery vzhledem k souřadnicovému systému světa. Původ souřadnicového systému kamery se nachází v jejím optickém středu, přičemž osy x a y definují rovinu obrazu.

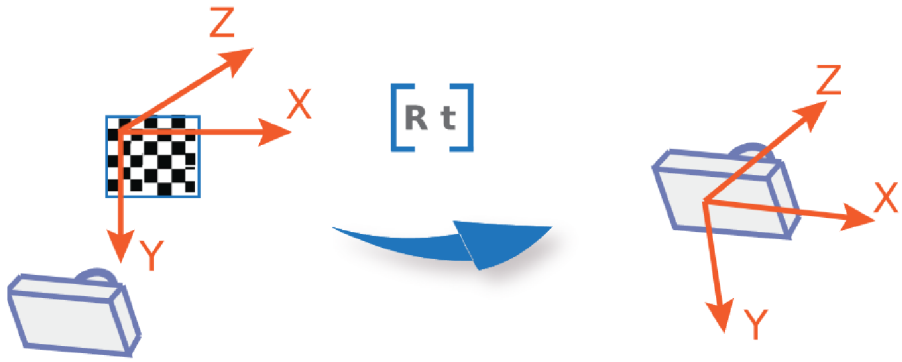
Pro transformaci bodů ze souřadnicového systému světa na souřadnicový systém kamery se používají matice rotace R a vektor translace t. Transformace bodu z souřadnicového systému světa na souřadnicový systém kamery se provádí pomocí vztahu:

$$X_{cam} = RX_{world} + t \tag{2.2}$$

kde X_{cam} a X_{world} jsou 3D body v souřadnicovém systému kamery a světa a R a t jsou matice rotace a vektor translace, které převádějí body mezi těmito souřadnicovými systémy.

Pokud jsou souřadnice bodu v kameře reprezentovány v homogenních souřadnicích jako X_{cam}, lze souřadnice v souřadnicovém systému světa X_{world} získat následovně:

$$X_{world} = [R|t]X_{cam} \tag{2.3}$$



Obrázek 2.12: Převod do souřadnicového systému kamery. Převzato z [9].

2.5.2 Kalibrační metody

SLS je technologie 3D skenování, která se skládá z projekčního zařízení a kamery. SLS využívá projekci vzorů na cílový povrch a následé snímání těchto vzorů kamerou. Dekódováním lze pomocí triangulace vypočítat vztah mezi fotoaparátem a projektorem. Kalibrace SLS

spočívá v nalezení vnitřních a vnějších parametrů obou zařízení, které popisují vztah mezi zachycenými obrazy a scénami, které zobrazují. [18]

Kalibrace se obvykle provádí fotografováním kalibračního objektu, který má známý vzor se známými vzdálenostmi mezi rozpoznatelnými značkami. Vnitřní parametry se vypočítají lokalizací a porovnáním těchto bodů na několika snímcích. Jelikož projektor nemůže zachytit značky stejně jako fotoaparát, bývá obvykle s projektorem nakládáno jako s inverzním fotoaparátem. [18]

Bod $P(X_W, Y_W, Z_W)$ na povrchu objektu v souřadnicovém systému světa je namapován na body (u_p, v_p) a (u_c, v_c) na rovině obrazu projektoru a kamery viz 2.13. Matice M popisuje vektory rotace a translace mezi kamerou a projektorem. Při zacházení s projektorem jako s inverzní kamerou může být použit perspektivní pinhole model pro popsání kamery, i projektoru jako:

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} M \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}, \tag{2.4}$$

kde (X_W, Y_W, Z_W) je trojrozměrná poloha bodu P scény a (u, v) je jeho projekce v obrazu. Z_c je měřítkový faktor, K označuje vnitřní matici a M je externí matice.

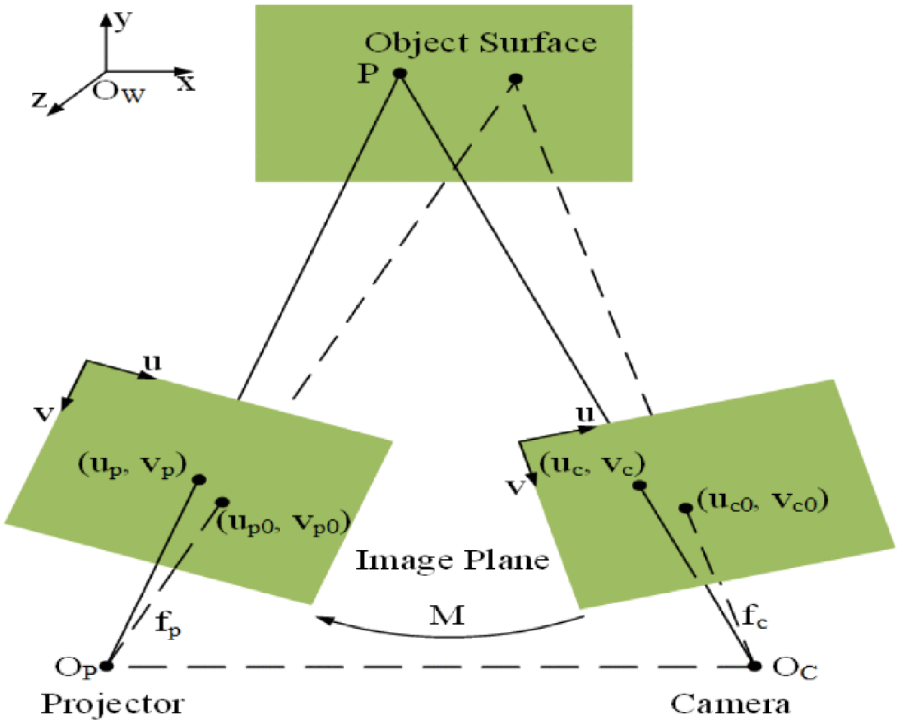
Pro dosažení vysoké přesnosti měření je obvykle nutné zahrnout do výpočtu i koeficienty zkreslení k_1, k_2, p_1, p_2, k_3, a to pomocí následujících vztahů pro výpočet nezkreslených obrazových souřadnic:

$$\begin{cases} x_r = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_r = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)' \end{cases} \tag{2.5}$$

$$\begin{cases} x_t = x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_t = y + [p_1(r^2 + 2y^2) + 2p_2 xy]' \end{cases} \tag{2.6}$$

kde $r^2 = x^2 + y^2$; (x, y) představují skutečné souřadnice se zkreslením; (x_r, y_r) a (x_t, y_t) představují souřadnice po aplikaci korekce zkreslení.

Rozdíl mezi různými metody kalibrace spočívá především v způsobu rozpoznání kalibračních značek a jak mapovat značky z obrazových souřadnic do souřadnic senzoru projektoru. [18]



Obrázek 2.13: Geometrické vyjádření SLS. Převzato z [18].

Metoda ProcamCalib

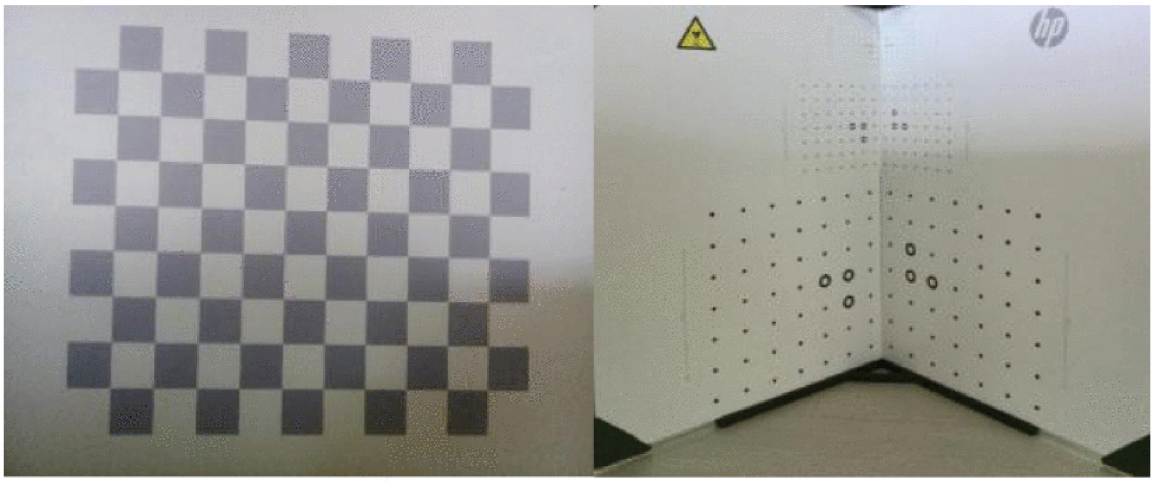
V této metodě se používá šachovnice. Nejprve je šachovnice zachycena kamerou. Za ponechání šachovnice ve stejné poloze je projekčním zařízením promítnut další vzor šachovnice, který je následně znovu zachycen kamerou. Opakováním této procedury a změnou polohy a orientace šachovnice lze získat skupinu kalibračních obrazů pro kameru i projektor. Fyzické obrazy šachovnic jsou použity pro kalibraci kamery. Lze tak určit 3D souřadnice vzoru šachovnice vzhledem k souřadnicovému systému kamery. Extrahováním pozice důležitých prvků promítnutého vzoru jsou poté vypočteny jeho 3D souřadnice na kalibrační rovině. Tím lze vyřešit rovnici 2.4 pro projektor. Tento způsob kalibrace vyžaduje pouze jednu šachovnici a jednu projekci projektorem, což usnadňuje manipulaci a zefektivňuje kalibrační postup. [18]

Metoda SLSCalib

Metoda využívá lokální homografie ke zlepšení přesnosti lokalizace příznaků. V této metodě se používá šachovnice. Nejprve je šachovnice zachycena kamerou. Za ponechání šachovnice ve stejné poloze je na šachovnici promítána sada ortogonálních Grayových kódů. Dekódováním kódu lze získat skupinu lokálních homografií, které umížňují najít vztah mezi body z kalibrační desky a jejich lokací v obrazové rovině projektoru. Poté lze projektor kalibrovat jako běžnou kameru. Tato kalibrační metoda využívá princip kódování strukturovaného světla, což umožňuje získat husté korespondence. Avšak obvykle trvá dlouhou dobu pro projekci vzoru a zpracování obrazu. [18]

Metoda 3DDCalib

Tato metoda používá narozdíl od ProcamCalib a SLSCalib 3D objekt. Tento objekt může být například kalibrační deska s pravům úhlem, na které jsou kruhové markery. Není nutné měnit polohu kalibrační desky kvůli určité ose z, která představuje pravý úhel desky, v souřadnicovém systému světa. Díky tomu stačí pořídit pouze jeden snímek, který následně umožní kalibraci kamery pomoci Tsaiovy metody [19]. Pro kalibraci projektoru stačí jedna pozice projektovaných borazů. Hlavním problémem je získání vztahu mezi pixelovými souřadnicemi kruhových markerů v promítaných obrazech a fyzickými souřadnicemi těchto markerů. V této metodě se také používá projekce Grayových kódů. Tato kalibrační metoda je efektivní a snadná na použití, ale vyžaduje speciální 3D objekt s vysokou výrobní přesností. [18]



Obrázek 2.14: a) Šachovnice použita v ProcamClib and SLSCalib ; b) A 3D kalibrační objekt použit v 3DCalib metodě. Převzato z [18].

Kapitola 3

Návrh

Tato kapitola popisuje jakým způsobem byl proveden návrh uživatelského rozhraní, jaké jsou jeho hlavní cíle a jak byly tyto cíle realizovány.

3.1 Cíl

Cílem promítaného uživatelského rozhraní je usnadnit práci se systémem Arcor2 2.4. Docílí toho tím, že umožní uživatelům efektivnější práci s virtuálními objekty a sníží pravděpodobnost chyb při jejich vkládání do scény.

Toto uživatelské rozhraní má doplňovat aplikaci Areditor2.4.3, u které byly uživatelským testováním zjištěny nedostatky v oblastech orientace v prostoru a vnímání hloubky. Promítané uživatelské rozhraní usnadní uživateli orientaci v prostoru virtuální scény, umožní mu lépe posoudit pozici vkládaného objektu a jaký bude jeho rozměr vůči stávajícím objektům.

3.2 UI

Hlavní myšlenkou je vytvořit přívětivé a intuitivní uživatelské rozhraní s co nejvyšší přesností a přehledností.

Uživatelské rozhraní má uživateli pomoct přesně určit pozici virtuálního objektu ve skutečné scéně. Při používání aplikace AREditor uživatel často neví, kde přesně se virtuální objekt nachází ve skutečném světě, jelikož z mobilní aplikace lze těžko určit, jak daleko se předmět od kamery nachází. K vyřešení tohoto problému může být použita technika promítání obrysů bojektů, která přesně určuje polohu v rovině pracoviště. Kombinací vizuálních informací, které podává AREditor a těch, které podává promítání obrysu těles, jsme schopni určit přesnou polohu tělesa v prostoru.

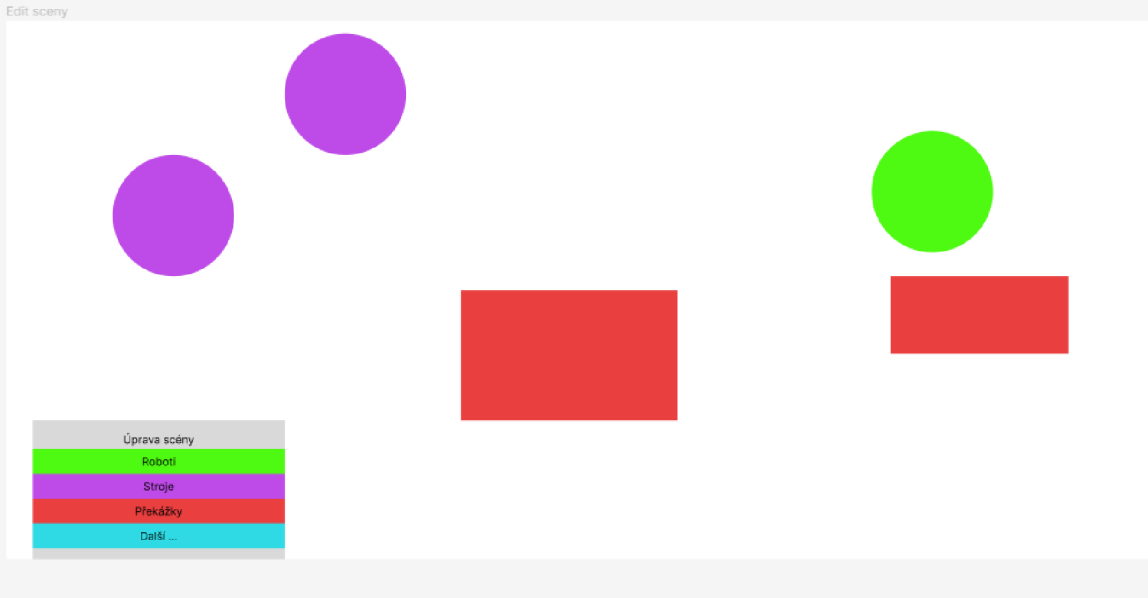
Kvůli přehlednosti je důležité, aby uživatelé mohli snadno rozlišit jednotlivé prvky a funkce virtuální scény. Proto bylo navrhnuto využití barev. Různé typy objektů ve scéně budou mít různé barvy, aby uživatelé mohli snadno rozlišit, o jaký typ objektu se jedná.

Tento přístup k návrhu uživatelského rozhraní zvyšuje jeho přehlednost a usnadňuje uživatelům práci s aplikací.

Samotné vytváření programu robota se skládá z několika fází. Pro každou z těchto fází byl vytvořen návrh. Jedná se o fáze:

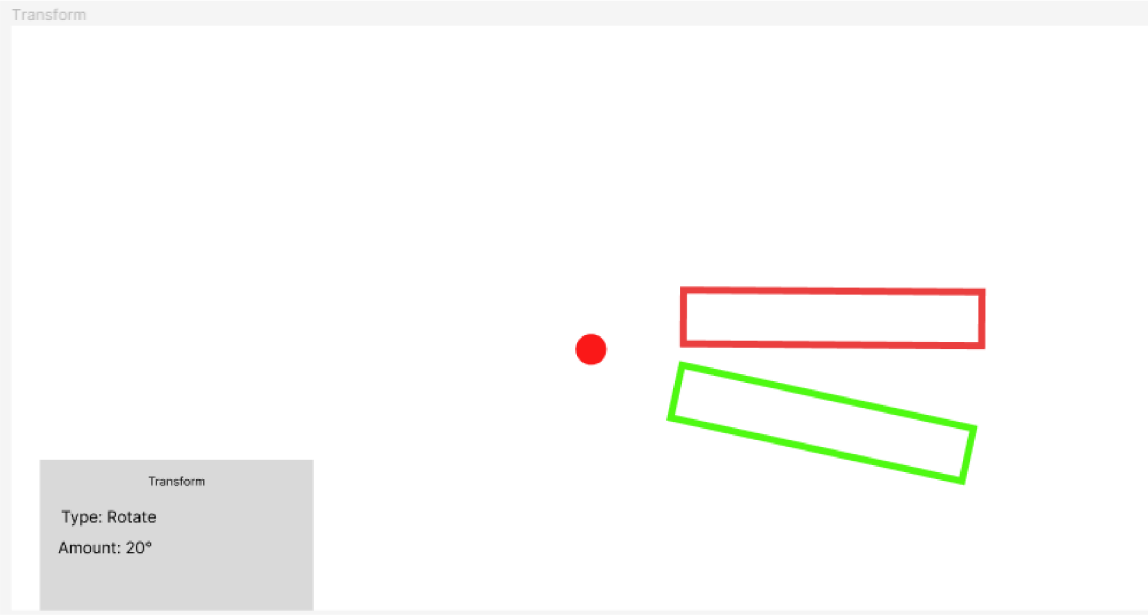
Definování scény

V této fázi se vkládají objekty do scény, je tedy potřeba vidět jaké objekty už scéna obsahuje a kde se nachází. Podle těchto požadavků byl vytvořen následující návrh.



Obrázek 3.1: Návrh vytváření scény

S objekty lze provádět operace jako škálování a rotace. Pro znázornění operace bylo navrhnuto vyznačení obrysu původního stavu objektu červenou barvou a obrysu cílového stavu objektu po provedení operace zelenou barvou. Zde je návrh zobrazující rotaci.



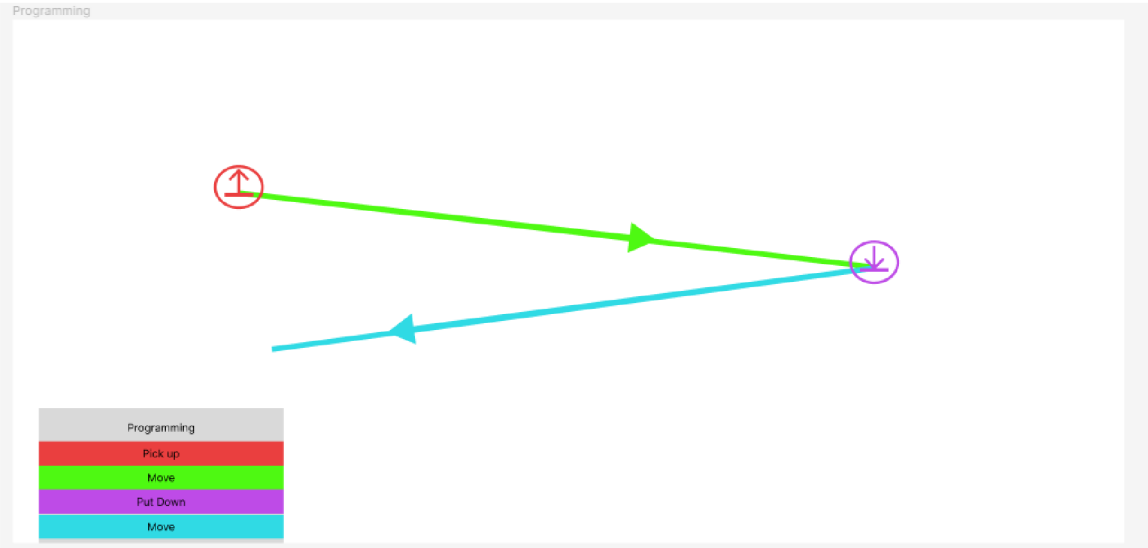
Obrázek 3.2: Návrh rotace objektu

Definování projektu

V první fázi se do vytvořené scény přidávají akční body, které vyznačují důležitou pozici, na které má dojít k nějaké akci. Zde stačí zobrazit jejich pozici.

Obrázek 3.3: Návrh vkládání akčních bodů

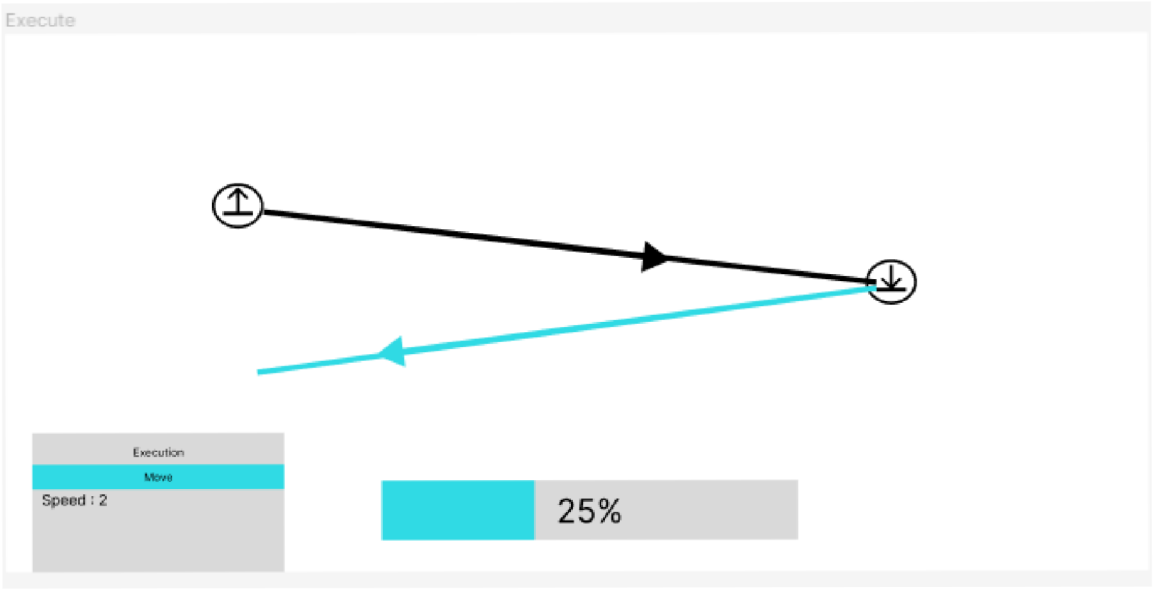
V následující fázi se k akčním bodům přidávají akce. Pro odlišení těchto akcí bylo navrhnuto použití značek symbolizující jednotlivé operace viz 3.4. Pro definování pořadí těchto operací jsem navrhnul barevně rozlišovat jednotlivé operace a jejich pořadí znázornit tabulkou v levém dolním rohu. Zde je návrh zobrazující vytvořené akce.



Obrázek 3.4: Návrh vkládání akcí

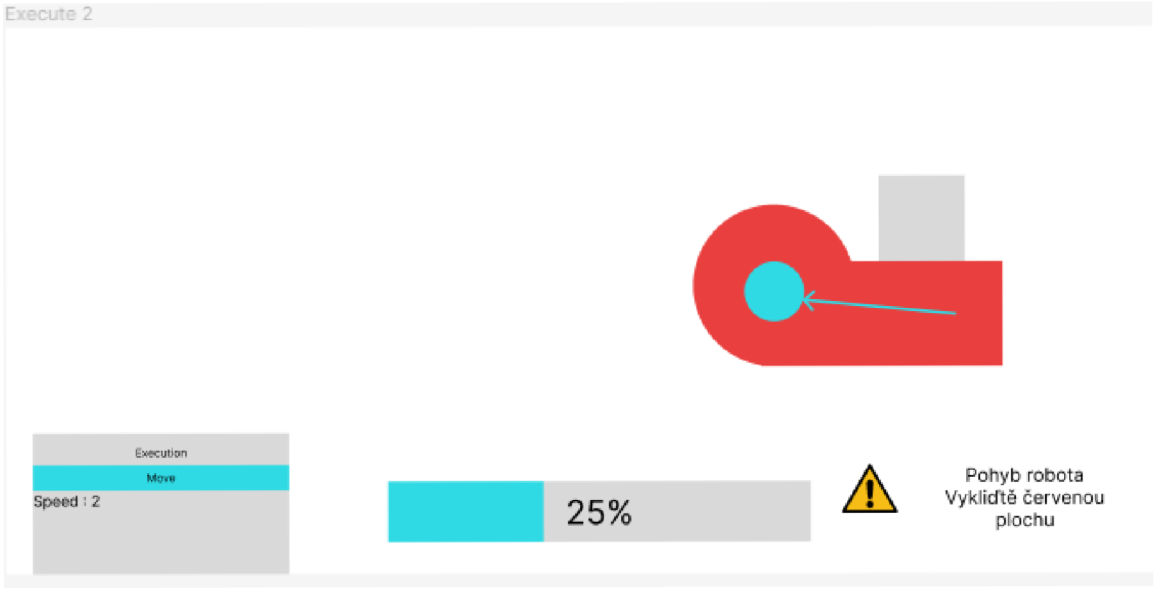
Provedení programu

V této fázi docházi k vykonání jednotlivých akcí definovaného programu. Pro odlišení právě vykonávané akce bylo navrženo použít odlišné barvy než u zbytku akcí. V levém dolním rohu budou zobrazeny parametry právě prováděné akce a uprostřed bude znázorněn průběh akce pomocí indikátoru načítání.



Obrázek 3.5: Návrh provádění akce

Pro informování uživatele o nadcházející akci robota bylo navrženo upozornění. Toto upozornění zahrnuje zvýraznění trasy, po které se robot bude pohybovat a plochu, kterou má uživatel vyklidit.



Obrázek 3.6: Návrh upozornění

3.3 Aplikace

Aplikace má zobrazovat aktuální stav scény přímo na ploše pracoviště, k tomu je nejprve potřeba získat data o scéně od ARServeru. Pro komunikaci lze využít websocket, přes který se aplikace připojí a následně přijímá informace o robotickém pracovišti. ARServer

informuje pouze o změnách, proto je třeba si lokálně ukládat stav scény a příchozí změny na ni aplikovat. Pro použití projekce jsou třeba kalibrační data2.5.1. Je nutné, aby scéna obsahovala instanci kamery, která má pozici a rotaci v souladu s jejím reálným protějškem. Od pozice kamery jsme pomocí vnějších parametrů2.5.1 schopni určit pozici a rotaci projektoru. Pokud známe polohu objektu vůči projektoru, jsme s využitím vnitřních parametrů2.5.1 projektoru schopni vypočítat souřadnice objektu na projekčním plátně tak, aby jeho projekce na pracovišti souhlasila s pozicí virtuálního objektu. Následně stačí na dané pozici na projekčním plátně zobrazit tvar odpovídající obrysu daného objektu.

3.4 Kalibrace

Pro kalibraci jsem vybral kalibrační metodu typu SLSCalib2.5.2, která je veřejně dostupná[1]. Metoda je založena na vědeckém článku [16] a je schopna získat kalibrační data pomoci promítání Grayových kódů na vzor šachovnice s libovolným počtem polí. Metoda se skládá z dvou skriptů. Skript `gen_graycode_imgs.py` se stará o vygenerování Grayových vzorů. `calibrate.py` se stará o zpracování kalibračních snímků a vypočítání kalibračních dat2.5.1.

[1] https://github.com/kamino410/procam-calibration

Kapitola 4

Implementace

Tato kapitola se zaměřuje na implementaci navrženého uživatelského rozhraní. Je zde popsáno jak byla realizována kalibrace projektoru, komunikace, základní princip aplikace a její architektura.

4.1 Kalibrace

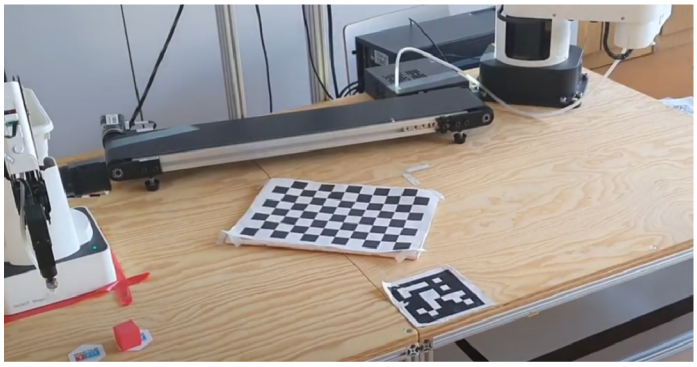
Proces kalibrace se skládá z několika částí.

generování Grayova kódů

Nejprve je potřeba vygenerovat sadu ortogonálních Grayových kódů[1]. Součástí Kalibrační metody 3.4 je skript `gen_graycode_imgs.py`, který vygeneruje tyto Grayovy kódy v závislosti na zadaném rozlišení projektoru.

příprava pracoviště

Aby bylo možné provést kalibraci správně, je třeba připravit pracoviště. To musí obsahovat šachovnici, která je viditelná jak projektorem, tak kamerou, Viz 2.13.



Obrázek 4.1: Pracoviště připravené ke kalibraci.

[1] https://cs.wikipedia.org/wiki/Gray%C5%AFv_k%C3%B3d

zachycení promítaných vzorů

Pro promítnutí a zachycení Grayových kódů jsem vytvořil skript v jazyce Python s názvem `captureProjection.py`, který prochází složku s vygenerovanými kódy a postupně je promítá do pracoviště. Jakmile je promítaný vzor připravený, provolá se služba Kinectu Azure, která je součástí systému ARCOR2 pro pořízení kalibračního snímku a tento snímek se uloží.

Pro přesnější kalibraci je doporučeno po skončení programu změnit polohu šachovnice a vytvořit další sadu kalibračních snímků.

Výpočet kalibračních dat

Po zachycení kalibračních vzorů je nutno tyto snímky zpracovat pomocí skriptu s názvem `calibrate.py`, který je součástí použité kalibrační metody 3.4. Tento skript zpracuje informace o rozlišení projektoru a parametrech šachovnice, a následně z kalibračních snímků vypočítá kalibrační data2.5.1 kamery a projektoru včetně koeficientů radiálního2.5 a tangenciálního2.5 zkreslení.

CalibrationData

Tato třída slouží k načítání kalibračních dat z xml souboru. Dá se skrze ni přistupovat k následujícím parametrům:

- Width: šířka výstupního obrazu v pixelech
- Height: výška výstupního obrazu v pixelech
- K1: koeficient radiálního zkreslení kamery
- K2: druhý koeficient radiálního zkreslení kamery
- K3: třetí koeficient radiálního zkreslení kamery
- P1: první koeficient tangenciálního zkreslení kamery
- P2: druhý koeficient tangenciálního zkreslení kamery
- Rotation: matice rotace projektoru
- Translation: vektor posunu projektoru
- ProjInt: vnitřní parametry projektoru

4.2 Komunikace

Pro správné fungování aplikace je zapotřebí od serveru získávat informace o aktuálním stavu systému. O to se stará skript `WebsocketManager.cs` za použití knihovny *NativeWebSocket*. Skript vytvoří websocket, pomoci kterého se připojí k ARServeru a následně přijímá příchozí zprávy o změnách stavu systému.

4.3 Implementace návrhu UI

Uživatelské rozhraní bylo vytvořeno jako aplikace v herním enginu Unity. Hlavní částí aplikace je plátno (Canvas), které obsahuje ovládací prvky pro připojení a samotnou reprezentaci scény.

Obrázek 4.2: Ovládací prvky

Připojení

Po stisknutí tlačítka s nápisem Connect, se rozhraní pokusí připojit k serveru na dané adrese a portu. Po připojení se ovládací prvky skryjí.

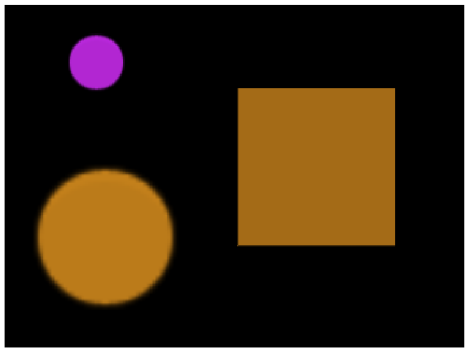
Zobrazení scény

Jakmile aplikace načte vlastnosti objektů (viz 4.4.2), je připravena načíst scénu. O otevřené scéně je aplikace informována zprávou typu "OpenScene", jejíž součástí je informace o obsahu scény. Obsah scény se zpracuje, pro každý objekt se na plátno vloží prefab4.4.3 reprezentující daný objekt a aktualizuje se jeho vizuální stránka podle typu objektu.

Obrázek 4.3: Reprezentace scény.

Zobrazení projektu

O otevřeném projektu je aplikace informována zprávou typu "OpenProject". Tato zpráva také obsahuje popis scény a rozmístění akčních bodů. Akční body mají vlastní prefab a jsou reprezentovány jinou barvou.



Obrázek 4.4: Reprezentace projektu.

Výpočet pozice na projekčním plátně

Pro správné zobrazení pozice objektů je nutné, aby scéna obsahovala kameru. V momentálním stavu systému musí být objekt typu "KinectAzure". K použití jiné kamery, by bylo potřeba doplnit metadata objektů o označení, že se jedná o kameru. Při zpracování tohoto objektu dojde k inicializaci projekce. Vytvoří se projektor, jehož pozice a orientace v prostoru se vypočítá z kalibračních dat a pozice kinectu.

Po inicializaci projekce dojde k výpočtu pozice objektů na plátně. Jelikož projekce neumožňuje zobrazovat objekty mimo projekční plochu, je nutno počítat se všemi objekty, jako kdyby byly ve výšce 0. Samotnou souřadnici objektu na projekční ploše vypočítáme následující funkcí, která převádí 3D souřadnice v prostoru na 2D souřadnice na projekčním plátně pomoci vnitřních parametrů kamery2.5.1.

```
public static Vector2 ManualWorldToScreenPoint(Vector3 point)
{
    CalibrationData calibData = ProjectionManager.Instance.calibrationData;
    //vnitrni parametry projektoru
    Matrix4x4 projIntrinsic = calibData.ProjInt;
    GameObject projector = ProjectionManager.Instance.projector;
    //prevod do souradnicoveho systemu projektoru
    Matrix4x4 worldToCam = projector.transform.worldToLocalMatrix;

    Vector4 localPoint = worldToCam * point;

    // vynasobeni lokalniho bodu vnitrni matici projektoru
    Vector4 temp = projIntrinsic * localPoint;

    if (temp.w == 0f)
    {
        // bod je na ohniskovem bode, neni definovan
        return Vector3.zero;
```

```
    }
    else
    {
        // prevod souradnic z clip space na souradnice platna
        temp.x = (temp.x / temp.w + 1f) * .5f * Width - Width / 2;
        temp.y = (temp.y / temp.w + 1f) * .5f * -Height + Height/2;
        return new Vector2(temp.x, temp.y);
    }
}
```

Změny oproti návrhu

Během vývoje aplikace bylo zjištěno, že na pracovní ploše není dostatek místa pro zobrazování všech informací. Na základě této skutečnosti bylo rozhodnuto, že se bude zobrazovat pouze pozice objektů a akčních bodů.

4.4 Architektura

V této sekci jsou popsány tři hlavní části, ze kterých se systén skládá. Jedná se o části:

1. Scéna
2. Herní objekty
3. Aplikační logika

4.4.1 Scéna

Většina objektů se zobrazuje na plátně, kinekt a projektor je ovšem potřeba vytvořit v prostoru, jelikož jejich poloha a rotace je důležitá k dalším výpočtům. Kvůli tomu je scéna rozdělena na dvě části. Jedna reprezentující projekční plochu (Canvas4.4.1) a jedna obsahující objekty v prostoru(World4.4.1).

Canvas

Canvas je herní objekt, který reprezentuje projekční plátno. Součástí tohoto objektu je komponenta Canvas(plátno). Všechny prvky uživatelského rozhraní, včetně samotných objektů, musí být potomci tohoto herního objektu, jinak nebudou zobrazeny. Plátno je nastaveno v renderovacím módu "Screen Space - Overlay", to má za následek, že plátno je použito jako překryv(overlay) obrazovky a má stejné rozlišení jako hlavní kamera projektu.

Tento objekt má dva potomky. Prvním potomkem je skupina ovládacích prvků používaných k připojení viz 4.2. Tyto kontrolní prvky jsou po připojení skryty. Druhý potomek má název "Scene"a jedná se o rodičovský prvek, jehož potomky se stávají objekty scény k zobrazení.

World

Tento herní objekt představuje rodičovský objekt objektů v prostoru. Tyto objekty tvoří abstraktní scénu, která není zobrazována, ale používá se pro výpočet pozice ostatních objektů na plátně. Tato scéna obsahuje pouze herní objekty reprezentující kinekt4.4.3 a projektor4.4.3 včetně jejich polohy a rotace v reálném světě.

4.4.2 Aplikační logika

Aplikační logika se skládá z několika hlavních správců (manager), pomocných tříd a skriptů popisujících chování herních objektů.

Každý správce se stará o specifickou část aplikace.

WebsocketManager

Tento správce je zodpovědný za kominukaci s ARServerem. Po přijetí zprávy je zde určeno, o jaký typ zprávy se jedná a přijaté informace jsou následně předány patřičné části systému na další zpracování. Tento skript byl převzat z aplikace AREditor[2].

GameManager

Tento správce je zodpovědný za načítání scény a projektu.

V procesu připojování k serveru může správce nabývat tří různých stavů, které udává WebsocketManager. Ve stavu připojování (Connecting) se čeká, jestli se podaří navázat spojení se serverem. Ve stavu připojeno (Connected) tento správce zažádá (skrze WebsocketManager) ARServer o parametry a akce dostupných objektů a tyto informace předá správci akcí (ActionsManager4.4.2). Ve stavu odpojeno (Disconnected) se postará o zavření otevřené scény nebo projektu.

Žádosti o otevření projektu nebo scény jsou předány tomuto správci. Pokud již byly načteny typy objektů, pošle se žádost správci scény 4.4.2 nebo projektu 4.4.2, podle typu požadavku, o vytvoření požadovaného prvku. Pokud tomu tak není, uloží se informace o scéně/projektu a počká se s jejich vytvořením, dokud nebudou načteny objekty.

Tento skript byl převzat z aplikace AREditor. Ze skriptu byly odebrány nepotřebné funkce.

ActionsManager

Tento správce má zodpovědnost za načtení a správu typů objektů včetně jejich metadat. Metadata obsahují informace o daném typu objektu, jako je například jeho kategorie (robot, kolizní objekt, akční objekt) nebo fyzický model.

Tento skript byl převzat z aplikace AREditor. Ze skriptu byly odebrány nepotřebné funkce.

SceneManager

Správce scény je zodpovědný za vytváření, úpravu a mazání scény.

Správce přijme informace o obsahu scény a pro každý objekt vytvoří patřičný model podle typu objektu. Tyto objekty mohou být několika typů, konkrétně robot, kolizní objekt a akční objekt. Všechny z nich jsou reprezentovány stejným herním objektem4.4.3, kterému jsou pouze měněny parametry jako je barva nebo tvar.

Po vytvoření objektu je provedena jeho inicializace, která zahrnuje nastavení pozice, orientace a grafické reprezentace. Objekt a jeho ID je následně přidáno do slovníku všech objektů ve scéně.

U akčních objektů je nutné zkontrolovat, jestli se jedná o akční objekt typu "KinectAzure", v tom případě je potřeba vytvořit jiný typ objektu, který není zobrazován a používá

[2]aplikace AREditor je dostupná na https://github.com/robofit/arcor2_areditor

se pro výpočet pozice ostatních objektů na plátně. Po vytvoření tohoto objektu provede ProjectionManager inicializaci projekce viz4.4.2.

V případě změny stavu objektu posílá ARServer informaci o této změně. Tento objekt se vyhledá ve slovníku objektů podle ID a jsou mu předány informace o požadované změně například změna pozice, rotace nebo smazání objektu.

Tento skript byl převzat z aplikace AREditor. Ze skriptu byly odebrány nepotřebné funkce a byla upravena funkce na vytváření objektů.

ProjectManager

Tento správce je zodpovědný za vytváření, úpravu a mazání akčních bodů.

Správce přijme požadavek na vytvoření projektu s informacemi o akčních bodech. Pro každý bod je vytvořena kopie prefab objektu ActionPoint.

Při vytváření akčního bodu je tento objekt a jeho ID přidáno do slovníku všech akčních bodů ve scéně.

V případě akčních bodů se pracuje se změnami obdobně jako při změnách akčních objektů.

Tento skript byl převzat z aplikace AREditor. Ze skriptu byly odebrány nepotřebné funkce.

ProjectionManager

Tento správce je zodpovědný za správné nastavení projekce a nahrání kalibračních dat po zapnutí aplikace.

Po vytvoření kinectu ve scéně dochází k inicializaci projekce. To znamená, že dojde k vytvoření instance projektoru4.4.3, nastavení rozlišení kamery projektoru a zobrazení projekčního plátna. K nastavení pozice a orientace projektoru dochází až po inicializaci kinectu a při každé další změně jeho polohy nebo orientace. Stará se o to následující funkce, která získá pozici a orientaci projektoru tím, že aplikuje kalibrační data na pozici a orientaci kinektu. Následující kód ukazuje nastavení pozice projektoru.

```
/// <summary>
/// Nastaveni pozice projektoru z kalibracnich dat.
/// </summary>
public void UpdateProjectorTransform()
{
    //pricteni posunu k aktualni pozici kinektu
    position = kinect.position + calibrationData.Translation;
    Matrix4x4 rotation = calibrationData.Rotation;
    //prevod matice na Quaternion
    rotQuat = LookRotation(rotation.GetColumn(2), rotation.GetColumn(1));
    //aplikovani rotace na rotaci kinectu
    projector.transform.rotation = kinect.transform.rotation * rotQuat;
    //prepocitani pozice vsech objektu na platne
    ResetAllPositions();
}
```

Po změně pozice nebo orientace projektoru je přepočítána pozice a orientace všech promítaných objektů.

4.4.3 Herní objekty

Tyto objekty reprezentují objekty pracoviště a obsahují popis jejich chování. Tyto objekty jsou vytvářeny klonováním komponent nazývaných prefab.

Prefab

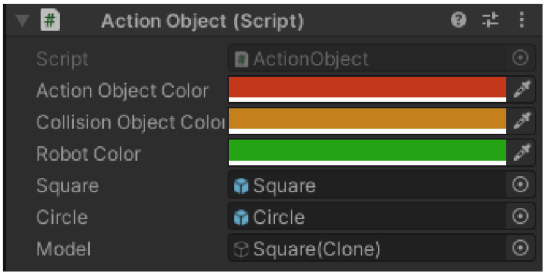
Jde o speciálním typ komponenty, která umožňuje uložit plně nakonfigurovaný objekt v projektu pro opakované použití. Tyto objekty se vytváří v editoru aplikace Unity. Každý typ objektu má vlastní prefab.

Následující sekce popisuje prefaby použité v projektu.

ActionObject

Tento herní objekt se používá pro reprezentaci veškerých objektů, které se mají zobrazit na projekčním plátně. Jedná se o prázdný objekt, který obsahuje pouze skript s názvem `ActionObject.cs`, jenž jej spravuje. Po vytvoření objektu dochází k jeho inicializaci, to zahrnuje definování id, typu, jména, pozice a orientace ve scéně a dalších metadat. Dále je vytvořena vizuální reprezentace objektu. Momentálně se používají dvě reprezentace modelu, a to čtverec a kruh4.4.3. Podle typu specifikovaného v metadatech, je vybrána příslušná reprezentace a je připojena jako potomek objektu.

Součástí vytváření modelu je také nastavení jeho barvy, která se liší pro akční objekty, kolizní objekty a roboty. Tyto barvy a modely se dají nastavit přímo v editoru tohoto objektu.



Obrázek 4.5: Nastavení barev a modelu.

Pozice a orientace objektu na projekčním plátně je přepočítána při každém posunu nejen samotného objektu, ale také projektoru. Tato pozice se vypočítá z 3D polohy objektu ve scéně. 3D poloha je definována při inicializaci objektu a přepisována při každé změně polohy. Samotný postup výpočtu je popsán zde 4.3.

Seznam všech akčních objektů scény je dostupný skrz správce scény4.4.2.

ActionPoint

Tento objekt slouží pro reprezentaci akčních bodů na projekčním plátně. Objekt obsahuje obrázek a skript `ActionPoints.cs`, který spravuje jeho pozici.

Po vytvoření objektu dochází k jeho inicializaci, to zahrnuje definování pozice ve scéně a nastavení pozice na projekčním plátně.

Výpočet pozice na plátně probíhá při každém posunu bodu nebo projektoru a má stejný průběh, jako u akčních objektů.

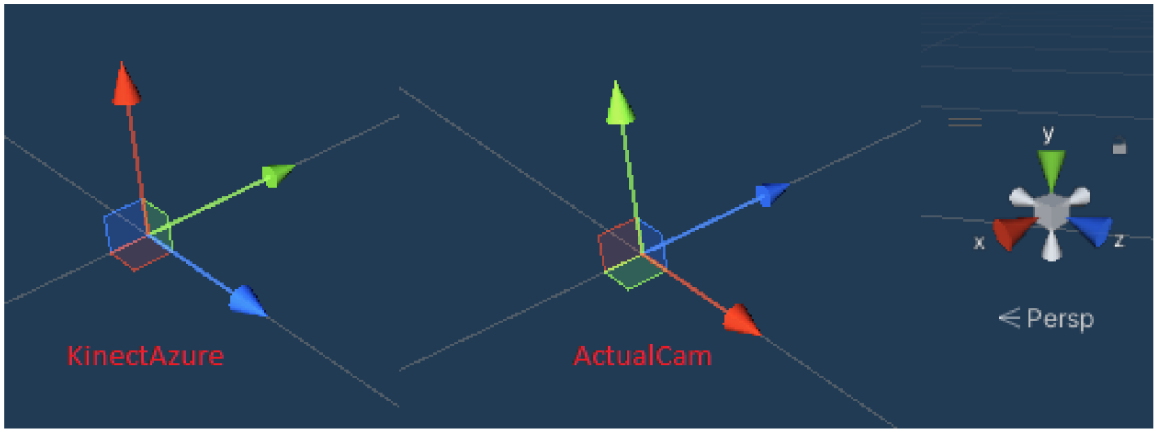
Seznam všech akčních bodů je dostupný skrz správce projektu 4.4.2.

KinectAzure

Herní engine Unity pracuje s jiným souřadnicovým systémem než ROS. To ovlivňuje nejen pozici, ale i rotaci. Z tohoto důvodu obsahuje tento objekt prázdný objekt nazvaný ActualCam, který je otočený tak, aby výsledná rotace byla v souladu s souřadnicovým systémem Unity. Konkrétně jde o rotaci $-90°$ okolo osy X a $-90°$ okolo osy Y. Objekt KinectAzure je tedy natočen tak, že kamera pozoruje scénu osou Y, zatímco vnitřní objekt ActualCam je natočen ke scéně osou Z. Při nastavování pozice kinektu je použita původní orientace rodičovského objektu KinectAzure. Pro výpočet pozice a orientace projektoru se používá vnitřní objekt ActualCam, který má rotaci v souladu s souřadnicovým systémem Unity.

Unity pracuje se systémem, ve kterém je směr dopředu definován jako kladný posun po ose Z, osa X směřuje doprava a osa Y nahoru.

Na následujícím obrázku lze na levé straně vidět rotaci samotného objektu KinectAzure, který má ratoci v souladu s ROS, zatímco napravo je rotace objektu ActualCam, který má rotaci v souladu s Unity. Červená šipka označuje osu X, zelená osu Y a modrá osu Z.



Obrázek 4.6: Úprava rotace KinectAzure.

Dále object KinectAzure obsahuje skript `KinectAzure.cs`, který je zodpovědný za jeho pozici a orientaci. Kinekt narozdíl od ostatních akčních objektů není zobrazován na projekčním plátně. Jeho pozice se používá pro výpočty, proto nemá žádnou vizuální reprezentaci a jeho pozice je pouze převedena ze souřadnicového systému ROS do souřadnicového systému Unity.

Po změně pozice nebo orientace kinektu je nutné přepočítat pozici a orientaci projektoru a tím pádem i pozici a orientaci všech objektů, které se mají zobrazovat na plátně, o což se stará správce projekce4.4.2.

Projector

Stejně jako KinectAzure4.4.3, je i tento herní objekt pouze abstraktní a nemá žádnou vizuální reprezentaci. Jde o jediný reálný objekt, který není vkládán prostřednictvím aplikaci AREditor2.4.3. Tento herní objekt je vytvořen po přidání kinektu a jeho pozice je nastavována správcem projekce viz4.4.2.

Projector obsahuje komponentu "Pixel Perfect Camera". Tato komponenta umožňuje ruční nastavení rozlišení kamery, které je nastaveno při inicializaci projekce na hodnoty uvedené v kalibračních datech4.1.

Objekt neobsahuje žádný skript, který by ho ovládal, používáse pouze jeho pozice a rotace pro výpočet pozice ostatních objektů na projekčním plátně.

Square a Circle

Tyto dva objekty reprezentují vizuální stránku promítaných objektů a jsou vždy vytvářeny jako součást akčního objektu4.4.3. Jedná se o UI element. To znamená, že tento objekt nemá 3D pozici a rotaci jako ostatní objekty, ale pouze souřadnici v rámci plátna, mimo které nemůže existovat.

V závislosti na tvaru objektu je vybrán jeden z těchto dvou prefabů pro jeho vizuální reprezentaci. Poté je na základě typu objektu stanovena jeho barva.

Obrázek 4.7: Vizuální reprezentace akčních objektů.

Kapitola 5

Testování

Tato kapitola se zaměřuje na testování promítaného uživatelského rozhraní. Testování bylo provedeno v reálném prostředí za účasti uživatelů, kteří měli možnost využívat systém a poskytnout zpětnou vazbu. Dále se zabývá zjištěnými problémy a připomínkami uživatelů, které byly následně zohledněny do návrhu budoucího vývoje systému.

Testovací skupina

Vzhledem k tomu, že promítané uživatelské rozhraní nebylo navrženo jako učební nástroj, ideálním uživatelem pro testování je ten, kdo má alespoň základní znalosti tohoto systému a umí s ním pracovat. Z tohoto důvodu jsem se rozhodl provést testování promítaného uživatelského rozhraní na dalších studentech, kteří pracovali se systémem Arcor2 při vývoji bakalářské práce, a na vedoucím práce.

Obrázek 5.1: Testování aplikace.

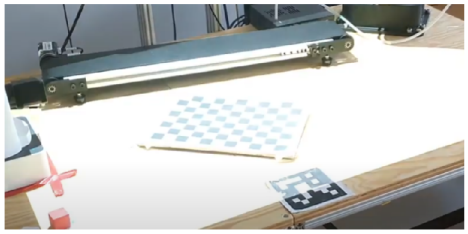
5.1 Poznámky

Během testování byly zjištěny následující problémy:

- Bylo zjištěno, že poloha promítaných objektů neodpovídá pozici skutečných objektů, což bylo způsobeno nepřesnou kalibrací projektoru.

  Hlavní příčinou jsou problémy s pořizováním snímků osvícené šachovnice, která v některých případech není detekovatelná kalibračním skriptem. Tato skutečnost vedla k nepřesné kalibraci projektoru a výsledkem byla nesprávná poloha promítaných objektů.

  Pro vyřešení tohoto problému by bylo nutné zlepšit způsob pořizování snímku šachovnice tak, aby byla viditelná i při osvícení projektorem, čehož se nepodařilo docílit ani změnou světelných vlastností v místnosti a změnou nastavení samotného projektoru.

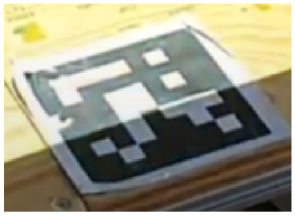


Obrázek 5.2: Osvícená šachovnice.

- Další problémy s kalibrací při projekci zahrnují obtíže s určením pozice kinektu a kalibrací AREditoru v souvislosti s osvětlením kalibračního vzoru. Tyto problémy mohou být způsobeny nerovnoměrným osvětlením kalibračního vzoru. Důsledkem může být neúspěšné zobrazení uživatelského rozhraní, nepřesné určení pozice kinektu ve scéně a další případy problémů se zpracování obrazu.

  Aby se tyto problémy minimalizovaly, je nutné zajistit aby nedocházelo k osvětlení jen některých částí kalibračního vzoru.

  Zde je ukázka osvětlení kalibračního vzoru, kvůli kterému nelze provést kalibraci AREditoru.



Obrázek 5.3: Osvícení kalibračního vzoru znemožňující kalibraci.

- Vypozorovaný problém spočívá v tom, že aktualizace promítané pozice objektu v uživatelském rozhraní se provádí až po dokončení transformace. To znamená, že pokud uživatel v AREditoru vybírá novou pozici objektu, promítaná pozice objektu se aktualizuje až po finálním provedení transformace. Tento postup může vést k nekonzistenci mezi pozicí objektu v AREditoru a jeho promítanou pozicí v uživatelském rozhraní.

  Řešením tohoto problému může být úprava procesu aktualizace pozice, tak aby se pozice objektu na straně serveru aktualizovala v reálném čase, tedy během změny pozice v AREditoru. To by zajišťovalo, že promítaná pozice objektu v uživatelském rozhraní bude vždy odpovídat aktuální pozici objektu v AREditoru.

- Při promítání uživatelského rozhraní občas dochází k blokování projekce samotným obsahem scény. Může docházet buď k částečnému, nebo kompletnímu překrytí projekce objektu. Takový objekt pak může mít zkreslený tvar popřípadě být celý zobrazen na špatném místě.

  Řešením tohoto problému může být využití hloubkové mapy kinektu, která umožňuje rozpoznat objekty ve scéně a na základě toho určit, které oblasti jsou překryté a které jsou volné pro projekci. Díky tomu bude promítaný objekt vždy kompletní a na správném místě, nebo se nebude zobrazovat

- Při vkládání objektu, který se fyzicky nachází ve scéně, dochází k problému, kdy promítaný objekt se objevuje na povrchu daného objektu, namísto toho, aby byl na projekční ploše. Tento jev způsobuje, že promítaný obraz může být zkreslený a jeho pozice nemusí být přesná.

  K řešení tohoto problému by se také mohla využít hloubková mapa kinektu, pomoci které by se zjistily rozměry a pozice objektu a projekce by byla upravena pro zobrazení na vyvýšeném povrchu.

Tyto problémy byly zohledněny v návrhu budoucího vývoje systému.

5.2 Vyhodnocení a plány do budoucna

Pomocí testování se podařilo lokalizovat hlavní problémy promítaného uživatelského rozhraní. Pro vyřešení těchto problémů je potřeba udělat změny v samotném systému ARCOR2. Následující sekce popisuje návrh změn systému pro vylepšení projekce.

Pořizování snímků

Pro správnou kalibraci systému a tím pádem i přesnou pozici promítaných objektů, by bylo potřeba upravit službu na pořizování snímků "Kinect Azure Service", aby byla schopna správně zachytit osvícenou scénu.

Vymezení zón, kde se nemá promítat

Na pracovišti jsou určitá místa, kam nechceme promítat. Jedná se zejména o místa, kde se nachází kaliřační vzory, nebo místa, kde je projekce blokována nějakým objektem. V budoucnu bych zamezil projekci na tyto místa jedním ze dvou řešení. První možností je přidáním dalšího herního objektu, který by byl černý a byl by v popředí scény. Tím pádem by zakryl veškerou projekci v daném místě. Výsledkém této metody by bylo ořezání objektů a zobrazení pouze některých částí, což může vést k nesprávnému pochopení tvaru objektu. Druhou možností je kontrolou pozice objektu. Pokud bude jeho pozice ve vymezené zóně, nebude zobrazena žádná jeho část.

Promítání v reálném čase

Při provádění transofmace objektu dochází ke změně na projekčním plátně až po ukončení operace, což má za následek, nutnost provádět operaci opakovaně, abychom viděli změnu. Toto je způsobeno tím, že AREditor posílá žádost o změnu stavu objektu až poté, co je

transformace dokončena. Možností, jak tento problém vyřešit, je upravit aplikaci AREditor, aby v daných intervalech posílala žádost o změnu stavu objektu již při provádění transofmace.

Projekce na povrch objektů

V současné době je aplikace schopna přesně zobrazovat objekty pouze na ploše pracoviště. Pro vyřešení tohoto problému navrhuji využít hloubkovou mapu získanou pomocí kinektu k určení výšky objektu. Následně bych upravil výpočet pozice objektu na projekčním plátně tak, aby byla zohledněna i výška objektu a zabránili jsme tak zkreslení projekce.

Zobrazení přesného tvaru objektu

Momentálně jsou veškeré objekty reprezentovány jedním ze dvou základních tvarů4.4.3. Chtěl bych, aby v budoucnu byl systém schopný přesněji reprezentovat obrys objektů. Toho bych chtěl docílit buď obsažením této informace v metadatech objektu, nebo vytvořením 2D reprezentace zpracováním 3D modelu.

Kapitola 6

Závěr

Cílem mojí práce bylo navrhnout a implementovat promítané uživatelské rozhraní s prvky rozšířené reality, které bude doplňovat aplikaci AREditor při práci s robotickým pracovištěm. Dále bylo třeba provést uživatelské testování, aby se nalezly nedostatky systému. Tyto cíle se mi podařilo splnit.

V rámci této bakalářské práce jsem se prvně zaměřil na využití rozšířené reality a promítaného uživatelského rozhraní v robotice. Provedl jsem porovnání využití různých typů rozhraní a zhodnotil jejich silné a slabé stránky.

Pro návrh vhodného uživatelského rozhraní jsem se musel seznámit s aplikací AREditor a jejími možnostmi. V aplikaci byly nalezeny nedostatky, které jsem zohlednil v návrhu promítaného uživatelského rozhraní.

Součástí práce byl také návrh kalibrace projektoru včetně určtění jeho pozice a rotace ve scéně. Správná kalibrace je důležitá pro dosažení přesné pozice objektů na projekční ploše, kterou v tomto případě tvoří samoté pracoviště. V této části jsem popsal jednotlivé parametry kamery a projektoru, a porovnal metody, jak lze tyto parametry získat.

Výslednou aplikaci jsem implementoval v prostředí Unity s využitím základu z aplikace AREditor. Rozhraní jsem testoval na studentech, kteří měli možnost ho využít při práci se systémem ARCOR2. Během testování se objevilo několik nedostatků uživatelského rozhraní i samotného systému ARCOR2, které jsem následně zohlednil při návrhu budoucího vývoje.

V budoucnu by bylo vhodné zlepšit přesnost projekce. Jedním způsobem, jak by se toho dalo dosáhnout, je změna kalibrační metody na takovou, která nepotřebuje šachovnici. Dalším krokem by byla aktualizace polohy objektů v reálném čase. Je třeba však mít na paměti, že tato změna by mohla zvýšit zátěž serveru. Vhodné by bylo také zahrnout 2D model v metadatech objektů, což by umožnilo jejich přesnější reprezentaci v promítaném rozhraní.

V dlouhodobém horizontu by bylo vhodné zvážit využití hloubkové mapy z Kinektu, což by umožnilo získávat více informací o obsahu scény a například detekovat objekty, které by mohly bránit projekci.

Literatura

[1] *Arcor2 AREditor wiki* [online]. Github [cit. 2023-04-12]. Dostupné z: https://github.com/robofit/arcor2_areditor/wiki.

[2] *Arcor2 wiki* [online]. Github [cit. 2023-04-12]. Dostupné z: https://github.com/robofit/arcor2/wiki.

[3] *Explained! What is Projected Augmented Reality? (with examples)* [online]. essentialpicks [cit. 2023-04-26]. Dostupné z: https://essentialpicks.com/projected-augmented-reality/.

[4] *OpenCV: Camera Calibration and 3D Reconstruction* [online]. [cit. 2023-04-02]. Dostupné z: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html.

[5] *What is a REST API?* [online]. IBM [cit. 2023-04-12]. Dostupné z: https://www.ibm.com/topics/rest-apis.

[6] *Kolaborativní roboti* [online]. Kinali, listopad 2008 [cit. 2023-04-02]. Dostupné z: https://www.kinali.cz/cs/technologie/kolaborativni-roboti/.

[7] *Co je to kolaborativní robot? 5 věcí, které byste o něm měli vědět* [online]. Factoryautomation, březen 2017 [cit. 2023-04-02]. Dostupné z: https://factoryautomation.cz/co-je-to-kolaborativni-robot-5-veci-ktere-byste-o-nem-meli-vedet/.

[8] *ARCOR2 - framework pro snadné programování robotů* [online]. Vysoké Učení Technické v Brně, 2021 [cit. 2023-04-12]. Dostupné z: https://www.fit.vut.cz/research/product/700/.cs?type=HABIL.

[9] *What Is Camera Calibration?* [online]. MathWorkds, 2023 [cit. 2023-04-02]. Dostupné z: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[10] Bhatt, D. *A comprehensive guide for Camera calibration in computer vision* [online]. Analytics Vidhya, říjen 2021 [cit. 2023-04-02]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-for-camera-calibration-in-computer-vision/.

[11] Collaborative, A. *ARCOR2: Framework for Collaborative End-User Management of Industrial Robotic Workplaces using AR.* Dostupné z: https://www.youtube.com/watch?v=RI1uiIEiPK8.

[12] Gao, Y. a Huang, C.-M. PATI: a projection-based augmented table-top interface for robot programming. *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 2019.

[13] Kapinus, M. *ARRIS - Augmented Reality Robot Interactive Space*. Dostupné z: https://www.youtube.com/watch?v=UqDFkKYTSG4.

[14] Koďousková, B. *ROZŠÍŘENÁ REALITA: VYUŽITÍ AR VE FIRMÁCH A STARTUPECH* [online]. Rascasone, březen 2023 [cit. 2023-04-02]. Dostupné z: https://www.rascasone.com/cs/blog/rozsirena-realita-ar-vyuziti-firmy-aplikace#s-jak-yacute-mi-typy-ar-aplikac-iacute-se-muzeme-setkat.

[15] Lin, C.-Y. a Lin, Y.-B. Projection-Based User Interface for Smart Home Environments. In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. 2013, s. 546–549. DOI: 10.1109/COMPSACW.2013.117.

[16] Moreno, D. a Taubin, G. Simple, Accurate, and Robust Projector-Camera Calibration. In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. 2012, s. 464–471. DOI: 10.1109/3DIMPVT.2012.77.

[17] Richard Hartley, A. Z. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. ISBN 9780521540513, 0521540518.

[18] Song, Y., Song, Z. a Wu, S. Calibration Methods of Projector-Camera Structured Light System: A Comparative Analysis. In: *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. 2018, s. 39–43. DOI: 10.1109/ICoIAS.2018.8493769.

[19] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000, sv. 22, č. 11, s. 1330–1334. DOI: 10.1109/34.888718.