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

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

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

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

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

Ve Zlíně dne 24. ledna 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, robota a několika ukázek řídícího kódu pro nějakou z disciplín v robotických soutěžích v simulátoru Webots, které budou využívány v kroužcích robotiky, které se konají 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 řídících kódů.

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

ABSTRACT

The aim of this bachelor thesis is to create a working environment, a robot and several examples of control code for some of the disciplines in robotic competitions in the Webots simulator, which will be used in robotics clubs held at FAI UTB in Zlín. The theoretical part of the thesis describes the Webots simulator itself. The practical part contains instructions for creating your own robot directly in the Webots simulator and a description of creating the working environment and control codes.

Translated with www.DeepL.com/Translator (free version)

Keywords: Webots, robotics competition, line following, 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

V dnešní době, kdy neustále roste zájem o informační technologie a mimo jiné i o programování robotů a různé robotické soutěže je zcela na místě tyto technologie a možnosti přivést mezi co nejvíce lidí. Hlavně na základní školy, kde by si děti v různých kroužcích mohly vyzkoušet základy programování. Obzvláště programování jednoduchých robotů by mohlo být velice atraktivní vzhledem k tomu, že to není jen monotónní psaní kódu, kde výsledek je jen nějaký výpis v konzoli. Programování jednoduchých robotů má tu výhodu, že stačí napsat pár řádků a robot se uvede do pohybu. Děti tím pádem vidí výsledky své práce na něčem fyzickém a určitě je to pro ně mnohem zábavnější.

Bohužel v dnešní době řada škol nemá k dispozici prostředky, jak tyto kroužky uskutečnit nebo nemá dostatečné kapacity v již otevřených kroužcích z důvodu nedostatku mikrokontrolerů. Řešením tohoto problému můžou být různé simulátory, které by umožňovaly programovat roboty bez nutnosti mít robota ve fyzické formě. Jedním z těchto simulátoru je Webots, který je naprosto zdarma a není nutná ani registrace. Díky tomu je možné, aby ho školy využívaly pro ony již zmíněné kroužky. Celkově se tento simulátor dá využít, jak pro učení naprostých základů, tak i pro pokročilé programování, kdy si programátor chce vyzkoušet nějaký kód ve virtuálním prostření, než přejde na fyzické testování, aby například vinou chyby v kódu nedošlo k nějakému poškození reálného robota.

Celkově 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 jako je například e-puck, různá robotická ramena, humanoidní roboti 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 možnostem 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 robota, vytvoření a úpravě pracovního prostředí simulátoru a popis vytvořených ukázek kontrolního kódu napsaných v programovacím jazyku C, tak aby bylo možné je využít při výuce nebo v dalších pracích. Jazyk C jsem zvolil mimo mé osobní preference taktéž

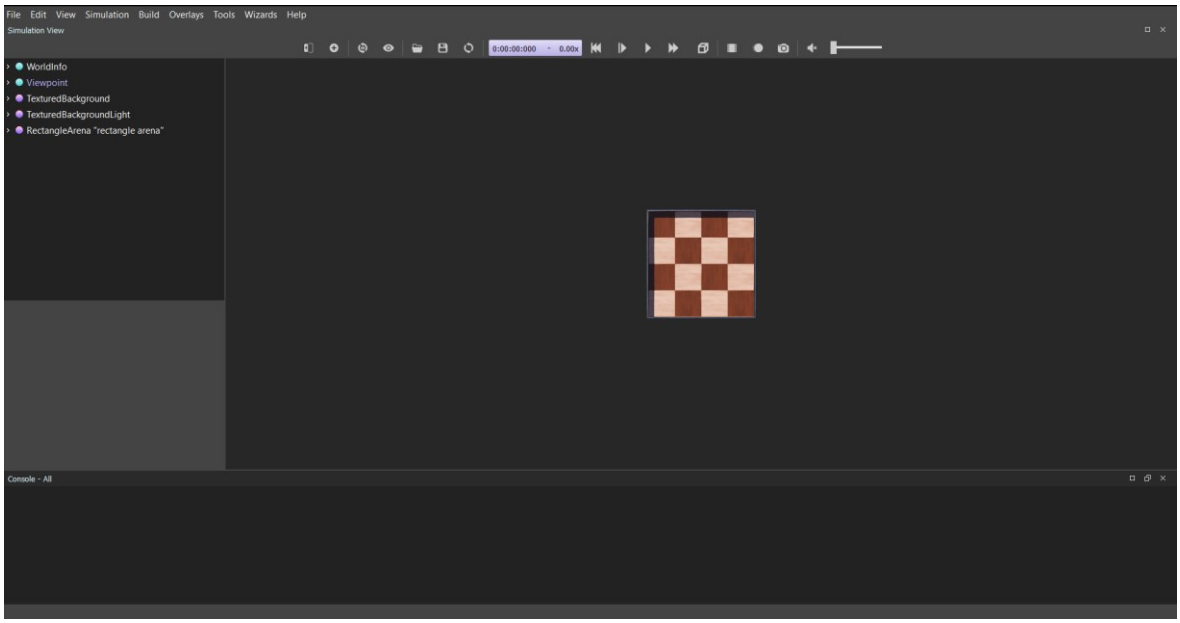
z důvodu, že většina jazyků používaných při programování robotů vychází právě z jazyka C.

I. TEORETICKÁ ČÁST

1 SIMULÁTOR WEBOTS

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

Zároveň obsahuje možnost upravovat prostředí, ve kterém bude vybraný robot či roboti pracovat. Opět s možností si do programu nahrát vlastní vytvořené prostředí v jiném programu. Taktéž Webots obsahuje i vlastní jednoduchý editor zdrojových kódů.



Obrázek 1 Nový projekt ve Webots

1.1 Historie Webots

Simulátor Webots byl původně vyvíjen jako nástroj, jehož účelem bylo zkoumání různých řídících 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 pádem je možné, aby tento program využil kdokoli, kdo má zájem s ním pracovat bez omezení.

1.2 Možnosti

Simulátor Webots nabízí spoustu možností, jak se zavděčit všem uživatelům. Od možnosti si kompletně upravit a vytvořit vlastní pracovní prostředí, přes širokou nabídku již před vytvořených robotů, až po vlastní editor zdrojových kódů, který nabízí programování v několika programovacích jazycích.

Úprava nebo dokonce vytvoření vlastního pracovního prostředí pro robota je programu řešeno především pomocí přidáváním a editací jednoduchých tvarů. Pokud ale uživatel chce může si do programu importovat vlastní pracovní prostředí vytvořené například v programu Blender. Stejně tak je možná přidávat vlastní objekty, takže když uživatel si chce do svého pracovního prostředí přidat například strom, tak mu stačí najít si nějaký model stromu a ten následně importovat do svého prostředí. Samotné vytváření prostředí nebo robotů je pěkně intuitivní a podobá se právě již zmiňovanému Blenderu. Takže uživatel má možnost svým vytvořeným objektům měnit parametry velikosti, barvy nebo může zvolenému objektu importovat texturu například cihel.

Vytváření vlastních robotů v programu Webots funguje na dost podobném principu jako vytváření pracovního prostředí. Tedy je primárně realizováno pomocí jednoduchých tvarů, které si pak uživatel může upravit, tak aby mu vyhovovali. Samotné součástky jako motory, senzory a tak dále nemají žádnou vlastní texturu nebo model, takže pokud uživatel chce lepší vizualizaci než jen osový kříž, který primárně slouží pro pozicování dané součástky, musí dané součástce přidat model například krychle nebo válce. Samozřejmě pokud uživatel chce může si do Webots opět importovat vlastní model robota.

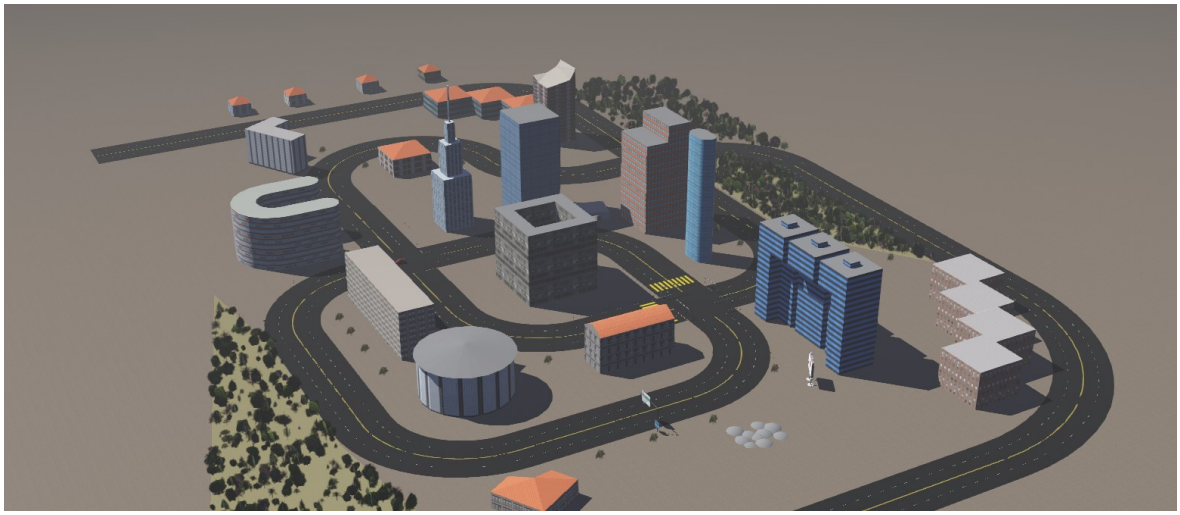
Webots také obsahují vestavěný editor zdrojového kódu, který je velmi jednoduchý a téměř bez našeptávače. Editor našeptává jen základní funkce jako for, while nebo if. Každopádně výhodou je možnost programovat v několika programovacích jazycích. Na výběr Webots umožňují z nabídky jazyky C, C++, Java, Python a Matlab. Takže 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, tedy skutečný fyzický robot, který na základě zdrojového kódu dělá danou práci, je často velmi užitečné a někdy i potřebné prvně provést potřebné simulace. Primární odůvodnění je to, že simulace je jednodušší připravit, jsou mnohdy rychlejší a pohodlnější ale hlavně jsou levnější. Vytvoření nových modelů robotů a nastavení simulace zabere jen několik hodin. Simulovaná robotická sestava je levnější než skuteční roboti a reálné sestavy, což umožňuje lepší zkoumání konstrukce. Simulace často běží rychleji než skuteční roboti a zároveň jsou všechny parametry snadno zobrazitelné na obrazovce. Simulace umožňují používat počítačově nákladné algoritmy, které by na reálných robotických mikrokontrolerech potřebovaly obrovské množství času, jako jsou genetické algoritmy. V neposlední řadě jsou výsledky simulace přenositelné na skutečné roboty. [4]

Obrovský potenciál využití poskytuje hejnová robotika [2], kde používaní roboti jsou velice nákladní. Proto je mnohem výhodnější dělat různé simulace právě v nějakém simulátoru, kde uživatel může mít libovolný počet robotů a je omezen ve své podstatě jen výkonem zařízení na, kterém je simulace prováděna. Následně může už hotový zdrojový kód požít v praxi na skutečných robotech. Další variantou využití může být simulace evoluční robotiky nebo 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. Zrovna autonomní vozidla jsou dobrým příkladem, kde je simulace před zkouškami v praxi velmi užitečná. Přeci jen by bylo velmi nepříjemné postavit autonomní vozidlo, které by mělo poměrně vysokou hodnotu, zničit při první zkoušce z důvodu nějaké chyby ve zdrojovém kódu.

Samozřejmě je i velice dobré využití ve výuce robotiky. Ať už se jedná o nějaký kroužek pro děti na základní škole nebo o hodiny programování na střední nebo vysoké škole. Výhodou opět jsou náklady, kdy 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, někdy s malou dopomocí nějakých dalších programů a nástrojů.

Využití různých simulátorů má v tomto technologickém odvětví opravdu hodně výhod. Konkrétně simulátor Webots umožňuje velké množství simulací, a to i pro běžné uživatele, kteří nepracují na nějakých velkých projektech pro obrovské společnosti zabívající se robotikou. Zároveň k Webots je zpracovaná oficiální dokumentace [1], kde se nachází popis programu jako takového, popis prostředí programu a jednotlivých oken. Zároveň obsahuje i seznam robotů, které program nabízí včetně informací o daném robotovi. Stejně tak je zpracován i seznam objektů pro pracovní prostředí robota nebo součástek, které uživatel může u svého robota použit. Dále se v dokumentaci nachází i tutoriál, který má několik částí a jeho účelem je během pár hodin naučit uživatele pracovat se samotným simulátorem Webots.

2 PROGRAMOVACÍ JAZYKY

V robotice se využívá velké množství programovacích jazyků. Tedy není problém, aby si každý programátor vybral jazyk, který mu vyhovuje nejvíce. Samotný simulátor Webots, jak už bylo psáno výše, obsahuje nejpoužívanější jazyky v robotice [3]. Těmi jsou C, C++, Java, Python a Matlab.

2.1 C

Jazyk C je procedurální programovací jazyk. Původně jej vyvinul Dennis Ritchie jako systémový programovací jazyk pro psaní operačního systému. Mezi hlavní vlastnosti jazyka C patří nízkoúrovňový přístup k paměti, jednoduchá sada klíčových slov a čistý styl, díky těmto vlastnostem je jazyk C vhodný pro systémové programování, jako je vývoj operačního systému. [5]

2.1.1 Historie

První vydání jazyka C vyšlo v roce 1972. Název byl zvolen z důvodu, že tento jazyk přebíral některé vlastnosti staršího programovacího jazyka B. Jazyk C byl původně vyvinut pro operační systém UNIX, aby překonal problémy předchozích jazyků, jako jsou B nebo BCPL. [6]

2.1.2 Využití v robotice

Většina začínajících programátorů pro roboty se učí právě jazyky C a C++. Oba jazyky josu považovány za velmi vyspělé programovací jazyky pro účely robotiky, protože umožňují snadnou interakci s nízkoúrovňovým hardwarem, který roboti potřebují. Jazyk C se využívá zejména, když robot má nějaké omezení v paměti. [3]

2.2 C++

Jazyk C++ je univerzální programovací jazyk, který se v dnešní době hojně v kompetitivním programování. Má imperativní, objektově orientované a generické programovací funkce. C++ běží na mnoha platformách, jako jsou Windows, Linux, Unix, Mac a dalších. [7]

2.2.1 Historie

Jazyk C++ vyvíjel Bjarne Stroustrup v Bellových laboratořích od roku 1979. Protože C++ je pokusem o přidání objektově orientovaných funkcí (a dalších vylepšení) do jazyka C,

dříve se mu říkalo "C s objekty". Jak se jazyk vyvíjel, Stroustrup jej v roce 1983 pojmenoval C++. Název C++ naznačuje, že se jedná o "inkrementované C" (připomeňme, že ++ je inkrementační operátor jazyka C). C++ byl zpřístupněn mimo Bellovy laboratoře v roce 1985. [8]

2.2.2 Využití v robotice

U C++ je využití v robotice dost podobné tomu, jaké je u klasického C. Pokud vývojáři nemají předem určený jazyk, tak většinou upřednostní C++ před C. Jednou z největších výhod jazyka C++ je, že tento jazyk přímo ovládá rozhraní API operačního systému. To znamená, že nepotřebuje žádné wrappery. Díky tomu vývojáři mohou používat knihovny specifické pro danou platformu. [3]

2.3 Java

Java je programovací jazyk, který byl poprvé vydán společností Sun Microsystems v roce 1995. Od skromných začátků se vyvinula do podoby spolehlivé platformy, na níž je postaveno mnoho služeb a aplikací, a dnes pohání velkou část digitálního světa. Na Javu se i nadále spoléhají nové inovativní produkty a digitální služby určené pro budoucnost. [9]

2.3.1 Historie

Historie Javy je velmi zajímavá. Java byla původně navržena pro interaktivní televizi, ale pro tehdejší digitální kabelovou televizi to byla příliš pokročilá technologie. Historie Javy začíná u Green Teamu. Členové týmu Java (známého také jako Green Team), iniciovali tento projekt s cílem vyvinout jazyk pro digitální zařízení, jako jsou set-top boxy, televizory atd. Nejlépe se však hodil pro programování na internetu. Později technologii Java začlenila společnost Netscape.

Zásady pro vytvoření programování v jazyce Java byly následující: "Jednoduchý, robustní, přenosný, nezávislý na platformě, zabezpečený, vysoce výkonný, vícevláknový, nezávislý na architektuře, objektově orientovaný, interpretovaný a dynamický". Javu vyvinul James Gosling, který je znám jako otec Javy, v roce 1995. James Gosling a členové jeho týmu zahájili projekt na počátku 90. let. [10]

2.3.2 Využití v robotice

Většina univerzit, které nabízejí kurzy robotiky, zahrnuje Javu jako povinný programovací jazyk. Java se dnes používá k vytváření mnoha vysokoúrovňových funkcí, které jsou v programování robotiky vyžadovány.

Virtuální stroj Java interpretuje instrukce během běhu kódu rychlým i spolehlivým způsobem. Je to v boji osvědčená a bezpečná technologie, která nabízí speciální funkce pro robotiku pro provádění úkolů podobných lidským. Například k přijímání a zpracování vizuálních obrazů se používá rozhraní API pro řeč nebo framework Java Media. [3]

2.4 Python

Python je univerzální interpretovaný, interaktivní, objektově orientovaný a vysokoúrovňový programovací jazyk. Vytvořil ho Guido van Rossum v letech 1985-1990. Stejně jako Perl jsou i zdrojové kódy Pythonu k dispozici pod licencí GNU General Public License (GPL). [11]

2.4.1 Historie

Guido Van Rossum zveřejnil první verzi kódu Pythonu (verze 0.9.0) na alt.sources v únoru 1991. Tato verze již obsahovala zpracování výjimek, funkce a základní datové typy list, dict, str a další. Byla také objektově orientovaná a měla systém modulů. Python verze 1.0 byl vydán v lednu 1994. Hlavními novinkami obsaženými v této verzi byly nástroje funkcionálního programování lambda, map, filter a reduce, které si Guido Van Rossum nikdy neoblíbil. [12]

2.4.2 Využití v robotice

Python je dnes stále populárnější díky své jednoduché syntaxi, vynikající dokumentaci a rozsáhlé komunitě. Protože je tento jazyk jednoduchý na používání, vyžaduje od vývojářů méně času na jeho osvojení. A když porovnáte jazyk Python s jinými objektově orientovanými programovými jazyky, jako jsou C, C++ nebo Java, zjistíte, že vám při psaní kódu ušetří spoustu času. Tento jazyk je vybaven několika užitečnými funkcemi, které z něj činí klíčového hráče na scéně robotiky. Příklady, které ukazují silné stránky jazyka Python v robotice, jsou Arduino nebo Raspberry Pi používané pro návrh vestavěných systémů. [3]

2.5 Matlab

MATLAB je zkratka pro "maticovou laboratoř". Zatímco jiné programovací jazyky obvykle pracují s čísly po jednom, Matlab pracuje s celými maticemi a poli. Základy jazyka zahrnují základní operace, jako je vytváření proměnných, indexování polí, aritmetika a datové typy. [13]

2.5.1 Historie

Matematickým základem první verze MATLABu byla řada výzkumných prací J. H. Wilkinsona a 18 jeho kolegů, které byly publikovány v letech 1965-1970 a později shromážděny v knize Handbook for Automatic Computation, Volume II, Linear Algebra, kterou editovali Wilkinson a C. Reinsch. V těchto článcích jsou uvedeny algoritmy, implementované v prostředí Algol 60, pro řešení maticových lineárních rovnic a úloh o vlastních hodnotách. [14]

2.5.2 Využití v robotice

Tento vysoce výkonný jazyk pro technické výpočty v sobě spojuje programování, vizualizaci a výpočty. Všechny jsou podávány ve snadno použitelném prostředí, kde mohou vývojáři vyjadřovat problémy a řešení ve známých matematických zápisech. MATLAB je poměrně oblíbený mezi začínajícími vývojáři a často se používá pro analýzu informací a budování řídicích systémů. Hodí se také pro modelování a simulace. MATLAB je snadno použitelný jazyk, který pomáhá eliminovat chyby při implementaci tím, že umožňuje vývojářům identifikovat problémy již ve fázi tvorby prototypu, a ne až později při výrobě (což je nákladnější). [3]

II. PRAKTICKÁ ČÁST

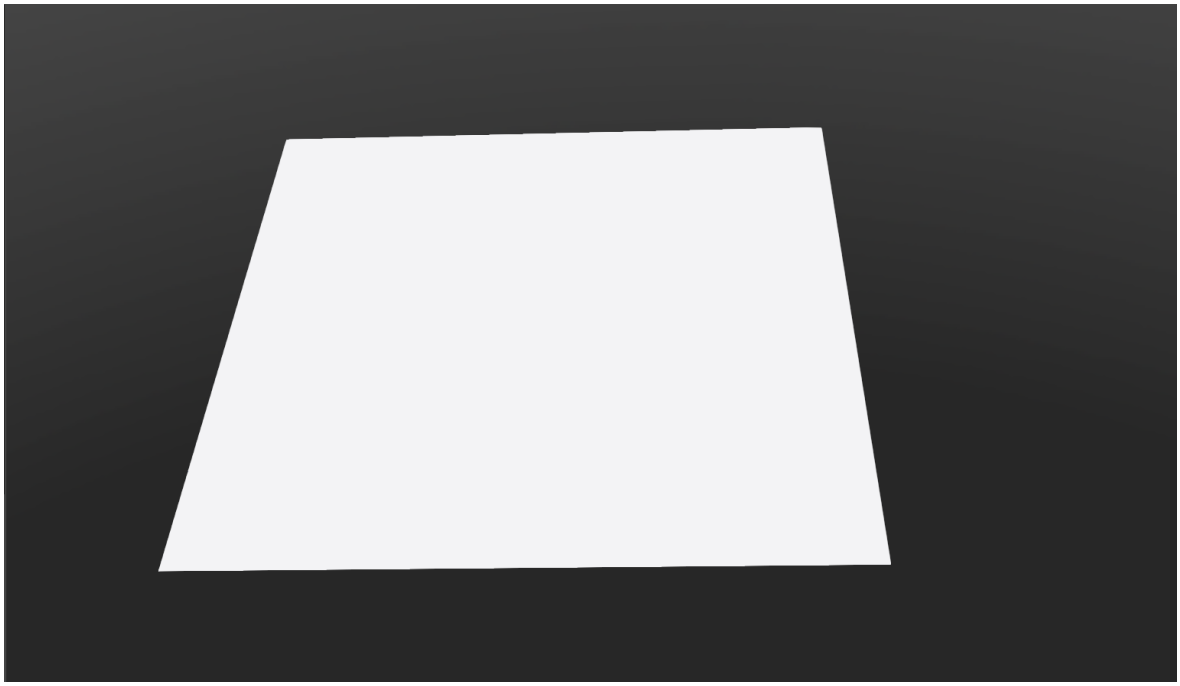
3 PROSTŘEDÍ

Jako první v rámci praktické části 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 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.

3.1 Základní prostředí

Jako základ jsem použil výchozí prostředí, které vytváří samotný program Webots. Tedy čtvercovou plochu ohraničenou zdí. Nejprve bylo potřeba změnit výchozí styl podlahy ze šachovnicového s texturou dřeva na čistě bílý. To jsem udělal změnou parametru vzhledu podlahy, kdy jsem odebral základní možnost a nastavil parametr tak, abych si mohl vzhled podlahy upravit podle sebe. Tento parametr má bílou barvu nastavenou jako výchozí, takže barvu jsem už nemusel měnit. Změnit jsem ale musel parametr „metalness“, tady kovový vzhled podlahy, kdy základně je nastaven na nevyšší možnou hodnotu 1. Tento parametr bylo potřeba změnit na hodnotu 0. Po této změně je podlaha už bílá tak, jak pro své účely potřebuji. Poté jsem už jen zvětšil velikost plochy, aby se na ní dalo lépe pracovat.

Následně jsem ještě zmenšil zdi kolem plochy, aby mi na pracovní ploše nezavazely při různých úhlech kamery. Vzhledem k tomu, že jsem využil před vytvořené prostředí, tak nelze zdi zcela odebrat, ale jsou zmenšit na tolik, aby nebyly viditelné.



Obrázek 4 Základ prostředí s bílou podlahou

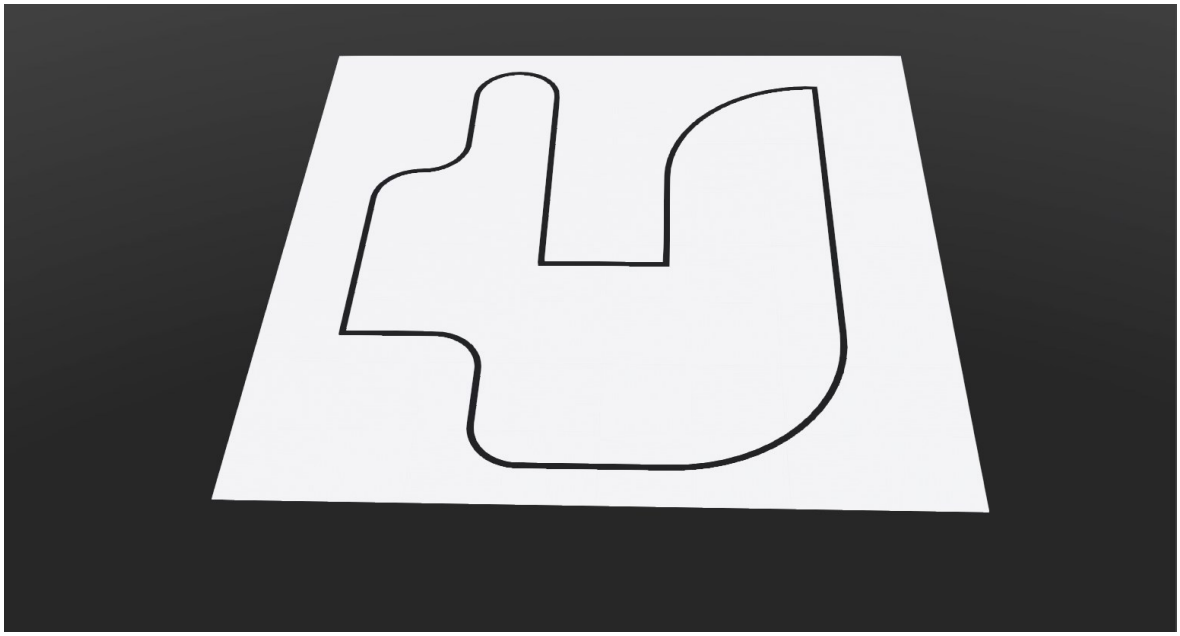
3.2 Čára

Pro vytvoření čáry jsem použil nástroj Tinkercad, což je nástroj, běžící v prohlížeči, pro tvorbu jednoduchých 3D modelů. 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 bílé ploše umístěn tak, aby nevyčníval. Následně jsem ještě musel upravit barvu čáry, která po exportu nebyla úplně černá. Tím byla tvorba prostředí pro disciplínu sledování čáry hotové.



Obrázek 5 Hotové prostředí disciplíny

4 ROBOT

Další částí byla tvorba robota. Vzhledem k tomu, že v zadání je bod, kde mám vytvořit návod pro tvorbu robota, tím pádem jsem musel nějakého robota vytvořit, tak jsem se rozhodl toho využít a vytvořit robota přímo pro sledování čáry. Robot je velmi jednoduchý ale funkční pro disciplínu sledování čáry. Každopádně se na něj dají přidat další senzory a jiné součástky tak, aby byl schopný dělat i další disciplíny.

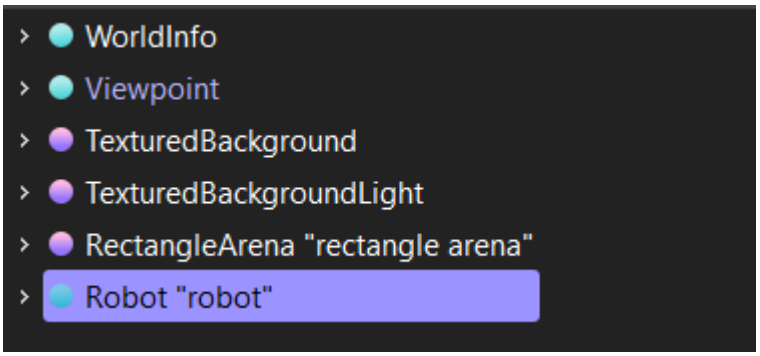
4.1 Návod na tvorbu robota

4.1.1 Základní tvar

Pro vytvoření svého vlastního nového robota musíme prvně kliknout ikonku „plus“, která se nachází v horní liště ikonek. Konkrétně je to druhá ikonka zleva.

Obrázek 6 Ikonka "plus" červeně označena

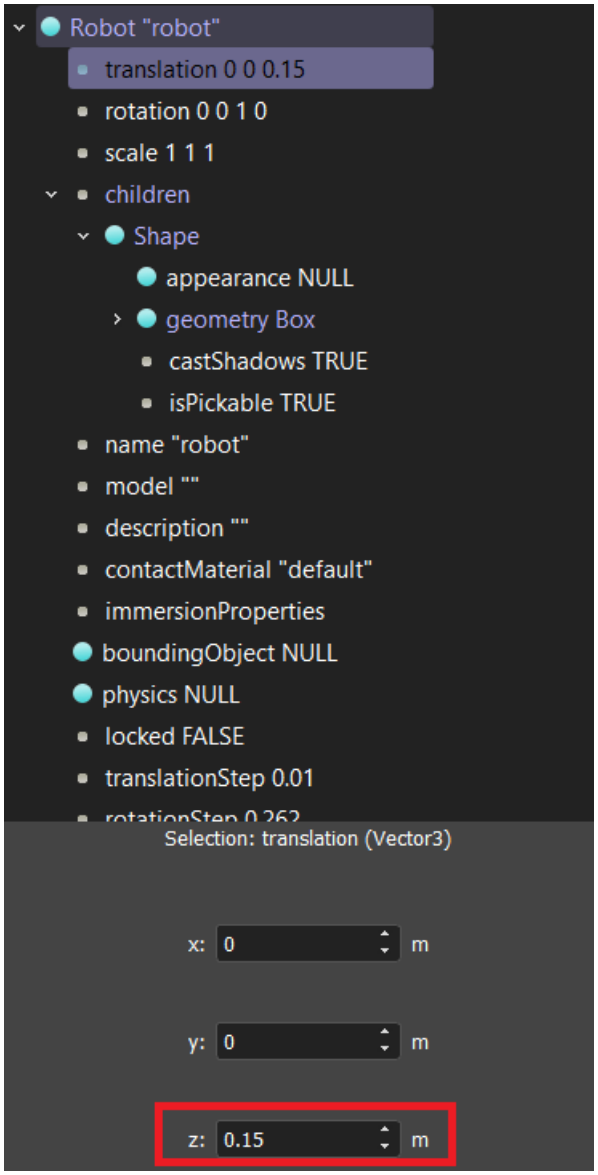
Otevře se nové dialogové okno. V nabídce tohoto dialogového okna rozklikneme hned první položku, kterou je „Base nodes“. Rozbalí se nám nabídka s několika možnostmi. V našem případě logicky zvolíme možnost „Robot“ a klikneme vpravo dole na tlačítko „Add“. Vizuálně nenastane téměř žádná změna, jen se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



Obrázek 7 Zvýrazněná položka "Robot"

Tuto položku rozklikneme. Zobrazí se nám dlouhá nabídka všemožných parametrů. Jedním z parametrů je například jméno, které se dá změnit například pro lepší přehlednost, když se v prostředí nachází více robotů. V našem případě to ovšem není zcela nutné, tím pádem já ponechám výchozí název „robot“. Parametr nebo spíše v tomto případě node který nás

zajímá je „children“. Na ten klikneme levým tlačítkem myši a označíme se 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 před tím. Opět vybereme první možnost „Base nodes“ a v nabídce vybereme možnost „Shape“ a opět potvrdíme tlačítkem „Add“. Opět se nic vizuálně nezmění ale robotovi jsme přidali tvar, zatím ale neví jaký. Rozklikneme si tedy node „Shape“ a dvojklikem klikneme na node „geometry“. Opět se nám objeví dialogové okno, ve kterém opět vybereme první položku „Base nodes“. Zde vybereme základní tvar robota. Volba je libovolná ale v našem případě použiji krychli neboli „Box“ a opět potvrdím 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 robota posuneme krychli tak, aby se nám s ní dobře pracovalo. U metody osového kříže stačí kliknout a držet na šipku, chcete-li osu, směřující na horu a táhnout směrem kam potřebujeme tedy nahoru. Pokud osový kříž není zobrazený stačí kliknout na krychli a kříž se zobrazí. U metody změny parametru „translation“ stačí změnit hodnotu souřadnice „z“. Krychli stačí posunout jen o maličko, tedy stačí úprava jen o pár setin například na 0.15.



Obrázek 8 Změna parametru "translation"

Jako první změníme velikost krychle. Rozklikneme tedy node „geometry“, kde máme 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.

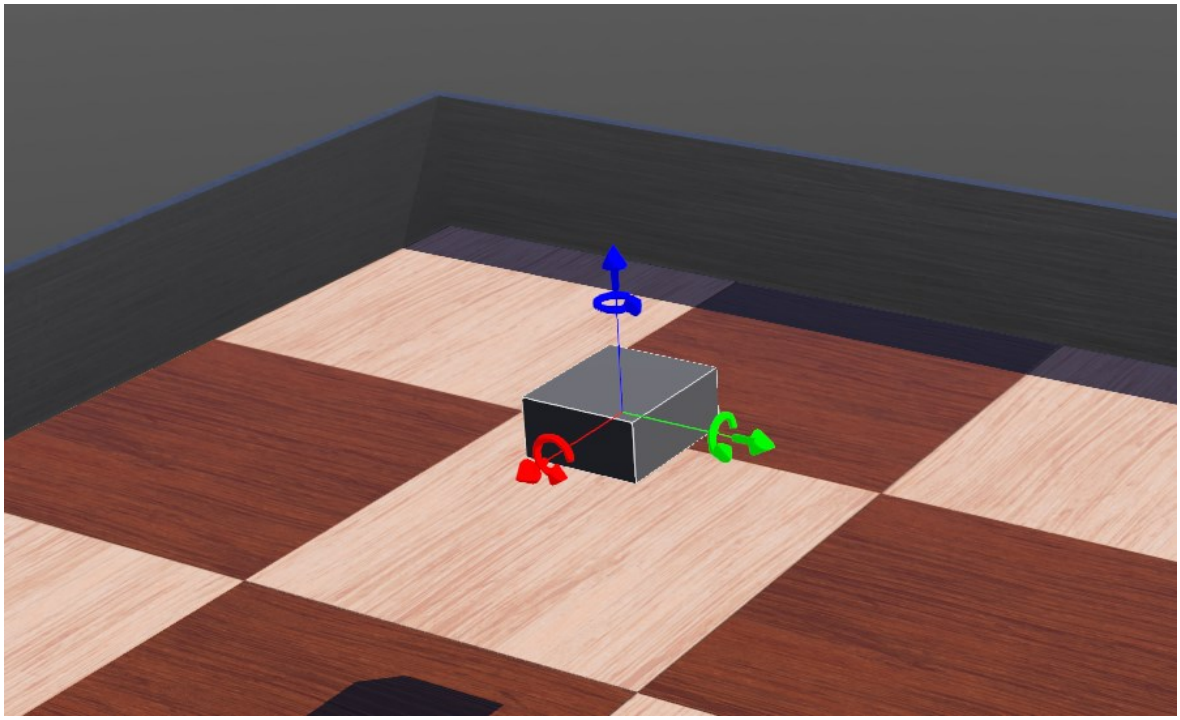
Nyní změníme barvu našeho kvádru. Sice to není vyloženě potřeba ale je to lepší pro následnou lepší přehlednost. Nad node „geometry“ se nachází node „appearance“. Klikneme na něj a v dialogovém okně opět zvolíme „Base nodes“ a z nabídky vybereme „PBRAppearance“. Následně si node rozklikneme a otevře se nám nabídka parametrů díky, kterým můžeme měnit vzhled objektu. V mém případě jsem změnil parametr „metalness“ na 0 a parametr „baseColor“ jsem ve všech políčkách (red, green, blue) změnil

na 0.3. Tím je úprava základního tvaru robota dokončena a můžeme zavřít nabídky parametrů u nodů, které jsme si otevřeli pomocí šipečky vlevo od názvu nodu.

Nyní klikneme na node „Shape“ a vlevo dole do políčka „DEF:“ napíšeme „BODY“. Následně v nabídce nodů robota najdeme node „boundingObject“ a klikneme na něj dvojklikem. V dialogovém okně zvolíme položku „USE“ a vybereme „BODY“. Díky tomu zajistíme, aby robot měl stejný „hitbox“ jako tvar, který jsme mu dali. Nyní doporučuji v horní liště ikonek pozastavit simulaci, pokud běží. To uděláme tak, že zhruba uprostřed lišty klikneme na ikonku pauzy. Simulace je pozastavená, pokud se tam nachází ikonka „play“.

Obrázek 9 Pozastavená simulace

Tohle jsme provedli z důvodu, že nyní přidáme robotovy fyziku. Kdybychom to neudělali, tak po přidání fyziky nám robot spadne na plochu tím pádem se dostane z pozice, ve které s ním chceme pracovat. Fyziku robotovi přidáme v nodu „physics“ hned pod nodem „boundingObject“. Opět na node poklepeme dvojklikem a opět v „Base nodes“ vybereme tentokrát jedinou položku „Physics“. Nyní má robot fyziku a měl by se tedy chovat podle fyzikálních zákonů.



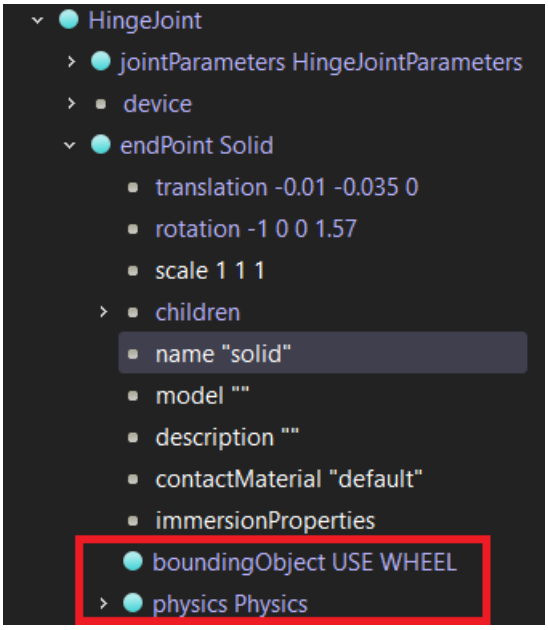
Obrázek 10 Základní tvar robota

4.1.2 Kola

Nyní robotovi přidáme kola. Postup je obdobný jako u tvoření základního tvaru. Tedy klikneme na node „children“ u robota 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“ nebo obdobně. Toto je tedy základ pro pravé kolo robota.

Teď vytvoříme samotný tvar kola. Třetí node se nazývá „endPoint“. Poklepeme na něj a 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 robota nastavíme u nodu „appearance“ možnost „PBRAppearance“ a následně upravíme vzhled, aby nám vyhovoval. Doporučuji dát Kolu jinou barvu, než má základ robota kvůli přehlednosti. V node „geometry“ doporučuji nastavit „Cylinder“ vzhledem k tomu, že vytváříme kolo. Následně upravíme velikost válce tak, aby velikost kola

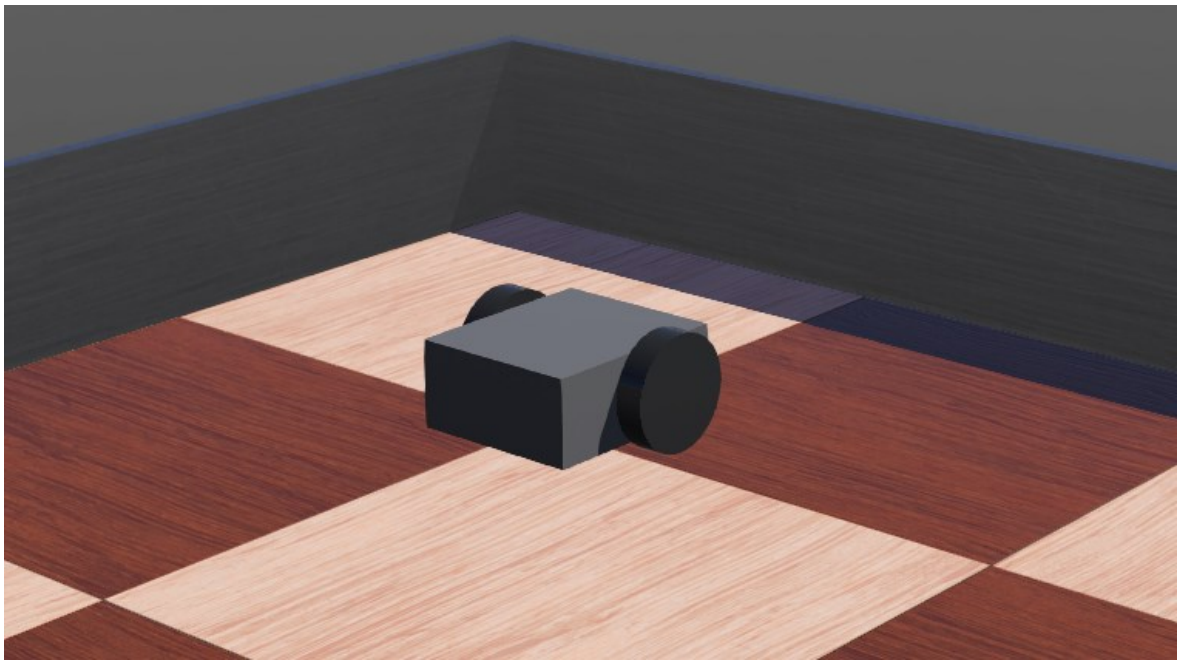
proporčně odpovídala velikosti robota. Úprava probíhá obdobně jako u základu robota. Výšku válce tedy „height“ jsem v mém případě zvolil 0.01 a „radius“ na 0.02. V tento moment se nám kolo ukrylo v robotovi je tedy potřeba ho posunout a otočit. První kolo otočíme. To zase můžeme provést pomocí osového kříže nebo u nodu „endPoint“ změníme parametr „rotation“ na x = 1, y = 0, z = 0 a angle na 1.5708. Následně kolo napozicujeme podle potřeby. Pomocí osového kříže to není dostatečně přesné, takže to uděláme pomoci „translation“. Je potřeba myslet na to, pokud jsme nepohnuli nějak kamerou, tak se díváme na robota z jeho přední strany. U „translation“ jsem 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“ zadáme stejné hodnoty jako jsme před chvílí zadali u parametru „translation“ tedy x = -0.01, y = -0.035 a z = 0. Následně parametr „axis“ upravíme tak, aby se y = 1 a jinde byla 0. Nyní už jen stačí kolu přidat „hitbox“ a fyziku. Uděláme to stejně jako jsme to udělali u základního tvaru robota. Tím bude jedno kolo hotové.



Obrázek 11 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, aby bylo na druhé straně. Skryjeme si opět pomocí šipeček vlevo vše, co už nebudeme potřebovat. Následně Pravím tlačítkem klikneme na node „HingeJoint“ a klikneme na „Copy“. Následně pravím 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 hotovy.



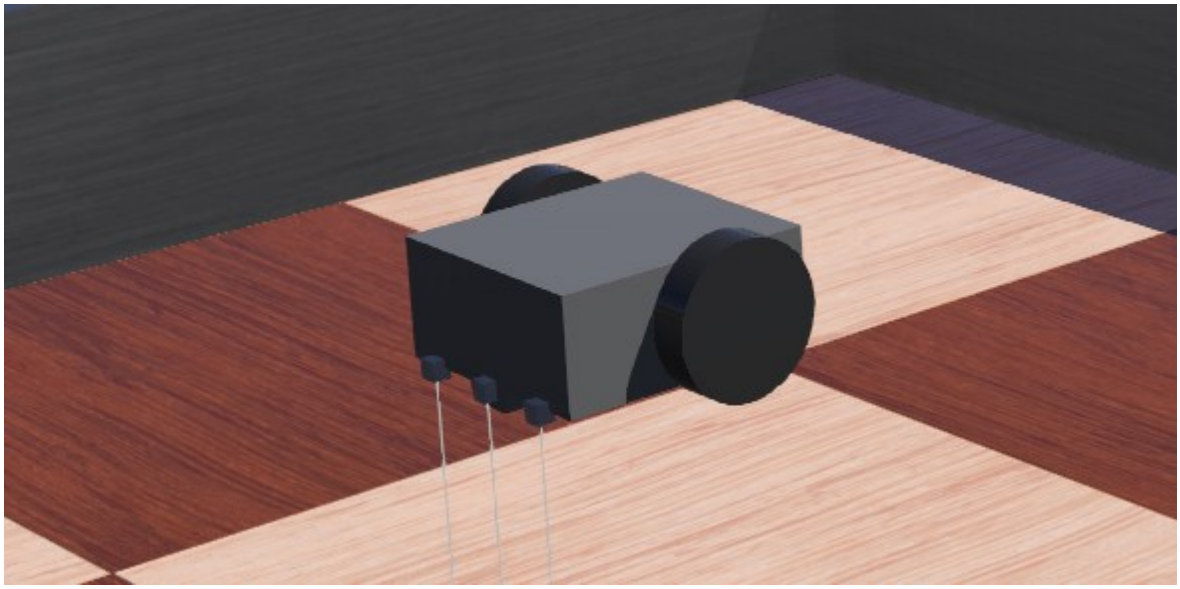
Obrázek 12 Robot s koly

4.1.3 Přední opora

Další částí bude opora v přední části, aby tělo robota bylo zhruba ve vodorovné poloze. Opět si přidáme do nodu „children“ přímo u robota nový node. To uděláme stejně jako v předchozích případech. Tentokrát z nabídky vybereme „Solid“. Následně v tomto vytvořeném nodu přidáme do nodu „children“ nový node stejným způsobem ale teď to bude node „Shape“. Tento node si rozklikneme a stejně jako v předchozích případech do nodu „appearance“ vložíme parametr „PBRAppearance“ a následně upravíme vzheld podle sebe. 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 nebo cokoli co uznáme za vhodné. Následně node „Shape“ můžeme pomocí šipečky zavřít a rovnou nastavit „hitbox“ a fyziku u nody „Solid“. To uděláme stejně jako v předchozích případech jen u parametru „boundingObject“ logicky vybereme název, který jsme dali před chvíli vytvořené krychli. Teď už stačí jen krachli napozicovat tak, aby plnila svůj účel. V nodu „Solid“ nastavíme parametr „translation“ na x = 0.03, y = 0, z = -0.015. Tím je další část dokončena.

4.1.4 Senzory

Poslední částí je vytvoření senzorů díky, kterým se robot bude držet čáry. Opět v nodu „children“ robota přidáme novou součástku naprosto stejnou cestou, jako v minulých případech. Z nabídky vybereme „DistanceSensor“. Jako první u senzoru změníme název. Klikneme na parametr „name“ senzoru a nějak rozumně si ho pojmenujeme. Senzorům není třeba dávat nějaký fyzický model, aby byly funkční ale pro přehlednost je dobré jim nějaký objekt nastavit, abychom přesně věděli, kde se senzor nachází. Nodu „children“ přidáme opět nodu „Shape“. Opět nastavíme parametr, abychom mohli upravovat vzhled a přidáme libovolný tvar. V mém případě jsem opět použil krychli, které jsem nastavil velikost všech stran na 0.005. Opět u nodu „Shape“ nastavíme „DEF:“ například na „SENSOR“. Opět nastavíme „hitbox“ a fyziku a napozicujeme senzor. Doporučuji zobrazit si 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 máme zobrazeno kam senzor směřuje a můžeme ho jednodušeji pozicovat. Parametr „translation“ změníme na x = 0.036, y = 0 a z = -0.015. Parametr „rotation“ změníme na x = 0, y = 1, z = 0 a angle = 1.5708. Nicméně vzhledem k tomu, že stavíme robota na 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 „inra-red“. Následně, obdobně jako u kol, stačí senzor nakopírovat, přejmenovat a upravit pozici tak, aby bylo možné jej použít na sledování čáry. Je potřeba tedy aspoň další dva senzory, každý na jedné straně robota. Tím je celý robot na sledování čáry dokončen, stačí jen vytvořit řídící algoritmus pro sledování čáry a v nodu robota najít parametr „controller“ a následně zvolit onen algoritmus.



Obrázek 13 Dokončený robot

5 ŘÍDÍCÍ ALGORITMY

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

5.1 Jednoduché algoritmy

Jednoduché algoritmy, které na základě podmínek určují kam robot pojede. Základem je porovnání hodnot získaných alespoň ze dvou senzorů. Dále jsem v několika ukázkách použil referenční hodnoty na určení kde se čára nachází vůči robotovi. Tyto referenční hodnoty jsem získal výpisem hodnot, které získali senzory, do konzole. Následně jsem je využil pro lepší řízení robota. Někdy bylo potřeba si pohrát s rychlostí, jakou robot jel vzhledem k tomu, že někdy nestíhal vyhodnocovat hodnoty ze senzorů dostatečně rychle a následně se nedokázal udržet na trati. I tak ovšem mají některé algoritmy občas problém, ale při mých zkouškách fungovali všechny bez větších obtíží. Níže můžete přikládám ukázku jednoho z řídích algoritmů, který se stará o to, aby robot sledoval čáru.

```
//určení polohy čáry na základě referenční hodnoty
  bool lineLeft = (leftSenVal < 250);
  bool lineRight = (rightSenVal < 250);
  //podmínky pro řízení robota
  if((leftSenVal > rightSenVal) && lineLeft)
  {
   leftSpeed = 0;
  }
  else if((leftSenVal < rightSenVal) && lineRight)
  {
   rightSpeed = 0;
  }
```

5.2 Složitější algoritmus

Dále jsem se rozhodl použít běžně používaný regulátor PID, který je velmi přesný a dokáže při správném nastavení pracovat dost efektivně. Chvíli mi trvalo, než jsem našel správné výchozí hodnoty pro regulátor, ale robot poté následoval čáru krásně plynule bez větších obtížích i při vyšších rychlostech. Celkově implementace trvala poměrně dlouho, protože bylo za potřebí najít správné hodnoty několika proměnných, a to vyžadovalo neustále testování a zkoušení, jak se robot drží čáry. Bylo to opravdu zdlouhavé obzvláště, když bylo nutné některé hodnoty nastavit opravdu velmi přesně. Níže přikládám ukázku kódu pro výpočet regulátoru PID.

```
P = Position - NEW_GS * 2;

I = P + pErr;

D = P - pErr;

PID = Kp * P + Ki * I + Kd * D;
```

Samotný kód se skládá ze dvou vytvořených funkcí, když pomineme základní funkci „main“. První funkce se stará o zpracování dat ze senzorů, aby se dali použít pro ve výpočtu pro PID regulátor. Druhá funkce se stará právě o výpočet regulátoru PID. Díky tomu je pak robot schopen sledovat čáru. Následná funkce „main“ už je velice jednoduchá a přehledná. Obzvláště kód v cyklu „while“, který se stará o chod robota, je velice jednoduchý. Níže přikládám ukázku.

```
if(!online){
    if(P == -NEW_GS){
      leftSpeed = -LFM_FS;
      rightSpeed = LFM_FS;
    }
    if(P == NEW_GS){
      leftSpeed = LFM_FS;
      rightSpeed = -LFM_FS;
    }
  }
```

ZÁVĚR

Měl jsem několik úkolů. Prvním bylo prostudování simulátoru Webots. Tahle část mě obzvláště bavila, protože jsem nebyl ničím omezován a mohl jsem si vyzkoušet spoustu věcí. Nakonec jsem u této části strávil možná až moc času.

Dalším úkolem bylo vytvořit prostředí pro některou z disciplín, které se vyskytují na robotických soutěžích. V rámci tohoto bodu bylo i najít nebo vytvořit model robot pro onu disciplínu. Vzhledem k dalšímu úkolu, kterým bylo zpracování návodu pro tvorbu robota ve Webots, jsem se rozhodl pro disciplínu vytvořit kompletně vlastního robota. Konkrétně jsem tedy zvolil disciplínu sledování čáry, takže jsem tvořil robota a zároveň návod pro právě tuhle disciplínu. Tvorba samotného prostředí nebyla nikterak náročná, ale tvorba robota byla trochu zdlouhavá vzhledem k tomu, že jsem tvořil i již zmíněný návod.

Dalším úkolem byla implementace několika zdrojových kódů, které měli definovat chování robota. Vzhledem k tomu že sledování čáry je poměrně jednoduchý úkol, tak nebylo tak obtížné vytvořit pár ukázek. Nejvíce času mi zabralo naprogramování sledování čáry pomocí regulátoru PID, kdy bylo potřeba najít velice přesné hodnoty pro některé proměnné, aby robot fungoval tak jak má.

Celkově mě tato práce bavila a 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 z nové roboty nebo jen součástky, které by byly různě poškozeny a tak podobně. Nebo se jedná 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.

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

[5] C Programming Language, [2022]. *Geeks for Geeks* [online]. Noida: Geeks for Geeks [cit. 2022-04-29]. Dostupné z: https://www.geeksforgeeks.org/c-programming-language/

[6] Explain the history of C language?, 2021. *Tutorialspoint* [online]. Kavuri Hills: Tutorialspoint [cit. 2022-04-29]. Dostupné z: https://www.tutorialspoint.com/explain-the-history-of-c-language

[7] C++ Programming Language, [2022]. *Geeks for Geeks* [online]. Noida: Geeks for Geeks [cit. 2022-04-29]. Dostupné z: https://www.geeksforgeeks.org/c-plus-plus/

[8] 1.3 BRIEF HISTORY OF C++, c2022. *O'Reilly* [online]. Farnham: O'Reilly UK [cit. 2022-04-29]. Dostupné z: https://www.oreilly.com/library/view/object-oriented-programming/9789332503663/xhtml/head-0045.xhtml

[9] What is Java technology and why do I need it?, [2021]. *Java* [online]. Austin: Oracle Corporation [cit. 2022-04-29]. Dostupné z: https://www.java.com/en/download/help/whatis_java.html

[10] History of Java, c2011-2021. *Javatpoint* [online]. Noida: Javatpoint [cit. 2022-04-29]. Dostupné z: https://www.javatpoint.com/history-of-java

[11] Python Tutorial, [2022]. *Tutorialspoint* [online]. Kavuri Hills: Tutorialspoint [cit. 2022-04-29]. Dostupné z: https://www.tutorialspoint.com/python/index.htm

[12] History and Philosophy of Python, [2022]. *Python courses* [online]. Singen: Python courses [cit. 2022-04-29]. Dostupné z: https://python-course.eu/python-tutorial/history-and-philosophy-of-python.php

[13] Language Fundamentals, c1994-2022. *MathWorks* [online]. Natick: MathWorks [cit. 2022-04-29]. Dostupné z: https://www.mathworks.com/help/matlab/language-fundamentals.html

[14] A Brief History of MATLAB, c1994-2022. *MathWorks* [online]. Natick: MathWorks [cit. 2022-04-29]. Dostupné z: https://www.mathworks.com/company/newsletters/articles/a-brief-history-of-matlab.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EPFL Swiss Federal Institute of Technology

OBJ Soubor 3D modelu

PID Proporcionální, integrační, derivační

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY

Obsah přiloženého CD:

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