Využití simulátoru Webots ve výuce

Martin Cypris

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Cypris**
Osobní číslo:	**A19011**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Využití simulátoru Webots ve výuce**
Téma práce anglicky:	**Using the Webots Simulator for Teaching Robotics**

Zásady pro vypracování

1. Prostudujte robotický simulátor Webots. Zaměřte se na způsob tvorby 3D modelů robotů, jejich pracovního prostředí a na implementaci kódu definujícího jejich chování.
2. Najděte nebo vytvořte 3D model robota a jeho prostředí pro některou z disciplín v robotických soutěžích.
3. Vytvořte návod pro tvorbu 3D modelů robotů, který bude dostatečně podrobný, ve stylu „krok za krokem", aby byl použitelný ve výuce kurzů a kroužků robotiky.
4. Implementujte několik ukázek řídicího kódu, definujícího chování robota a celé simulace.
5. Zdrojové kódy ukázek podrobně komentujte a doplňte vysvětlujícím textem, aby výsledek byl použitelný ve výuce nebo v budoucích kvalifikačních pracích.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. KIRTAS, M., K. TSAMPAZIS, N. PASSALIS a A. TEFAS. Deepbots: A Webots-Based Deep Reinforcement Learning Framework for Robotics. In: Ilias MAGLOGIANNIS, Lazaros ILIADIS a Elias PIMENIDIS, ed. Artificial Intelligence Applications and Innovations [online]. Cham: Springer International Publishing, 2020, s. 64–75. IFIP Advances in Information and Communication Technology. ISBN 978-3-030-49186-4.
2. MORDENTI, Andrea. Programming Robots with an Agent-Oriented BDI-based Architecture: Explorations using the JaCa and WeBots platforms. S.l.: LAP LAMBERT Academic Publishing, 2013. ISBN 978-3-659-33303-3.
3. STARY, Vadim a Lukas GACHO. Modelling and Simulation of Missile Guidance in WEBOTS Simulator Environment. In: 2020 19th International Conference on Mechatronics – Mechatronika (ME): 2020 19th International Conference on Mechatronics – Mechatronika (ME) [online]. Prague, Czech Republic: IEEE, 2020, s. 1–5 [vid. 2021-12-03]. ISBN 978--1-72815-602-6.
4. CARBONELL, Vanessa Cruz a Ricardo Andrés Castillo ESTEPA. Simulation of a Quadrupedal Bioinspired Modular Robot Using Webots. International Review on Modelling and Simulations (IREMOS) [online]. 2019, 12(2), 94–102. ISSN 2533-1701.
5. PACHECO, Julio a Francesc BENITO. Development of a Webots Simulator for the Lauron IV Robot. In: Proceedings of the 2005 conference on Artificial Intelligence Research and Development. NLD: IOS Press, 2005, s. 347–354. ISBN 978-1-58603-560-0.
6. MICHEL, Olivier. Webots: Symbiosis Between Virtual and Real Mobile Robots. In: Jean-Claude HEUDIN, ed. Virtual Worlds [online]. Berlin, Heidelberg: Springer, 1998, s. 254–263. Lecture Notes in Computer Science. ISBN 978-3-540--68686-6.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Oponent bakalářské práce: **Ing. Jiří Zátopek**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **22. července 2022**
Termín odevzdání bakalářské práce: **19. srpna 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 25. července 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;

beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Martin Cypris v.r.
podpis studenta

ABSTRAKT

Cílem této bakalářské práce je vytvoření pracovního prostředí, robotů a několika ukázek řídicího kódu pro některou z disciplín v robotických soutěžích v simulátoru Webots. Tyto ukázky budou dále využívány v kroužcích robotiky na FAI UTB ve Zlíně. V teoretické části práce je popsán samotný simulátor Webots. Praktická část obsahuje návod na vytvoření vlastního robota přímo v simulátoru Webots a popis vytvoření pracovního prostředí a řídicích kódů.

Klíčová slova: Webots, robotická soutěž, sledování čáry, robosumo, C

ABSTRACT

The aim of this bachelor thesis is to create simulation of a robotic competition including several examples of control code for some of the disciplines using the Webots simulator. This examples will be used in robotics clubs held at FAI TBU in Zlin. The theoretical part of the thesis describes the Webots simulator itself. The practical part contains instructions for creating a robot in the Webots simulator and a description of creating its work environment and control code.

Keywords: Webots, robotics competition, line following, robosumo, C

Rád bych poděkoval vedoucímu této bakalářské práce doktoru Dulíkovi za vedení práce. Dále bych chtěl poděkovat všem učitelům na FAI UTB, kteří mě učili a vedli ke znalostem díky kterým jsem byl schopen se dostat ve studiu tak daleko. Taktéž bych chtěl poděkovat mé rodině za jejich věčnou trpělivost, kterou se mnou měli.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **9**
I TEORETICKÁ ČÁST **11**
1 SIMULÁTOR WEBOTS **12**
1.1 HISTORIE WEBOTS 12
1.2 MOŽNOSTI 13
1.3 VYUŽITÍ 14
1.4 NASTAVENÍ FYZIKY 16
1.5 PRÁCE S PROGRAMEM 18
1.5.1 Hlavní nabídka 19
1.5.2 Hlavní lišta ikonek 21
1.5.3 Vedlejší lišta ikonek 22
1.5.4 Panel objektů 22
1.5.5 Implementace kódu 23
1.6 PID REGULÁTORY 26
1.6.1 Proporcionální část 27
1.6.2 Integrační část 27
1.6.3 Derivační část 27
II PRAKTICKÁ ČÁST **28**
2 FYZIKA **29**
2.1 VYTVOŘENÍ PROSTŘEDÍ 29
2.2 GRAVITACE 31
2.3 TŘENÍ 32
3 LINE FOLLOWER **34**
3.1 ZÁKLADNÍ PROSTŘEDÍ 34
3.1.1 Čára 35
3.2 NÁVOD NA TVORBU ROBOTU 36
3.2.1 Základní tvar 37
3.2.2 Kola 40
3.2.3 Přední opora 42
3.2.4 Senzory 43
3.3 ŘÍDICÍ ALGORITMY 44
3.3.1 Jednoduché algoritmy 44
3.3.2 Složitější algoritmus 45
4 ROBOSUMO **46**
4.1 ZÁKLADNÍ PROSTŘEDÍ 46
4.2 ROBOT 47
4.3 ŘÍDICÍ ALGORITMY 48

ZÁVĚR **50**
SEZNAM POUŽITÉ LITERATURY **52**
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK **53**
SEZNAM OBRÁZKŮ **54**
SEZNAM PŘÍLOH **55**

ÚVOD

V dnešním moderním světě roste nejen zájem, ale i potřeba ovládat informační technologie. Tyto technologie nás obklopují na každém kroku. To je důvodem, proč se učit informační technologie ovládat už od útlého dětství. Dobrým základem jsou již základní školy, na kterých by se děti mohly učit základům programování. Programováním se člověk dokáže sblížit s danými technologiemi a zároveň pochopit i to, jak fungují. Obzvláště u dětí je potřeba, aby je učení bavilo. Programování robotů by mohlo být způsobem, jak zajistit, aby se děti pro programování nadchly. Programování jednoduchých robotů má tu výhodu, že stačí napsat pár řádků pro uvedení robotu do pohybu. Děti tím pádem vidí výsledky své práce na něčem pohyblivém. Určitě je to pro ně mnohem zábavnější než psaní suchého kódu.

Pro školy bude mnohem jednodušší využít jeden z dostupných simulátorů. Nemusely by v tom případě pořizovat roboty fyzicky a případně zajišťovat celou novou učebnu. Jedním z dostupných simulátorů je Webots, který je zdarma a nevyžaduje registraci. Díky tomu je možné ho bezproblémově využít pro již zmíněné kroužky robotiky. Simulátor má další výhodu v tom, že si ho děti mohou naistalovat do svého počítače. To umožňuje, aby si programování robotů zkoušely i mimo školní prostředí. Celkově se tento simulátor dá využít nejen pro učení naprostých základů programování, ale i zkoušení různých pokročilých kódů. Zejména u složitějších kódů je výhodou možnost vývoje ve virtuálním prostředí, které umožňuje nastavit prostředí robota a počáteční podmínky tak, aby se určitý aspekt činnosti robota dal opakovaně a co nejrychleji odladit.

Simulátor Webots nabízí spoustu možností od programování jednoduchých robotů až po simulování autonomního řízení vozidel. Zároveň obsahuje velké množství příkladů a ukázkových kódů, velký počet před-vytvořených pracovních prostředí a spoustu modelů existujících robotů. Příkladem můžou být například roboti e-puck, různá robotická ramena, humanoidní roboty nebo dokonce model robotického vozítka využívaného NASA při průzkumu Marsu. Webots také umožňuje programovat v několika programovacích jazycích, takže uživatelé mají možnost výběru. Díky všem těmto vlastnostem je možné ve Webots nasimulovat téměř vše, což je obrovským plusem.

Má práce se věnuje v teoretické části samotnému simulátoru Webots a jeho využití ve výuce. Dále v praktické části práce obsahuje návod na vytvoření jednoduchého robotu, ke tvorbě a úpravě pracovního prostředí simulátoru. Jejím obsahem je také popis vytvořených

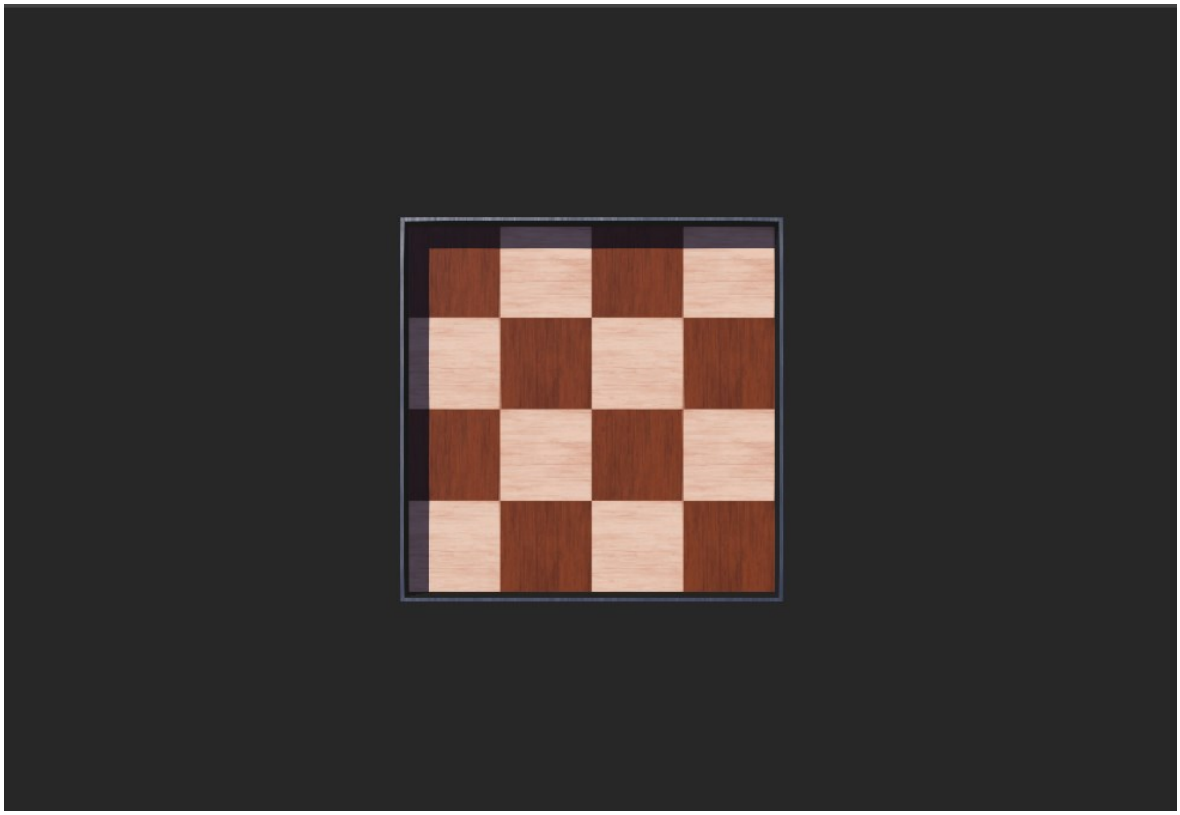
ukázek řídicího kódu. Kód je psán v jazyce C, protože je to jednoduchý jazyk, který je základem pro mnoho jiných programovacích jazyků. Jazyk C je velmi účinný, protože má velice blízko k hardwarové úrovni.

I. TEORETICKÁ ČÁST

1 SIMULÁTOR WEBOTS

Webots je program, který se dá jednoduše popsat jako simulátor robotů. Simulátor obsahuje řadu modelů robotů, které lze využít pro své projekty. Obsahuje možnost si vytvořit vlastního robotu v prostředí Webots nebo nahrát 3D model robotu vytvořený v nějakém jiném programu.

Osahuje možnost upravovat prostředí, ve kterém bude vybraný robot či roboty pracovat. Do Webots lze nahrát prostředí vytvořené i v jiném programu. Zároveň obsahuje i vlastní jednoduchý editor zdrojových kódů. Samotný simulátor Webots je freeware, který je volně stažitelný bez nutnosti registrace.



Obrázek 1 Nový projekt ve Webots

1.1 Historie Webots

Simulátor Webots byl původně vyvíjen jako nástroj, jehož účelem bylo zkoumání různých řídicích algoritmů v mobilní robotice [1]. Vývoj tohoto nástroje započal Dr. Olivier Michel, který pracoval pro EPFL v Lausanne ve Švýcarsku. V roce 1998 se z nástroje stal komerční produkt, který byl prodán na více jak 400 univerzit a výzkumných center po celém světě [2].

V prosinci roku 2018 byl simulátor Webots vydán jako open-source program [1]. Tím bylo umožněno, aby tento program využil kdokoli, kdo má zájem s ním pracovat.

1.2 Možnosti

Simulátor Webots obsahuje mnoho možností, které využijí i nároční uživatelé. Od možnosti si kompletně upravit a vytvořit vlastní pracovní prostředí, přes širokou nabídku již předvytvořených robotů, až po vlastní editor zdrojových kódů, který nabízí programování v několika programovacích jazycích.

Úprava nebo vytvoření vlastního pracovního prostředí pro roboty je v programu řešeno pomocí přidávání a editací jednoduchých tvarů. Pokud uživatel chce, může si do programu importovat vlastní pracovní prostředí. Toto prostředí je možné vytvořit v různých programech například v programu Blender. Do simulátoru je možné přidávat i vlastní objekty. Například pokud se uživatel rozhodne si do prostředí importovat vlastní strom, nebude žádný problém využít již vytvořený. Samotná tvorba prostředí nebo robotů je uživatelsky přívětivé a podobá se již zmiňovanému Blenderu. Takže uživatel má možnost svým vytvořeným objektům měnit parametry obsahem, kterých jsou různé vlastnosti objektu, jako velikost, barvy, textury a tak dále. Například může uživatel objektu přiřadit texturu cihel. Do Webots se dají importovat objekty ve formátu Collada, STL a OBJ.

Tvorba vlastních robotů v programu Webots funguje na podobném principu jako vytváření pracovního prostředí. Primárně je realizována pomocí jednoduchých tvarů, které si uživatel může upravit do konečné podoby, tak jak potřebuje. Samotné součástky, jako motory, senzory, kamery, čidla a tak dále, nemají žádnou vlastní texturu nebo model. Pokud uživatel chce, tak může dané součástce přiřadit zvolený tvar. Například motoru, který otáčí kolem přiřadí tvar válce. Samozřejmě pokud uživatel chce, může si do Webots opět importovat vlastní model robotu. Import vlastního robotu funguje na principu, že si uživatel robotu v simulátoru ve své podstatě složí. V jiném programu si vymodeluje jednotlivé součástky, které následně importuje jako modely pro konkrétní součástky v simulátoru.

Webots taky obsahují vestavěný editor zdrojového kódu. Ten je ale velmi jednoduchý a téměř bez našeptávače. Editor našeptává jen základní programové prostředky jako for, while nebo if. Naopak výhodou je možnost programovat v několika programovacích jazycích. Na výběr Webots umožňují z nabídky jazyky C, C++, Java, Python a Matlab. To

znamená Webots umožňuje výběr z nejpoužívanějších programovacích jazyků pro programování robotů a robotických zařízení [3].

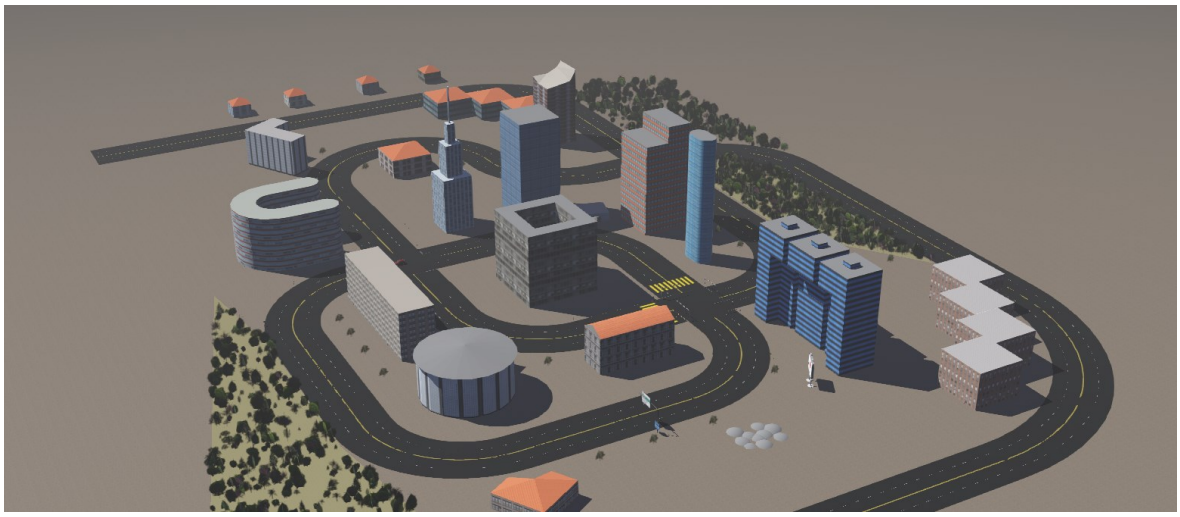
Obrázek 2 Upravené prostředí ve Webots

1.3 Využití

Ačkoli konečným cílem je vždy skutečná robotika, skutečný robot, který na základě zdrojového kódu dělá danou práci, je velmi často užitečné, a i potřebné nejdříve provést potřebné simulace. Simulace je jednodušší připravit, jsou mnohdy rychlejší a pohodlnější, obsahují větší možnost ladění robotu, ale hlavně jsou levnější. Vytvoření nových modelů robotů a nastavení simulace zabere jen několik hodin. Simulovaná robotická sestava je řádově levnější než skutečné roboty a reálné sestavy, což umožňuje lepší zkoumání konstrukce. Simulace často běží rychleji než skutečné roboty. Důvodem je především výpočetní rychlost počítače, na kterém je simulace prováděna. V simulaci veškeré výpočty a vyhodnocování provádí klasický počítač, který je mnohonásobně výkonnější, něž mikrokontroler, který totéž provádí v samotném robotu. Zároveň jsou všechny parametry a hodnoty ze simulace snadno zobrazitelné na obrazovce. Simulace umožňují používat výpočetně náročné algoritmy, které by na reálných robotických mikrokontrolerech potřebovaly obrovské množství času, jako jsou genetické algoritmy. V neposlední řadě jsou výsledky kvalitní simulace přenositelné na skutečné roboty. Bohužel žádná simulace

není dokonalá a například tření v kloubech by se velmi obtížně počítalo. Proto ani samotná přenositelnost není stoprocentní a je vždy třeba provést ladění kódu pro reálný robot. [4]

Obrovský potenciál využití poskytuje hejnová robotika [2], kde používané roboty jsou velice nákladné. Proto je mnohem výhodnější dělat různé simulace právě v nějakém simulátoru, kde uživatel může mít libovolný počet robotů. Prakticky jediným omezením simulátoru je výpočetní výkon. To znamená, že počet robotů v simulaci je limitován prakticky jen výkonem počítače, na kterém se simulace provádí. Po dokončení simulací může být už hotový zdrojový kód použit v praxi na skutečných robotech. Další variantou využití může být například simulace adaptivního chování.



Obrázek 3 Model města pro simulaci autonomních vozidel ve Webots

Samotný simulátor Webots už po stažení obsahuje ukázkový příklad města s autonomními vozidly. Autonomní vozidla jsou dobrým příkladem, kdy je simulace před zkouškami v praxi velmi užitečná. Simulace se provádí z důvodu, aby se předešlo chybám. Pokud nastane chyba v simulátoru, tak se dá simulace po opravě chyb bez problémů provést znovu. V realitě by došlo k poškození vozidla a objektů v jeho blízkosti.

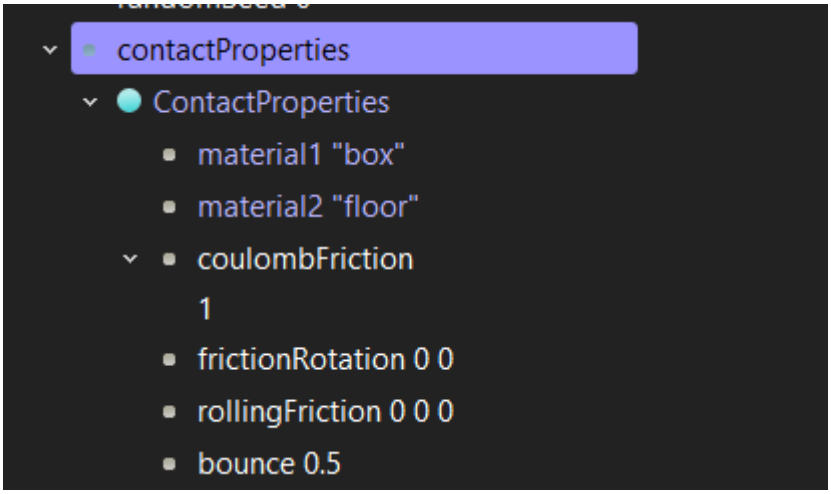
Webots samotný má ovšem velmi velký potenciál využití ve výuce. Využití může být například v kroužcích robotiky pro základní školy, ve výuce programování na středních školách nebo dokonce na vysokých školách. Výhodou jsou opět náklady. Webots je, jak už bylo zmíněno kompletně zdarma. Zároveň se učitelé nemusí bát, že by robot, který by byl žákům nebo studentům zapůjčen, byl jakkoli zničen. Zároveň není vyžadován nějaký zvláštní prostor, kde by se funkčnost robotů testovala. Vše se dá jednoduše vytvořit a nastavit přímo v prostředí Webots. Pro lepší výsledky je dobré použít i jiné programy,

které například mají lepší možnosti pro vytvoření pracovního prostředí. Další výhodou je to, že odpadá nutnost ladění a seřizovaní samotného robotu, který bude využíván.

Využití simulátorů je v tomto technologickém odvětví časté a široké. Konkrétně simulátor Webots obsahuje obrovské množství ukázek a příkladů, které může využít běžný uživatel, ale i zkušený programátor. K simulátoru Webots je dostupná velmi rozsáhlá oficiální dokumentace [1], ve které se nachází popis programu jako takového, popis prostředí programu a jednotlivých oken. Zároveň obsahuje i seznam robotů, které program nabízí, včetně informací o daném robotu. Stejně tak je zpracován i seznam objektů pro pracovní prostředí robotu nebo součástek, které uživatel může do svého robotu použít. Dále se v dokumentaci nachází i textový tutoriál. Samotný tutoriál je rozdělen na několik částí. U každé části je uvedena i její časová náročnost. Většina částí má časovou náročnost maximálně 30 minut. Nejnáročnější části mají 60 minut. Samozřejmě je to odhadovaná délka a reálná časová náročnost bude pro každého jiná. Stejně tak má Webots i poměrně rozsáhlou fanouškovskou základnu. Díky ní je možné najít na internetu množství rad, tipů a triků. Případně se zeptat na různých fórech na nějaký konkrétní problém.

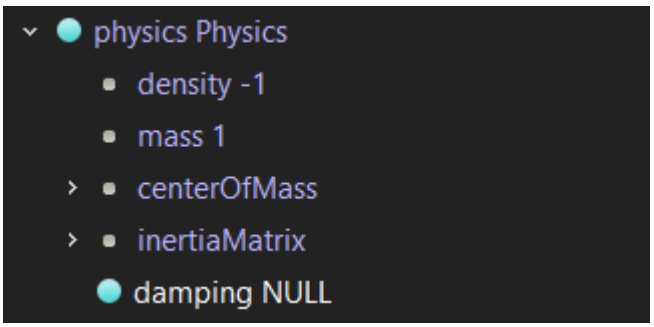
1.4 Nastavení fyziky

V každém projektu Webots je automaticky přítomen objekt/uzel (node) „WorldInfo“. V tomto objektu lze nastavit základní parametry fyziky prostředí. Například hodnota gravitační konstanty je automaticky nastavena na hodnotu 9,81 m/s^2, odpovídající gravitačnímu zrychlení na povrchu Země. Dále je možné v tomto nodu přidat další node „ContactProperties“. V tomto přidaném nodu lze nastavit parametry materiálu, například parametry tření. Těchto nodů může uživatel vytvořit libovolný počet, pro každou dvojici objektů jeden. Samotné nastavení tření uživatel provede pojmenováním jednotlivých materiálů. Následně v nodech zvolených objektů přiřadí parametru „contactMaterial“ příslušný název materiálu. Uživateli je umožněno tření nastavit velice detailně díky různým parametrům, které node „ContactProperties“ nabízí. Déle je zde možné nastavit, jak moc se od sebe budou materiály odrážet. S tím je spjatý parametr minimální rychlosti, která je potřebná, aby se materiály od sebe odrazily.[1]



Obrázek 4 Nastavení nodu ContactProperties

Fyzikální model jednotlivých objektů se nastavuje pomocí nodu „physics“. Tento node umožňuje zadávat parametry pro podsystém simulátoru fyziky. V nodu „physics“ je možné nastavit objektu parametry váhy, těžiště nebo například rozložení váhy. Při nastavování váhy je potřeba myslet na to, že váha se do programu zadává v kilogramech. Dalším zajímavým parametrem je inertiaMatrix. Tento parametr obsahuje dva vektory. Na základě těchto vektorů program dokáže simulovat setrvačnost objektu. Díky tomu je možné, aby fyzikální engine vypočítal reálné síly, které na objekt budou působit. Využití nodu „physics“ je dobrovolné. Uživatel tento node nemusí využít. V případě nevyužití tohoto nodu má objekt kinematické chování. Pokud ale uživatel node „physics“ využije, objekt se chová podle fyzikálního modelu. [1]



Obrázek 5 Nastavení nodu physics

U kloubů robotů je možné nastavit parametr „springConstant“ neboli konstantu pružnosti. Nastavení tohoto parametru může sloužit např. jako simulace tlumičů u kolových robotů. Bohužel simulátor Webots neumožňuje nastavení tohoto parametru na základě předem definovaných materiálů. Uživatel si konstantu musí někde vyhledat. Samotný fyzikální

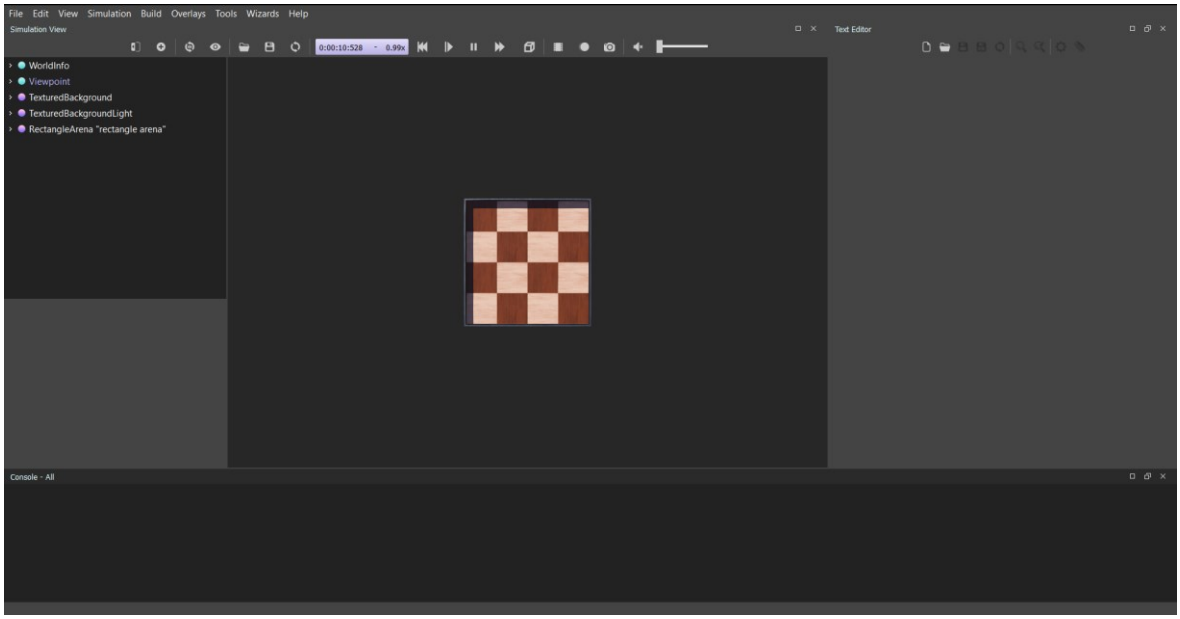
engine je schopný na základě konstanty pružnosti schopný vypočítat sílu pružiny. Tento výpočet provede pomocí Hookova zákona. Klidová poloha pružiny je pak v bodě, kde se pružina nacházela při jejím vytvoření. To znamená, že při tvorbě robotu uživatel určí i klidovou polohu pružiny. [1]

Pokud uživateli nevyhovuje základní fyzikální chování objektů, má možnost si naprogramovat vlastní plugin. Ve Webots lze tento plugin programovat jen v jazyce C a C++. Vlastní fyzikální plugin umožňuje uživateli vytvořit vlastní model chování například na základě aerodynamiky u létajících robotů. To stejné platí pro roboty ve vodě – uživatel si může vytvořit plugin, který bude simulovat chování na základě nějakého modelu hydrodynamiky. Tyto modely program Webots nenabízí standartně a uživatel si je musí sám vytvořit. Pomocí vlastního pluginu si uživatel může naprogramovat i lepší systém detekce kolizí, tření a tak dále. [1]

V této bakalářské práci jsou ukázky základního nastavení simulace fyziky popsány v praktické části – v kapitole 2, začínající na straně 29.

1.5 Práce s programem

Práce se samotným programem je velice jednoduchá a uživatel si ji rychle osvojí. Program ale neobsahuje českou lokalizaci. Proto je nezbytná aspoň základní znalost angličtiny. To by ale dnes mělo být samozřejmostí. Samotnému učení práce s programem napomáhá již zmíněný oficiální tutoriál, který je zpracován v rámci oficiální dokumentace. Pracovní plocha programu je rozdělena na několik částí. V horní části se nachází lišta s hlavní nabídkou, hlavní lištou ikonek, případně i s vedlejší lištou ikonek. Pod touto lištou se ve středu obrazovky zobrazuje panel, ve kterém se zobrazuje samotná simulace. Tedy pracovní prostředí s robotem případně roboty. V pravé části obrazovky se zobrazuje panel pro jednoduchou práci s jednotlivými objekty v simulaci. V levé části obrazovky se nachází textový editor pro úpravu kódu. Ve spodní části je umístěna konzole pro výpis například hlášek o sestavení kódu, chybových hlášek nebo dat, která si uživatel nechá vypsat. Program taky obsahuje možnost, po dvojkliku na robot, zobrazit dodatečný panel. Tento panel se zobrazí v levé části obrazovky. Uživatel zde může vidět data ze senzorů, které se na robotu nachází. Tato data jsou zobrazována v grafu, pokud je to možné.



Obrázek 6 Základní pracovní plocha

1.5.1 Hlavní nabídka

Obrázek 7 Hlavní nabídka programu

Hlavní nabídka se nachází v horní části pracovního okna. Je to hned první vodorovná nabídka pod záhlavím okna. V nabídce se nachází základní záložky pro práci s programem. Po kliknutí na vybranou záložku se zobrazí seznam funkcí, které uživatel může využít.

V nabídce „File“ jsou umístěny základní funkce programu, například vytvoření nového světa (pracovního prostředí) nebo načtení už vytvořeného, uložení světa, otevření textového editoru pro úpravu kódů, importování 3D modelů nebo spuštění nahrávání videa.

Nabídka „Edit“ obsahuje základní funkce, jako je kopírování, vystřižení nebo vložení. Tyto funkce je možné využít pro některý z objektů pracovního prostředí nebo kód v textovém editoru. Nabídka obsahuje i funkci pro vyhledávání slov v kódu, případně jejich nahrazení.

„View“ je nabídka, která obsahuje seznam funkcí pro vykreslování nebo změnu pohledu. Dá se zde přepnou vykreslování na wireframe, takže se vykreslí jen obrysy jednotlivých objektů. V nabídce se nachází i funkce pro přepínání perspektivy. Pro většinu uživatelů nejužitečnější je nabídka „Optional Rendering“. V této nabídce se nachází možnost spustit vykreslování různých doplňků, které mohou usnadnit práci. Například vykreslování

paprsků senzorů, aby bylo možné je lépe a přesněji umístit, osy kloubů, těžiště objektů a tak dále.

Nabídka „Simulation“ obsahuje základní funkce pro ovládání simulace. Spuštění simulace v reálném čase, pozastavení simulace, udělat jeden krok simulace a zrychlení simulace jsou funkce, které se v této nabídce nachází.

V nabídce „Build“ se nachází funkce pro práci s kódem. Například funkce „Build“ pro sestavení kódu nebo funkce „Clean“ pro automatickou vizuální úpravu kódu. Pro uživatele programující v jazyce Java je zde připravená funkce pro vytvoření souboru JAR.

Nabídku „Overlays“ využijí ti, kteří na svůj robot přidají kameru nebo displej. Nabídka obsahuje funkce pro zobrazení nebo schování okna případně oken, které v reálném čase vykreslují, co vidí kamera nebo ukazuje displej umístěný na robotu.

Nabídka „Tools“ nabízí možnosti zobrazit a skrýt základní panely pro práci s programem. Takže je zde možnost skrýt panel textového editoru pro úpravu kódu, nabídku pro práci se samotným prostředím a robotem. V případě potřeby lze skrýt i panel, na kterém se vykresluje samotná simulace. Dále se zde nachází možnost otevřít novou konzoli programu nebo vyčistit stávající. Zároveň se zde nachází položka „Preferences…“, která otevírá dialogové okno pro nastavení programu. Je možné zde nastavit lokalizaci programu, barevný motiv programu nebo kvalitu vykreslování textur.

V nabídce „Wizards“ se nachází jen tři položky. Nachází se zde možnost vytvoření adresáře pro nový projekt, vytvoření nového controlleru, tedy kódu pro řízení robotu, nebo vytvoření nového vlastního pluginu pro fyziku podle, které se roboty budou chovat.

Velice užitečná je nabídka „Help“, kde si uživatel může zobrazit základní informace o programu nebo OpenGL. Je zde možné zobrazit i rady, jak se pohybovat kamerou v 3D prostoru, jak pohybovat jednotlivými objekty, nebo jak aplikovat dodatečnou sílu nebo točivý moment na objekt. Uživatel si zde může ověřit, jestli je jeho verze Webots aktuální.

Nejzajímavější možností je zde „Webots Guided Tour…“, která otevře dialogové okno. V otevřeném dialogovém okně se nachází na výběr dvě nabídky. Nabídka, která se opět nazývá „Webots Guided Tour“ obsahuje asi 20 ukázek různých simulací. Nachází se zde automatické simulace například robotická ramena, která z pásu sbírají plechovky a umisťují je do krabic nebo robot, který dokáže přenést objekty z jednoho místa na druhé. Zá zmínku ještě stojí model města s automobilem, který se vyhýbá překážkám na vozovce. Taky se zde nachází příklady s roboty, které může ovládat uživatel pomocí klávesnice,

například létající dron nebo kolové vozítko pro přepravu nákladu. Všechny simulace jsou plně funkční včetně řídicích algoritmů. Uživatel je může využít pro procvičení svých programovacích schopností nebo jako základ pro svůj projekt.

Druhou možností v dialogovém okně je „Devices Tour“, kde se nachází jednoduché ukázky pro jednotlivá vstupní a výstupní zařízení, která je možné na robota umístit. Pokud se uživatel chce naučit, jak například pracovat s displejem na robotu, může využít funkční ukázku s jednoduchým kódem. To platí i pro kamery, světelný senzor, senzor vzdálenosti, radar, motory, gyroskop a další zařízení.

1.5.2 Hlavní lišta ikonek

Obrázek 8 Hlavní lišta ikonek

Hlavní lišta ikonek se nachází nahoře uprostřed obrazovky pod hlavní nabídkou. V této liště se nachází ikonky nejpoužívanějších funkcí, aby je uživatel nemusel hledat v hlavní nabídce. Usnadňují tedy práci s programem. Ikonky jsou rozděleny do několika sekcí rozdělených svislou čárou.

První ikonka zleva, v první sekci, slouží pro skrytí nabídky pro práci s prostředím a robotem. Pomocí druhé ikonky uživatel otevře dialogové okno, díky kterému může do pracovního prostředí přidávat nové objekty. Další dvě ikonky se nachází v další sekci. Umožňují nastavení pohledu kamery. První vrátí pohled do výchozího nastavení, druhá ikonka otevře nabídku, ve které si uživatel může vybrat pohledy z různých úhlů například pohled shora, z boků nebo ze spodu. První ikonka ve třetí sekci má podobu složky. Ta slouží pro načtení světa (pracovního prostředí). Ikonka diskety naopak slouží pro ukládání světa. Poslední ikonka v této sekci, dvě šipky tvořící kruh, vrátí všechny provedené neuložené změny. V další sekci se nachází panel ukazující čas simulace. Vpravo od něho je zobrazena skupina ikonek pro práci se simulací. První ikonka vrátí simulaci na začátek. Druhá provádí jeden krok simulace. Třetí spouští simulaci v reálném čase a čtvrtá spouští zrychlenou simulaci. Poslední ikonka zapíná nebo vypíná vykreslování simulace. Po vypnutí vykreslování se uživateli objeví černá obrazovka s nápisem „No Rendering“. Další sekce tří ikonek slouží pro zaznamenání simulace. První ikonka zaznamená video simulace. Uživatel si může vybrat kvalitu, ve které výsledné video bude nahráno. Výsledné video se uloží ve formátu mp4. Další ikonka vytvoří webovou stránku, na které se bude

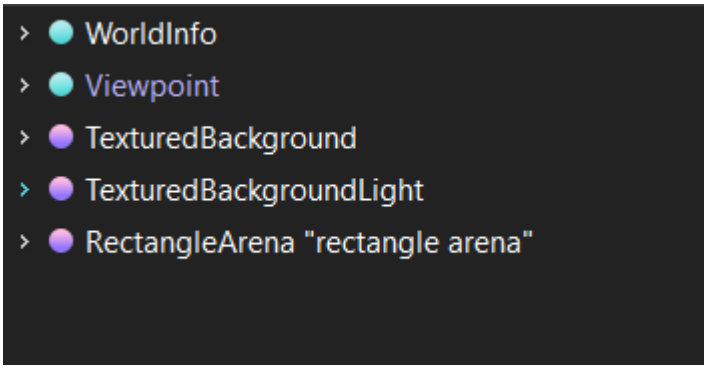
zobrazovat video záznam z dané simulace. Třetí ikonka vytvoří screenshot simulace. Poslední sekce slouží jen pro úpravu hlasitosti dané simulace. Některé roboty mohou mít na sobě součástky, které vydávají zvuk. Například reproduktory nebo i některé motory.

1.5.3 Vedlejší lišta ikonek

Obrázek 9 Vedlejší lišta ikonek

Vedlejší lišta ikonek, pokud je zobrazená, se nachází vlevo od hlavní lišty ikonek. Zobrazuje se pouze tehdy pokud má uživatel zobrazený textový editor pro úpravu kódu. Stejně jako hlavní lišta ikonek je i tato rozdělena na sekce. V první sekci jsou ikonky pro vytvoření nového textového souboru, otevření již existujícího souboru, uložení, uložení jako a vrácení všech neuložených změn. Ve druhé sekci jsou ikonky pro vyhledání slova v otevřeném souboru případně jeho nahrazení. Ve třetí a poslední sekci je ikonka pro sestavení kódu a jeho estetickou úpravu.

1.5.4 Panel objektů



Obrázek 10 Panel objektů

Panel pro práci s objekty simulace se nachází v levé části pracovního plochy. Nachází se v něm jednotlivé uzly neboli nody. Tyto nody obsahují parametry nebo mohou obsahovat další nody. Tvoří tak stromovou strukturu. V základním prostředí se nachází výchozí nody vytvořeny samotným programem. Node „WorldInfo“ obsahuje základní parametry vytvořeného světa (pracovního prostředí). Nachází se zde například nastavení gravitační síly. Výchozí nastavení odpovídá podmínkám reálného světa. Další node „Viewpoint“ zase obsahuje parametry pro nastavení pohledu kamery. Parametry v nodech „TexturedBackgroudnd“ a „TexturedBackgroundLight“ nastavují texturu pozadí a světlo.

„RectangleArea“ je node, který obsahuje parametry popisující základní pracovní prostředí. Jsou zde parametry pro úpravu velikosti plochy prostředí nebo úpravu výšky a šířky stěn. Dále se zde nachází nody, které obsahují parametry pro změnu textury podlahy nebo stěny kolem pracovního prostředí.

Další nody vytváří samotný uživatel. Nody vytvořené uživatelem symbolizují jednotlivé objekty nacházející se v pracovním prostředí. Nachází se zde parametry objektu například pozice (translation), otočení (rotation) nebo název objektu (name). Dále se zde nachází další nody, které obsahují dodatečné parametry. Například node „physics“, který uživatel musí aktivovat. To provede pomocí dvojkliku a v dialogovém okně vybere položku „Physics“. Následně může objektu nastavit váhu, hustotu nebo polohu těžiště. Uživatel má možnost přidat objektu vlastní nody, díky kterým si může přizpůsobit objekt podle sebe. V nodu objektu se nachází položka „children“, do které může uživatel přidávat další nody. Díky těmto přidaným nodům může uživatel například měnit tvar objektu.

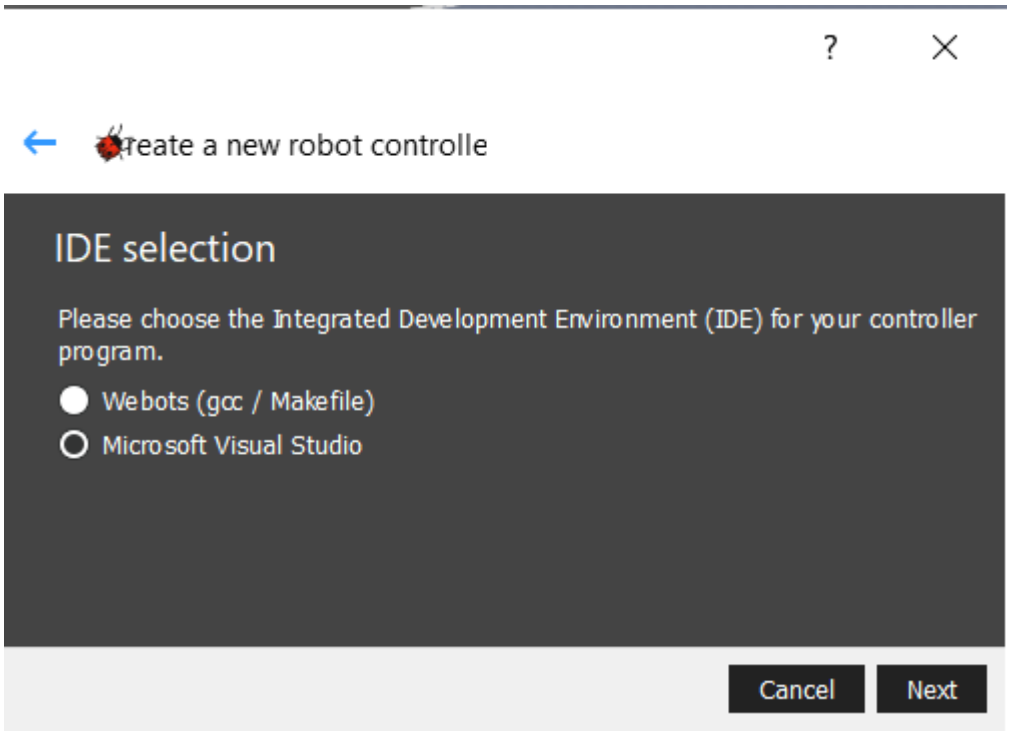
Uživatel nemusí s objekty pracovat jen přes tento panel. Může s nimi pracovat přímo v pracovním prostředí. Samozřejmě to má jen omezené možnosti. Uživatel může přímo v pracovním prostředí měnit polohu objektu, jeho otočení nebo velikost. Tato možnost ale není dostatečně přesná. Navíc úprava velikosti objektu je poměrně špatně ovladatelná.

1.5.5 Implementace kódu

Webots podporuje práci v libovolném vývojovém prostředí. Uživatel tedy nemusí programovat v editoru, který nabízí Webots. Může tak využít své oblíbený vývojové prostředí. Pro práci s Webots musí prostředí správně nastavit. Především je nutné, aby spustitelný soubor EXE měl stejný název jako adresář, ve kterém se nachází. Zároveň je nutné, aby uživatel dodržel předem určenou cestu k souboru. Pokud uživatel tuto cestu nedodrží, Webots soubor neuvidí. Cesta k souboru musí být následující: „(WebotsProjekt)/controllers/(NázevSouboru)/(NázevSouboru).exe“. Texty v závorkách může uživatel nahradit vlastními názvy. Bohužel nelze popsat nastavení všech vývojových prostředí. Z toho důvodu jsem vybral tři oblíbená vývojová prostředí. [1]

První vývojovým prostředím je Visual Studio. Toto vývojové prostředí je možné použít pro psaní kódů pro Webots v jazyce C nebo C++. Samotný simulátor Webots nabízí při tvorbě nového controlleru v C nebo C++ možnost automaticky vytvořit projekt ve Visual Studiu. Uživatel se tak nemusí starat o správný název souboru nebo jeho umístění. Vše je provedeno automaticky. Pokud uživatel nemá ve svém Visual Studiu nainstalované

rozšíření pro práci s jazyky C a C++, bude vyzván k nainstalování rozšíření. Po případné instalaci rozšíření se uživateli otevře projekt ve Visual Studiu. V projektu by mělo být vše automaticky nastaveno a neměl by nastat žádný problém. Je ale možné, že při úplně prvním spuštění bude potřeba Visual Studio restartovat. Následně uživatel může programovat kód. Pro přiřazení algoritmu robotu musí uživatel v nodu „Robot“ u parametru „controller“ vybrat možnost „<extern>“. Následné spuštění kódu uživatel provede přímo ve Visual Studiu. Pro případné ladění je možné využít i debugger ve Visual Studiu. V případě jakéhokoli problému je v oficiální dokumentaci popsáno detailní nastavení Visual Studia. [1]

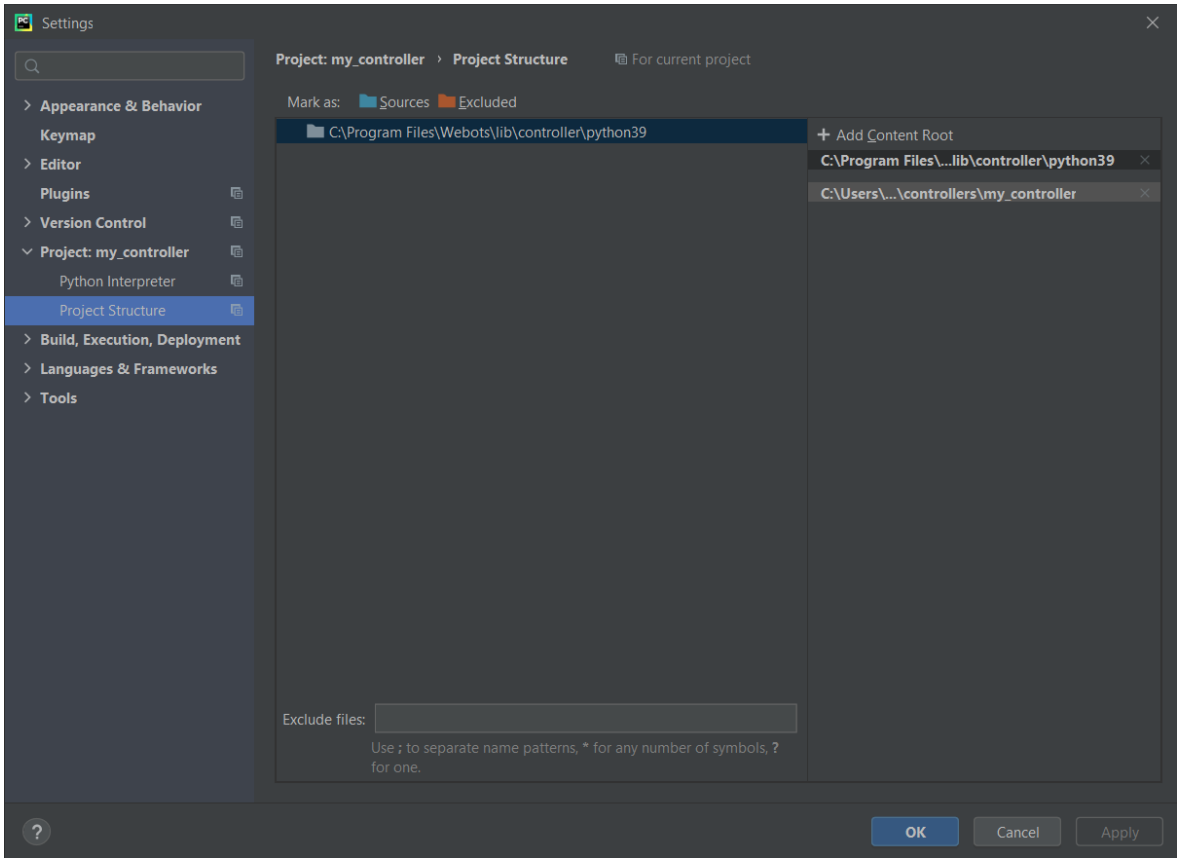


Obrázek 11 Vytvoření controlleru pro Visual Studio

Qt Creator je opět vhodný pro jazyky C a C++. Nejdříve musí uživatel vytvořit projekt v Qt Creatoru a následně změnit nastavení, aby projekt splňoval požadavky popsané v prvním odstavci této kapitoly. Následně je potřeba, aby uživatel vytvořil soubor „qmake.pro“, pokud soubor není už vytvořen. Do toho souboru uživatel vloží kód z oficiální dokumentace Webots [1]. Tento kód je umístěn v sekci „Development Environments/Using Your IDE“. Následné přiřazení kódu robotu a jeho spuštění probíhá obdobně jako u Visual Studia. Opět je možné využít debugger přímo v Qt Creatoru.

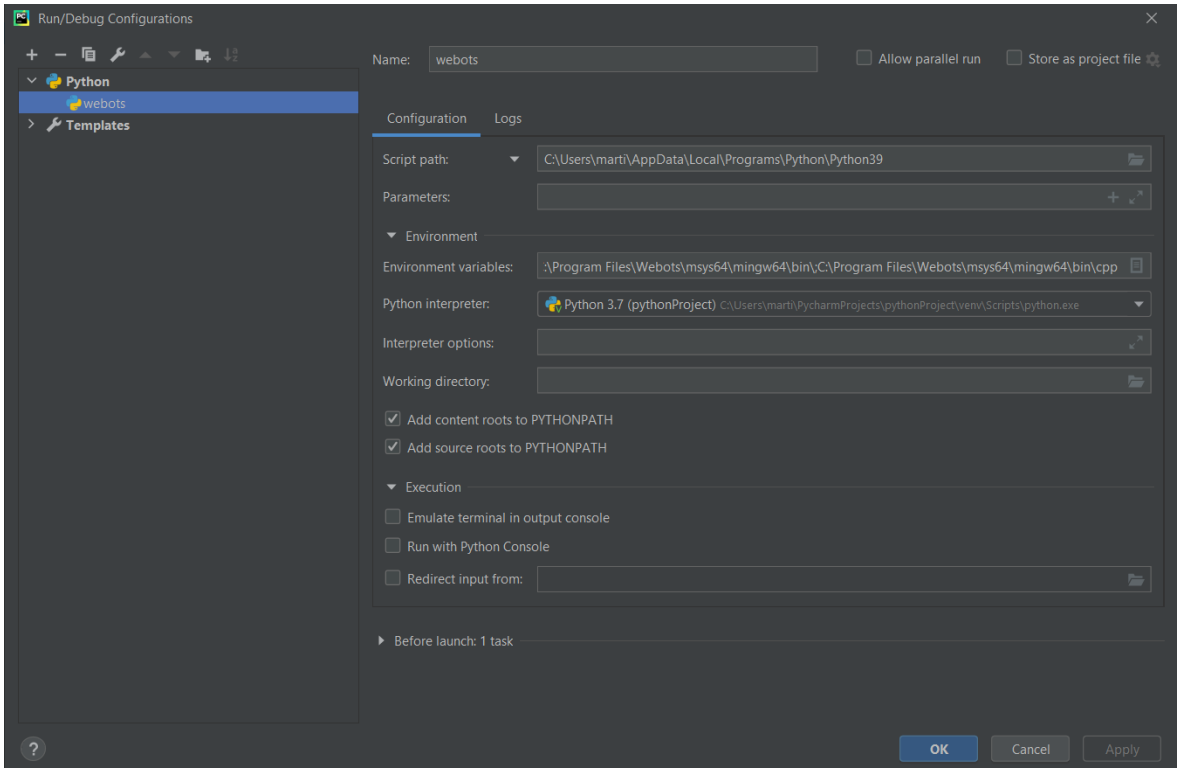
Jako poslední vývojové prostředí bude popsán PyCharm. Toto vývojové prostředí slouží pro vývoj programů v jazyce Python. Pro vývoj Webots kódu je nutné vytvořit controller

v prostředí Webots. Následně je možné v prostředí PyCharm otevřít složku s controllerem. Nyní uživatel musí nastavit vývojové prostředí. V nastavení PyCharmu najde položku „Project Structure“. Uživatel klikne na tlačítko „Add Content Root“ a nastaví cestu na složku Pythonu v domovském adresáři Webots.



Obrázek 12 Nastavení cesty Pythonu

Dále je potřeba nastavit překladač PyCharmu. Pro otevření nastavení uživatel klikne na položku „Run“ v hlavní nabídce PyCharmu. Následně vybere možnost „Edit Configurations“. Po přidání nového nastavení překladače musí uživatel vyplnit pole „Script Path“. Do tohoto pole zadá cestu ke složce Pythonu v počítači. Do pole „Environment variables“ uživatel vloží cestu, kterou opět nalezne na oficiální dokumentaci Webots [1] v sekci „Development Environments/Using Your IDE“. Po těchto krocích je možné začít vyvíjet kód. Spuštění kódu je obdobné jako u předchozích vývojových prostředí.

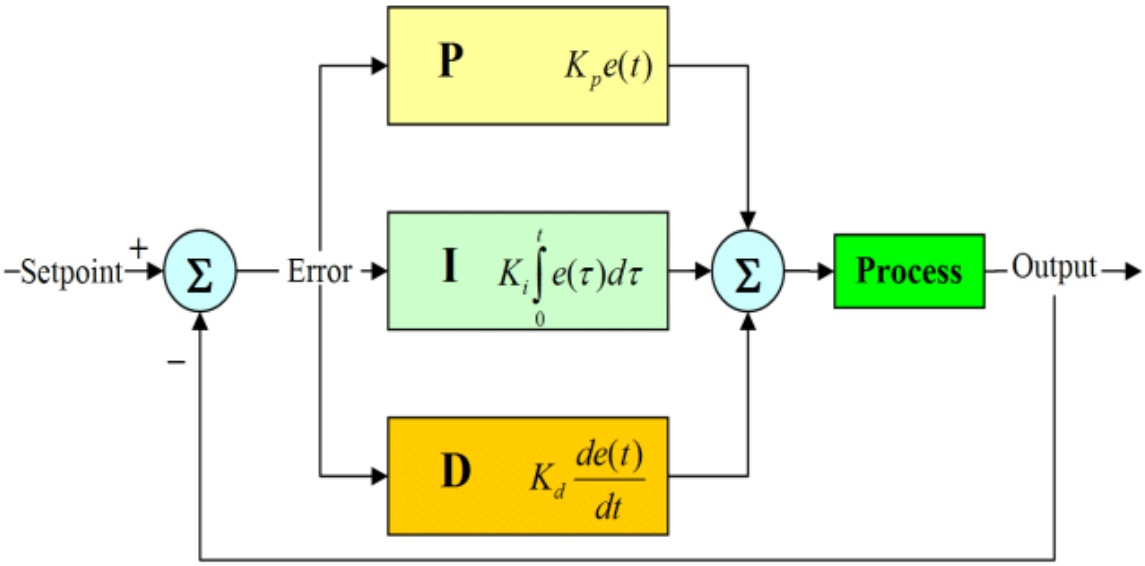


Obrázek 13 Nastavení překladače PyCharmu

1.6 PID regulátory

PID regulátor je běžně využíván k řízení procesů a strojů včetně vozidel, robotů, ale třeba i raket. V oblasti automatizace a robotiky je to jeden ze základních a často používaných algoritmů řízení. Oblíbený je hlavně díky své robustnosti a univerzálnosti. [6]

Samotný regulátor se skládá ze tří částí. To je proporcionální, integrační a derivační část. Jednotlivé části se nastavují tak, aby se dosáhlo optimální regulace dané soustavy. Účelem regulátoru je zajistit, aby hodnota na výstupu sestavy byla stejná jako nastavená žádaná hodnota (set-point). Toto porovnání je prováděno pomocí zpětné vazby. Jednotlivé části regulátoru se musí individuálně vyladit, aby regulátor správně fungoval. Hodnota na výstupu je součet výstupních veličin jednotlivých částí regulátoru. Pomocí zpětné vazby se pak výstupní hodnota porovná se vstupní hodnotou. Rozdíl těchto dvou hodnot je pak chyba, se kterou regulátor pracuje. [7]



Obrázek 14 Blokové schéma PID regulátoru [8]

1.6.1 Proporcionální část

Proporcionální část regulátoru je nejjednodušší. Výstupní veličina této části je přímo úměrná veličině na vstupu. Chyba, se kterou regulátor pracuje se vynásobí koeficientem K_p. Tento koeficient se dá taky označit jako konstanta proporcionálního zesílení. Pokud je tato konstanta příliš vysoká, celý systém může být nestabilní. Pokud je konstanta naopak nízká, systém je málo citlivý. [6][9]

1.6.2 Integrační část

Integrační část je přímo úměrná velikosti chyby a délce trvání chyby. Výsledkem je, že i malá chyba způsobí růst integrační složky. Tento člen nepracuje jen s aktuální hodnotou chyby, ale i s předcházejícími hodnotami, přesněji s jejich integrálem. Hodnotu integrálu následně vynásobí konstantou integračního zesílení K_i. Tato část regulátoru stabilizuje oscilace, které vznikají vlivem proporcionální části regulátoru. Vzhledem k faktu, že integrační člen pracuje s minulými hodnotami, může se snadno stát, že integrovaná chyba požadovanou hodnotu překročí. [6][9]

1.6.3 Derivační část

Derivační část vypočítá derivaci chyby – „zkoumá“, jak rychle se chyba změnila. Vypočtenou derivaci následně vynásobí konstantou derivačního zisku K_d. Díky derivační části se regulátor může rychleji ustálit na požadované hodnotě. [6][9]

II. PRAKTICKÁ ČÁST

2 FYZIKA

V této kapitole popíšu několik jednoduchých ukázek, jak pracuje fyzika v programu Webots. V těchto ukázkách budu měnit některé parametry, které ovlivní chování objektu, například gravitaci prostředí, váhu objektu nebo tření mezi povrchy. Pro tyto účely jsem vytvořil jednoduché pracovní prostředí, které pro základní ukázky zcela dostačuje. Kapitola je pojata jako návod pro uživatele, aby si vyzkoušeli práci s programem a fyzikou.

2.1 Vytvoření prostředí

Vytvoříme si nový projekt. V hlavní nabídce zvolíme položku „Wizards“ a hned první volbu „New Project Directory“. V dialogovém okně vybereme umístění projektu, název projektu a název světa (pracovního prostředí). Po vytvoření projektu klikneme pravým tlačítkem myši na prázdné prostředí. Zvolíme jedinou možnost, kterou lze vybrat. Tou je možnost „Add New“. V dialogovém okně zvolíme možnost „Base Nodes“ a vybereme položku „Solid“. Následně volbu potvrdíme tlačítkem „Add“. V panelu objektů vlevo rozklikneme dvojklikem nově vytvořený node „Solid“. Pak dvojklikem poklepeme na možnost „children“. Otevřelo se nám stejné dialogové okno jako před chvílí. Dále zvolíme možnost „Base Nodes“, ale tentokrát vybereme možnost „Shape“ a opět potvrdíme tlačítkem „Add“. Následně si otevřeme node „Shape“. V něm si vyhledáme node „geometry“. Na tento node poklepeme a v dialogovém okně vybereme možnost „Plane“. Následně u tohoto nodu změníme parametr „Size“ na hodnoty x = 2 a y = 0,5. Node „geometry“ zavřeme pomocí šipečky vlevo od něj. Poté klikneme na node „Shape“ a dole v panelu objektů do pole „DEF“ zadáme slovo „FLOOR“. I tento node následně zavřeme. V nodu „Solid“ vyhledáme node „boundingObject“. Dvojklikem si opět otevřeme dialogové okno. Tentokrát v nabídce vybereme možnost „USE“ a následně vybereme možnost „FLOOR“. Tím jsme objektu, který tvořila jen průchozí textura, přidali „hitbox“. Díky tomu, že jsme jej přidali přes možnost „USE“, „hitbox“ dědí parametry od nodu „Shape“. To znamená, že vždy bude mít stejnou velikost a tvar. Nyní už jen změníme parametr „name“ nodu „Solid“, aby prostředí bylo přehlednější.

Obrázek 15 Vytvořená plocha

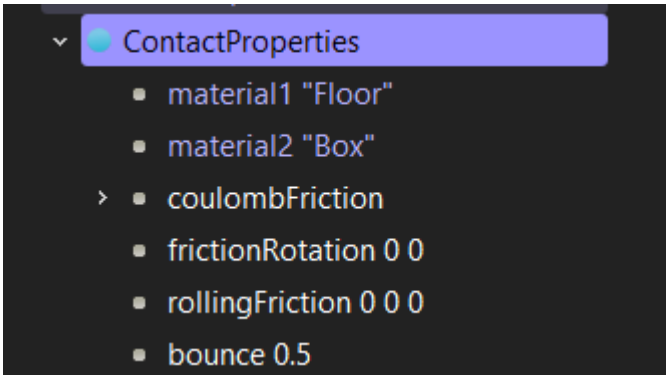
Dále si vytvoříme objekt, na kterém následně budeme testovat fyzikální model programu. Tvorba tohoto objektu probíhá stejně jako vytváření podlahy. Rozdíly ve tvorbě nastanou až po vytvoření nodu „Shape“. U tohoto nodu nodu „geometry“ přidáme tvar „Box“. Velikost necháme přednastavenou. Dále nodu „appearance“ přidáme možnost změnit vzhled objektu. Dvojklikem si otevřeme dialogové okno a vybereme možnost „PBRAppearance“. Následně v parametru „baseColor“ změníme hodnoty RGB, aby objekt měl jinou barvu než podlaha. Další postup je obdobný jako u vytváření podlahy. U nodu „Shape“ do pole „DEF“ zadáme slovo „BOX“. Pomocí toho pak v nodu „Solid“ nastavíme node „boundingObject“. Následně v pracovní ploše klikneme na vytvořenou krychli a pomocí osového kříže ji posuneme na okraj podlahy. Zároveň ji zvedneme mírně do výšky. V horní liště ikonek následně zastavíme simulaci. Opět se vrátíme do panelu objektů. V nodu krychle změníme parametr „name“ pro přehlednost. Poté poklepeme na node „physics“ a v dialogovém okně zvolíme jedinou možnost „Physic“. V tomto nodu budeme následně měnit některé vlastnosti v rámci ukázek.

Obrázek 16 Prostředí s objektem

2.2 Gravitace

V nodu „WorldInfo“ si vyhledáme parametr „gravity“. Výchozí hodnotu 9,81 změníme na 0. Po následném spuštění simulace by krychle měla zůstat ve vzduchu. Poté parametru „gravity“ změníme hodnotu na 1. Krychle by měla začít padat na plochu. Následně si simulaci vrátíme na začátek a parametr „gravity“ změníme zpět na hodnotu 9,81. Po následném spuštění simulace by měla rychlost padání krychle být jednoznačně vyšší než při hodnotě gravitační konstanty 1 m/s^2.

Následně si v nodu „WorldInfo“ vyhledáme položku „contactProperties“. Poklepeme na ni a v dialogovém okně vybereme jedinou možnost „ContactProperties“. Tento nově vytvořený node si otevřeme. V parametrech „material1“ a „material2“ změníme výchozí název „default“ na „Floor“ u prvního a „Box“ u druhého parametru. Tím si v podstatě pojmenujeme dva materiály. Následně tyto dva materiály musíme přiřadit jednotlivým objektům. Otevřeme si node podlahy a v něm vyhledáme parametr „contactMaterial“. Do tohoto parametru zadáme název prvního materiálu, takže „Floor“. Obdobně nastavíme materiál i krychli. Samozřejmě do parametru „contactMaterial“ zadáme název druhého materiálu, takže „Box“.



Obrázek 17 Node ContactProperties

Poté se vrátíme k nodu „ContactProperties“. V něm vyhledáme parametr „bounce“. Tento parametr je automaticky nastaven na hodnotu 0,5. Tento paramentr udává, jak moc se od sebe budou jednotlivé materiály odrážet. Parametr změníme na hodnotu 0 a znovu spuštěním simulace necháme krychli spadnout na plochu. Pokud bylo vše nastaveno správně, krychle se od podlahy vůbec neodrazí. Simulaci si vrátíme na začátek a parametr „bounce“ vrátíme na původní hodnotu. Opět spustíme simulaci. Tentokrát by se krychle měla od povrchu mírně odrazit a po opětovném dopadu zůstat v klidu. Následně si simulaci zase resetujeme a parametr „bounce“ změníme na hodnotu 1. Krychle by se po dopadu na

plochu měla odrazit mnohem více než v předchozím případě a odrážet se opakovaně do menší a menší výšky. Nakonec by se po několika sekundách měla přestala odrážet a zůstat v klidu na ploše.

Parametr „bounce“ necháme nastavený na hodnotě 1. Změníme ale váhu krychle. V nodu „physiscs“ u krychle si najdeme parametr „inertiaMatrix“. Dvojklikem něj si otevřeme dialogové okno. V něm si zvolíme možnost „Bounding object based“. Díky tomu program, na základě velikosti a tvaru objektu, vypočítá vektor setrvačnosti, váhu a polohu těžiště objektu. Tyto hodnoty ale můžeme bez problémů změnit. Kdybychom zvolili druhou možnost v dialogovém okně, všechny tyto parametry bychom museli zadat ručně. Váhu objektu si necháme na výchozí hodnotě 1. Váha krychle bude tedy 1 kg. Opět si spustíme simulaci. Výsledek simulace by měl být stejný jako v předchozím případě. Lze tedy usuzovat, že program s těmito hodnotami pracoval ještě, než jsme je krychli přiřadili. Simulaci si restartujeme, ale tentokrát si váhu krychle, tedy parametr „mass“ změníme na hodnotu 100. Krychle tedy bude mít při této simulaci váhu 100 kg. V tomto případě by se krychle měla v podstatě zabořit do plochy. Při samotném doteku s plochou ale dojde ke zpomalení pohybu krychle.

2.3 Tření

Pro ukázku tření si vrátíme váhu krychle na 1 kg. Následně v nodu „ContactProperties“, který se nachází v nodu „WorldInfo“, vrátíme parametr „bounce“ taky na původní hodnotu 0,5. Poté si otevřeme parametr „coulombFriction“. Tento parametr udává hodnotu smykového tření. Výchozí hodnota je 1. Tuto hodnotu si pro zatím ponecháme. Následně si krychli položíme, pomocí osového kříže, na plochu. Při pozastavené simulaci na klávesnici podržíme klávesy „Ctrl“ a „Alt“, pak klikneme na krychli levým tlačítkem myši a tlačítko budeme také držet. V místě, kde jsme kliknuli na krychli, se vytvořil vektor síly. Pohybem myši můžeme nastavit jeho velikost a směr. Sílu si nastavíme přibližně na 2 N. V tento moment můžeme pustit všechna tři tlačítka. Po spuštění simulace by se krychle neměla ani trochu pohnout. Simulaci si opět pozastavíme. Stejným způsobem nastavíme sílu na 10 N. Krychle by se po spuštění simulace měla pohnout mírně ve směru vektoru. Opět si pozastavíme simulaci a sílu nastavíme na 100 N. Následně, po spuštění simulace, by se chování krychle mělo dát přirovnat vržení hrací kostky. Krychle by se tak měla překutálet přes celou plochu.

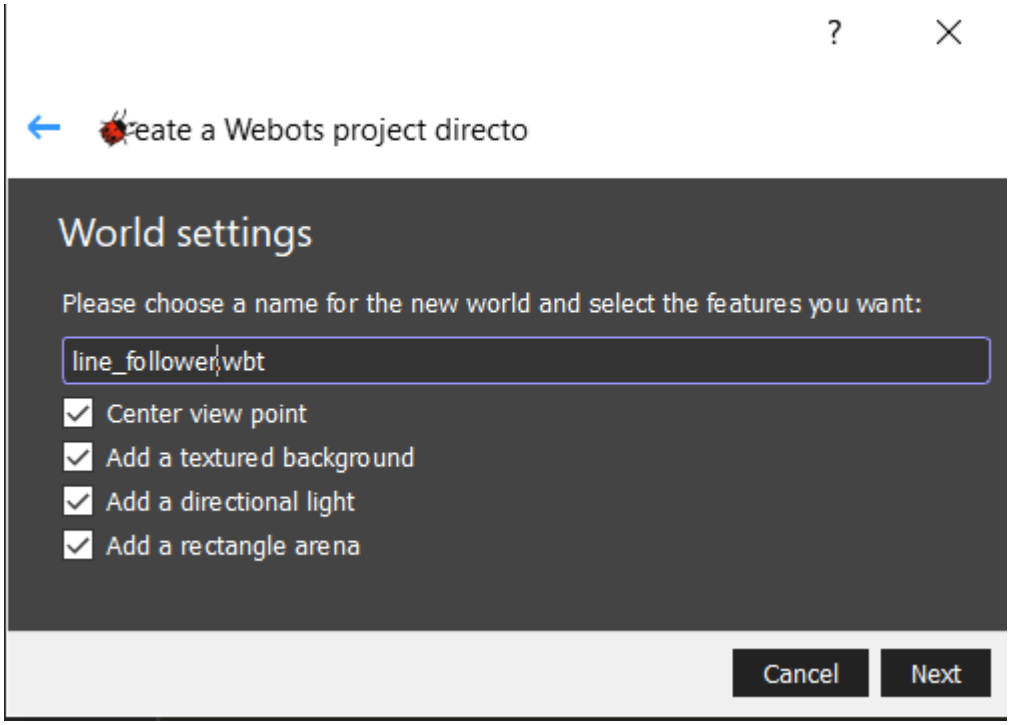
Simulaci vrátíme zpět do výchozí pozice. Parametr tření si tentokrát nastavíme na hodnotu 0. Opět nastavíme vektor síly působící na krychli, tentokrát na hodnotu 1 N. Po spuštění simulace by krychle měla začít pomalu klouzat ve směru vektoru síly. Po několika sekundách by krychle se měla doklouzat na konec plochy bez jakýchkoli známek zpomalení. Následně si zkusíme zvýšit hmotnost krychle. Nastavíme ji na 10 kg. Opět si nastavíme vektor síly na 1 N. Krychle by měla začít pomalu klouzat. Klouzání tentokrát bude mnohem pomalejší. Zkusíme si tedy zvýšit sílu na 10 N. Krychle by měla začít klouzat přibližně stejnou rychlostí jako při prvním pokusu, kdy měla hmotnost 1 kg. Pokus budeme opakovat ještě jednou. Tentokrát si sílu vektoru nastavíme na 50 N. Krychle by měla začít klouzat o poznání rychleji než v předešlých případech.

3 LINE FOLLOWER

Jako první jsem vytvořil prostředí, ve kterém bude robot následně pracovat. Za úkol jsem měl vytvořit prostředí pro jednu z disciplín, které se vyskytují na robotických soutěžích. Já jsem si vybral první disciplínu sledování čáry (line follower), kdy má robot za úkol projet trať za co nejkratší čas. Tratí v tomto případě je černá čára, která je kvůli kontrastu na bílém podkladu. Následně jsem vytvořil robot pro tuto disciplínu. Zároveň jsem i vytvářel návod pro tvorbu robotu. Dále jsem vytvořil několik ukázek řídicích algoritmů pro robot.

3.1 Základní prostředí

Jako první jsem vytvořil nový projekt. V hlavní nabídce jsem si otevřel záložku „Wizards“. Následně jsem vybral možnost „New Project directory“. V dialogovém okně jsem nejprve zvolil umístění složky. Dále jsem přejmenoval svět (pracovní prostředí). Na stránce dialogového okna „World settings“ se nachází čtyři checkboxy. První tři by měly být automaticky zaškrtnuté. Poslední checkbox „Add a rectangle arena“ jsem zaškrtnul ručně. Díky tomu se mi automaticky vytvořilo výchozí pracovní prostředí, se kterým jsem dále pracoval. Tímto prostředím je prázdná plocha ohraničená zdí.

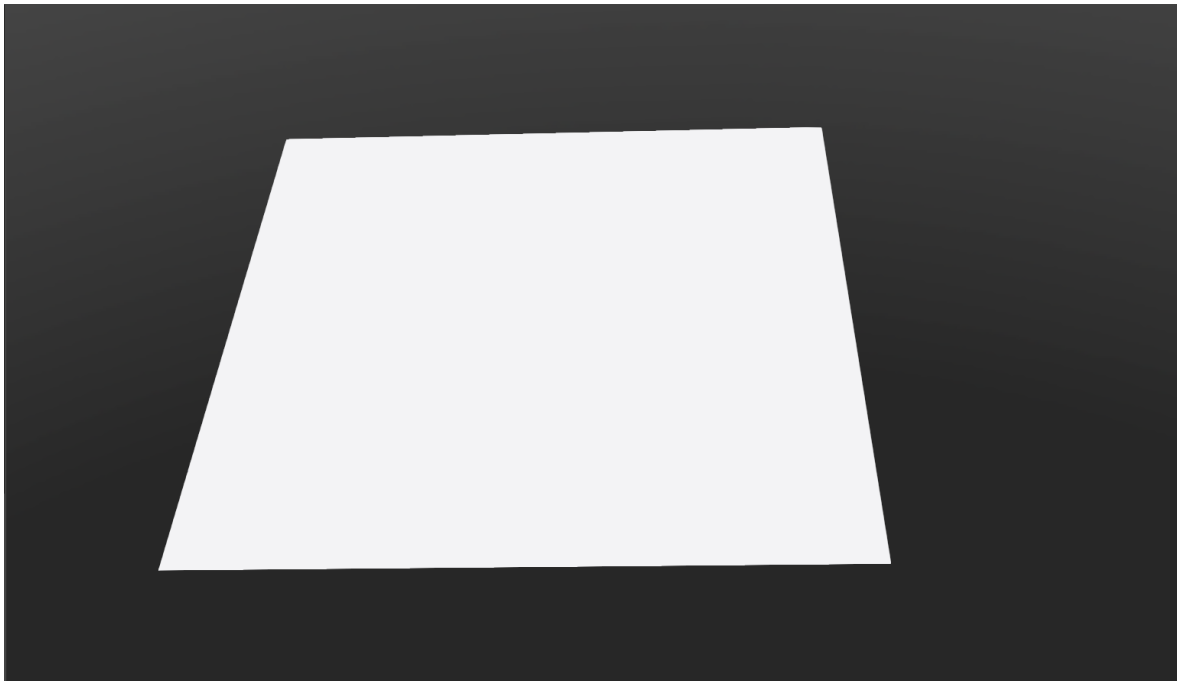


Obrázek 18 Dialogové okno pro vytvoření projektu

Nejprve jsem chtěl změnit výchozí styl podlahy ze šachovnicového s texturou dřeva na čistě bílý. Rozkliknul jsem node „RectangleArena“. Následně jsem pravým tlačítkem myši

kliknul na node „floorAppearance“ a v nabídce vybral možnost „Convert to Base Node“. Node jsem si levým tlačítkem myši otevřel a opět pravým tlačítkem myši jsem kliknul na node „baseColorMap“. V nabídce jsem vybral možnost „Reset to Default Value“. Tímto krokem jsem dokončil bílou plochu. Poté jsem už jen zvětšil velikost plochy, aby se na ní dalo lépe pracovat. To jsem udělal pomocí parametru „floorSize“, ve které jsem nastavil velikost v ose x a v ose y.

Po tomto kroku jsem ještě zmenšil zdi kolem plochy, aby mi na pracovní ploše nezavazely při různých úhlech kamery. To jsem udělal pomocí parametru „wallHeight“. Webots neumožňuje nastavení tohoto parametru na 0, takže jsem musel nastavit velmi malou výšku. Zároveň jsem stejným způsobem upravil i parametr „wallThickness“. Tím jsem zdi zmenšil natolik, aby nebyli viditelné a nepřekážely mi při práci.



Obrázek 19 Základ prostředí s bílou podlahou

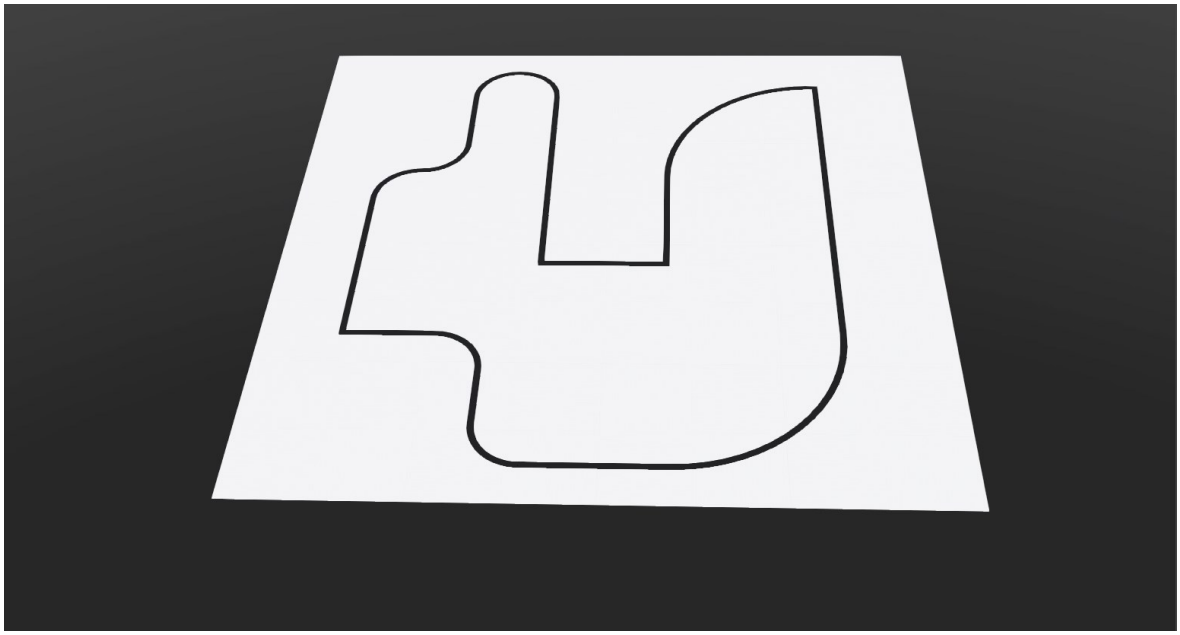
3.1.1 Čára

Pro vytvoření čáry jsem použil nástroj Tinkercad, což je nástroj, běžící v prohlížeči, pro tvorbu jednoduchých 3D modelů. Tento nástroj jsem využil, protože samotné vytváření čáry ve Webots by bylo náročné. Tinkercad byl pro mě mnohem přívětivější v tomto směru. Nástroj sice vyžaduje registraci, ale jinak je zcela zdarma, takže nebyl problém ho využít pro mé účely.

Čáru jsem vytvořil pomocí základních objektů, kterým jsem jejich výšku nastavil na nejnižší možnou hodnotu a další parametry jsem upravil podle potřeby. Nejtěžší bylo vytvořit zatáčky v podobě oblouku. U těch bylo potřeba pomocí dvou válců vytvořit prstenec a následně pomocí krychle oříznout podle potřeby.

Jakmile jsem byl s dráhou spokojený, tak jsem všechny objekty, ze kterých byla dráha tvořena, spojil do jednoho. Poté už stačilo jen exportovat dráhu ve formátu OBJ. Následně stačilo soubor importovat do prostředí Webots.

Ve Webots už objekt stačilo napozicovat, tak aby byl na dříve vytvořené bílé ploše a zároveň, aby z ní nevyčníval. Následně jsem ještě musel upravit barvu čáry, která po exportu nebyla úplně černá. Tím byla tvorba prostředí pro disciplínu sledování čáry hotova.



Obrázek 20 Hotové prostředí disciplíny

3.2 Návod na tvorbu robotu

Další částí byla tvorba robotu. V zadání se nachází bod, který říká, že mám vytvořit návod pro tvorbu robotu. Proto jsem paralelně tvořil tento návod a robot. Robot je velmi jednoduchý, ale funkční pro disciplínu sledování čáry. Samozřejmě je možnost přidat další senzory a jiné součástky tak, aby byl robot schopný dělat i další disciplíny. Můj základní robot je dvojkolový. Dvě kola jsem zvolil, protože na tuto disciplínu se nejčastěji využívají dvojkolové roboty. Na robot jsem dal tři infračervené senzory, které na základě odrazu

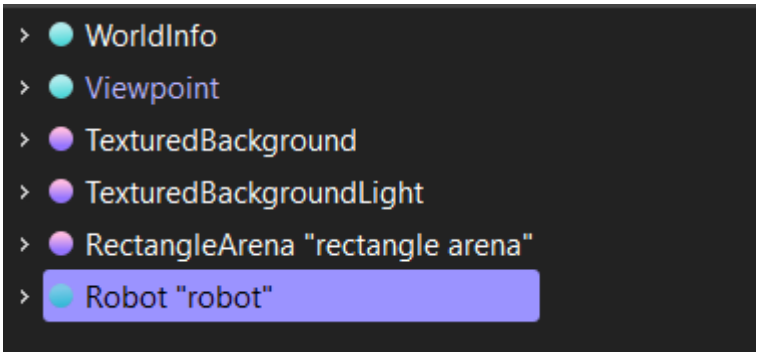
světla dokážou rozpoznat kontrast. Tři senzory jsem zvolil z důvodu, že to pro jednoduchou ukázku stačí.

3.2.1 Základní tvar

Pro vytvoření vlastního nového robotu musíme nejprve kliknout ikonku „plus“, která se nachází v horní liště ikonek.

Obrázek 21 Ikonka "plus" červeně označena

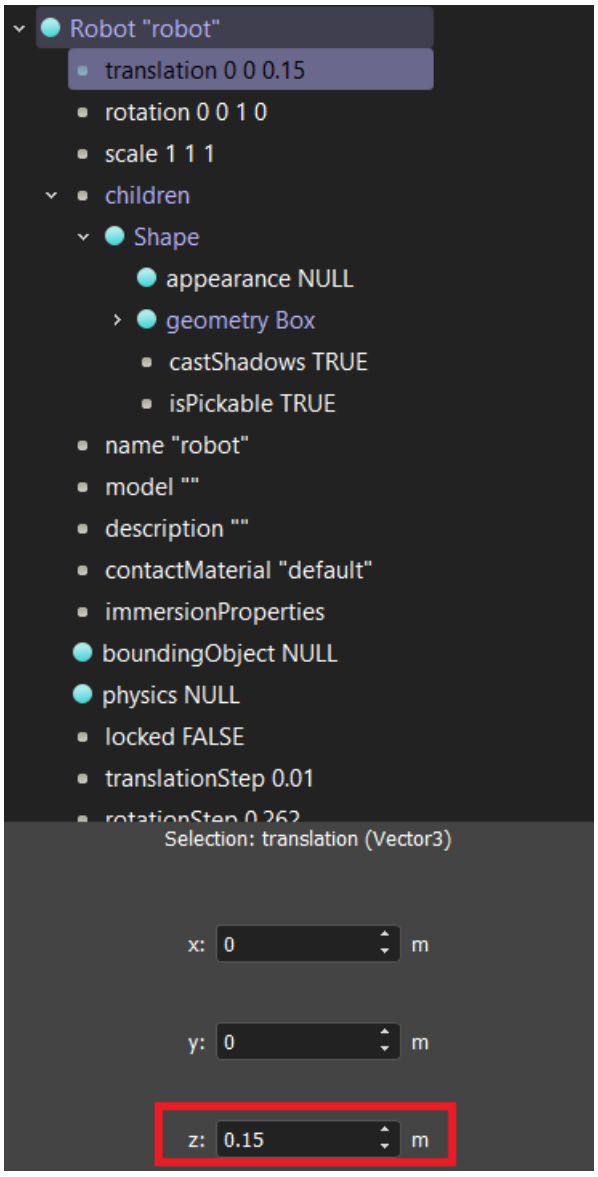
Otevře se nám nové dialogové okno. V nabídce tohoto dialogového okna rozklikneme hned první položku, kterou je „Base nodes“. Rozbalí se nám nabídka s několika možnostmi. V našem případě logicky zvolíme možnost „Robot“ a klikneme vpravo dole na tlačítko „Add“. Vizuálně nenastane téměř žádná změna, jen se uprostřed pracovní plochy objeví osový kříž. Nicméně se nám v panelu objektů objevila nová položka „Robot“.



Obrázek 22 Zvýrazněná položka "Robot"

Tuto položku rozklikneme. Zobrazí se nám dlouhá nabídka všemožných parametrů. Jedním z parametrů je jméno, které je vhodné změnit pro lepší přehlednost, když se v prostředí nachází více robotů. V našem případě to ovšem není zcela nutné, takže si ponecháme výchozí název „robot“. Parametr nebo spíše v tomto případě node, který nás zajímá, je „children“. Na ten klikneme levým tlačítkem myši a označíme si ho. Poté opět klikneme na ikonku „plus“ v horní liště. Alternativně můžeme kliknout na node „children“ pravým tlačítkem a následně v nabídce vybrat „Add new“. Obě varianty otevřou opět dialogové okno, které jsme měli otevřené už dříve. Opět vybereme první možnost „Base nodes“ a v nabídce vybereme možnost „Shape“ a zase potvrdíme tlačítkem „Add“. Opět se nic vizuálně nezmění, ale robotu jsme přidali tvar, zatím pouze abstraktní. Rozklikneme si tedy node „Shape“ a dvojklikem klikneme na node „geometry“. Zase se nám objeví

dialogové okno, ve kterém opět vybereme první položku „Base nodes“. Zde vybereme základní tvar robotu. Volba je libovolná, ale v našem případě použijeme krychli neboli „Box“ a opět potvrdíme tlačítkem „Add“. V prostředí se nám objeví krychle, jejíž střed je přesně uprostřed výchozí plochy, tím pádem je krychle z poloviny zapuštěná do oné plochy. Buď pomocí osového kříže, který máme zobrazený nebo pomocí parametru „translation“ u robotu posuneme krychli tak, aby se nám s ní dobře pracovalo. U metody osového kříže stačí kliknout na šipku a držet osu, směřující nahoru a táhnout směrem, kam potřebujeme. Pokud osový kříž není zobrazený, stačí kliknout na krychli a kříž se zobrazí. U metody změny parametru „translation“ stačí změnit o několik setin hodnotu souřadnice „z“, protože hodnota se uvádí v metrech.



Obrázek 23 Změna parametru "translation"

Nejdříve změníme velikost krychle, aby se její tvar podobal robotu. Rozklikneme tedy node „geometry“, ve které se nachází parametr „size“. Ten opět upravíme, aby nám to vyhovovalo. V mém případě jsem x změnil na 0.07, y na 0.06 a z na 0.03. Tím máme upravenou velikost a z krychle nám vznikl kvádr.

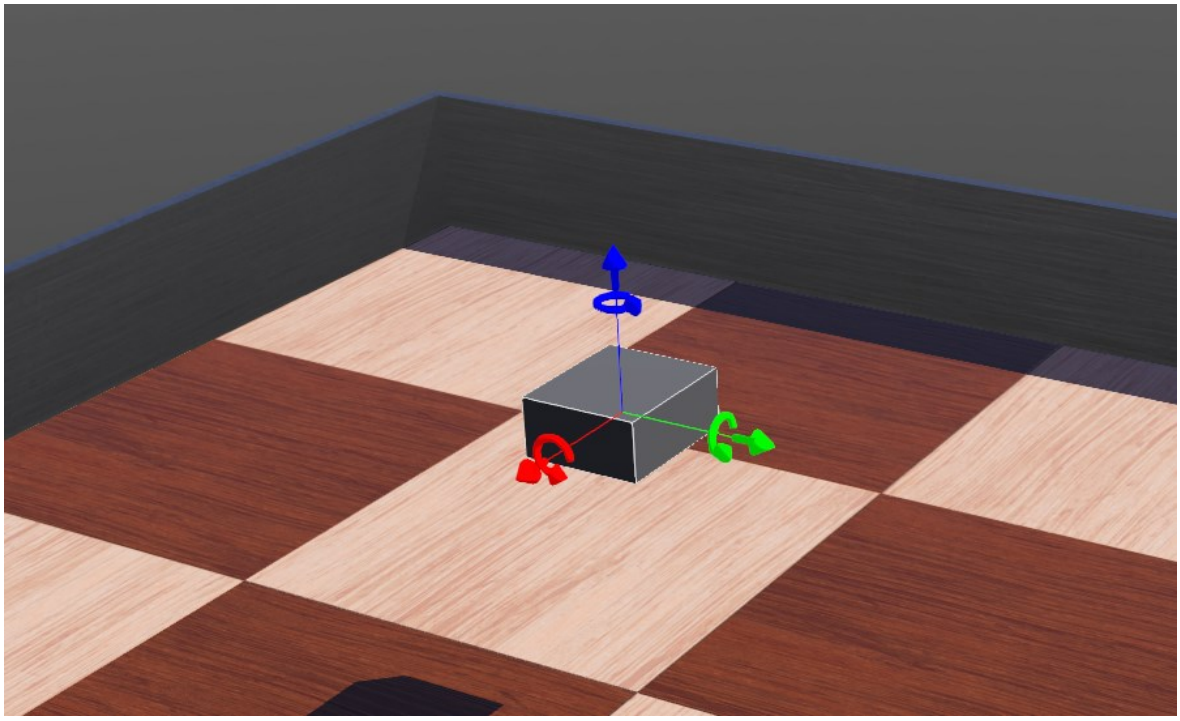
Nyní změním barvu našeho kvádru. Nad nodem „geometry“ se nachází node „appearance“. Klikneme na něj a v dialogovém okně opět zvolíme „Base nodes“ a z nabídky vybereme „PBRAppearance“. Tento node v sobě obsahuje parametry pro změnu vzhledu objektu. Následně si node rozklikneme a otevře se nám ona nabídka parametrů, díky kterým můžeme měnit vzhled objektu. V mém případě jsem změnil parametr „metalness“ (kovový vzhled) na 0. Parametr „baseColor“ jsem ve všech políčkách (red, green, blue) změnil na 0.3. Tyto parametry si každý může upravit podle sebe. Tím je úprava základního tvaru robotu dokončena a můžeme zavřít nabídky parametrů u nodů, které jsme si otevřeli pomocí šipečky vlevo od názvu nodu.

Vzhledem k tomu, že zatím jsme vytvořili jen texturu těla robotu, musíme mu přiřadit další vlastnosti, aby jej bylo možné použít. Klikneme na node „Shape“ a vlevo dole do políčka „DEF:“ napíšeme „BODY“. Následně v nabídce nodů robotu vyhledáme node „boundingObject“ a klikneme na něj dvojklikem. V dialogovém okně zvolíme položku „USE“ a vybereme „BODY“. Tím zajistíme, aby robot měl stejný „hitbox“ jako tvar, který jsme mu přiřadili. Nyní je třeba v horní liště ikonek pozastavit simulaci, pokud běží. To uděláme tak, že zhruba uprostřed lišty klikneme na ikonku pauzy. Jako ujištění, že je simulace opravdu pozastavená, slouží například to, že se v lište nachází ikonka „play“ nebo se zastaví čas simulace.

Obrázek 24 Pozastavená simulace

Nyní robotu přidáme fyziku. Pozastavení simulace je v tento moment výhodné pro usnadnění práce. V případě, že by simulace dál běžela, po přidání fyziky by robot spadnul na plochu. Byl by tedy v pozici, ve které by se s ním špatně pracovalo. Fyziku robotu přidáme v nodu „physics“ hned pod nodem „boundingObject“. Opět na node poklepeme dvojklikem a v „Base nodes“ vybereme tentokrát jedinou položku „Physics“. Tento node si rozklikneme a dvojklikem poklepeme na parametr „inertialMetrix“. V dialogovém okně vybereme možnost „Bounding object based“. Program si následně vypočítá, na základě

velikosti objektu, jeho váhu a nastaví těžiště. Výsledná váha by měla být necelých 130 gramů. To je váha, která může odpovídat reálnému robotu. Program zároveň vypočítal i hodnoty pro vektor setrvačnosti.



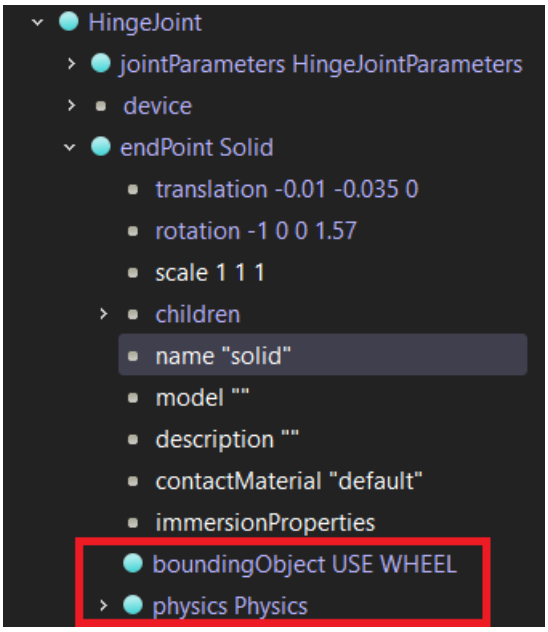
Obrázek 25 Základní tvar robotu

3.2.2 Kola

Nyní robotu přidáme kola. Postup je obdobný, jako u tvoření základního tvaru. Klikneme na node „children“ u robotu a přidáme novou položku. V dialogovém okně si opět rozklikneme položku „Base nodes“ a tentokrát zvolíme „HingeJoint“. Rozklikneme si nově vytvořený node a poklepeme na node „jointParameters“, aby se nám otevřelo dialogové okno. Opět zvolíme „Base nodes“ a vybereme „HingeJointParameters“. Obdobně to uděláme i u nodu „device“, kde zvolíme možnost „RotationalMotor“. Následně si tento node rozklikneme a hned první parametr „name“ změníme tak, aby se nám později dobře používal v kódu například „wheelRight“. Máme hotov základ pro pravé kolo robotu.

Teď vytvoříme samotný tvar kola. Nyní využijeme třetí node, který se nazývá „endPoint“. Poklepeme na něj a v dialogovém okně opět zvolíme „Base nodes“ a následně „Solid“. Node si rozklikneme a nalezneme node „children“, kde stejným způsobem jako v předchozích případech přidáme „Shape“. Rovnou můžeme u tohoto nodu nastavit „DEF:“ na „WHEEL“. Opět jako u základu robotu nastavíme u nodu „appearance“

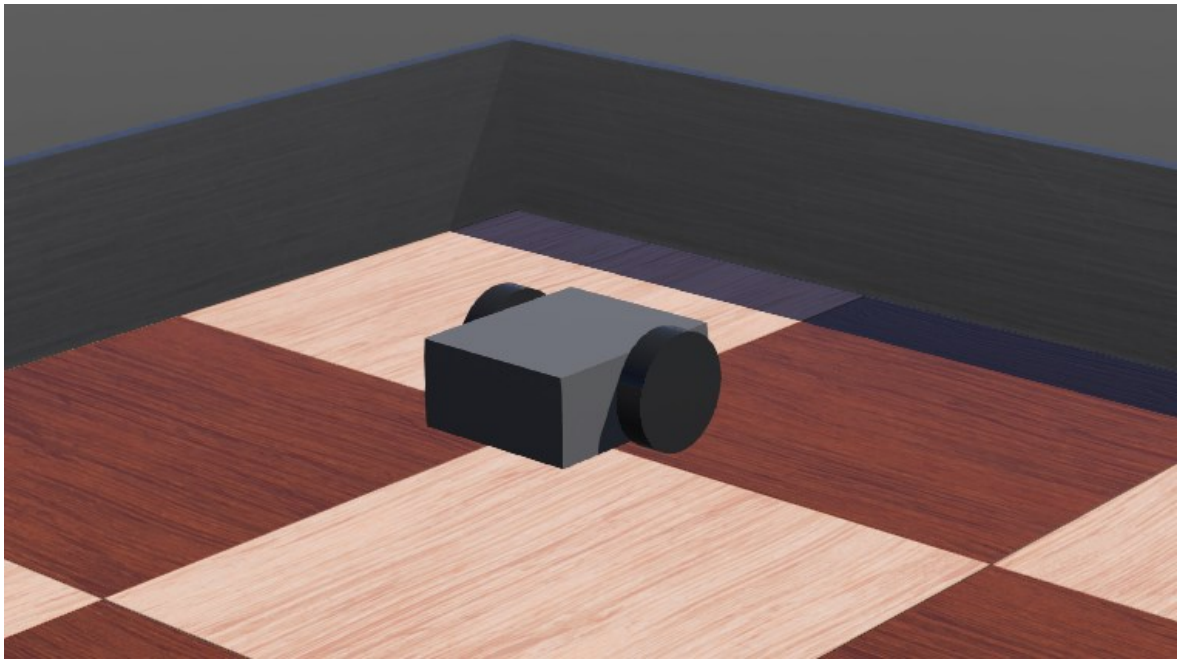
možnost „PBRAppearance“ a následně upravíme vzhled, aby nám vyhovoval. Je vhodné dát kolu jinou barvu, než má základ robotu kvůli přehlednosti. V node „geometry“ je třeba nastavit „Cylinder“. Následně upravíme velikost válce tak, aby velikost kola proporčně odpovídala velikosti robotu. Úprava probíhá obdobně jako u základu robotu. Výšku válce tedy „height“ jsem v mém případě zvolil 0.01 a „radius“ na 0.02. V tento moment se nám kolo ukrylo v robotu je tedy potřeba ho posunout a otočit. První kolo otočíme. To zase můžeme provést pomocí osového kříže nebo u nodu „endPoint“ změníme parametr „rotation“ na x = 1, y = 0, z = 0 a angle na 1.5708. Hodnotu úhlu jsem zjistil otočením kola pomocí osového kříže. Následně kolo napozicujeme podle potřeby. To uděláme pomocí parametru „translation“, protože pomocí osového kříže nedocílíme potřebnou přesnost. V parametru „translation“ jsem v mém případě změnil hodnoty x na -0.01, y na -0.035 a z jsem nechal na 0. Tím máme kolo na pozici, kde ho chceme mít. Následně se vrátíme k nodu „jointParameters“, který rozklikneme. V parametru „anchor“ zadám stejné hodnoty jako jsem před chvílí zadal u parametru „translation“ tedy x = -0.01, y = -0.035 a z = 0. V případě, že máte v parametru „translation“ jiné hodnoty, použijete ty. Následně parametr „axis“ upravíme na x = 0, y = 1 a z = 0. Nyní už jen stačí kolu přidat „hitbox“ a fyziku. To uděláme stejně, jako jsme to udělali u základního tvaru robotu. Tím je jedno kolo hotové.



Obrázek 26 Nastavení "hitboxu“ a fyziky kola

Druhé kolo můžeme udělat naprosto stejným způsobem nebo můžeme první kolo zkopírovat a následně jen upravit jeho název a pozici tak, aby bylo na druhé straně. Skryjeme si opět pomocí šipeček vlevo vše, co už nebudeme potřebovat. Pravým tlačítkem

klikneme na node „HingeJoint“ a klikneme na „Copy“. Následně pravým tlačítkem klikneme na node „children“ a klikneme na „Paste“. Následně změníme jméno u „device“, aby odpovídalo levému kolu, změníme polohu kola tedy „translation“, aby se kolo nacházelo na druhé straně a u „jointParameters“ změníme „anchor“, aby odpovídal poloze kola. Tím jsou kola hotova.



Obrázek 27 Robot s koly

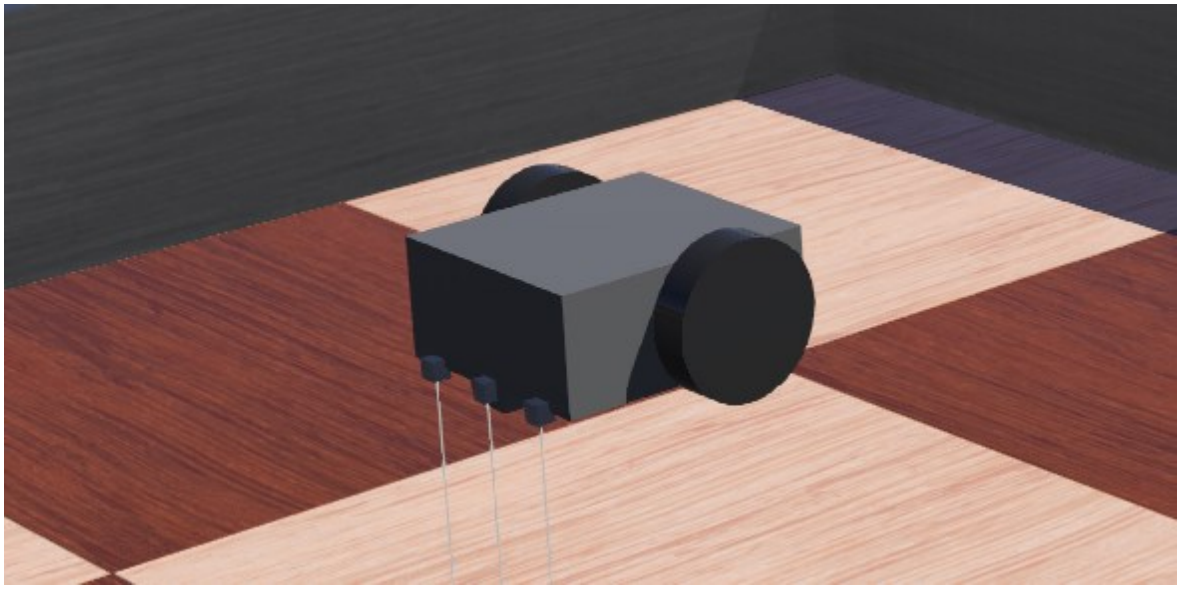
3.2.3 Přední opora

Další částí bude opora v přední části, aby tělo robotu bylo zhruba ve vodorovné poloze. Opět si přidáme do nodu „children“ přímo u robotu nový node. To uděláme stejně jako v předchozích případech. Tentokrát z nabídky vybereme „Solid“. Následně v tomto vytvořeném nodu přidáme do nodu „children“ stejným způsobem nový node „Shape“. Tento node si rozklikneme a stejně jako v předchozích případech do nodu „appearance“ vložíme parametr „PBRAppearance“ a následně upravíme vzhled, jak potřebujeme. Do nodu „geometry“ přidáme parametr „Box“ a následně upravíme jeho velikost. Já velikost krychle nastavil ve všech rozměrech na 0.01. Následně klikneme zpět na node „Shape“ a do políčka „DEF:“ napíšeme název. Následně nodu „Shape“ můžeme pomocí šipečky zavřít a rovnou nastavit „hitbox“ a fyziku u nodu „Solid“. To uděláme stejně jako v předchozích případech jen u parametru „boundingObject“ logicky vybereme název, který jsme dali před chvílí vytvořené krychli. Teď už stačí jen krychli umístit do pozice, aby plnila svůj účel. V nodu „Solid“ nastavíme hodnoty parametru „translation“ na x = 0.03, y

= 0, z = -0.015. Pokud se liší rozměry robotu, budou se lišit i souřadnice. Tím je další část dokončena.

3.2.4 Senzory

Poslední částí je vytvoření senzorů, díky kterým se robot bude držet čáry. Opět v nodu „children“ robotu přidáme novou součástku naprosto stejnou cestou, jako v minulých případech. Z nabídky vybereme „DistanceSensor“. Jako první u senzoru změníme název. Klikneme na parametr „name“ a pojmenujeme jej. Senzorům není třeba dávat nějaký fyzický model, aby byly funkční, ale přesto je vhodné jim model nastavit, aby bylo přesně vidět, kde se nachází. Nodu „children“ přidáme nodu „Shape“. Opět nastavíme parametr, abychom mohli upravovat vzhled a přidáme libovolný tvar. V mém případě jsem použil krychli. Této krychli jsem nastavil velikost všech stran na 0.005. U nodu „Shape“ nastavíme „DEF:“ například na „SENSOR“. Opět nastavíme „hitbox“ a fyziku a napozicujeme senzor. Pro přehlednost je vhodné si zobrazil paprsky, které ukazují, kam senzor směřuje. V horní liště nabídek klikneme na „View“, následně kurzorem najedeme na „Optional Rendering“ a dále klikneme na možnost „Show DistaceSensor Rays“. Nyní vidíme, kam senzor směřuje a můžeme ho jednodušeji pozicovat. Parametr „translation“ změním na x = 0.036, y = 0 a z = -0.015. Tyto hodnoty mohou být různé, záleží na rozhodnutí uživatele, kam che senzor umístit. Parametr „rotation“ změníme na x = 0, y = 1, z = 0 a angle = 1.5708. Nicméně vzhledem k tomu, že stavíme robot pro sledování čáry, nedává moc smysl senzor měřící vzdálenost. Musíme tedy změnit typ senzoru. V nodu „DistaceSensor“ najdeme parametr „type“, který má výchozí nastavení na „generic“. Klikneme tedy na parametr a vlevo dole klikneme na políčko, ve kterém je naspáno „generic“ a z nabídky vybereme „infra-red“. Obdobně jako u kol, stačí senzor zkopírovat, přejmenovat a upravit pozici tak, aby bylo možné jej použít pro sledování čáry. Přidáme ještě další dva senzory, každý na jedné straně robotu. Tím je celý robot pro sledování čáry téměř dokončen, stačí jen vytvořit řídicí algoritmus pro sledování čáry a v nodu robotu najít parametr „controller“ a vytvořený algoritmus v něm zvolit.



Obrázek 28 Dokončený robot

3.3 Řídicí algoritmy

Aby byl robot plně funkční, je nutné naprogramovat algoritmus, který bude jeho určovat jeho chování. Pro programování jsem využil editor přímo v programu Webots, nicméně není problém kód naprogramovat v jakémkoli jiném vývojovém prostředí, například Visual Studio Code. Vytvořil jsem několik ukázek algoritmů, které roboty řídí. Ty jednodušší vychází čistě z hodnot, které získávají senzory. Využil jsem různé kombinace senzorů a následně porovnával jejich hodnoty. Dále jsem použil složitější, ale běžně používaný algoritmus s využitím PID regulátoru.

3.3.1 Jednoduché algoritmy

Jednoduché algoritmy určují na základě podmínek, kam robot pojede. Základem jednoduchých algoritmů je porovnání hodnot získaných alespoň ze dvou senzorů. V několika ukázkách jsem použil referenční hodnoty pro určení, kde se čára nachází vůči robotu. Tyto referenční hodnoty jsem získal výpisem hodnot, které získaly senzory, do konzole. K tomu jsem využil základní funkci printf(). Následně jsem referenční hodnoty využil pro lepší řízení robotu. Někdy bylo potřeba doladit rychlost pohybu robotu, protože občas nestihl vyhodnotit data ze senzorů. To mělo za následek, že se nedokázal udržet na trati. To je nevýhoda u tohoto typu algoritmů. Níže přikládám ukázku jednoho z řídicích algoritmů, který se stará o to, aby robot sledoval čáru.

```
1     //určení polohy čáry na základě referenční hodnoty
2     bool lineLeft = (leftSenVal < 250);
3     bool lineRight = (rightSenVal < 250);
4     //podmínky pro řízení robotu
5     if((leftSenVal > rightSenVal) && lineLeft)
6     {
7       leftSpeed = 0;
8     }
9     else if((leftSenVal < rightSenVal) && lineRight)
10    {
11      rightSpeed = 0;
12    }
```

3.3.2 Složitější algoritmus

Dále jsem se rozhodl použít algoritmus, který vychází z běžně používaného regulátoru PID. Tento regulátor je velmi přesný a při správném nastavení pracuje velice efektivně. Pracné bylo nalezení správných výchozích hodnot pro regulátor. Po jejich doladění robot následoval čáru plynule bez větších obtíží i při vyšších rychlostech. Celkově pro implementaci byla potřeba složité a zdlouhavé ladění. Ladění vyžadovalo zdlouhavé testování a zkoušení, jak se robot drží čáry. Konstanty regulátory bylo nutno nastavil velmi přesně. Níže přikládám ukázku kódu pro výpočet regulátoru.

```
1   P = Position - NEW_GS * 2; //Výpočet složky P
2   I = P + pErr; //Výpočet složky I
3   D = P - pErr; //Výpočet složky D
4   PID = Kp * P + Ki * I + Kd * D; //Výpočet PID
5   pErr = P; //definice chyby
```

Samotný kód se skládá ze dvou vytvořených funkcí, když pomineme základní funkci „main“. První funkce se stará o zpracování dat ze senzorů, aby se dala použít ve výpočtu pro regulátor. Druhá funkce se stará právě o výpočet regulátoru. Díky tomu je pak robot schopen sledovat čáru. Níže přikládám ukázku.

```
1     //Definice proměnných rychlosti
2     leftSpeed = motorSpeed[0];
3     rightSpeed = motorSpeed[1];
4     //Podínky pro řízení robotu
5     if(!online){
6       if(P == -NEW_GS){
7         leftSpeed = -LFM_FS; //výpočet rychlosti pro levé kolo
8         rightSpeed = LFM_FS; //výpočet rychlosti pro pravé kolo
9       }
10    if(P == NEW_GS){
11        leftSpeed = LFM_FS;
12        rightSpeed = -LFM_FS;
13      }
14    }
```

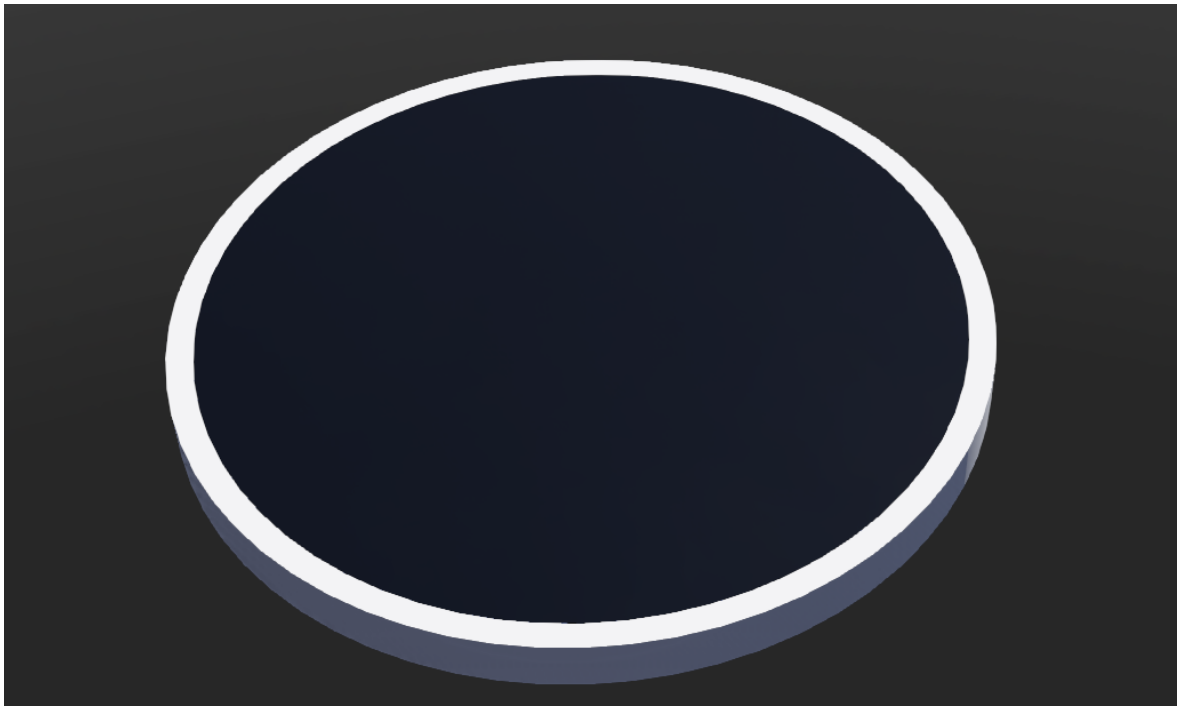
4 ROBOSUMO

Druhou disciplínou, kterou jsem vybral, je robotické sumo. V této disciplíně jde o to, aby jeden robot vytlačil druhého z arény. Aréna je černý kruh, který je ohraničen bílou čárou na jeho obvodu. Kruh má celkový průměr 154 cm.

4.1 Základní prostředí

Jak je psáno v úvodu kapitoly, základní prostředí pro disciplínu robosumo je kruhová aréna. Toto prostředí jsem vytvářel přímo v programu Webots. Jeho tvorba je proti prostředí pro sledování čáry značně jednodušší. Vytvořil jsem nový projekt ve Webots s prázdným pracovním prostředím. Dále vytváření prostředí probíhalo podobně jako prostředí pro ukázky fyziky. Vytvořil jsem node „Solid“. Tomu jsem přiřadil node „Shape“. Jako tvar jsem logicky vybral válec. Parametr „radius“ válce jsem nastavil na hodnotu 0,77. Tento parametr udává poloměr válce v metrech. Změnil jsem taky parametr „subdivision“. Tento parametr udává počet polygonů, které tvoří válec. Čím vyšší je počet, tím je zaoblenější okraj válce. Výchozí hodnota tohoto parametru je 32. Já tuto hodnotu zdvojnásobil. Zadal jsem tedy hodnotu 64. Tato hodnota vytvořila dostatečně zaoblený okraj válce. Následně jsem pomocí nodu „PBRAppearance“ změnil barvu válce na bílou a jeho kovový vhled změnil na 0. Následně jsem stejným způsobem jako v minulých případech přidal válci „hitbox“. A změnil jeho název pro lepší přehlednost.

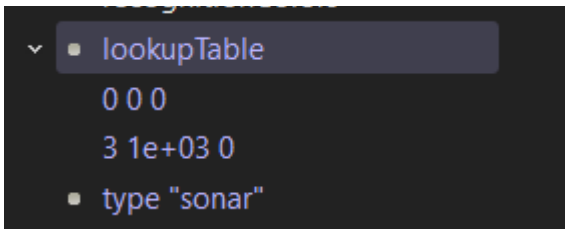
Nyní jsem musel vytvořit druhý válec, který bude tvořit vnitřní černou část arény. Tvorba probíhala stejným způsobem jako u válce předchozího. Velikost poloměru válce jsem zadal 0,72, aby byl průměr černé plochy arény o 10 cm menší než celkový průměr arény. Opět jsem zmenil parametr „subdivision“ na hodnotu 64. Pomocí nodu „PBRAppearance“ jsem odstranil kovový vzhled válce a jeho barvu změnil na černou. Dále jsem přidal „hitbox“ válci a pro přehlednost ho přejmenoval. Vzhledem k tomu, že se oba válce dokonale překrývaly, překrývaly se i samotné barvy. Proto jsem musel prostřední černý válec mírně zvednout nad bílý válec. To jsem udělal pomocí parametru „translation“. U tohoto parametru jsem změnil souřadnici „z“ tak, aby se barvy válců nepřekrývaly a prostřední část byla kompletně černá. Tím je aréna zkompletovaná.



Obrázek 29 Aréna pro robosumo

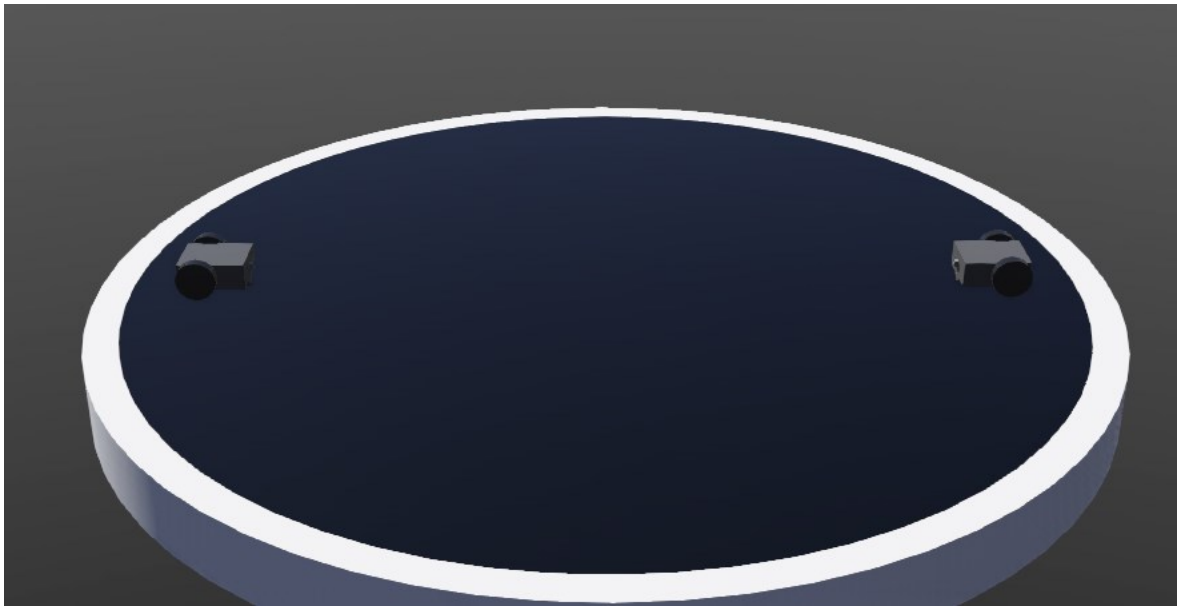
4.2 Robot

Pro tuto disciplínu jsem využil zase vlastní robot. Použil jsem stejný základ jako u robotu pro sledování čáry. Změnil jsem celkovou velikost robotu. Robot pro disciplínu sumo je tedy větší a těžší. Zároveň jsem na robotu použil jen jeden senzor vzdálenosti. Tentokrát jsem typ senzoru změnil na sonar. Tento senzor bude vyhledávat objekt, ke kterému pak robot pojede. U senzoru jsem v parametru „lookupTable“ změnil hodnotu „x“ v druhém řádku na 3 m. To je vzdálenost, kam senzor dohlédne. Hodnotu 3 metry jsem zadal na základě skutečných sonarů pro podobné roboty. Nevýhodou je, že sonar vysílá signál jen v jednom paprsku. Pokud je úhel odrazu paprsku sonaru větší než 22,5°, paprsek se nevrátí zpět. Proto má robot problém před sebou rozeznat šikmé plochy.



Obrázek 30 Nastavení sonaru

Celková tvorba robotu je popsána v kapitole Line Follower, takže ji tady nebudu dál podrobně popisovat. Druhý robot pro robo sumo je kopií prvního. Umístil jsem jej na druhou stranu arény. Roboty se ale liší v řídicím algoritmu.



Obrázek 31 Roboty připravené na ploše

4.3 Řídicí algoritmy

Řídicí algoritmy jsou poměrně jednoduché. Ve své podstatě robot musí najít svého protivníka a pak k němu vyrazit. Obecně vyhrává robot, který najde toho druhého dříve. Vzhledem k tomu, že roboty jsou před startem umisťovány na startovních čarách ve vzájemné vzdálenosti 20 cm, mohou se najít po startu velmi rychel. Pokud se ale hned od startu nevidí, začnou prozkoumávat okolí. Pohybují se po ploše, dokud nenajdou protivníka. V kódu jsem proto použil pomocnou proměnnou „counter“. Počítá, kolikrát byl proveden hlavní řídicí cyklus „while“. Poté jsem pomocí podmínek tuto proměnnou využil na určení času jednotlivých pohybů robotu při hledání protivníka. Tato doba se bude lišit na základě trvání jednoho průchodu cyklem „while“.

Jakmile sonar začne získávat jinou hodnotu než 1000, zaznamenal před sebou objekt. V tomto případě protivníka, kterého se pak snaží vystrčit z arény. Hodnota 1000 je výchozí hodnota, kterou sonar získává, když v jeho dosahu není jiný objekt. Čím je objekt blíže, tím sonar získává nižší hodnoty. Níže přikládám část kódu pro vyhledání protivníka.

```
1  if(sonarValue >= 1000) //1000 - hodnota ze senzoru, když není objekt v
   dosahu
2      {
3        if(counter < 20) //hodnota závisí na době, za kterou se vykoná
4                                 //jeden průchod cyklem while
5        {
6          //nastavení proměnných rychlosti
7          leftSpeed = 0.2 * MAX_SPEED;
8          rightSpeed = -0.2 * MAX_SPEED;
9          //nastavení rychlosti motorům
10         wb_motor_set_velocity(wheels[0], leftSpeed);
11         wb_motor_set_velocity(wheels[1], rightSpeed);
12       }
```

ZÁVĚR

Zadání této práce obsahuje několik úkolů. Prvním bylo prostudování simulátoru Webots. Pro splnění této části jsem nastudoval celý tutoriál v oficiální dokumentaci. Díky tomu jsem se naučil v prostředí Webots vytvářet prostředí pro roboty a vytvářet vlastní roboty. Práci s programem si lze osvojit velice rychle. Problém může činit výchozí editor kódu. Ten se spíše podobá poznámkovému bloku. Sice obsahuje našeptávač, ale ten napovídá jen několik základních příkazů, takže je většinou nutné kód psát ručně. Editor pracovního prostředí robotů je také velice jednoduchý. Jednoduchá prostředí v něm lze vytvořit poměrně bez problémů. Pro většinu simulací, které bude dělat běžný uživatel nebo studenti při výuce, bude tento editor dostačující. U tohoto bodu zadání jsem strávil asi nejvíce času. Chtěl jsem program prozkoumat co nejvíc do hloubky a vyzkoušet co nejvíce funkcí. Navíc jsem se věnoval i ukázkám funkčnosti fyziky. Celkově fyzika v programu funguje poměrně dobře. Pro základní simulace bude určitě postačující. Je ale nutné dodat, že Webots je jednoduchý program, který je zdarma. To znamená, že určitě nemá fyziku propracovanou do nejmenšího detailu. Kdo hledá simulátor, který bude s fyzikou pracovat co nejrealističtěji, musí najít nějaký vhodnější program. Nebo zkušenější programátoři mohou vytvořit svůj vlastní fyzikální engine pro Webots.

Dalším úkolem bylo vytvořit prostředí pro některou z disciplín, které se vyskytují na robotických soutěžích. Zvolil jsem dvě disciplíny. První zvolenou disciplínou bylo sledování čáry. K této disciplíně jsem vytvořil vlastní pracovní prostředí. Dále jsem vytvořil vlastní robot. Současně jsem i zpracoval návod na tvorbu robotu.

Dalším úkolem byla implementace několika zdrojových kódů, které měly definovat chování robotu. Nejvíce času mi zabralo naprogramování sledování čáry pomocí regulátoru na základě PID. Musel jsem zadat poměrně přesné hodnoty konstant, aby robot správně fungoval.

Dále jsem se zaměřil na disciplínu Robosumo. Opět jsem vytvořil vlastní pracovní prostředí a vlastní robot. Tvorba robotu tentokrát už nebyla tak zdlouhavá. Robot jsem tvořil už po několikáté a zároveň jsem se nemusel věnovat tvorbě návodu. Nakonec jsem pro tuto disciplínu naprogramoval několik řídicích algoritmů. Celkově jsem se snažil vymyslet co nejjednodušší algoritmy, aby byly co nejvíce pochopitelné pro naprosté nováčky. Bohužel u této disciplíny jsem narazil na menší problém. Fyzikální engine

programu nezvládal propočítávat kolize dvou robotů. Několikrát se mi stalo, že roboty po kolizi od sebe náhodně odletěly.

Myslím, že ať už Webots nebo jiné obdobné simulátory mají obrovskou škálu využití. Od testování nových algoritmů pro autonomní vozidla, přes hejnovou robotiku až po běžného člověka, který se jen chce naučit programovat roboty, ale nemá k tomu dostatečné prostředky. Samozřejmě nesmíme opomíjet využití ve výuce, kdy simulátory přináší obrovskou řadu výhod. Ať už se jedná o šetření peněz, které by byly utraceny za nové roboty nebo jen součástky, které by byly různě poškozeny nebo o rozšíření kapacit pro různé kroužky a třídy. Těch výhod je opravdu mnoho a myslím si, že si časem simulátor Webots nebo jiný simulátor najde do výuky svou cestu. Dětem simulátor může rozšířit obzory. Mohou díky jemu najít nový koníček. Zároveň jim poskytuje možnost, jak se zábavně naučit programovat.

SEZNAM POUŽITÉ LITERATURY

[1] Cyberbotics: Webots User Guide [online], c1998-2022. Lausanne: Cyberbotics [cit. 2022-04-28]. Dostupné z: https://cyberbotics.com/doc/guide/index

[2] MORDENTI, Andrea, 2013. Programming Robots with an Agent-Oriented BDI-based Architecture: Explorations using the JaCa and WeBots platforms. 1. Bologna: LAP LAMBERT Academic Publishing. ISBN 978-3659333033.

[3] Top 8 Programming Languages for Robotics, 2021. In: Codete [online]. Kraków: Spółka Komandytowa [cit. 2022-04-28]. Dostupné z: https://codete.com/blog/top-8-programming-languages-for-robotics

[4] MICHEL, Olivier, 2004. Cyberbotics Ltd. Webots ™: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems* [online]. **2004**(1), 1-4 [cit. 2022-04-29]. ISSN 1729-8814. Dostupné z: doi:10.5772/5618

[6] PID Theory Explained, [2022]. *Engineer Ambitiously* [online]. Austin, Texas: NATIONAL INSTRUMENTS [cit. 2022-08-01]. Dostupné z: https://www.ni.com/cs-cz/innovations/white-papers/06/pid-theory-explained.html

[7] How Does a PID Controller Work, c2003-2022. *OMEGA Engineering* [online]. Manchester: OMEGA Engineering [cit. 2022-08-01]. Dostupné z: https://www.omega.co.uk/prodinfo/how-does-a-pid-controller-work.html

[8] PID Block Diagram, c2008-2022. In: *ResearchGate* [online]. Berlin: ResearchGate [cit. 2022-08-01]. Dostupné z: https://www.researchgate.net/figure/PID-Block-Diagram-PID-stands-for-Proportional-Integral-Derivative-control-A-PID_fig1_316709017

[9] PID controller, c2013. *Embedded system* [online]. Nottingham: Sierra Software [cit. 2022-08-01]. Dostupné z: https://www.ssla.co.uk/pid-controller/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EPFL	Swiss Federal Institute of Technology
COLLADA	Formát pro ukládání 3D modelů
STL	Formát pro ukládání 3D modelů
OBJ	Formát pro ukládání 3D modelů
EXE	Formát spustitelného souboru
PID	Proporcionální, integrační, derivační

SEZNAM OBRÁZKŮ

Obrázek 1 Nový projekt ve Webots 12
Obrázek 2 Upravené prostředí ve Webots 14
Obrázek 3 Model města pro simulaci autonomních vozidel ve Webots 15
Obrázek 4 Nastavení nodu ContactProperties 17
Obrázek 5 Nastavení nodu physics 17
Obrázek 6 Základní pracovní plocha 19
Obrázek 7 Hlavní nabídka programu 19
Obrázek 8 Hlavní lišta ikonek 21
Obrázek 9 Vedlejší lišta ikonek 22
Obrázek 10 Panel objektů 22
Obrázek 11 Vytvoření controlleru pro Visual Studio 24
Obrázek 12 Nastavení cesty Pythonu 25
Obrázek 13 Nastavení překladače PyCharmu 26
Obrázek 14 Blokové schéma PID regulátoru [8] 27
Obrázek 15 Vytvořená plocha 30
Obrázek 16 Prostředí s objektem 30
Obrázek 17 Node ContactProperties 31
Obrázek 18 Dialogové okno pro vytvoření projektu 34
Obrázek 19 Základ prostředí s bílou podlahou 35
Obrázek 20 Hotové prostředí disciplíny 36
Obrázek 21 Ikonka "plus" červeně označena 37
Obrázek 22 Zvýrazněná položka "Robot" 37
Obrázek 23 Změna parametru "translation" 38
Obrázek 24 Pozastavená simulace 39
Obrázek 25 Základní tvar robotu 40
Obrázek 26 Nastavení "hitboxu“ a fyziky kola 41
Obrázek 27 Robot s koly 42
Obrázek 28 Dokončený robot 44
Obrázek 29 Aréna pro robosumo 47
Obrázek 30 Nastavení sonaru 47
Obrázek 31 Roboty připravené na ploše 48

SEZNAM PŘÍLOH

Příloha P I: Přenosné médium CD

PŘÍLOHA P I: PŘENOSNÉ MÉDIUM CD

Obsah přiloženého CD:

- BP v elektronické podobě
- Adresář projektu ve Webots lineFollower
    - controllers – řídicí algoritmy
    - worlds – pracovní prostředí robota (spustitelné soubory ve Webots)
- Adresář projektu ve Webots sumo
    - controllers – řídicí algoritmy
    - worlds – pracovní prostředí robota (spustitelné soubory ve Webots)