



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

VIRTUÁLNÍ PROSTŘEDÍ PRO VIZUALIZACI A SIMULACI V MOBILNÍ ROBOTICE

VIRTUAL ENVIRONMENT FOR VISUALIZATION AND SIMULATION IN MOBILE ROBOTICS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Filip Pár

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Petr Gábrlík, Ph.D.

BRNO 2023



Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Filip Pár

ID: 221311

Ročník: 3

Akademický rok: 2022/23

NÁZEV TÉMATU:

Virtuální prostředí pro vizualizaci a simulaci v mobilní robotice

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vizualizace a simulace reálných robotů ve virtuálním prostředí, zahrnuje zejména tvorbu modelů robotů a map prostředí.

1. Seznamte se s frameworkem ROS2 (Robot Operating System 2), vizualizačním nástrojem Rviz a simulačním prostředím Gazebo.
2. Na systému Linux (Ubuntu) dané nástroje nainstalujte a naučte se je používat.
3. Vytvořte modely čtyř mobilních robotů z pracoviště ÚAMT pro účely vizualizace i simulace.
4. Vytvořte dvourozměrnou mapu (occupancy grid) části pracoviště ÚAMT pro účely navigace.
5. Vytvořte trojrozměrný model části pracoviště ÚAMT pro účely simulace.
6. V bodech 3.–5. úzce spolupracujte s vedoucím, výsledné soubory odevzdejte formou přehledného GIT repositáře respektující běžnou strukturu ROS projektů.

DOPORUČENÁ LITERATURA:

PYO, YoonSeok, HanCheol CHO, RyuWoon JUNG a TaeHoon LIM. ROS Robot Programming. Republic of Korea: ROBOTIS Co., 2017. ISBN 979-11-962307-1-5.

Termín zadání: 6.2.2023

Termín odevzdání: 22.5.2023

Vedoucí práce: Ing. Petr Gábrlík, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Náplní této bakalářské práce je seznámení se s frameworkem ROS2. Dále je v této práci popsán podrobný postup tvoření modelů robotů z pracoviště ÚAMT pro vizualizační nástroj Rviz a simulační prostředí Gazebo. Následně je v práci popsán postup tvoření dvourozměrné mapy pro účely navigace a trojrozměrného modelu části pracoviště ÚAMT pro použití v simulačním prostředí Gazebo.

Klíčová slova

ROS2, RViz, Gazebo, URDF, SDF, Xacro

Abstract

The purpose of this bachelor thesis is to get acquainted with the ROS2 framework. Furthermore, this thesis describes the detailed procedure of creating robot models of the ÚAMT workplace for the Rviz visualization tool and the Gazebo simulation environment. Subsequently, the thesis describes the procedure of creating a two-dimensional map for navigation purposes and a three-dimensional model of a part of the ÚAMT workplace for the use in the Gazebo simulation environment.

Keywords

ROS2, RViz, Gazebo, URDF, SDF, Xacro

Bibliografická citace

PÁR, Filip. Virtuální prostředí pro vizualizaci a simulaci v mobilní robotice [online]. Brno, 2023 [cit. 2023-05-15]. Dostupné z: <https://www.vut.cz/studenti/zav-prace/detail/151652>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Petr Gábrlík.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta: *Filip Pár*

VUT ID studenta: *221311*

Typ práce: *Bakalářská práce*

Akademický rok: *2022/23*

Téma závěrečné práce: *Virtuální prostředí pro vizualizaci
a simulaci v mobilní robotice*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 22. května 2023

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petrovi Gábrlíkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 22. května 2023

podpis autora

Obsah

SEZNAM OBRÁZKŮ.....	9
ÚVOD.....	10
1. POPIS PROGRAMŮ	11
1.1 ROS2.....	11
1.2 RVIZ.....	11
1.3 GAZEBO	11
2. TVORBA ROBOTY	12
2.1 PŘEDLOHA ROBOTY	12
2.2 VYTVOŘENÍ ROBOTY DO RVIZ	12
2.2.1 <i>Knihovna transformací tf2</i>	12
2.3 XACRO	13
2.4 NASTAVENÍ PROSTŘEDÍ	13
2.5 TVOŘENÍ URDF.....	13
2.6 DEFINOVÁNÍ JEDNOTLIVÝCH TVARŮ	14
2.6.1 <i>Tvoření modelu z mesh souborů</i>	15
2.6.2 <i>Nastavení barvy pro jednotlivé části robota</i>	17
2.7 ZOBRAZENÍ ROBOTY VE 3D SIMULÁTORU GAZEBO	17
2.7.1 <i>Kolizní model</i>	17
2.7.2 <i>Fyzikální vlastnosti robota</i>	18
2.7.3 <i>Pluginy</i>	19
2.7.4 <i>Spouštěcí soubor</i>	20
3. VYTVÁŘENÍ SVĚTŮ A MAP PRO POUŽITÍ V GAZEBO A RVIZ	22
3.1 VYTVÁŘENÍ SVĚTŮ PRO SIMULACI V GAZEBO	22
3.1.1 <i>SDF soubor</i>	22
3.2 ZPŮSOBY TVOŘENÍ SVĚTA	23
3.2.1 <i>Tvoření modelů pomocí kódu</i>	23
3.2.2 <i>Grafický editor implementovaný v Gazebo</i>	24
3.3 TVOŘENÍ VLASTNÍCH MODELŮ	25
3.3.1 <i>Vytvoření modelu</i>	26
3.3.2 <i>Struktura složky pro funkční model v Gazebo</i>	28
3.3.3 <i>Soubor model.config</i>	28
3.3.4 <i>SDF formát</i>	29
3.3.5 <i>OGRE material script</i>	30
3.4 TVORBA GAZEBO SVĚTA	31
3.4.1 <i>World file</i>	31
3.4.2 <i>Základní definování souboru</i>	32
3.4.3 <i>Nastavení fyzikálních vlastností</i>	32
3.4.4 <i>Osvětlení světa</i>	32
3.4.5 <i>Vkládání modelů do světa</i>	32
3.4.6 <i>Možnost tvoření světa v simulačním prostředí Gazebo</i>	32
3.5 MŘÍŽKA OBSAZENOSTI	33
3.5.1 <i>Výroba z půdorysu budovy</i>	33

3.5.2	<i>Kolekce dat ze senzorů robotů</i>	34
4.	VÝSLEDNÉ BALÍČKY	35
4.1	BALÍČEK S ROBOTY	35
4.2	BALÍČEK 3D SVĚTA.....	36
5.	ZÁVĚR	39
	SEZNAM SYMBOLŮ A ZKRATEK	42

SEZNAM OBRÁZKŮ

Obrázek 1.1	Foxy Fitzroy [1]	11
Obrázek 2.1	Robot Orpheus X4 [3]	12
Obrázek 2.2	Detailní model kola robota	15
Obrázek 2.3	Zjednodušený model kola robota	16
Obrázek 2.4	Porovnání vizuálního (vlevo) a kolizního (vpravo) modelu.....	18
Obrázek 2.5	Výsledný model robota zobrazený v prostředí Gazebo	21
Obrázek 2.6	Výsledný model robota v prostředí RViz	21
Obrázek 3.1	Půdorys budovy	22
Obrázek 3.2	Inspektor zdí v Gazebu.....	24
Obrázek 3.3	Vytvořený půdorys místností.....	25
Obrázek 3.4	Vytvořený model budovy	25
Obrázek 3.5	Vytvořený model skříně	26
Obrázek 3.6	UV mapa modelu skříně.....	27
Obrázek 3.7	UV mapa vyexportovaná do obrázku	27
Obrázek 3.8	Otexturovaná UV mapa skříně.....	28
Obrázek 3.9	Výsledný model skříně zobrazený v Gazebu	31
Obrázek 3.10	Půdorys budovy připravený na konverzi formátu.....	33
Obrázek 3.11	Nasnímaná mřížka obsazenosti.....	34
Obrázek 4.1	Výsledný robot Orpheus AC v prostředí Gazeba (vpravo) a Rvizu (vlevo).....	35
Obrázek 4.2	Výsledný robot Orpheus X4 v prostředí Gazeba (vpravo).....	36
Obrázek 4.3	Výsledný robot Loki v prostředí Gazeba (vpravo)	36
Obrázek 4.4	Výsledný model prostředí na základě předlohy pracoviště ÚAMT	37
Obrázek 4.5	Detailní pohled na model prostředí 1.....	37
Obrázek 4.6	Detailní pohled na výsledný model prostředí 2	38

ÚVOD

Dnešní doba jde velmi rychle kupředu a s ní i nové technologie. Vývoj nových technologií ale stojí vývoj velké peníze. Snížení nákladů při vyvíjení technologií v průmyslu za pomoci testování ve virtuálním prostředí získalo z toho důvodu v posledním desetiletí velké popularity. Tím se velkou mírou zasadila nezisková společnost Open Robotics, která pravidelně každoročně vydává novou open source verzi programu ROS2. Díky této skutečnosti má velkou komunitu a je jednoduché najít pomoc při vývoji vlastního robota, čímž se tato práce zabývá.

Obsahem práce je seznámení se frameworkem ROS2 i s jeho vizualizačním a simulačním prostředím. Následující částí je vytvoření modelů robotů Orpheus X4, Orpheus AC a Loki. Následně převedení laboratoří robotizace do simulačního prostředí.

V hlavní části je podrobněji rozebrán postup tvoření modelu robota krok po kroku. V navazujících částech jsou popsány odlišné postupy a metody vytváření světů v Gazebu dále také vytváření 2D mřížky obsazenosti, pro budoucí použití k navigaci robota.

1. POPIS PROGRAMŮ

1.1 ROS2

ROS2 neboli Robot Operating System je volně přístupná sada softwarových knihoven a nástrojů, které pomáhají vytvářet robotické aplikace. Procesy jsou zpracovány v nodech, neboli uzlech, které odesílají data. Aktuální jsou podporovány dvě distribuce vydání a to Foxy Fitzroy a Humble Hawksbill. Tato práce je vypracována na distribuci Foxy Fitzroy, později na Humble Hawksbill. [1]



Obrázek 1.1 Foxy Fitzroy [1]

1.2 Rviz

Je 3D vizualizační nástroj pro aplikace ROS2. Poskytuje pohled na váš model robota, zachycuje informace ze senzorů robota a sdílí zachycená data. Dokáže zobrazovat data z kamery, laserů, z 3D a 2D zařízení. [2]

1.3 Gazebo

Gazebo je 3D robotický simulátor. Jeho úkolem je simulování robota, při co největší snaze k realističnosti, což usnadňuje vývoj a při přechodu do reálného prostředí není zapotřebí dělat velké změny v kódu. Jelikož je Gazebo vyvíjeno stejnou společností jako ROS2 je jejich propojení velmi přímočaré a bezproblémové. [2]

2. TVORBA ROBOTA

V této kapitole bude popsán postup tvoření robota, který bude možné zobrazit v Rvizu a později i v Gazebu.

2.1 Předloha robota

Jako předloha pro vytvoření robota ve virtuálním prostředí byl zvolen středně velký robot Orpheus X4 se čtyřmi koly, který je vybaven diferenciálním pohonem.



Obrázek 2.1 Robot Orpheus X4 [3]

2.2 Vytvoření robota do Rviz

Pro tvoření modelu robota byl zvolen formát Unified Robot Description (URDF), což je soubor psaný jazykem XML, který reprezentuje model robota. Robot byl tvořen pomocí balíčku Navigation2.

Aby tento robot fungoval správně, musí být popsány transformace mezi hlavním tělem robota (link) a ostatními částmi robota. Tyto transformace dále tvoří takzvaný transformační strom. Při pouze jednom spoji například mezi `first_link` a `second_link` je transformační strom jednoduchý a k realizaci nám pouze jeden publisher. Při tvorbě složitějšího robota s větším počtem linků je zapotřebí vytvořit velký počet publisherů, což může být velmi zdlouhavé. Z tohoto důvodu je vhodné použít balíček Robot State Publisher, který se stará o sdílení transformací. [4]

2.2.1 Knihovna transformací tf2

Tato knihovna je druhá generace transformační knihovny, která umožňuje uživateli sledovat více souřadnicových rámců v průběhu času. Balíček Robot state publisher

spolupracuje s tf2 balíčkem a spolu sdílí všechny nezbytné transformace, které lze přímo odvodit z geometrie a struktury robota. Ke správné funkčnosti stačí dodat korektně napsaný URDF soubor vašeho robota, o zbytek se postarají tyto balíčky. [5]

2.3 Xacro

Důležitou vlastností URDF je možnost použít balíček Xacro (XML Macros). Jedná se o balíček, pomocí kterého je možné používat při psaní v jazyce XML makra, díky kterému je možné zpřehlednit a zkrátit daný kód robota. Pomocí Xacra lze vytvořit konstanty, které lze použít opakovaně v souboru. Je tedy jednodušší provést změnu parametru robota, jelikož stačí přepsat hodnotu pouze na jednom řádku.

2.4 Nastavení prostředí

Pro vytvoření robota bylo zapotřebí nainstalovat balíčky Navigation2, joint state publisher a Xacro. Nainstalování balíčků bylo provedeno zadáním těchto příkazů do konzole.

```
sudo apt install ros-foxy-navigation2
sudo apt install ros-foxy-nav2-bringup
sudo apt install ros-foxy-joint-state-publisher-gui
sudo apt install ros-foxy-xacro
```

Jako další byla vytvořena složka našeho balíčku, s názvem orpheus_x4.

```
ros2 pkg create --build-type ament_cmake orpheus_x4_description
```

2.5 Tvoření URDF

V tuto chvíli máme připraveno vše pro započítí tvoření našeho robota. V podsložce src/description byl vytvořen soubor orpheus_x4_description.urdf do kterého byla vepsána definice souboru a element robota.

```
<?xml version="1.0"?>
<robot name="orpheus_x4" xmlns:xacro="http://ros.org/wiki/xacro">
</robot>
```

V dalším kroku byly do souboru pomocí Xacro elementů vloženy makro konstanty robota.

```
<!-- Konstanty robota -->
<xacro:property name="base_width" value="0.43"/>
<xacro:property name="base_length" value="0.557"/>
<xacro:property name="base_height" value="0.11"/>
```

```

<xacro:property name="wheel_radius" value="0.20"/>
<xacro:property name="wheel_width" value="0.06"/>
<xacro:property name="wheel_ygap" value="0.033"/>
<xacro:property name="wheel_zoff" value="0.005"/>
<xacro:property name="wheel_xoff" value="0.23"/>

```

Hodnoty value udávají rozměr v metrech. Elementy s názvem base_* slouží ke specifikování rozměrů těla robota. Elementy wheel_radius a wheel_width udávají velikost kola. Pro vytvoření mezery kol na ose y od robota byl specifikován element wheel_ygap. Pro změnu pozice na ose x a ose z od středu robota byly specifikovány elementy wheel_xoff a wheel_zoff.

Aby bylo možné 3D model vizuálně zobrazit, bylo nutné v elementu visual určit tvar a barvu. Element visual znázorňuje pouze vzhled objektu. Pro nastavení fyzikálních vlastností a kolizí je zapotřebí elementu inertial, který bude probrán později v kapitole vkládání robota do programu Gazebo.

2.6 Definování jednotlivých tvarů

Dále bylo zapotřebí definovat krabici, znázorňující tělo robota. K vytvoření kváдру byly využity konstanty, které jsme si vytvořili v předchozích bodech.

```

<!-- Tělo robota -->

<link name="base_link">
  <visual>
    <geometry>
      <box size="${base_length} ${base_width} ${base_height}"/>
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
  <collision>
    <geometry>
      <box size="${base_length} ${base_width} ${base_height}"/>
    </geometry>
  </collision>

```

Jako další krok definujeme base_footprint. Tento link je pouze virtuální a nemá žádné vizuální ani kolizní prvky. Používají jej některé další navigační balíčky, které si z něj určují střed robota. Kolem tohoto bodu se vytvoří zóna, nejčastěji kruh, který znázorňuje půdorys robota. Tento půdorys robota se poté využívá například k vyhýbání se překážkám.

Pokračujeme definováním jointu mezi base_link a base_footprint. Joint znázorňuje kinematické a dynamické vlastnosti propojení souřadnicových rámců. Mezi typy jointů neboli kloubů se řadí:

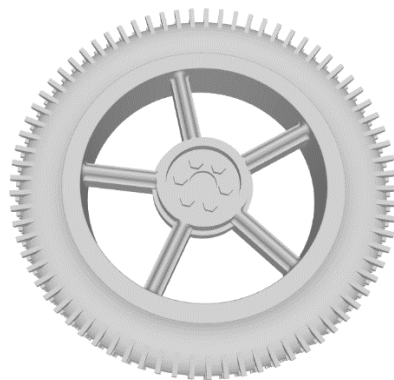
- Continuous – joint, který se otáčí kolem jedné osy bez limitovaného rozsahu otáčení.
- Revolute – který je stejný jako continuous, ale má dané horní a spodní limity otáčení.
- Prismatic – joint, který se posouvá po jedné ose s limity posuvu.
- Fixed – má všechny stupně volnosti zakázány, nejedná se tedy o kloub, ale o pevný spoj.

V tomto případě je link pevně svázaný s tělem robota, z tohoto důvodu byl použit fixed joint. Element parent reprezentuje jméno linku na který je nadřazeným linkem pro náš joint a element child reprezentuje podřadným linkem. Origin element značí posun a změnu natočení podřadného linku vůči nadřazenému. [6]

```
<link name="base_footprint"/>
  <joint name="base_joint" type="fixed">
    <parent link="base_link"/>
    <child link="base_footprint"/>
    <origin xyz="0.0 0.0 ${-(wheel_radius+wheel_zoff)}" rpy="0 0 0"/>
  </joint>
```

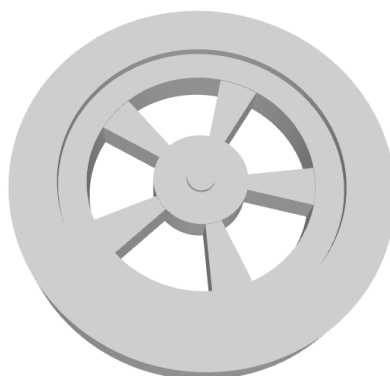
2.6.1 Tvoření modelu z mesh souborů

Jako další bylo zapotřebí vytvořit kola robota. V tomto případě se již jedná o složitější tvar a z toho důvodu nebylo možné vytvořit 3D model. Tento 3D model byl vytvořen v Fusion 360, což je cloudový software pro 3D modelování určený k navrhování a výrobě 3D modelů. Formát URDF akceptuje dva typy 3D modelů, STL a OBJ. Pro toto využití byl vybrán formát STL, ve kterém jsou modely sice méně detailní, ale výsledný soubor má menší velikost.



Obrázek 2.2 Detailní model kola robota

Při pozdějším testování se ukázalo, že je model až příliš detailní a náročný pro výpočetní model Gazebo. Z tohoto důvodu byl vytvořen zjednodušený model kola, pomocí kterého se podařilo snížit náročnost a zlepšit chod simulace.



Obrázek 2.3 Zjednodušený model kola robota

Jelikož je pozice všech 4 kol symetrická od středu báze robota, bylo vytvořeno makro pro každé kolo, pomocí kterého bylo možné zrcadlit jednotlivá kola od středu těla robota. Díky vytvoření tohoto makra nebylo zapotřebí psát kód pro každé kolo zvlášť a bylo tím docíleno zkrácení kódu. V makru byl specifikován prefix, což je název daného kola, `x_reflect` zrcadlení na ose `x` a `y_reflect` zrcadlení na ose `y`. Jelikož se jedná o kolo, musel být joint specifikován jako `continuous`, aby se mohl otáčet. U `continuous` jointu bylo potřebné specifikovat `element axis`, který udává osu, okolo které se kolo bude točit.[4]

```

<!-- Kola -->
  <xacro:macro name="wheel" params="prefix x_reflect y_reflect">
    <link name="${prefix}_link">
      <visual>
        <origin xyz="0 0 0" rpy="${pi/2} 0 0"/>
        <geometry>
          <mesh scale=" 0.001 0.001 0.001" filename="file://$(find
robot_models)/meshes/orpheus_x4/Wheel_v3.stl"/>
        </geometry>
        <material name="Black">
          <color rgba="0.15 0.15 0.15 1.0"/>
        </material>
      </visual>

      <joint name="${prefix}_joint" type="continuous">
        <parent link="base_link"/>
        <child link="${prefix}_link"/>
        <origin xyz="${x_reflect*wheel_xoff}
${y_reflect*(base_width/2+wheel_ygap)} ${-wheel_zoff}" rpy="0 0 0"/>
        <axis xyz="0 1 0"/>
      </joint>
    </xacro:macro>
  <!-- Nastavení pozic jednotlivých kol -->
  <xacro:wheel prefix="drivewhl_rl" x_reflect="-1" y_reflect="1" />
  <xacro:wheel prefix="drivewhl_rr" x_reflect="-1" y_reflect="-1" />
  <xacro:wheel prefix="drivewhl_l" x_reflect="1" y_reflect="1" />

```

```
<xacro:wheel prefix="drivewhl_r" x_reflect="1" y_reflect="-1" />
```

2.6.2 Nastavení barvy pro jednotlivé části robota

V základním nastavení při neudání atribut material má určená část modelu v Rvizu světle šedou barvu. Při potřebě změny barvy je možné přidat ve visual elementu barvu. Barva se udává v RGBA formátu, což je RGB formát rozšířený o alfa složku, která určuje průhlednost barvy, kde 0 znamená plně průhledná a 1 neprůhledná. Jednotlivé RGB složky se zadávají v rozsahu 0 – 1 (0 – 255), udávající intenzitu barvy.[15]

```
<material name="Black">
  <color rgba="0.15 0.15 0.15 1.0"/>
</material>
```

Stejným principem byly vloženy další části robota, jakožto deska a platforma pro připevnění příslušenství. Soubor URDF pro vizuální zobrazení robota v Rviz je hotový. Aby bylo možné využít použité balíčky bylo nutné v souboru package.xml použité balíčky specifikovat.

```
<exec_depend>joint_state_publisher</exec_depend>
<exec_depend>robot_state_publisher</exec_depend>
<exec_depend>rviz</exec_depend>
<exec_depend>xacro</exec_depend>
```

2.7 Zobrazení robota ve 3D simulátoru Gazebo

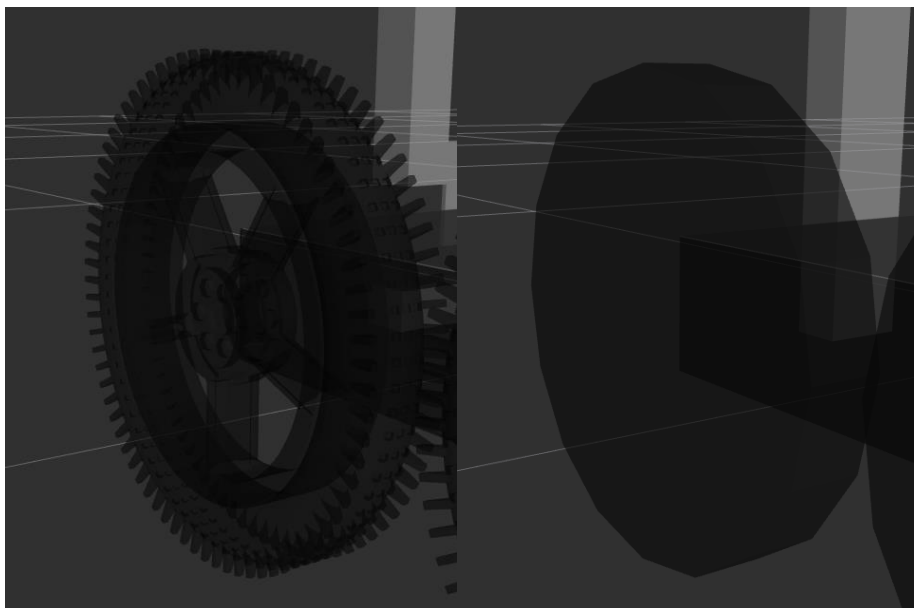
Gazebo používá pro specifikaci robota v simulačním prostředí Simulation Description Format (SDF). Naštěstí Gazebo umí přeložit URDF formát na SDF formát. Aby bylo možné robota zobrazit v 3D simulátoru, bylo nutné modelu robota dodat fyzikální vlastnosti a kolizní model, tak aby simulátor věděl, jak se robot má chovat ve virtuálním prostředí.

V této části se bude pracovat s balíčkem gazebo_ros_pkgs, z tohoto důvodu bylo nutné ho nainstalovat pomocí tohoto příkazu do konzole:

```
sudo apt install ros-foxy-gazebo-ros-pkgs
```

2.7.1 Kolizní model

Dosud byly linky specifikovány pouze vizuálně, aby však ve 3D simulaci v Gazebo fungovala detekce kolize, musíme také definovat kolizní element. Tento element se vkládá do elementu link a má nejčastěji stejné rozměry jako vizuální element. V Případech, kdy je tvar složitější, bývá v kolizním elementu nahrazen jednodušším tvarem. Tímto se zmenší náročnost na výpočetní techniku.



Obrázek 2.4 Porovnání vizuálního (vlevo) a kolizního (vpravo) modelu

Často se takto také zvyšuje bezpečnost, kdy chceme omezit, nebo snížit pohyb okolo citlivých částí našeho robota. [7]

```
<collision>
  <origin xyz="0 0 0" rpy="{pi/2} 0 0"/>
  <geometry>
    <cylinder radius="{wheel_radius}" length="{wheel_width}"/>
  </geometry>
</collision>
```

2.7.2 Fyzikální vlastnosti robota

Pro každou část robota je nutné vytvořit vlastní inerciální element. Tímto elementem je určena setrvačnost jednotlivých částí robota. Tato setrvačnost je definována maticí, která se pro každý tvar liší. Pro psaní inerciálních vlastností bylo opět využito makro. [7]

Setrvačnost válcovitého tvaru je definována pomocí této matice: [8]

$$I = \begin{bmatrix} \frac{1}{12}m(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \quad (2.1)$$

Kde m váha [kg]
 r poloměr [m]
 h výška [m]

```
<xacro:macro name="cylinder_inertia" params="m r h">
  <inertial>
```

```

    <origin xyz="0 0 0" rpy="{pi/2} 0 0" />
    <mass value="{m}" />
    <inertia ixx="{(m/12) * (3*r*r + h*h)}" ixy = "0" ixz = "0" iyy="{(m/12)
* (3*r*r + h*h)}" iyz = "0" izz="{(m/2) * (r*r)}" />
    </inertial>
</xacro:macro>

```

Setrvačnost krabicovitého tvaru je definována touto maticí: [8]

$$I = \begin{bmatrix} \frac{1}{12}m(h^2 + d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2 + d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2 + h^2) \end{bmatrix} \quad (2.2)$$

Kde m váha [kg]
w šířka [m]
h výška [m]
d hloubka [m]

```

<xacro:macro name="box_inertia" params="m w h d">
    <inertial>
    <origin xyz="0 0 0" rpy="{pi/2} 0 {pi/2}" />
    <mass value="{m}" />
    <inertia ixx="{(m/12) * (h*h + d*d)}" ixy="0.0" ixz="0.0" iyy="{(m/12)
* (w*w + d*d)}" iyz="0.0" izz="{(m/12) * (w*w + h*h)}" />
    </inertial>
</xacro:macro>

```

2.7.3 Pluginy

Pro komunikaci mezi ROS2 a Gazebo se musí dynamicky připojit knihovna ROS2, která dá instrukce Gazebo, které má provést.

Gazebo podporuje tři typy pluginů:

1. Modelové pluginy – starající se o fyzikální stránku modelu
2. Sensorové pluginy – přidavné senzory k modelu
3. Vizualní pluginy – upravují vizualní stránku modelu [9]

V našem případě použijeme plugin differential drive ModelPlugin aby bylo možné s robotem v pohybovat. Pluginy se vkládají na konec souboru. V pluginu definujeme počet párů kol, vzdálenost mezi koly, průměr kol a maximální povolenou akceleraci a točivý moment.

```

<gazebo>
  <plugin name='diff_drive' filename='libgazebo_ros_diff_drive.so'>
    <ros>
      <namespace>/demo</namespace>
    </ros>
  <!-- Počet párů kol -->

```

```

    <num_wheel_pairs>2</num_wheel_pairs>

<!-- Kola -->
  <left_joint>drivewhl_l_joint</left_joint>
  <right_joint>drivewhl_r_joint</right_joint>
  <left_joint>drivewhl_rl_joint</left_joint>
  <right_joint>drivewhl_rr_joint</right_joint>

<!-- kinematika -->
  <wheel_separation>0.44</wheel_separation>
  <wheel_separation>0.44</wheel_separation>
  <wheel_diameter>0.4</wheel_diameter>
  <wheel_diameter>0.4</wheel_diameter>

<!-- limity -->
  <max_wheel_torque>20</max_wheel_torque>
  <max_wheel_acceleration>1.0</max_wheel_acceleration>

<!-- výstup -->
  <publish_odom>>true</publish_odom>
  <publish_odom_tf>true</publish_odom_tf>
  <publish_wheel_tf>true</publish_wheel_tf>

  <odometry_frame>odom</odometry_frame>
  <robot_base_frame>base_link</robot_base_frame>
</plugin>
</gazebo>

```

V tuto chvíli je model robota dokončen a připraven na zobrazení.

2.7.4 Spouštěcí soubor

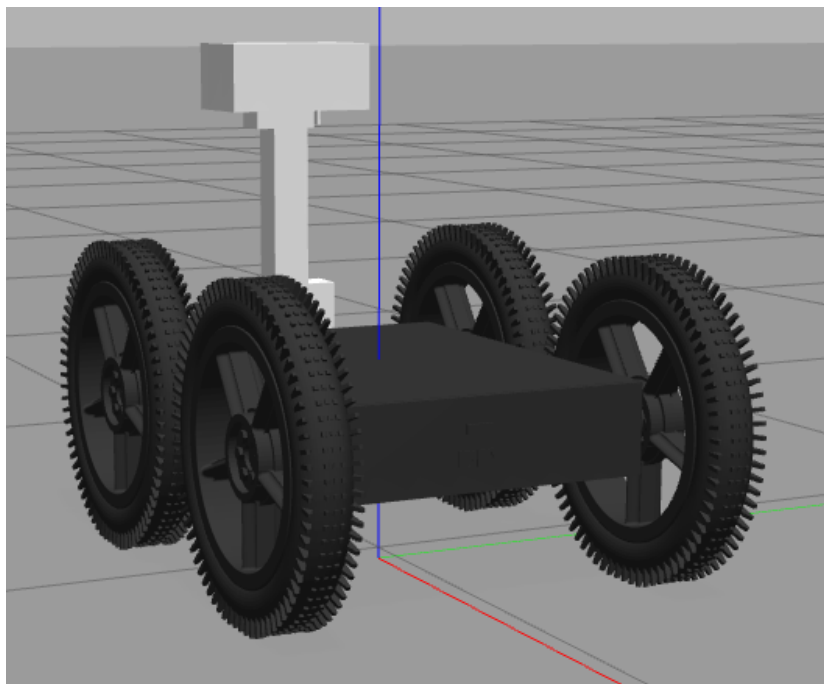
Pro zobrazení robota se všemi funkcemi by bylo nutné vypsát enormně velký počet řádků do konzole. Z tohoto důvodu se pro spuštění robota využívá spouštěcí soubor (launch file), který se postará o spuštění. V tomto souboru byla definována adresa našeho balíčku a adresa modelu robota. Dále jsou definovány jednotlivé nody (uzly), které je zapotřebí spustit pro správnou funkci robota. Launch file byl upraven z [4].

Pro zobrazení robota byl zadán příkaz k sestavení našeho balíčku.

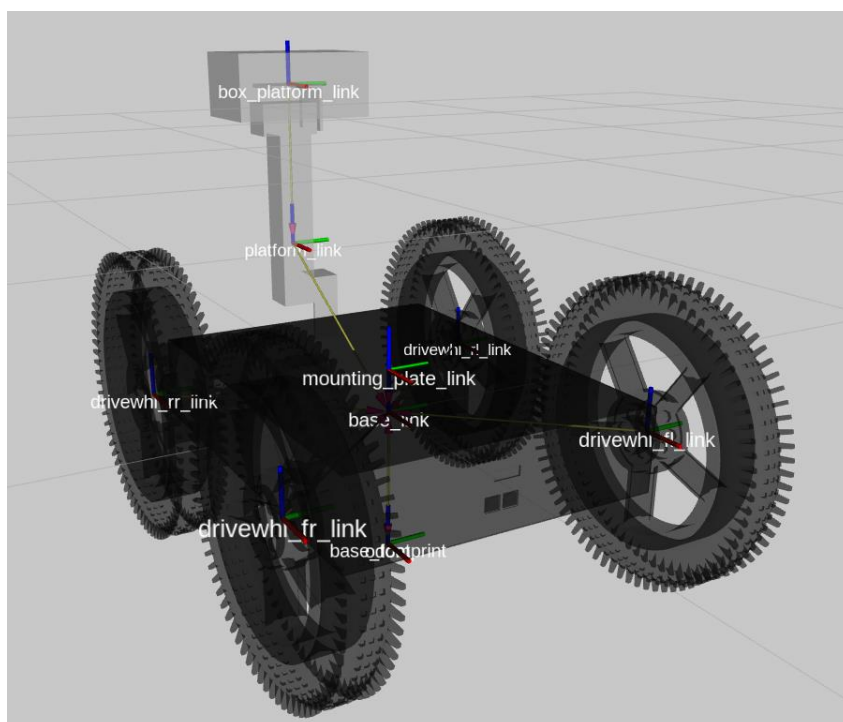
```
colcon build
. install/setup.bash
```

V tuto chvíli nám již nic nebrání k zobrazení našeho robota. Zobrazení provedeme příkazem:

```
ros2 launch robot_models display.launch.py
```



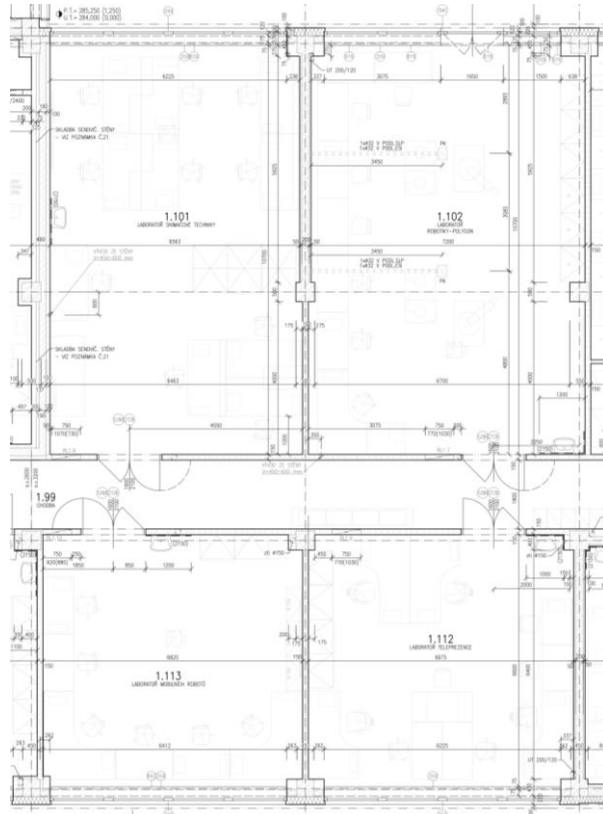
Obrázek 2.5 Výsledný model robota zobrazený v prostředí Gazebo



Obrázek 2.6 Výsledný model robota v prostředí RViz

3. VYTVÁŘENÍ SVĚTŮ A MAP PRO POUŽITÍ V GAZEBU A RVIZ

Mojí úlohou bylo vytvoření virtuálního světa, laboratoří robotizace, konkrétně místností SE 1.112, SE 1.113 a SE 1.102 a přilehlé chodby. Jako podkladem mi byl náčrt půdorysu budovy.



Obrázek 3.1 Půdorys budovy

3.1 Vytváření světů pro simulaci v Gazebu

Pro zobrazení světa v Gazebu je zapotřebí soubor formátu SDF (Simulation Description Format) v jazyce XML.

3.1.1 SDF soubor

Syntaxe SDF souboru je velmi podobná s URDF. SDF soubor na rozdíl od URDF umí navíc specifikovat i objekty, které nejsou roboty. Lze pomocí něj popsat vše, co se týče simulovaného světa až po robota samotného. Je vysoce škálovatelný a extrémně snadný na přidávání a upravování prvků. [11]

3.2 Způsoby tvoření světa

V této kapitole budou probrány styly, kterými je možné tvořit modely do světa Gazebo a poté tvoření samotného Gazebo světa.

3.2.1 Tvoření modelů pomocí kódu

První způsob tvoření modelů je psaním kódu a princip je velmi podobný, jako u tvoření robota. Nevýhodou tohoto stylu tvoření je absence vizuálního zobrazení aktuálně tvořeného modelu. Pro vizuální kontrolu je nutné soubor nejdříve uložit, poté spustit Gazebo a zobrazit, což na slabších počítačích trvá příliš dlouho. Modely lze tvořit z jednoduchých tvarů typu box, koule a další.

```
<link name='Wall_0'>
  <collision name='Wall_0_Collision'>
    <geometry>
      <box>
        <size>4 0.15 2.5</size>
      </box>
    </geometry>
    <pose>0 0 1.25 0 -0 0</pose>
  </collision>
  <visual name='Wall_0_Visual'>
    <pose>0 0 1.25 0 -0 0</pose>
    <geometry>
      <box>
        <size>4 0.15 2.5</size>
      </box>
    </geometry>
    <material>
      <script>
        <uri>file://media/materials/scripts/gazebo.material</uri>
        <name>Gazebo/Grey</name>
      </script>
      <ambient>0.921569 0.807843 0.615686 1</ambient>
    </material>
    <meta>
      <layer>0</layer>
    </meta>
  </visual>
  <pose>-2.46639 -3.07195 0 0 -0 0</pose>
</link>
```

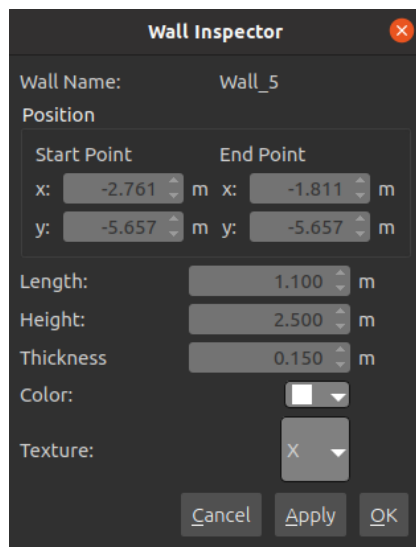
Oproti URDF se zde objevují nové elementy. Element *pose* udává počáteční bod, ze kterého dané těleso vychází. Element *material*, obsahuje sub-elementy, pomocí kterých definujeme, ze kterého materiálu je těleso postavené a jestli pohlcuje či vyzařuje světlo. Pomocí elementu *layer* je možné určit vrstvu, neboli patro, ve kterém se má objekt zobrazit. Pro určení zdali bude těleso statické je použit element *static*.

3.2.2 Grafický editor implementovaný v Gazebu

Gazebo má v sobě implementovaný grafický editor, který je možný použít jak na tvoření robotů, tak také na tvoření budov a světů. Pro otevření editoru stačí v horní liště Gazeba kliknout na Edit a poté na Building editor.

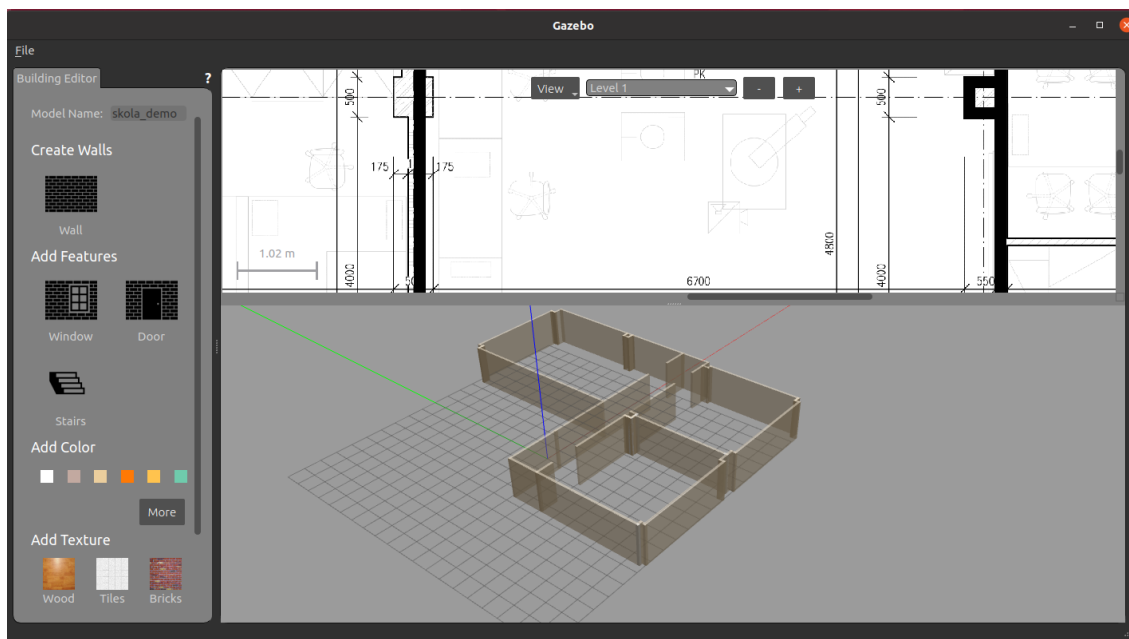
Editor je rozdělen na 2D pohled a 3D pohled pro rychlejší tvoření. Velikou pomocí pro tvoření budov je možnost importování obrázku půdorysu jako podkladu pro tvoření ve 2D pohledu. Při importování obrázku do Gazeba lze nastavit měřítko, ve kterém se obrázek vloží, aby bylo možné svět tvořit s co největší přesností.

Pro vytvoření zdi stačí kliknout na 2D pole, posunout myš požadovaným směrem a poté znovu kliknout. Vytvořenou zeď je možné tahem posunout. Dále je možné na zeď dvakrát kliknout a otevřít inspektora zdi, ve kterém lze upravit počáteční a koncový bod, rozměry a vzhled zdi.



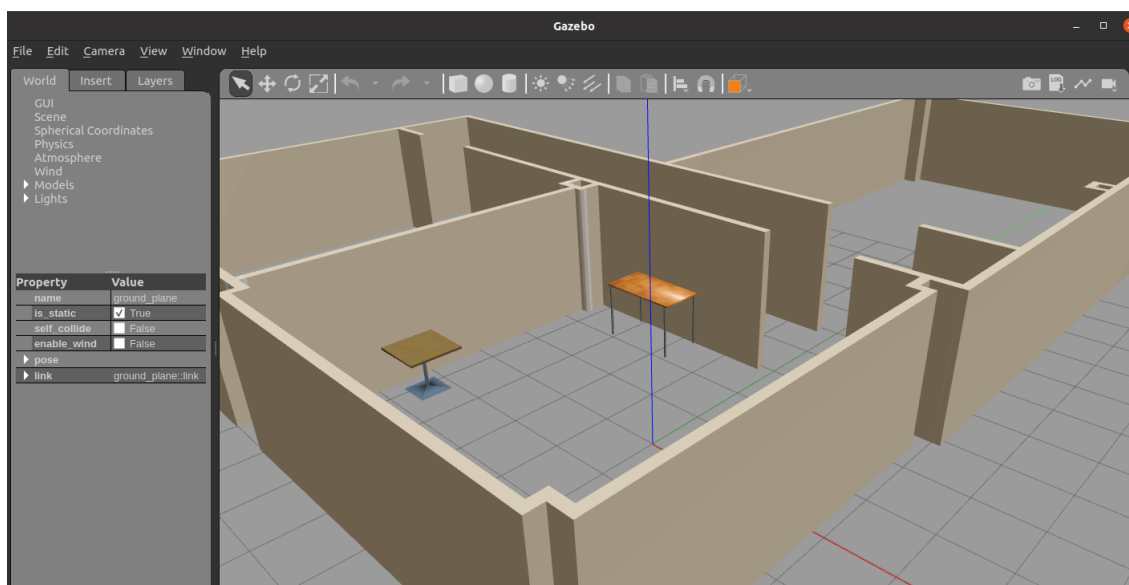
Obrázek 3.2 Inspektor zdi v Gazebu

Jelikož byla tato práce tvořena na virtuálním počítači, nebylo možné využít grafickou kartu. Z tohoto důvodu byl editor velmi zasekaný a při pokusech vložení dveří či oken se editor zhroutil, což bylo zapříčiněno spíše špatnou optimalizací tohoto editoru. Uložením vznikne soubor model.sdf ve kterém je uložený náš model a model.config ve kterém jsou uloženy administrativní informace.



Obrázek 3.3 Vytvořený půdorys místností

Výsledný model lze poté uložit možností File a poté Save. Pomocí možnosti insert lze do prostředí vložit 3D objekty. Gazebo má již v základu velkou knihovnu 3D modelů. V případě, že požadovaný objekt nenajdeme, je možné prozkoumat veřejně přístupné knihovny 3D modelů a objekt importovat do Gazeba.



Obrázek 3.4 Vytvořený model budovy

3.3 Tvoření vlastních modelů

Pro vytvoření 3D světa, který by se co nejvíce blížil reálnému vzezření vybraných prostor bylo nadále vyrobit modely stolů, oken, dveří a dalších doplňků vyskytujících se v daných

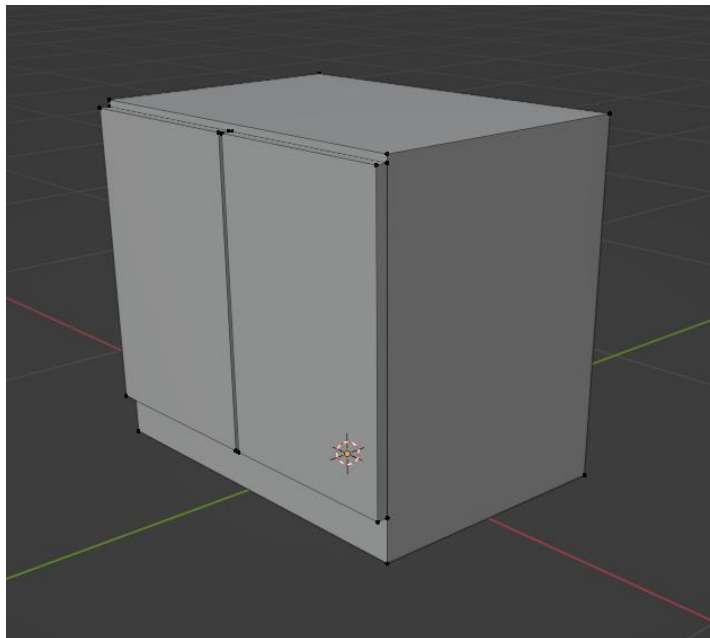
prostorech.

Jelikož jde již o složitější modely, které se často skládají z více materiálů, nebylo již možné tvořit modely v jednoduchém STL formátu, který neobsahuje žádné textury. Z tohoto důvodu bylo nutné tvořit modely ve formátu DAE, které již mohou obsahovat i textury. V následujících kapitolách si projdeme postup tvoření těchto modelů.

3.3.1 Vytvoření modelu

Celý postup si popíšeme na tvorbě modelu skříně. Pro tvorbu modelů byl vybrán program Blender, jelikož se jedná o nejvíce rozšířený a používaný opensource program na vytváření 3D modelů a simulací.

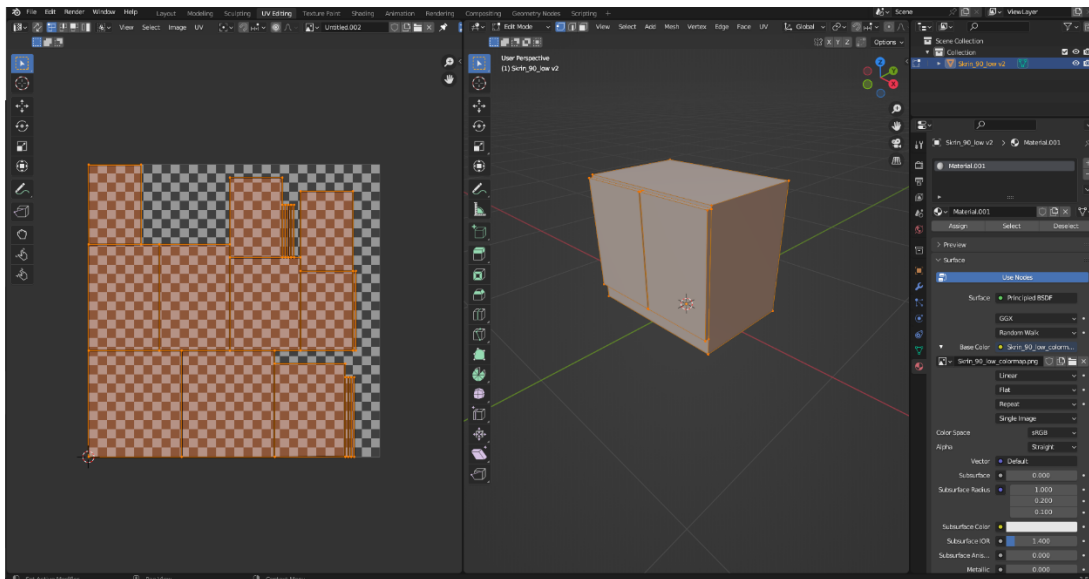
V první řadě je nutné vytvořit 3D model o požadovaných rozměrech a detailech. Vzhledem ke snaze o plynulý chod simulace bylo potřeba udělat u jednotlivých modelů kompromis mezi počtem detailů a náročností na simulaci.



Obrázek 3.5 Vytvořený model skříně

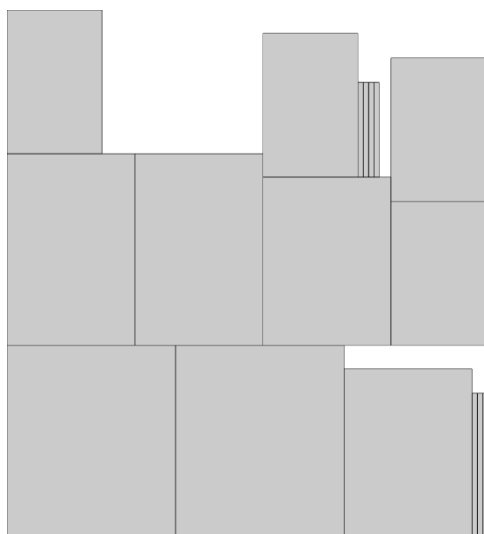
V dalším kroku vytvoříme takzvanou UV mapu, což je rozložení 3D sítě modelu do 2D obrázku. Toto nám dále umožní vytvořit texturu pro daný model.

V hlavní nabídce vybereme *UV Editing*, následně označíme celý model a v sekundární nabídce vybereme *UV -> Smart UV project* a potvrdíme tlačítkem *Ok*. V tuto chvíli máme vytvořené obrysy textur. Následně musíme obrysy textur vyexportovat možností *UV -> Export UV layout*.



Obrázek 3.6 UV mapa modelu skříně

Vyexportovaný obrázek otevřeme v malování nebo jiném obrázkovém editoru a dané plochy vyplníme požadovanou barvou, nebo překryjeme obrázkem požadovaného stylu. Obrázek uložíme.



Obrázek 3.7 UV mapa vyexportovaná do obrázku

V záložce texture paint v možnosti materiál properties vytvoříme nový materiál a jako base color zvolíme námi vytvořený obrázek.



Obrázek 3.8 Otexturovaná UV mapa skříně

Model máme hotový a stačí ho vyexportovat ve formátu DAE.

3.3.2 Struktura složky pro funkční model v Gazebu

Pro správnou funkci modelu v Gazebu je nutné dodržet tuto strukturu složek a souborů. Jejich funkci si v dalších kapitolách popíšeme

```
├─ skrin_90_low/  
  ├─ materials/  
    ├─ scripts/  
    │   └─ skrin_90_low.material  
    └─ textures/  
        └─ Skrin_90_low_colormap.png  
  ├─ meshes/  
  │   └─ model.dae  
  ├─ model.config  
  └─ skrin_90_low.sdf
```

3.3.3 Soubor model.config

Soubor model.config je ve formátu XML. Je v něm uveden název modelu, jeho aktuální verze a odkaz na SDF soubor daného modelu. Dále je zde uvedeno autorství modelu a jeho krátký popis. [15]

```
<?xml version="1.0"?>  
  
<model>  
  <name>skrin_90_low</name>  
  <version>1.0</version>  
  <sdf version="1.5">skrin_90_low.sdf</sdf>  
  <author>  
    <name>Filip Par</name>  
    <email>xparfi00@vutbr.cz</email>  
  </author>  
  
  <description>  
    skrin 90 nizka  
  </description>  
</model>
```

3.3.4 SDF formát

SDF format (Simulation Description Format), někdy zkráceně SDF, je formát XML, který popisuje objekty a prostředí pro simulátory robotů, vizualizaci a řízení. SDF format byl původně vyvinut jako součást simulátoru robotů Gazebo a byl navržen s ohledem na vědecké aplikace robotů. V průběhu let se SDF format stal stabilním, robustním a rozšiřitelným formátem schopným popsat všechny aspekty robotů, statických a dynamických objektů, osvětlení, terénu i fyziky.[16]

```
<?xml version="1.0"?>
<sdf version="1.5">
<model name="skrin_90_low">
  <pose>0 0 0 0 0 0</pose>
  <static>true</static>
  <link name="skrin_90_low_collision">
    <collision name="skrin_90_low_collision">
      <geometry>
        <mesh>
          <uri>model://skrin_90_low/meshes/model.dae</uri>
        </mesh>
      </geometry>
    </collision>
  <visual name="skrin_90_low">
    <cast_shadows>>false</cast_shadows>
    <geometry>
      <mesh>
        <uri> model://skrin_90_low/meshes/model.dae</uri>
      </mesh>
    </geometry>
    <material>
    <script>
      <uri> model://skrin_90_low/material/scripts</uri>
      <uri> model://skrin_90_low/material/textures</uri>
      <name>skrin_90_low </name>
    </script>
    </material>
  </visual>
</link>
</model>
</sdf>
```

Formát tohoto souboru je velmi podobný formátu URDF. Elementem *model* udáváme, že se jedná o fyzikální objekt. V elementu *pose* lze upravit posunutí a rotaci modelu vůči nulovým souřadnicím. Elementem *static* lze booleovou hodnotou předat fyzikálnímu modelu simulátoru, zda se jedná o statický model se kterým je možno pohybovat.

V elementu *collision* se ukrývá subelement *geometry*, ve kterém je uveden odkaz na námi vytvořený DAE soubor, který se nachází ve složce *meshes*.

Element utvářející vizuální část modelu obsahuje totožný subelement *geometry*. Oproti koliznímu elementu lze zde nastavit, zda bude model generovat stíny. V našem případě jsme nechali stíny vypnuté, jelikož je to pro účel naší simulace nepotřebné a zbytečně by byla zatížena výpočetní technika. Dále je zde subelement *materiál*, ve kterém

je uveden název a odkaz na materiální skript s textury. Tento skript zařídí implementaci textur na daný model.

3.3.5 OGRE material script

Pro zobrazení textury na modelu v Gazebu se používá OGRE material script, který se stará o přiřazení materiálu na náš model. Tento script je psaný v jazyce C++.

```
material skrin_90_low
{
    recive_shadows off
    technique
    {
        pass
        {
            ambient 1 1 1 1.000000
            diffuse 1 1 1 1.000000
            specular 0.03 0.03 0.03 1.000000
            texture_unit
            {
                texture Skrin_90__low_colormap.png
            }
        }
    }
}
```

V tomto souboru je specifikován název materiálu *material skrin_90_low*, odkaz na texturu *texture Skrin_90__low_colormap.png*. Atribut *recive_shadows* udává, zda-li se mají na materiálu objevovat textury.

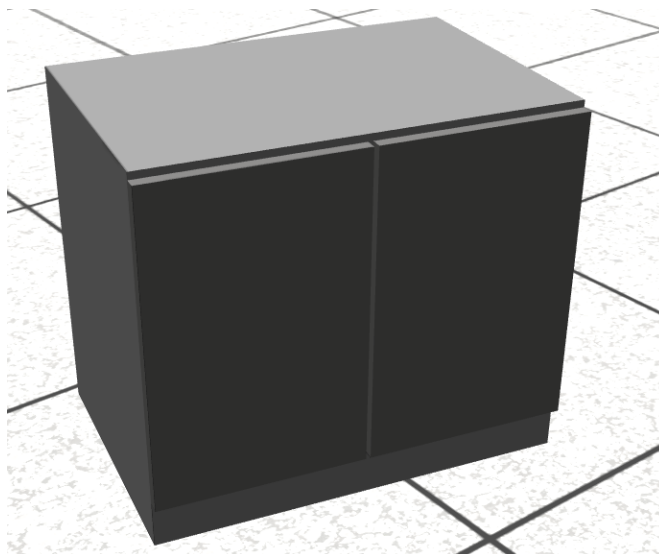
Výsledná barva objektu závisí jak na materiálu, tak na světlech, která na něj svítí. Hodnoty na světlech představují intenzitu vyzařovaného světla. Hodnoty na materiálu představují procento světla, které objekt odráží. Stejný materiál může mít různé barvy v závislosti na světle, které na něj dopadá.

Uvažujme například materiál s difúzní barvou RGBA(1, 1, 0,5, 1,0). Pod bílým světlem by vypadal žlutě. Pokud by na něj dopadalo světlo vyzařující difuzní barvu RGBA(0 1 0 1), vypadal by zeleně.

Ambient - Ambientní osvětlení je barva objektu, když na něj nesvítí žádné světlo. Je kolem objektu zcela rovnoměrné. Ambientní světlo má aproximovat osvětlení, které se odrazilo už tolikrát, že je těžké určit, odkud pochází.

Diffuse - Jedná se o barvu objektu pod čistě bílým světlem. Vypočítává se pomocí směru světla a normály povrchu, na který světlo dopadá. Část objektu s normálou protilehlou ke zdroji světla bude z této složky nejjasnější.

Složka *Specular* je barva a intenzita světla ze zrcadlového odrazu Vyšší hodnoty způsobují, že objekt vypadá lesklejší. Leštěný kovový povrch bude mít velmi velkou hodnotu spekulární složky, zatímco kus papíru nebude mít téměř žádnou.[17]



Obrázek 3.9 Výsledný model skříně zobrazený v Gazebu

V tuto chvíli máme model hotový a je možné jej vložit do našeho Gazebo světa.

3.4 Tvorba Gazebo světa

V této kapitole si popíšeme tvorbu samotného balíčku Gazebo světa, který jsem nazval *map_model*. Jelikož jednotlivé simulační světy často obsahují vyšší desítky modelů, nebylo by praktické vkládat jednotlivé modely postupně do simulačního prostředí. Pro načtení Gazebo světa s modely byl z tohoto důvodu vytvořen balíček, který nahraje veškeré modely naráz s určitými pozicemi. Pro správnou funkci balíčku musíme dodržet určitou strukturu.

```
├── models/
│   ├── stol_150
│   ├── stol_200
│   └── ..
├── worlds/
│   └── skola.world
├── launch/
│   └── load_world_into_gazebo.launch.py
├── package.xml
└── CMakeLists.txt
```

Součástí struktury balíčku jsou tři složky. Položky *package.xml*, *CMakeLists.txt* a *launch file* jsou již probrány u tvorby robotu. Složka *models* obsahuje podsložky, ve kterých se vyskytují jednotlivé modely vytvořené v předchozích kapitolách. Veškeré důležité informace jsou uvedeny v souboru s příponou *world*.

3.4.1 World file

Pro vytvoření Gazebo světa je potřeba dané prvky světa specifikovat v souboru ve formátu SDF.

3.4.2 Základní definování souboru

Nejdříve v souboru definujeme, že se jedná o SDF soubor, poté elementem *world* určíme, že jde o soubor specifikující celý simulační svět.

```
<sdf version='1.5'>
  <world name='default'>
```

3.4.3 Nastavení fyzikálních vlastností

```
  <physics type='ode'>
    <gravity>0 0 -9.8</gravity>
  </physics>
```

Pro specifikaci fyzikálních vlastností je použit element *physics* a jeho subelement *gravity*, který udává sílu a směr působení gravitační síly simulačního prostředí.

3.4.4 Osvětlení světa

Pro specifikování osvětlení světa se využívá element *light*, který obsahuje atributy *name*, což je název našeho světelného zdroje a *type*, který určuje typ světla. Na výběr je ze tří typů a to bodové, směrové a spotlightové. V našem případě bylo vybráno světlo směrové.

```
<light name='sun' type='directional'>
  <pose>0 0 3 0 0 0</pose>
  <direction>0.1 0.1 -0.9</direction>
</light>
```

Element *direction* udává vektor, ve kterém světelný zdroj září a element *pose* udává jeho pozici vůči počátečnímu bodu světa.

3.4.5 Vkládání modelů do světa

Pro funkčnost vložení jednotlivých modelů do světa je nutností mít vkládaný model uložený ve složce *models*, ke které má balíček přístup. V jiném případě se při načtení světa nedostupný model nezobrazí.

```
<include>
  <uri>model://stul_150</uri>
  <name>stul_150_SE_1_112_1</name>
  <pose>-1.2 -1 0 0 0 1.57</pose>
</include>
```

Model vložíme pomocí elementu *include*, ve kterém specifikujeme adresu vkládaného modelu, posunutí a natočení oproti počátečnímu bodu světa a v neposlední řadě si jej pojmenujeme. Pojmenovat si jej lze dle vaší libosti, pro opakované vložení jednoho modelu do stejného světa je zapotřebí zadat vždy odlišné jméno.

3.4.6 Možnost tvoření světa v simulačním prostředí Gazebo

Při sestavování modelu světa psaním textu bylo velmi často obtížné správně určit požadovanou pozici jednotlivých modelů. Jednalo se o velmi zdlouhavý proces, jelikož bylo nutné vždy simulaci zapnout, zkontrolovat pozice přidávaných modelů a při potřebě modelu změnit pozici bylo nutné přespát soubor *world* a poté simulaci načíst znova. Naštěstí simulační prostředí Gazebo disponuje možností do již existujícího světa přidávat

nové modely a přemísťovat modely, které jsou již součástí světa. Postup je jednoduchý. Spustíme náš již vytvořený balíček konzolovými příkazy k sestavení našeho balíčku.

```
colcon build
. install/setup.bash
```

A následným spuštěním našeho balíčku.

```
ros2 launch map_model load_world_into_gazebo.launch.py
```

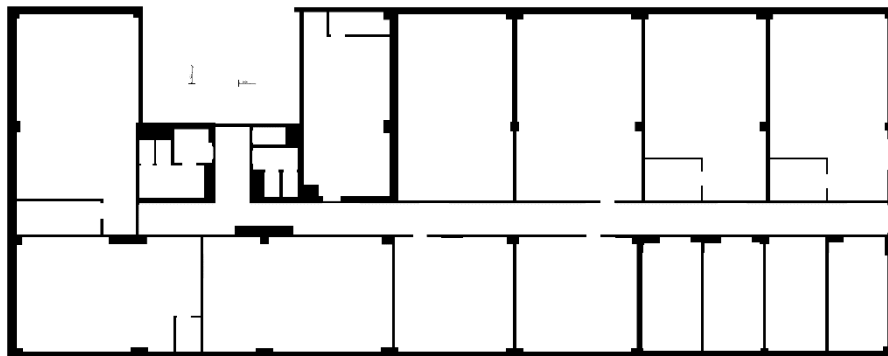
Po načtení našeho balíčku rozklikneme nabídku *Insert*, ve které si vybereme model, který bychom chtěli vložit a poté na něj klikneme. Poté jej dalším kliknutím vložíme do prostředí. Následně vložený model můžeme model přesunout na námi požadované místo. Tento proces můžeme opakovat, dokud nejsme spokojeni s vizuální podobou našeho světa. Daný svět můžeme možností *File -> Save World As* uložit do našeho souboru *skola.world*. Nevýhodou tohoto principu je nepřehlednost a délka výsledného souboru, jelikož Gazebo vygeneruje soubor s elementy, které nejsou zapotřebí k chodu simulace a jsou při nevypsání v souboru v základním nastavení.

3.5 Mřížka obsazenosti

Mapa obsazenosti neboli occupancy grid je 2D mapa pravděpodobnostního výskytu překážky. Nejčastěji je výsledkem sběru dat z laserového scanneru robota. Je uložena v souborech pgm, což je soubor obrázku, který obsahuje stupně šedi v rastru. Každý pixel v mřížce je reprezentován jedním nebo dvěma bajty. Může dosahovat 256 odstínů šedi, kde 0 reprezentuje pixel bez překážky, čím vyšší odstín šedi, tím je pravděpodobnost obsazení pixelu překážkou větší. Neznámý, či neprozkoumaný pixel má hodnotu -1, je tedy průhledný. Robot má tuto mapu v sobě uloženou, například z předchozího průjezdu, a je díky tomu schopný si trasu dopředu naplánovat. Tvorba těchto map je možná dvěma způsoby. [12]

3.5.1 Výroba z půdorysu budovy

Prvním způsobem je možnost vytvoření z obrázku půdorysu. Obrázek je vhodné před konvertováním zvýraznit stěny a zbavit se případných kót a šumu.



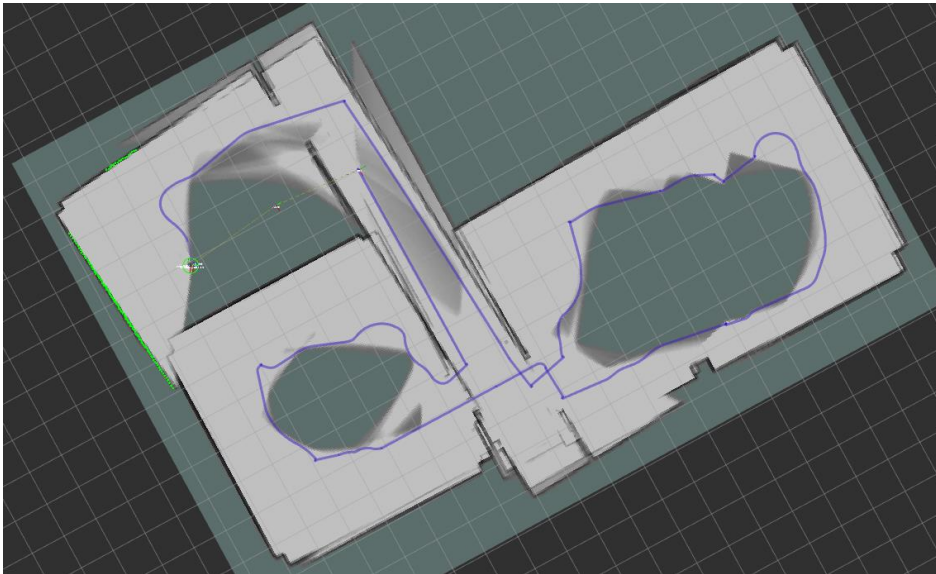
Obrázek 3.10 Půdorys budovy připravený na konverzi formátu

K výslednému obrázku je nutné přiložit specifikační soubor YAML, ve kterém lze určit rozlišení a určení hranic pro obsazený a volný pixel. [13]

3.5.2 Kolekce dat ze senzorů robotů

V této kapitole bylo vyzkoušeno vytvoření mřížky obsazenosti pomocí balíčku ROS 2 Turtlebot 3 Map Explorer dostupného zde [14].

Mapa je tvořena pomocí balíčku SLAM, který pomocí informací z laserového scanneru a polohy tvoří 2D mapu v kombinaci s balíčkem Google Cartographer. Navigace byla prováděna pomocí balíčku Navigation2.



Obrázek 3.11 Nasnímaná mřížka obsazenosti

Do úložiště balíčku byl přidán svět, který byl v předchozích krocích vymodelován. Poté byl balíček spuštěn. Balíček fungoval velmi dobře do chvíle, kdy po určitém čase začalo docházet k deformaci obrazu mapy. Balíček byl několikrát spuštěn znovu se stejným výsledkem.

4. VÝSLEDNÉ BALÍČKY

Výsledkem této bakalářské práce jsou dva balíčky. První balíček obsahuje 3 vytvořené roboty pro použití ve vizualizačním prostředí Rviz a simulačním prostředí Gazebo. Druhý balíček obsahující 3D svět pro simulaci v Gazebo. Výsledné balíčky jsou volně dostupné v mém osobním GitHub repozitáři na adrese <https://github.com/xparfi00>.

4.1 Balíček s roboty

Balíček s názvem *robot_models* obsahuje tři modely robotů, vytvořených na základě reálných robotů ÚAMT. Konkrétně se jedná o roboty s názvy Orpheus X4, Orpheus AC a Loki. Balíček vždy načte jednoho robota. Typ robota který chceme načíst vybereme aktivací daného řádku, ve spouštěcím souboru *display.launch.py*.

```
def
generate_launch_description():pkg_share=launch_ros.substitutions.FindPackageS
hare(package='robot_models').find('robot_models')

#default_model_path=os.path.join(pkg_share,'urdf/loki/loki_description.urdf')

#default_model_path=os.path.join(pkg_share,'urdf/orpheus_x4/orpheus_x4_descri
ption.urdf')

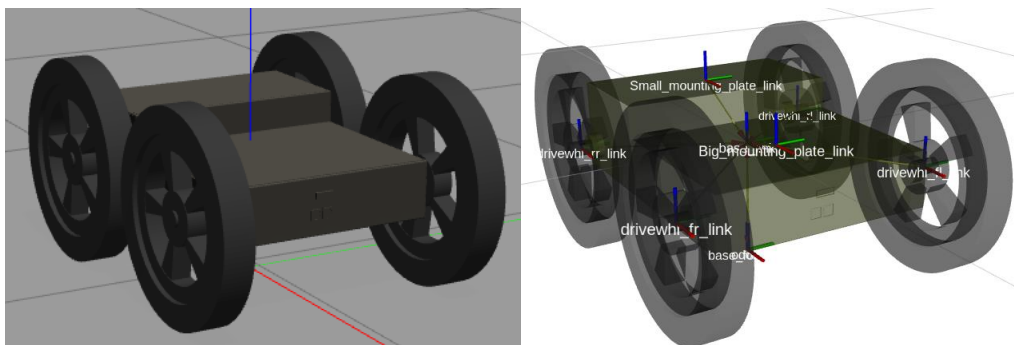
default_model_path=os.path.join(pkg_share,'urdf/orpheus_ac/orpheus_ac_descrip
tion.urdf')
```

Pro spuštění balíčku musíme do konzole vypsát příkaz k sestavení balíčku

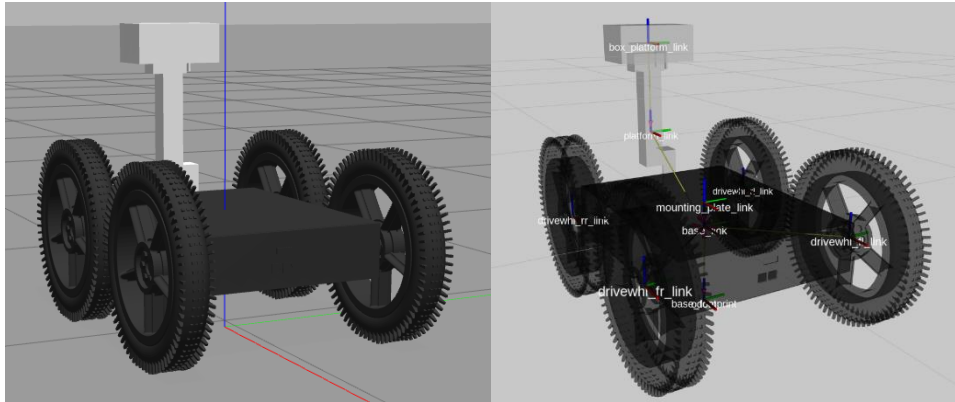
```
colcon build
. install/setup.bash
```

V tuto chvíli nám již nic nebrání k zobrazení našeho robota. Zobrazení provedeme příkazem:

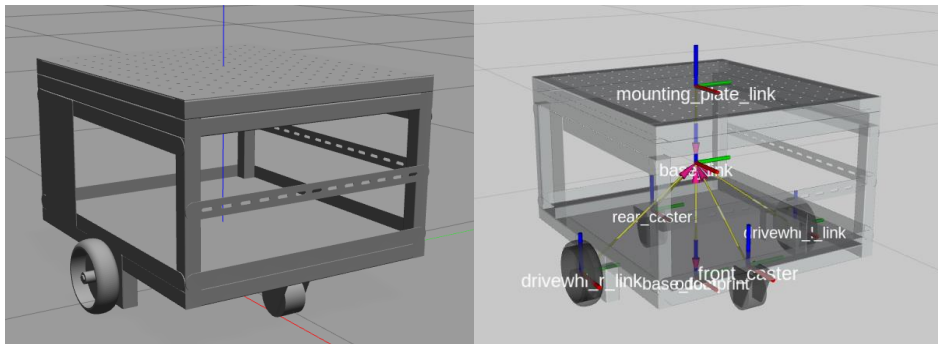
```
ros2 launch robot_models display.launch.py
```



Obrázek 4.1 Výsledný robot Orpheus AC v prostředí Gazeba (vpravo) a Rvizu (vlevo)



Obrázek 4.2 Výsledný robot Orpheus X4 v prostředí Gazeba (vpravo)



Obrázek 4.3 Výsledný robot Loki v prostředí Gazeba (vpravo)

4.2 Balíček 3D světa

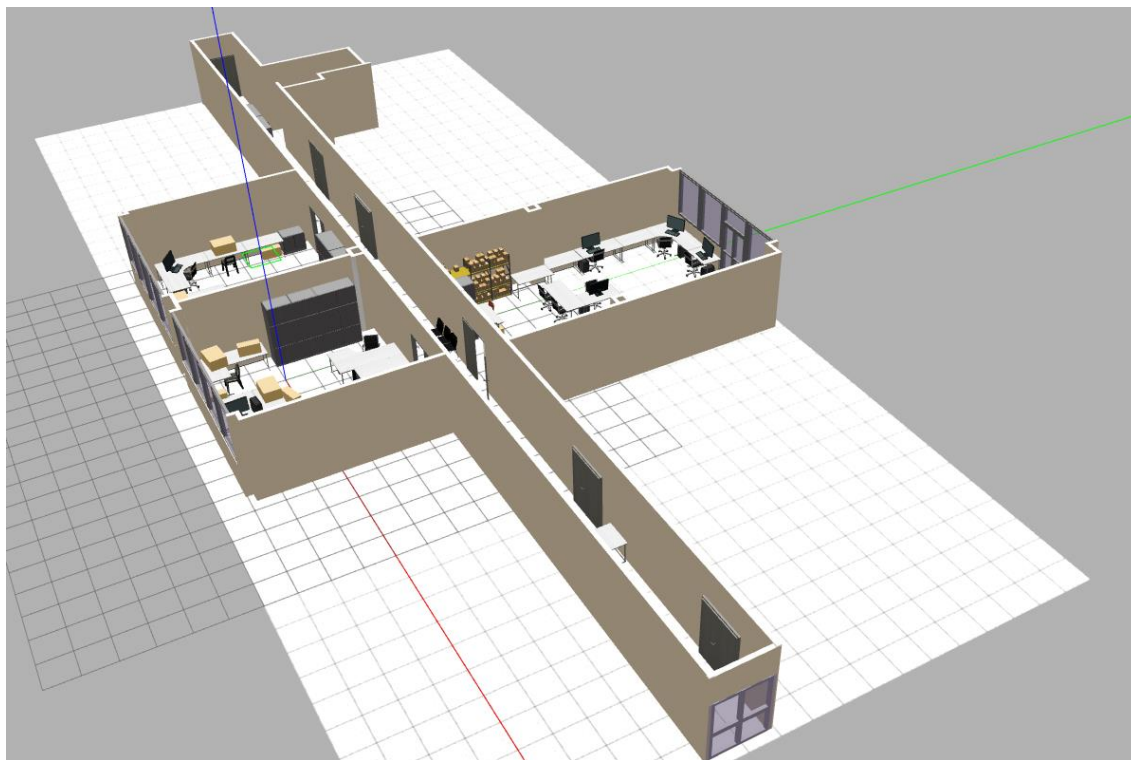
Tento balíček s názvem *map_model* obsahuje trojrozměrný svět určený pro účely simulace v simulačním prostředí Gazebo vytvořený na základě předlohy pracoviště ÚAMT. Tento svět obsahuje modely nábytku a doplňků vytvořených na základě předlohy reálného vybavení daného pracoviště. Svět byl zpestřen modely volně dostupných z knihovny Gazeba.[18] Tyto modely mají uvedeného autora v souboru *model.config*.

Pro spuštění balíčku do konzole vypíšeme příkaz k sestavení balíčku

```
colcon build
. install/setup.bash
```

Spuštění provedeme příkazem:

```
ros2 launch map_model load_world_into_gazebo.launch.py
```



Obrázek 4.4 Výsledný model prostředí na základě předlohy pracoviště ÚAMT

Při tvoření tohoto světa bylo nutné dbát na kompromis náročností na výpočetní techniku a detailností jednotlivých modelů, dále jejich počtem.



Obrázek 4.5 Detailní pohled na model prostředí 1



Obrázek 4.6 Detailní pohled na výsledný model prostředí 2

5. ZÁVĚR

Tato práce byla zaměřena na seznámení se s frameworkem ROS2, vizuálním prostředím RViz a simulačním prostředím Gazebo. Dále na vytvoření robota, 3D světa a 2D Mřížku obsazenosti.

V kapitole 2 byl podrobně popsán postup tvorby robota do vizuálního prostředí RViz a simulačního prostředí Gazebo. Byl úspěšně sestaven balíček s názvem *robot_models*, obsahující tři modely robotů, vytvořených na základě reálných robotů ÚAMT. Konkrétně se jedná o roboty s názvy Orpheus X4, Orpheus AC a Loki. U těchto modelů bylo nutné dojít ke kompromisu, jelikož při více detailnějších modelech jednotlivých částí robota docházelo k neplynulému chodu simulace z důvodu vysokého nároku na výpočetní techniku.

Dalším úkolem bylo vytvoření 3D světa pro účel simulace na základě předlohy pracoviště ÚAMT. V tomto bodě jsme si popsali postupy tvoření modelů třemi způsoby, kde byl popsán celý postup tvoření vlastního modelu. Z těchto vytvořených modelů byl dále vytvořen balíček světa pro simulaci v simulačním prostředí Gazebo. I v tomto bodě bylo nutné myslet na kompromis mezi kvalitou jednotlivých modelů a výslednou zátěží pro simulační prostředí.

Poté bylo nutné vytvořit 2D mřížku obsazenosti, což bylo provedeno za pomoci konverze obrázku na typ *pgm* a přiřazením jednoduchého specifikačního souboru *YAML*. Byl také vyzkoušen volně přístupný mapovací balíček, pomocí kterého bylo možné nasnímat 3D mapu do 2D mřížky obsazenosti. Bohužel po určité době běhu simulace došlo k zasekávání, čímž se rozhodil transformační rámec a došlo k posunutí obrazu a výsledná 2D mřížka obsazenosti byla nepoužitelná. Výsledná mřížka obsazenosti je tvořena pouze z obvodových zdí budovy. Pro správnou funkci je vhodné doplnit tuto mřížku o překážky vyskytující se v budově, jakožto nábytek a další vybavení.

Jako rozvinutí této práce je možné zauvažovat nad modelováním detailnějšího robota pomocí *mesh* souborů, vytvoření detailnějšího 3D světa s nábytkem, či psáním skriptů.

Během tvoření těchto balíčků jsem úzce spolupracoval s vedoucím mé bakalářské práce. Výsledné balíčky jsou dostupné v mém veřejně přístupném GitHub repozitáři dostupném na adrese <https://github.com/xparfi00>.

Literatura

- [1] *ROS 2 Documentation. Ros 2 Documentation: Foxy* [online]. 2022 [cit. 2022-01-02]. Dostupné z: <https://docs.ros.org/en/foxy/index.html>
- [2] *What is the Difference Between rviz and Gazebo? Automatic Addison* [online]. 2020 [cit. 2022-01-03]. Dostupné z: <https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/>
- [3] *Orpheus X4. Laboratory of telepresence and robotics* [online]. 2021 [cit. 2022-01-03]. Dostupné z: <http://www.orpheus-project.cz/orpheus-x/>
- [4] *Setting Up The URDF. Navigation 2* [online]. 2020 [cit. 2022-01-03]. Dostupné z: https://navigation.ros.org/setup_guides/urdf/setup_urdf.html
- [5] *Tf2. ROS Index* [online]. 2021 [cit. 2022-01-03]. Dostupné z: <https://index.ros.org/p/tf2/>
- [6] *Urdf/XML/joint. ROS Wiki* [online]. [cit. 2022-01-02]. Dostupné z: <http://wiki.ros.org/urdf/XML/joint>
- [7] *Adding Physical and Collision Properties to a URDF Model. ROS Wiki* [online]. [cit. 2022-01-02]. Dostupné z: <http://wiki.ros.org/urdf/Tutorials/Adding%20Physical%20and%20Collision%20Properties%20to%20a%20URDF%20Model>
- [8] *List of moments of inertia. Wikipedia* [online]. [cit. 2022-01-02]. Dostupné z: https://en.wikipedia.org/wiki/List_of_moments_of_inertia#List_of_3D_inertia_tensors
- [9] *Gazebo plugins in ROS. Gazebosim* [online]. 2014 [cit. 2023-05-03]. Dostupné z: http://gazebosim.org/tutorials?tut=ros_gzplugins
- [10] *Adding Physical and Collision Properties to a URDF Model. Github* [online]. 2021 [cit. 2022-01-03]. Dostupné z: https://github.com/ros-planning/navigation2_tutorials/tree/master/sam_bot_description/launch
- [11] *SDF vs URDF what should one use? GAZEBO ANSWERS* [online]. 2016 [cit. 2022-01-03]. Dostupné z: <https://answers.gazebosim.org/question/62/sdf-vs-urdf-what-should-one-use/>
- [12] *Occupancy Grid Mapping with Webots and ROS2. Towards data science* [online]. 2021 [cit. 2023-04-03]. Dostupné z: <https://towardsdatascience.com/occupancy-grid-mapping-with-webots-and-ros2-a6162a644fab>
- [13] *Map_server. ROS Wiki* [online]. 2020 [cit. 2022-01-03]. Dostupné z: http://wiki.ros.org/map_server
- [14] *Ros2_explorer. Github* [online]. 2021 [cit. 2023-04-27]. Dostupné z: https://github.com/DaniGarciaLopez/ros2_explorer
- [15] *Color And Texture Models. GAZEBO* [online]. 2020 [cit. 2023-05-05]. Dostupné z: https://classic.gazebosim.org/tutorials?tut=color_model&cat=
- [16] *SDFFormat. SDFFormat* [online]. 2020 [cit. 2023-05-05]. Dostupné z: <http://sdformat.org/>
- [17] *Materials. OGRE Wiki* [online]. VinculumOne, 2012 [cit. 2023-05-05]. Dostupné z: <https://wiki.ogre3d.org/Materials>

[18] *Models. GAZEBO* [online]. 2022 [cit. 2023-05-05]. Dostupné z:
<https://app.gazebosim.org/fuel/models>

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

ROS2	Robot Operating System 2. Software pro vytváření robota
tf2	Transform library generation 2. Druhá generace transformační knihovny
XML	Extensible Markup Language. Obecný značkovací jazyk
Xacro	XML Macros. XML Makra
URDF	Unified Robot Description. Jednotný formát popisu robota
SDF	Simulation Description Format. Simulační popisný formát
YAML	YAML Ain't Markup Language. Formát pro sterilizaci strukturovaných dat
STL	Standard Triangle Language format
DAE	Digital Asset Exchange file

Symboly:

m	váha	(kg)
r	poloměr	(m)
h	výška	(m)
w	šířka	(m)
d	hloubka	(m)