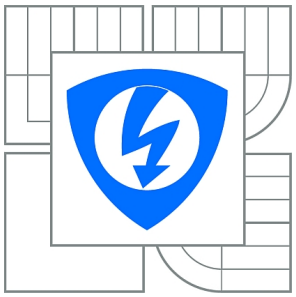
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SPECIALIZOVANÝ INTERPRET JOYSTICKU PRO RDS

SPECIALIZED JOYSTICK INTERPRET FOR RDS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

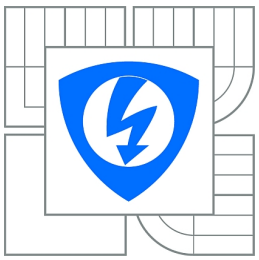
AUTOR PRÁCE — DÁVID MUZIKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, Ph.D.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Dávid Muzika	***ID:***	115242
Ročník:	3	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Specializovaný interpret joysticku pro RDS

POKYNY PRO VYPRACOVÁNÍ:

- Vypracujte přehled robotických simulátorů, ve kterých by bylo možné testovat různé interprety ovládání jednoosého diferenciálně řízeného robota pomocí joysticku;
- seznamte se s prostředím Robotics Developer Studio (RDS) a vypracujte manuál popisující tvorbu scény a objektů;
- popište univerzální interpret joysticku používaný v RDS;
- navrhněte specializovaný interpret joysticku pro úlohu průjezdu bludištěm;
- vytvořte prostředí bludiště v RDS s možností průjezdu jednoosého diferenciálně řízeného mobilního robota;
- proveďte a vyhodnoťte experimenty s univerzálním a specializovaným interpretem joysticku.

DOPORUČENÁ LITERATURA:

Ding D, Cooper RA, Spaeth D.:Optimized joystick controller. Conf Proc IEEE Eng Med Biol Soc. 2004;7:4881-3.

Termín zadání:	7.2.2011	***Termín odevzdání:***	30.5.2011

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

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabýva návrhem a implementací specializovaného interpretu joysticku v prostředí Microsoft Robotic Developer Studio. V práci je popsáno vývojové prostředí Microsoft Robotics Developer Studio a také podrobněji popsaná jeho simulační součást Visual Simulation Environment. V práci je navrženo nekolik specializovaných interpretů pro úlohu průjezd bludiště. Ve výsledku práce jsou vyhodnocené experimenty s univerzálním a specializovaným interpretem.

KLÍČOVÁ SLOVA

VSE, RDS, specializovaný interpret, univerzální interpret

ABSTRACT

The main subjects of this bachelor thesis is implementation of specialized joystick interpret in environment Microsoft Robotics Developer Studio. The work describes environment of Microsoft Robotics Developer Studio and also described in more detail the simulation part the Visual Simulation Environment. The work propose several specialized interpreter for the passage of the maze. As a result, work is evaluated by experiments with universal and specialized interpreter.

KEYWORDS

VSE, RDS, specialized interpred, univesal interpred

MUZIKA, Dávid *Specializovaný interpret joysticku pro RDS*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2011. 45 s. Vedoucí práce byl Ing. Petr Honzík, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Specializovaný interpret joysticku pro RDS“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Děkuji vedoucímu práce Ing. Petru Honzíkovi, Ph.D. za trpělivost, ochotu při řešení problému a za vlídný a přátelský přístup. Také bych chtěl poděkovat Vojtěchu Erbenovi za pomoc a další cenné rady při zpracování mé bakalářské práce

OBSAH

Úvod **10**

1 Simulátory **11**
1.1 Microsoft Robotics Developer Studio 11
1.2 Simbad 11
1.3 Gazebo 12
1.4 anyKode Marilou 12

2 Microsoft Robotic Developer Studio **14**
2.1 Seznam programů a služeb 14
2.2 Spuštění Microsoft Visual Simulation Environment 14
2.2.1 Microsoft DSS Manifest Editor 14
2.2.2 Visual Programming language 15
2.3 Vytvoření základní simulační scény 18

3 Vytvoření simulačního prostředí pomocí VSE **21**
3.1 Editace scény VSE 21
3.2 Vytvoření vlastní entity v VSE 23
3.3 Spuštění simulace a módy zobrazení 26

4 Vytvoření simulačního prostředí pomocí kódu v C# **29**
4.1 Simulace vytvořená pomocí služby 29
4.2 Použité služby 29
4.2.1 MyRobot 30
4.2.2 SimpleDashboard 30
4.3 Vytvoření vlastní entity v C# 30

5 Vytvoření testovacího prostředí **32**
5.1 Vytvoření bludiště 32

6 Joystick v RDS **35**
6.1 Obsluha joysticku v RDS 35
6.2 Návrh specializovaného interpretu pro joystick 35
6.2.1 Základní řízení 35
6.2.2 Integrální řízení 36
6.2.3 Integrální řízení s deaktivovanou osou X 37
6.2.4 Ortogonální řízení 38

7 Dosažené výsledky **40**
7.1 Testování 40
7.2 Výsledky testování 40

8 Závěr **43**

Literatura **44**

Seznam symbolů, veličin a zkratek **45**

SEZNAM OBRÁZKŮ

1.1 MRDS 11
1.2 simbad 12
1.3 gazebo 12
1.4 Marilou 13
2.1 Simulation IRobot manifest 15
2.2 Visual Simulation Environment 15
2.3 Generic Differential drive 16
2.4 Connections 16
2.5 Data Connections 17
2.6 Simulated differencial drive 17
2.7 EntityUI.manifest 18
2.8 EntityUI 19
2.9 Empty Scene 19
3.1 Edit VSE 21
3.2 Move Menu 22
3.3 VSE Edit Mode 23
3.4 New Entity 24
3.5 Construction Parameteres 24
3.6 Shape 25
3.7 Shape Completed 26
3.8 Box 26
3.9 Run VSE 27
3.10 Physic Render 27
3.11 Wireframe 28
4.1 Simulovaná scéna 31
5.1 Simulovaná scéna 34
6.1 Popis os joysticku 35
6.2 Integrální řízeni 36
6.3 Pouze Rz řízeni 37
6.4 Ortogonální řízení 38
7.1 User Dashboard 40

SEZNAM TABULEK

2.1 Ovládání 20
7.1 Test 42

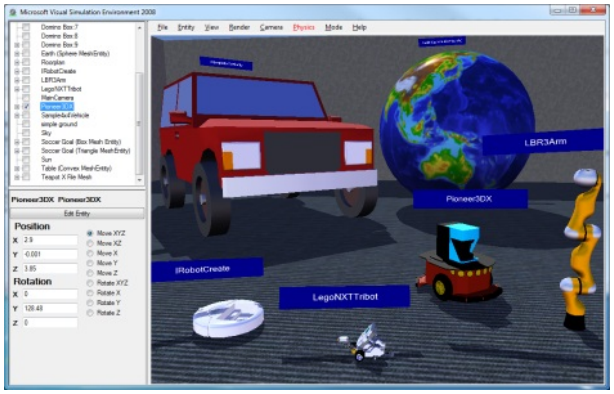
ÚVOD

Specializovaný interpret joysticku je softwarové upravení dat přijímaných od uživatele pro efektivnější a rychlejší řešení specifické úlohy. Specializace v případě zadané úlohy spočívá v úpravě dat přijímaných z ovladače tak, aby bylo pro uživatele jednodušší splnit úlohu projetí bludiště jednoosým diferenciálně řízeným robotem. Specializovaný interpret využívá tří stupňů volnosti joysticku X, Y a R_Z. Pro realizaci bude použit Microsoft Robotic Developer Studio. Práce obsahuje popis tohoto prostředí a návod na práci v simulačním prostředí Visual Simulation Environment (VSE). Bude navrženo několik specializovaných interpretů a testovací dráha, která bude představovat cestu bludištěm. Testovací úlohou bude projet tuto dráhu v co nejkratším čase s co nejmenším počtem kolizí. Uživatel bude otestován jak na univerzálním tak i na specializovaném interpretu a výsledky budou porovnány. Cílem práce je seznámit se s prostředím Robotic Developer Studio (RDS), vypracovat manuál popisující tvorbu scény a objektu a navrhnout a otestovat různé specializované interprety joysticku.

1 SIMULÁTORY

1.1 Microsoft Robotics Developer Studio

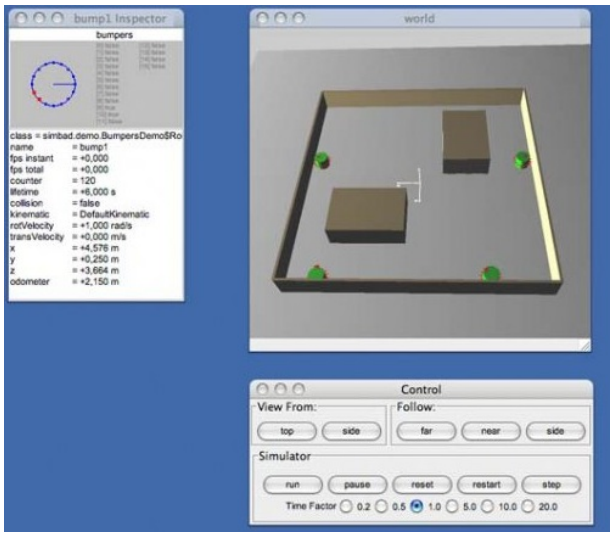
Microsoft Robotic Developer Studio (MRDS) (viz. [4]) je vývojové prostředí pro operační systém Windows. Je určeno pro univerzitní a amatérský vývoj robotických aplikací pro velkou škálu robotů. Skládá se ze dvou programových součásti: z Visual Programming Language (VPL) a VSE. VPL slouží k blokovému programování robotických aplikací. VSE slouží k vytvoření simulační scény a testování robotických aplikací.



Obr. 1.1: MRDS

1.2 Simbad

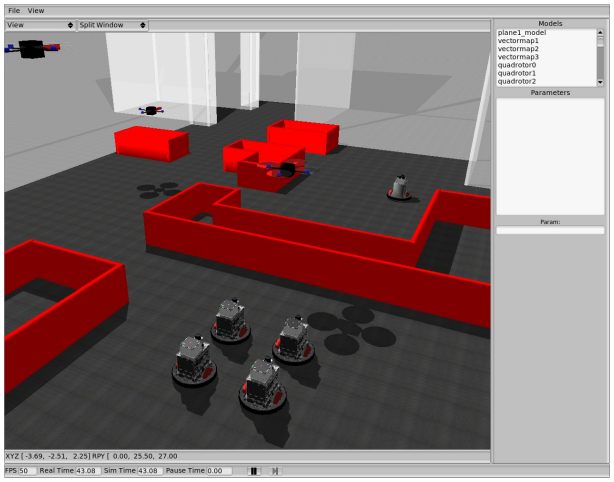
Simbad [6](obr. 1.2) je 3D robotický simulátor využívající prostředí Java. Je určen pro vzdělávací a vědecké účely, programování a výzkumu umělé inteligence a strojového učení. Simbad umožňuje programátorům vytvořit vlastní ovládání robota, upravovat prostředí a používat dostupné senzory.



Obr. 1.2: simbad

1.3 Gazebo

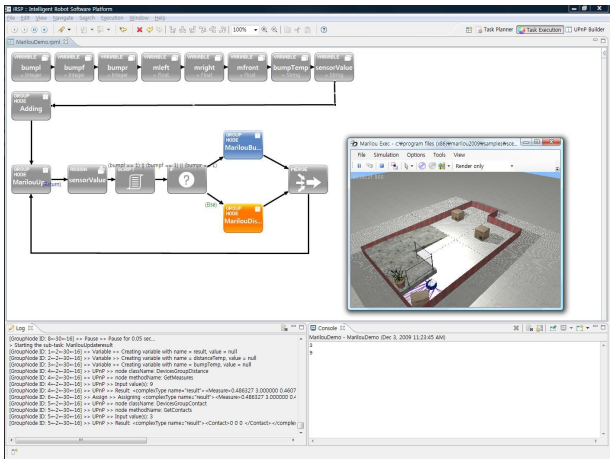
Gazebo [7] je robotický simulátor který dokáže simulovat více robotů a objektů ve venkovním prostředí. Vytváří realistickou odezvu senzorů a fyzickou interakci mezi objekty



Obr. 1.3: gazebo

1.4 anyKode Marilou

anyKode Marilou [1](obr. 1.4) je modelovací simulační prostředí, které simuluje práci robota v reálném světě. Je používán ve výzkumných centrech a také v průmyslu pro simulování různých projektů.



Obr. 1.4: Marilou

2 MICROSOFT ROBOTIC DEVELOPER STUDIO

2.1 Seznam programů a služeb

Visual Programming Language (VPL) je vývojové prostředí založené na grafickém programování, které je použito k vývoji aplikací pro programování a simulování robotů.

Visual Simulation Environment (VSE) umožňuje vytvářet a testovat vytvořené robotické aplikace ve vizuálním engine a s fyzikálním modelem.

Decentralized Software Services (DSS) umožňuje vytvářet programové moduly, které spolupracují s robotem a připojeným PC použitím jednoduchého otevřeného protokolu.

DSS Manifest Editor (DSSME) je nástroj sloužící k vytváření nebo úpravě simulačních scén a dalších manifestů

Concurency and Coordination Runtime (CCR) umožňuje aplikacím vytvářet asynchronní vlákna bez programové úpravy.

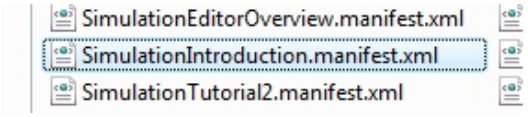
2.2 Spuštění Microsoft Visual Simulation Environment

Tato kapitola čerpá ze zdrojů [5] [3]. V této kapitole jsou popsány základní způsoby spuštění VSE a postupy pro vytvoření základní simulační scény v tomto prostředí.

VSE je prostředí ve kterém se odehrává samotná simulace. Toto prostředí pracuje pod aplikačním rozhraní Directx 9 a fyziku zpracovává Aegia PhysX společnosti Nvidia.

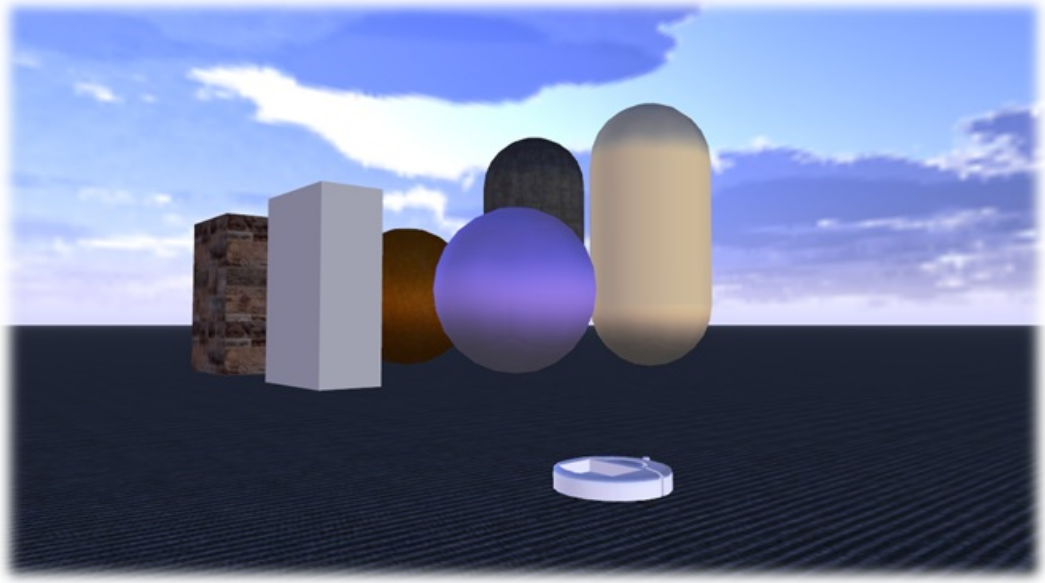
2.2.1 Microsoft DSS Manifest Editor

Jedním ze způsobů jak spustit VSE je použití programu Microsoft DSS Manifest Editor. Tento způsob je rychlý a není zapotřebí mít předem vytvořený projekt ve VPL. Po spuštění Manifest Editoru se zobrazí okno, které slouží k úpravám manifestů. Pro práci je výhodnější otevřít si již vytvořenou prázdnou scénu (prostředí vytvořené ve VSE). Pro úpravu je vhodný IRobot.Create.Simulation.Manifest.xml ve složce s nainstalovaným RDS `\samples\Config` (viz Obr. 2.1)



Obr. 2.1: Simulation IRobot manifest

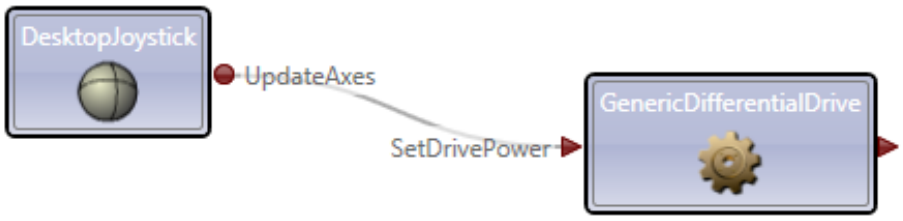
Po načtení tohoto souboru se spustí základní manifest, který slouží k zobrazení scény v prostředí VSE. Po spuštění simulace (tlačítkem F5) se zobrazí prostředí, ve kterém budeme dále pracovat.viz Obr. 2.2



Obr. 2.2: Visual Simulation Environment

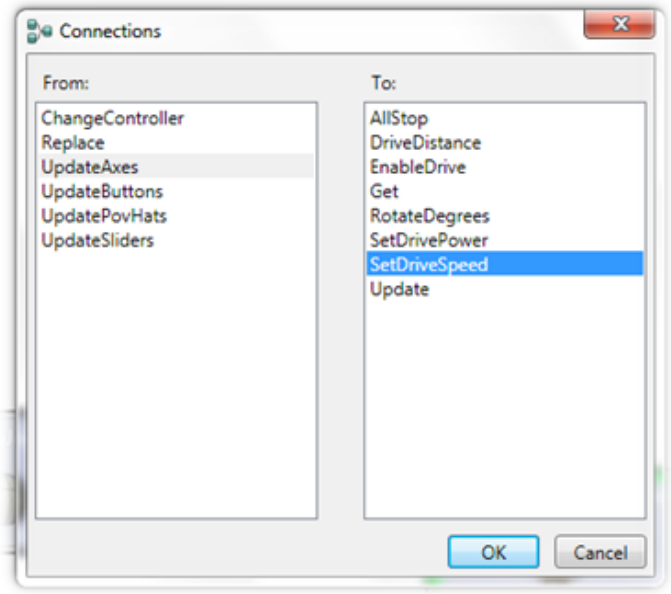
2.2.2 Visual Programming language

Dalším způsobem jak otevřít VSE pro účely úpravy simulačního prostředí je použitím programu VPL. Pro spuštění VSE v prostředí VPL je nejjednodušší vložit dva bloky z nabídky služeb a to DesktopJoystick a GenericDifferentialDrive. Po propojení dle obrázku.2.3 se zobrazí menu. Toto menu slouží k nastavení načítaných dat z joysticku a jejich použití v simulaci.



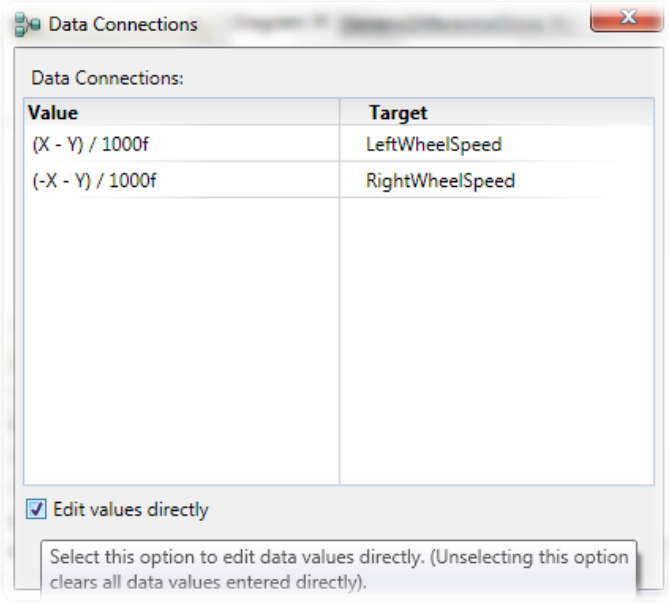
Obr. 2.3: Generic Differential drive

Pro simulaci je vhodné načítat z joysticku změnu os (Update Axes) a tuto hodnotu reprezentovat v simulaci jako rychlost (set drive speed), viz Obr. 2.4



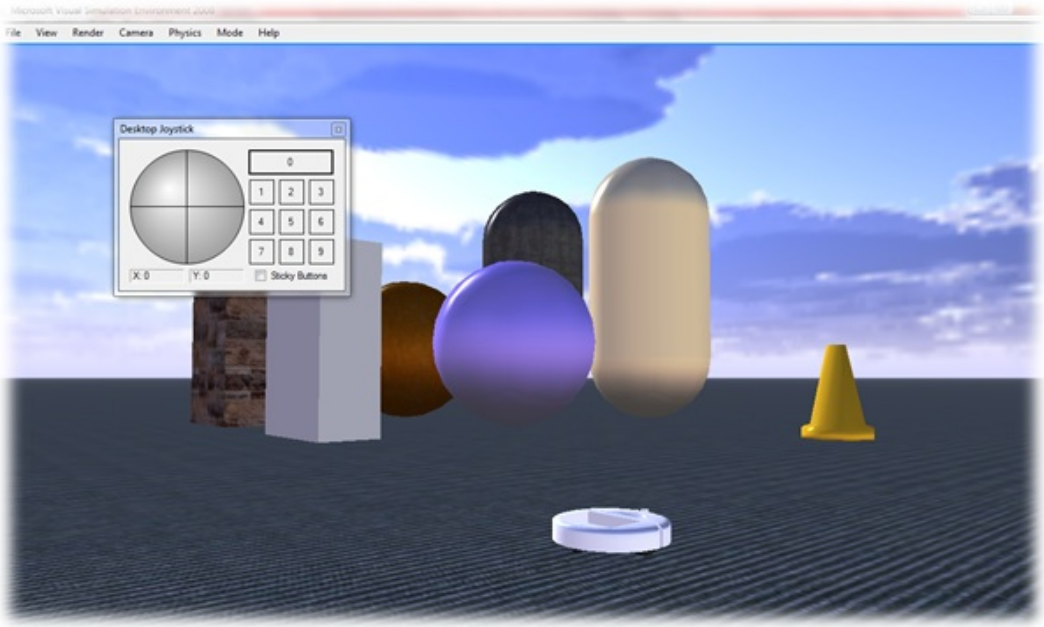
Obr. 2.4: Connections

Dále je nutné nastavit převodní vztah mezi bloky na hodnoty uvedené na Obr.2.5, což se provede v bloku Data Connections. Před spuštěním simulace je nezbytné nastavit manifest, ve kterém se bude simulace odehrávat. Pro výběr manifestu je zapotřebí otevřít nabídku bloku SimpleDifferentialDrive(dvojklikem) a v kolonce Set Configuration je třeba zvolit možnost Use a manifest. Tlačítkem Import Manifest lze vybrat např. IRobot.Create.Simulation.Manifest.xml, jeden z ukázkových manifestů nabízených firmou Microsoft.



Obr. 2.5: Data Connections

Po spuštění,stiskem tlačítka F5, se spustí simulační prostředí VSE. Nyní i s robotem kterého lze ovládat (Obr. 2.6).



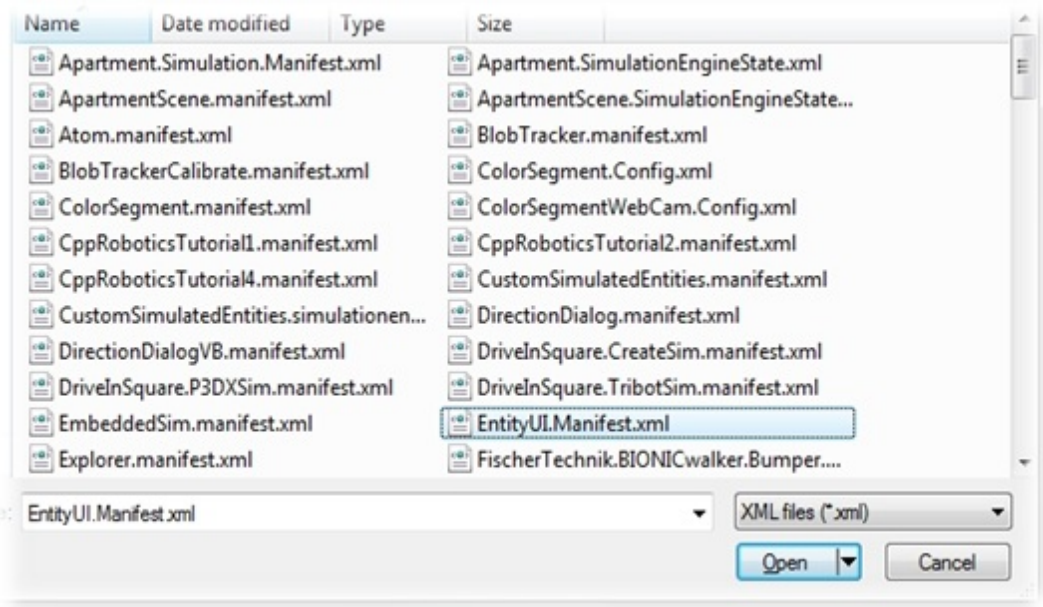
Obr. 2.6: Simulated differencial drive

Díky bloku Desktop Joystick je možné nyní robota ovládat i bez připojeného ovladače (Joysticku, gamepadu). Robota lze ovládat s pomocí klávesnice, nebo myši. Při použití klávesnice tlačítkem „w“ robot akceleruje, tlačítko „s“ robota zabrzdí a

poté reverzuje jeho chod, tlačítka „a“ a „d“ zatáčí s robotem doleva a doprava. K ovládání lze také použít myš, kde pohybem na virtuálním ovladači lze v okně Desktop Joystick ovládat pohyb robota.

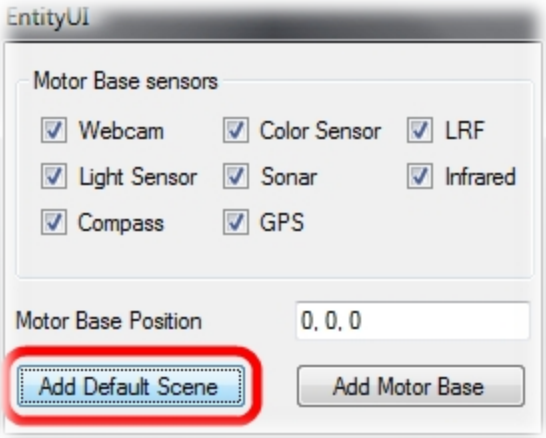
2.3 Vytvoření základní simulační scény

Nejjednodušší způsobem vytvoření základní simulační scény je otevřením souboru `\Microsoft Robotics Dev Studio 2008 R3\samples\Config\Simulation EmptyProject.manifest.xml` v Manifest editoru. Po spuštění se otevře prázdná scéna do které je zapotřebí vložit základní prvky jako jsou podlaha a obloha. Tyto prvky lze do simulace vložit pomocí EntityUI.manifest.xml. Manifesty se v prostředí VSE otevírají rozkliknutím kontextové nabídky v hlavním menu pod položkou File volbou Open Manifest. EntityUI.manifest.xml se nachází `\config\EntityUI.manifest.xml` (obr.2.7).

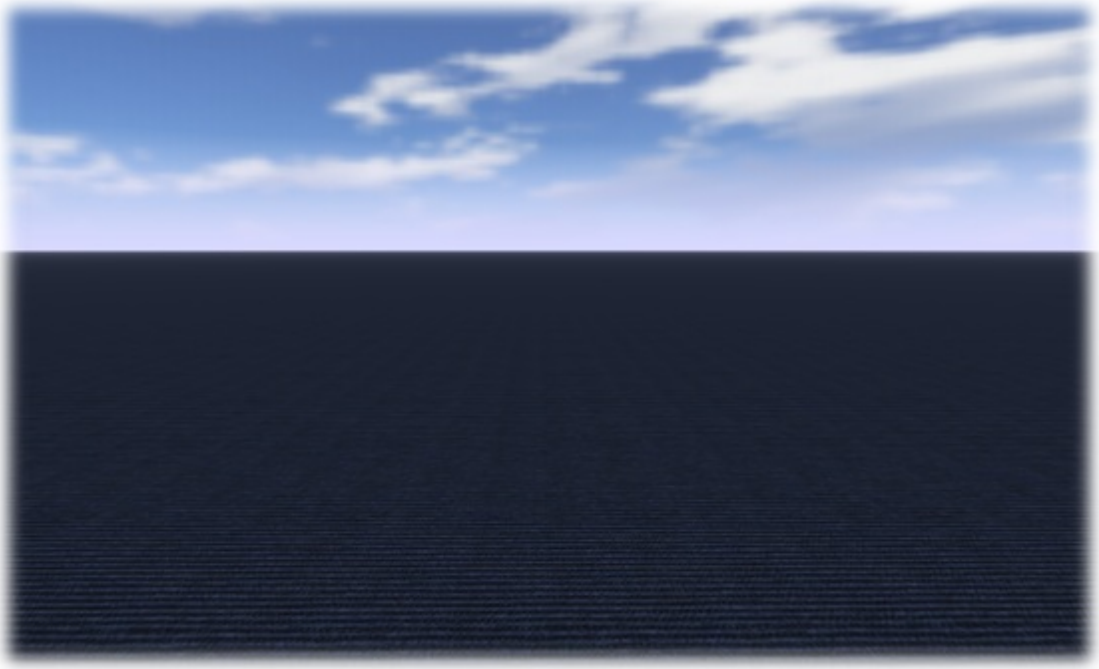


Obr. 2.7: EntityUI.manifest

Po načtení se zobrazí menu EntityUI (obr.2.8). Pro vytvoření základních prvků v simulaci (podlaha, obloha) je zapotřebí vybrat položku Add Default Scene. Přes toto menu je možné přidat do simulace i robota, ale takto vytvořeného robota by nebylo možné ovládat, proto se robot přidává do simulace později. Ve výsledku by v simulační scéně měly být přítomny prvky: Podlaha (Ground), Obloha(Sky) a kamera (maincamera) viz Obr.2.9



Obr. 2.8: EntityUI



Obr. 2.9: Empty Scene

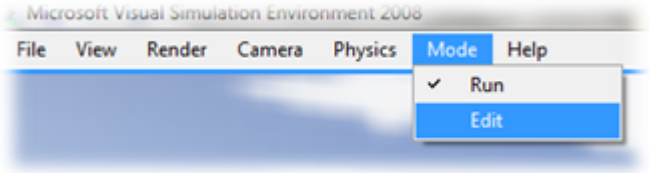
A	šipka vlevo	Num Pad 4	Pohyb vlevo
D	šipka vpravo	Num Pad 6	Pohyb doprava
W	šipka nahoru	Num Pad 8	Pohyb vpřed
S	šipka dolů	Num Pad 2	Pohyb zpět
Q	Page Up	Num Pad 9	Vertikální pohyb nahoru
E	Page Down	Num Pad 3	Vertikální pohyb dolů
Levý shift	pravý shift		Přidržením urychluje pohyb
Home			Reset kamery do počáteční pozice

Tab. 2.1: Ovládání

3 VYTVOŘENÍ SIMULAČNÍHO PROSTŘEDÍ POMOCÍ VSE

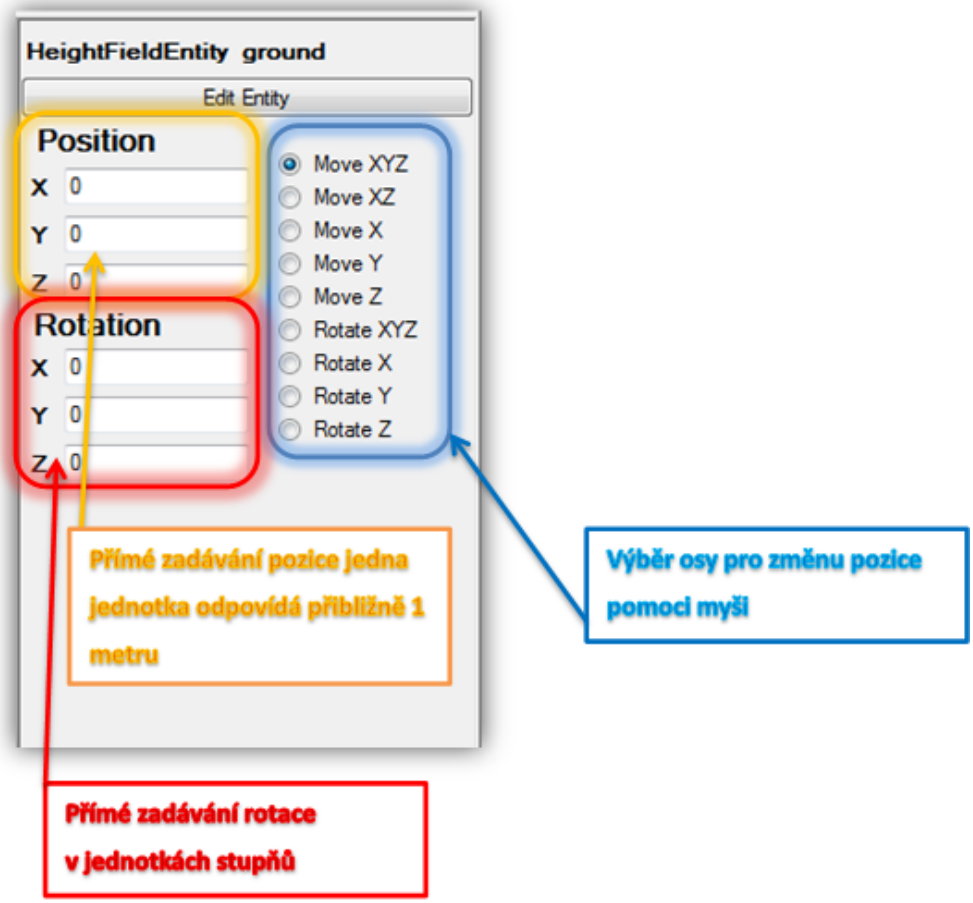
3.1 Editace scény VSE

Prostředí VSE slouží nejenom pro spouštění simulace, ale také k editaci simulace, přidávání objektu a k úpravě jejich fyzikálních vlastností. Pro úpravu objektů v simulaci je třeba simulaci přepnout do tzv. editačního módu. Položka pro přepnutí simulace do editačního modu v menu je MODE→EDIT (Obr.3.1). Po přepnutí do editačního režimu se simulace automaticky pozastaví a objeví se nabídka s již existujícími entitami a nabídka se základními vlastnostmi právě vybrané entity.

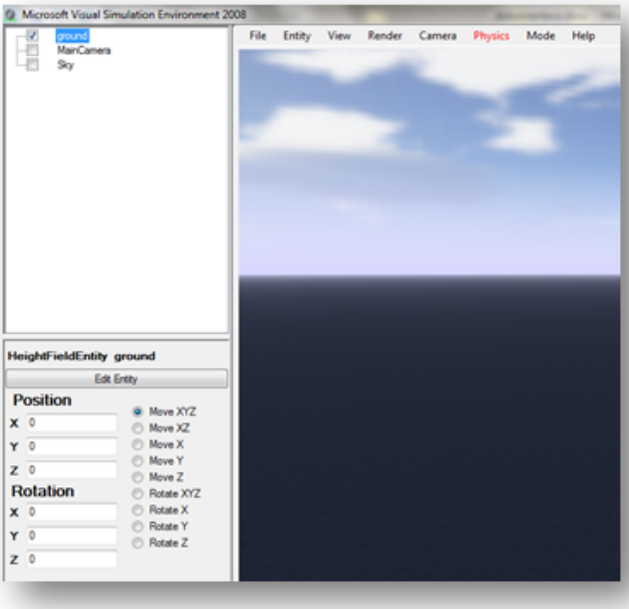


Obr. 3.1: Edit VSE

V levém horním rohu editačního okna je zobrazený seznam již vložených entit v simulačním prostředí. Při označení některé z vložených entit se zobrazí v levém dolním rohu nabídka, která zobrazuje aktuální pozici právě vybraného objektů a možnost změny jeho pozice (obr.3.2). Pozici objektu lze měnit dvěma různými způsoby, jedním z nich je přímé zadávání souřadnic v osách x, y, z a rotace ve směru os x, y a z. Dalším způsobem změny pozice objektu pomocí této nabídky je ovládaní pomocí kombinace klávesnicí a myši. Nejprve se volí osa, ve které se bude objekt přesunovat nebo rotovat. Po výběru osy z menu (například Move Y) se přidržením tlačítka „CTRL“ a pohybem myši pohybuje právě vybraným objektem ve zvolené ose (obr.3.3).



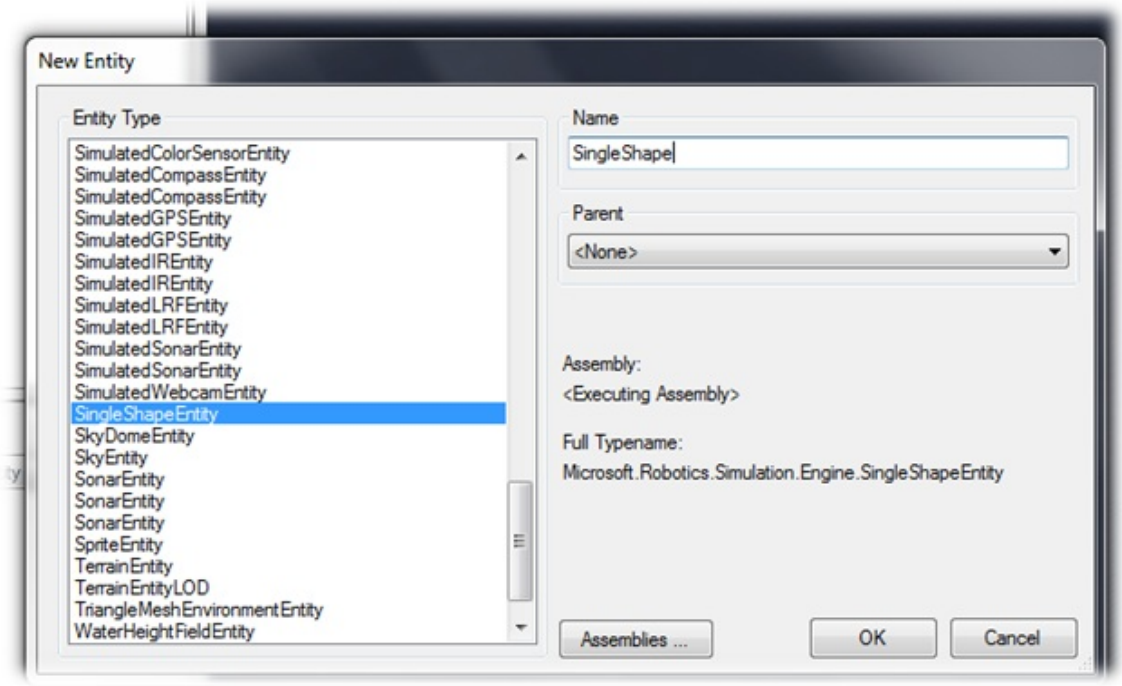
Obr. 3.2: Move Menu



Obr. 3.3: VSE Edit Mode

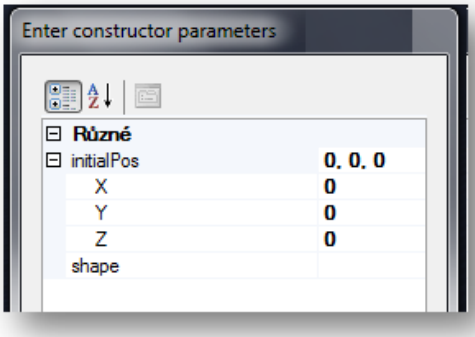
3.2 Vytvoření vlastní entity v VSE

Pro přidání jednoho z elementárních tvarů do simulace, například krychle, se musí do simulace vložit nová entita. Před přidáváním objektů je nutné nejprve vytvořit základní prvky simulace, o tom pojednává kapitola 2.3. Nová Entita se do simulace vkládá v menu Entity→New, po načtení databáze entit je možné vybrat SingleShape-Entity (objekt jednoduchého tvaru) (obr.3.4). Tím je do simulačního prostředí vložen bezrozměrný objekt.



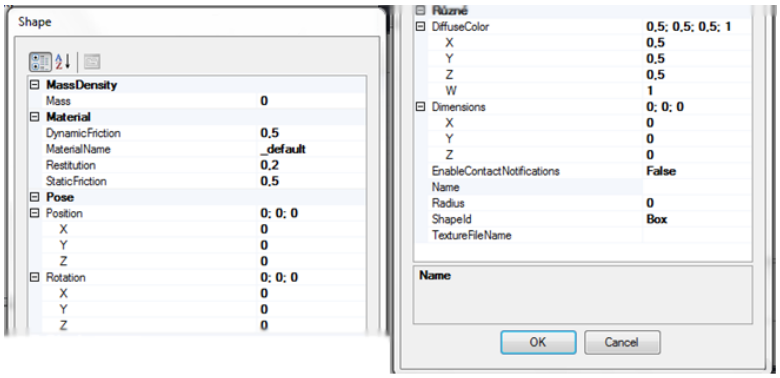
Obr. 3.4: New Entity

Dále je možné upravit jméno objektu a také určit rodiče objektu. Rodiče slouží k snadnému slučování a spojování objektů do jednoho celku. Rodičem hmotného objektu může být pouze hmotný objekt, což objekt ground ani sky není. Po potvrzení se zobrazí okno editace fyzických vlastnosti objektu (Obr.3.5).



Obr. 3.5: Construction Parameteres

Položka construction parameters slouží k zadávání počátečních souřadnic objektu a také určujeme tvar objektu. Menu k nastavování tvaru objektu se nachází v položce shape (Obr.3.6)



Obr. 3.6: Shape

Mass určuje hmotnost objektu, hmotnost je zadávaná v kilogramech.

Dimensions určuje délku stran při použití tvaru Box, při zadání 1 lze srovnat s 1m (nastavit na 1;1;1).

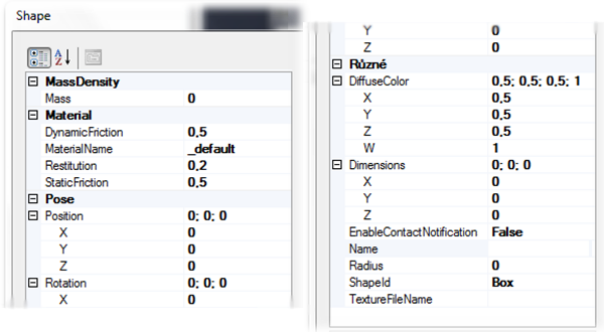
Radius určuje poloměr při použití tvaru Sphere.

ShapeId výběr tvaru objektu.

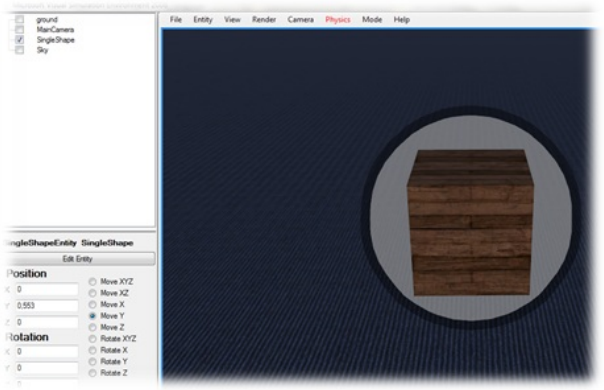
TextureFileName výběr textury objektu, roz-klikneme a z nabídky vybereme jednu texturu.

Po zadání parametrů by měl výsledek vypadat takto (Obr.3.7). Dále je zde možné upřesnit vlastnosti materiálu, například nastavit statické tření položkou StaticFriction nebo dynamické tření objektu položkou DynamicFriction, nebo také restituci materiálu (odrazivost materiálu). Položka position nastavuje posunutí od výchozí pozice (výchylka), rotation nastavuje výchylku od výchozí polohy. Položka diffuze color nastavuje zbarvení objektu v RGB, dimension nastavují proporcionální zvětšení objektu v měřítku směrových os, například při zadání (2;2;2) bude objekt zvětšen ve všech osách v poměru 2:1.

Po potvrzení se vykreslí krychle, která je zapuštěná do podlahy(viz Obr.3.8). Aby tomu tak nebylo, lze upravit pozici v nabídce objektu nebo kliknutím na objekt pravým tlačítkem myši a výběrem moveY. Nyní je možné přidržením tlačítka CTRL a pohybem myši pohybovat celým objektem. Pokud je zapotřebí vytvořit složitější objekty přímo ve VSE, lze je vytvořit vhodným složením základních tvarů a jejich spojení, nebo použít multishape entity v nabídce entit. Je zde ale také možnost importovat objekty vytvořené v 3D modelovacích studiích ve formátu .xml a .dae pro fyzické modely, nebo .obj a .x pro grafické objekty. Tyto objekty se importují do VSE pomocí tlačítka load entity v hlavním menu pod záložkou Entity. Takto lze importovat pouze fyzické objekty typu .xml a .dae



Obr. 3.7: Shape Completed

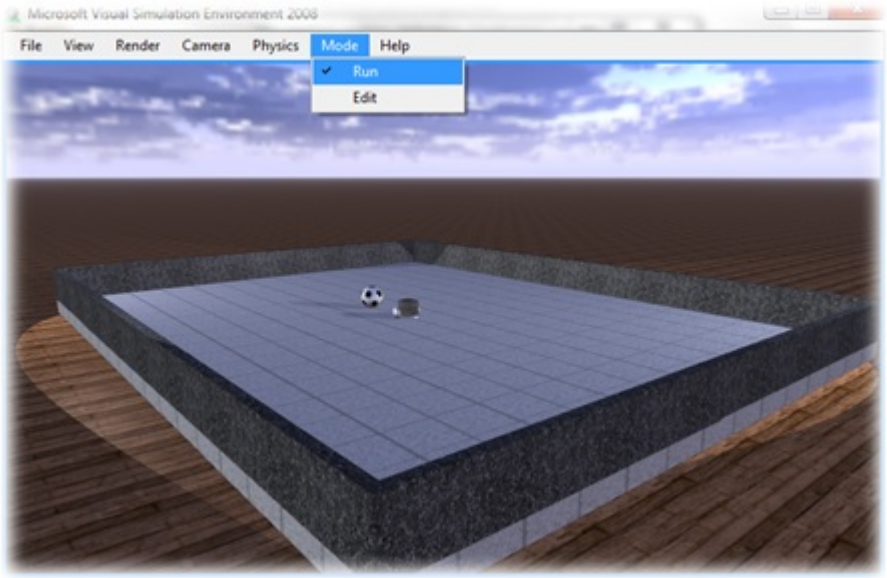


Obr. 3.8: Box

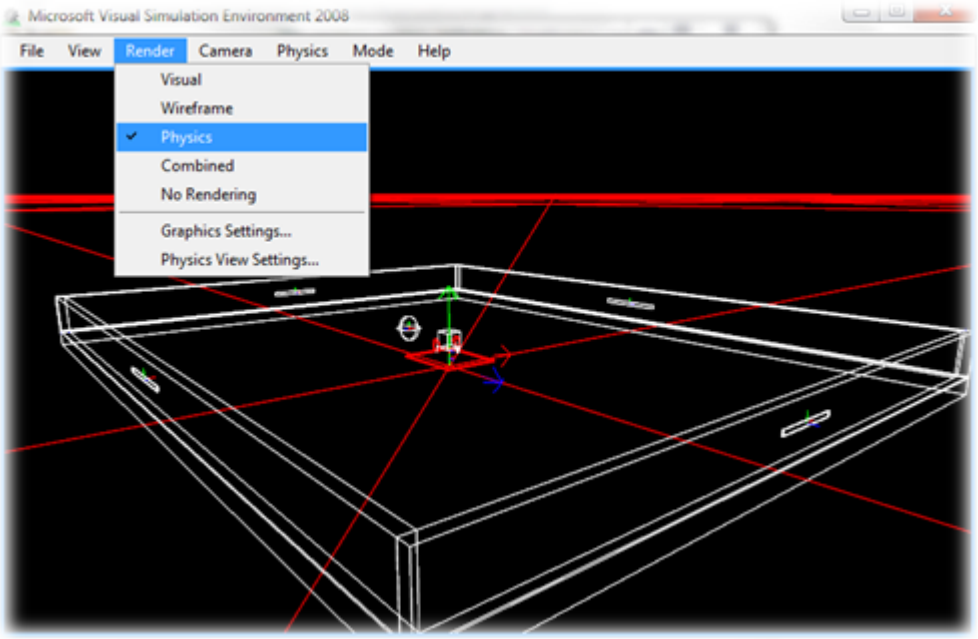
Po spuštění simulace mode→run začnou působit na simulované prostředí fyzikální zákony. Pro přidání robota, který může být řízen, je zapotřebí stejným postupem přidat Entitu jménem MotorBaseWithDrive. Aby bylo možné vytvořenou scénu použít, je nutné uložit právě vytvořený manifest File → Save Scene As do složky s nainstalovaným RDS.

3.3 Spuštění simulace a módy zobrazení

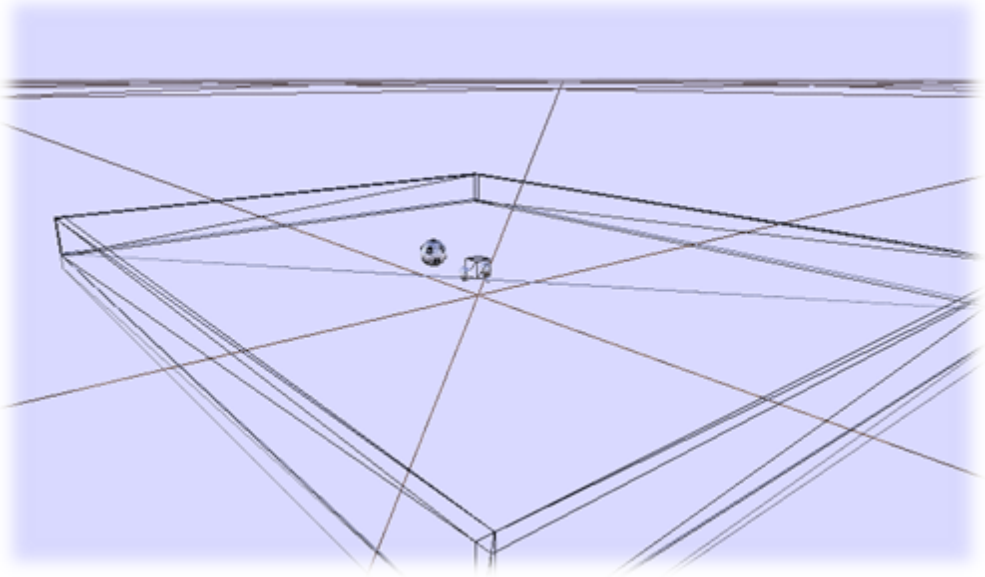
Po spuštění simulace (Obr.3.9) se automaticky aktivuje i výpočet fyziky. Fyzika ovlivňuje všechny aktivní objekty na scéně. Graficky znázorněný výpočet fyziky lze zobrazit pomocí různých zobrazovacích módů. K čistému zobrazení aktuálně působících sil na objekty slouží režim Physics viz Obr.3.10. S takto znázorněnými modely probíhá výpočet vzájemných interakcí objektů. Dalším používaným módem zobrazeni je drátový model (wireframe), zjednodušeně by se dalo říci, že toto zobrazení zobrazuje hrany všech objektů (Obr.3.11), ale tento mód zobrazení neodpovídá fyzickému modelu objektu.



Obr. 3.9: Run VSE



Obr. 3.10: Physic Render



Obr. 3.11: Wireframe

4 VYTVOŘENÍ SIMULAČNÍHO PROSTŘEDÍ POMOCÍ KÓDU V C#

4.1 Simulace vytvořená pomocí služby

Tato kapitola čerpá ze zdroje [2]. Při vytváření projektu ve VPL lze tento projekt zkompilovat jako službu. Kompilaci lze spustit pomocí položky Compile As a Service v nabídce menu Build. Tímto lze sloučit jednotlivé bloky programu do jednoho celku. Tato služba se po kompilaci zobrazí v nabídce služeb a také ji bude možné editovat v jazyce C# v Microsoft Visual Studio. Simulační scénu lze také vytvořit programově v Microsoft Visual Studiu. Nejprve je zapotřebí vytvořit novou službu pomocí Microsoft Visual Studia. Pro vytvoření nové služby je zapotřebí vytvořit novy projekt v Microsoft Visual Studio. Pro programování v prostředí VPL je při vytváření projektu zapotřebí zvolit položku Microsoft Robotics a jazyk C# a také je zde nutné vybrat šablonu DSS Service. Po vytvoření nového projektu je zapotřebí přidat reference. Reference se projektu přiřazují pravým kliknutím na položku References→Add Reference. Aby se program zkompiloval bez chyb, je nutné mu přiřadit následující reference :

PhysicsEngine - přístup do fyzikálního engine
RoboticsCommon - definice fyzikálních charakteristik robotů.
SimulationCommon - reference používané fyzikálním enginem a simulátorem.
SimulationEngine - komunikuje se simulací.
SimulationEngine.proxy
SimulatedDifferentialDrive.2006.M06
SimulatedDifferentialDrive.2006.M06.Proxy
Tyto reference je zapotřebí deklarovat na začátku souboru s implementací tříd

```
using Microsoft.Robotics.Simulation.Physics;
using Microsoft.Robotics.PhysicalModel;
using Microsoft.Robotics.Simulation;
using Microsoft.Robotics.Simulation.Engine;
using engineproxy = Microsoft.Robotics.Simulation.Engine.Proxy;
```

4.2 Použité služby

Pro řešení práce byly použity a upraveny služby MyRobot a SimpleDashboard.

4.2.1 MyRobot

MyRobot spouští všechny součásti potřebné pro běh programu jako i ostatní služby použité v tomto programu. Vytváří všechny entity v simulaci jako podlahu, oblohu, kameru a robota. Služba je použita při vytváření projektu v VPL.

4.2.2 SimpleDashboard

SimpleDashboard je součástí RDS, ale byla upravena pro potřeby práce. Obsahuje funkce jako komunikace s joystickem, zjišťování pozice robota, funkce měření času a výběr typu řízení. Tato služba obsahuje i uživatelské rozhraní, které bylo upraveno pro obsluhu těchto funkci (obr. 7.1).

4.3 Vytvoření vlastní entity v C#

Při vytváření simulačního prostředí je nutné si nejprve vytvořit třídy objektů, které v ní budou použity. Vytvořené třídy reprezentují všechny objekty použité v simulaci. Vytvoření základních prvků prostředí se realizuje pomocí tříd AddSky() a AddGround(), které jsou součástí RDS.

```
1  private void AddBoxKinematic(Vector3 shape, Vector3 position)
2          {
3              BoxShapeProperties cBoxShape = null;
4              SingleShapeEntity cBoxEntity = null;
5
6              cBoxShape = new BoxShapeProperties(10f,new Pose(),shape);
7              cBoxShape.Material = new MaterialProperties("gbox",0.5f,0.4f,0.5f);
8              cBoxShape.DiffuseColor = new Vector4(0.5f,0.5f,0.5f,1.0f);
9              cBoxShape.TextureFileName = "Concrete.jpg";
10
11             cBoxEntity = new SingleShapeEntity(new BoxShape(cBoxShape),position);
12             cBoxEntity.State.Flags = EntitySimulationModifiers.Kinematic;
13             cBoxEntity.State.Name = "greybox"+(++boxes);
14             SimulationEngine.GlobalInstancePort.Insert(cBoxEntity);
15         }
```

Kód 4.1: Vytvoření nepohyblivého materiálu ve tvaru kvádru.

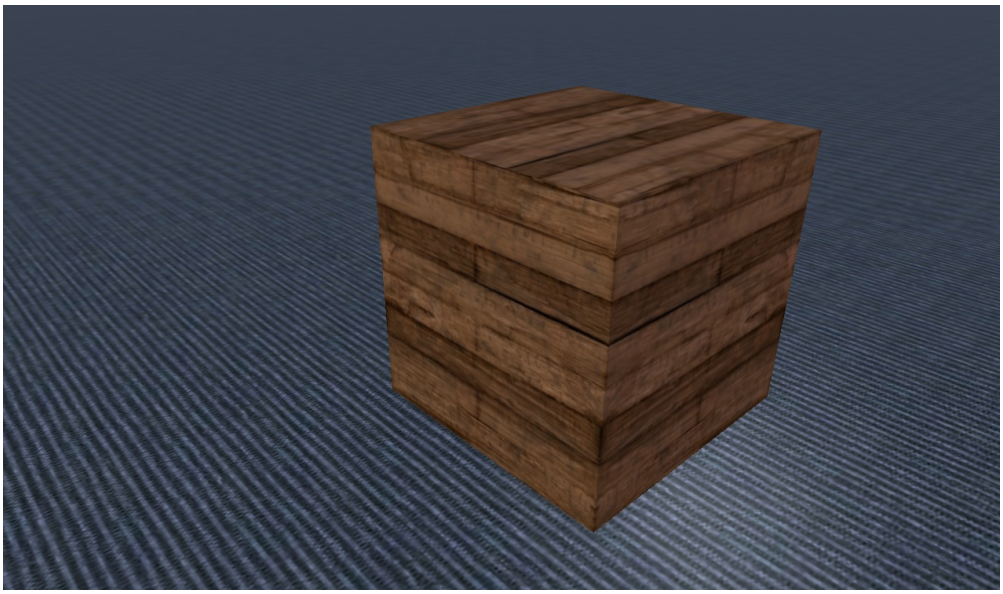
Tato metoda byla použita k vytvoření podlahy v konečné simulaci. Řádek 11 určuje, která entita bude použita pro vytvoření objektu. 12. řádek kódu způsobí, že objekt vytvořený touto třídou bude kinematický. To znamená, že bude fyzicky reagovat na kolize s okolními předměty, ale nebude pohyblivý. Další důležitá část kódu je řádek 13, který pojmenuje vytvořený objekt názvem greybox a číslo které určuje pořadové číslo.

```
1  private void CreateWorld()
2          {
```

```
3                 AddSky();
4                 AddGround();
5                 AddBoxKinematic(new Vector3(1f, 1f, 1f), new Vector3(0f, 0.5f, 0f));
6             }
```

Kód 4.2: Create World

Po vytvoření metody AddBoxKinematic je možné nyní vytvořit simulační scénu stejnou jako v kapitole 3.2



Obr. 4.1: Simulovaná scéna

5 VYTVOŘENÍ TESTOVACÍHO PROSTŘEDÍ

5.1 Vytvoření bludiště

Při vytváření simulačního prostředí je nutné si nejprve vytvořit třídy objektů, které v ní budou použity. Vytvořené třídy reprezentují všechny objekty použité v simulaci. Vytvoření základních prvků prostředí se realizuje pomocí tříd AddSky() a AddGround(), které jsou součástí RDS.

Další třídou použitou na vytvoření stěn je AddBoxKinematicWall. Tato třída je shodná s třídou AddBoxKinematic (viz. 4.1) pouze používá jinou texturu povrchu.

Metoda pro vytvoření robota

```
1  void AddRobot(Vector3 position)
2  {
3       robotBase=CreateMotorBase(ref position);
4       SimulationEngine.GlobalInstancePort.Insert(robotBase);
5  }
6  private DifferentialDriveEntity CreateMotorBase(ref Vector3 position)
7  {
8       DifferentialDriveEntity robotBase=new MotorBase(position);
9      robotBase.State.Flags=EntitySimulationModifiers.LockCenterOfMass;
10     robotBase.ChassisShape.State.DiffuseColor = new Vector4(0.5f,0.5f 0.5f,1.0f);
11     robotBase.ChassisShape.State.TextureFileName="cellfloor.jpg";
12     robotBase.ChassisShape.State.Dimensions=new Vector3(0.08f,0.08f,0.08f);
13     robotBase.LeftWheel.WheelShape.WheelState.LocalPose.Position=new Vector3(-0.05f
           ,0.1f,-0.0f);
14     robotBase.LeftWheel.WheelShape.WheelState.Radius=0.03f;
15     robotBase.RightWheel.WheelShape.WheelState.LocalPose.Position=new Vector3(0.05f
           ,0.1f,-0.0f);
16     robotBase.RightWheel.WheelShape.WheelState.Radius=0.03f;
17     robotBase.CasterWheelShape.State.LocalPose.Position=new Vector3(0f,0.0855f
           ,-0.025f);
18     robotBase.CasterWheelShape.State.Radius=0.015f;
19     robotBase.Rotation = new Microsoft.Xna.Framework.Vector3(0f,90f,0f);
20     robotBase.MotorTorqueScaling=15f;
21     robotBase.Flags=VisualEntityProperties.DoCompletePhysicsShapeUpdate;
22     robotBase.State.Name="MotorBase";
23     drive.Contract.CreateService(ConstructorPort,
24          Microsoft.Robotics.Simulation.Partners.CreateEntityPartner(
25              "http://localhost/"+robotBase.State.Name)
26      );
27      return robotBase;
28 }
```

Kód 5.1: Add Robot

Tato metoda slouží k vytvoření robota, který bude použit v simulaci. Použitý robot musí mít své parametry jako rozměry, rozchod kol, otáčky motoru a různé fyzikální vlastnosti. Pro určení velikosti základny robota slouží řádek 12, ve kterém je určeno že základna robota je krychle o rozměrech 8 centimetru. Dále je nutné určit rozměry a pozici kol vůči základně robota. Toto obstarává řádek 13.-16. Tyto řádky určují,

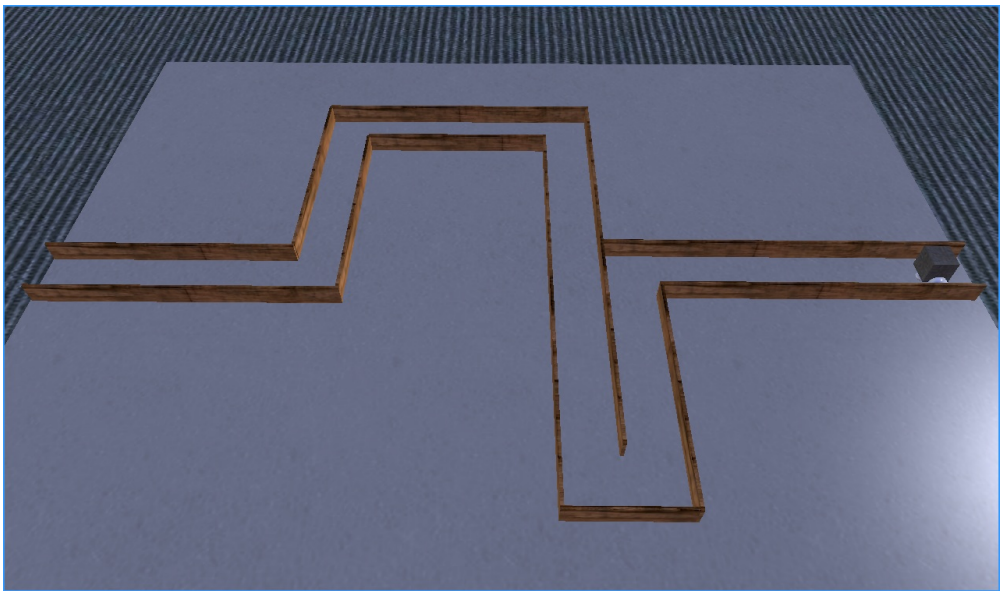
že kola robota budou mít poloměr 3 centimetry a že budou vzdáleny 5 centimetrů v ose X a 1 centimetr v ose Y od středu jeho základny. Kvůli stabilitě robota je nutné do jeho konstrukce zahrnout i samostavné kolečko. Velikost tohoto kolečka a jeho pozici určují řádky 17 a 18. Pro úpravu rychlosti (otáček motoru) robota slouží řádek 20. Toto je bohužel jediné nastavení pohonu robota v RDS, nelze zde nastavit točivý moment, zrychlení ani brzdění tudíž robot svými pohonnými vlastnostmi spíše připomíná tank. Dále je vhodné, kvůli vlastnostem pohonu, uzamknout těžiště, což způsobí, že robota není možné převrhnout.

Po vytvoření všech potřebných tříd jednotlivých objektů simulace je nyní možné jejich vhodným poskládáním vytvořit simulační dráhu (Bludiště).

```
1  private void SetupCamera()
2  {
3   CameraView view = new CameraView();
4   view.EyePosition = new Vector3(0.03f, 2.02f, 1.55f);
5   view.LookAtPoint = new Vector3(0.02f, 1.19f, 1.03f);
6   SimulationEngine.GlobalInstancePort.Update(view);
7  }
8
9  private void CreateWorld()
10 {
11  AddSky();
12  AddGround();
13
14  AddRobot(new Vector3(1.36f, 0.01f, 0f));
15
16  AddBoxKinematic(new Vector3(3f,0.01f,2.5f),new Vector3(0f,0.005f,0f));
17  AddBoxKinematicWall(new Vector3(1.17f,0.07f,0.01f),new Vector3(0.9f,0.05f,-0.09f))
18          :
19          :
20          :
21  AddBoxKinematicWall(new Vector3(0.975f,0.07f,0.01f),new Vector3(-1f,0.05f,0.12f));
22 }
```

Kód 5.2: Create World

Před vytvořením simulované scény je vhodné umístit kameru a určit směr jejího pohledu. To obstarává třída SetupCamera. Tato třída na 4. řádku nastavuje pozici kamery a na 5. řádku směr jejího pohledu. Při volání vytvořených tříd se do jejich parametrů zadávají rozměry objektů, zadávané podle velikosti v jednotlivých dimenzích, a jako druhý parametr je pozice v simulačním prostředí zadávaná jako souřadnice (X,Y,Z). Správným poskládáním objektů správných rozměrů lze vytvořit bludiště (viz. řádek 16 až 21).



Obr. 5.1: Simulovaná scéna

6 JOYSTICK V RDS

6.1 Obsluha joysticku v RDS

Joystick komunikuje s RDS tak, že nastavuje celočíselné proměnné X,Y a Rz (Obr.6.1) v rozmezí -1000 až 1000. S daty načtenými z joysticku pracuje třída input.JoystickState. Tato třída je součástí instalace RDS. V klidové pozici joysticku jsou hodnoty dat přečtených z jeho os nulové, po naklonění dopředu se okamžitě v třídě input.JoystickState změní hodnota public int Y; na -1000.

Obr. 6.1: Popis os joysticku

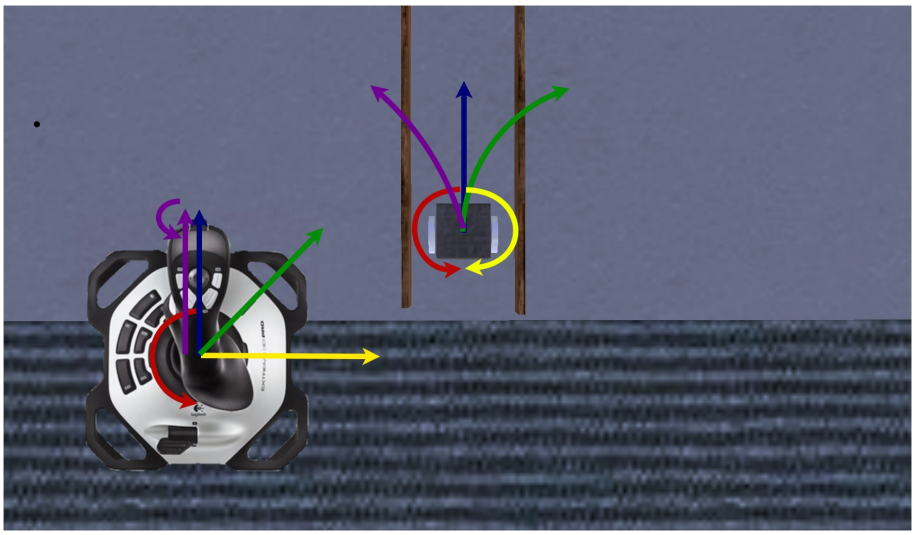
6.2 Návrh specializovaného interpretu pro joystick

6.2.1 Základní řízení

Jedná se o řízení, kdy je robot ovládán přímo nakloněním joysticku. V základním ovládání není obsažena osa R_Z

6.2.2 Integrální řízení

Jedná se o klasické řízení, kdy je rychlost robota ovládána osou Y a pohyb do stran osami X a R_Z s upravenou citlivostí.



Obr. 6.2: Integrální řízeni

Způsob ovládání robota znázorňuje obrázek 6.2. Při naklonění joysticku v kladném směru osy Y (modrá šipka) začne robot akcelerovat v přímém směru. Velikost naklonění osy Y určuje rychlost. Při záporném směru naklonění joysticku začne robot nejprve brzdit a při nulové rychlosti couvat. Při vychýlení joysticku v ose X (žlutá šipka) začne robot rotovat ve směru, ve kterém byl joystick vychýlen, stejnou úlohu má i osa R_Z (červená šipka). Při současném vychýlení osy Y a osy X (zelená šipka) začne robot opisovat kruh, jehož velikost je určena rychlostí robota a měrou vychýlení osy X. Pokud je joystick vychýlen z osy Y a zároveň je s ním i rotováno (Osa R_Z) (fialová šipka), robot se bude chovat stejně jako v předešlém případě. Osy R_Z a X jsou zaměnitelné a v tomto typu řízení vykonávají stejnou úlohu.

Řízení je přizpůsobené na hladké projetí pravoúhlé zatáčky při plné výchylce osy Y. Rovnice 6.1 popisuje výkon levého a pravého kola robota při pohybu vpřed nebo vzad. Rovnice 6.2 popisuje výkon kol při nulové nebo malé výchylce osy Y (rotace na místě).

$|\mathbf{Y}| \geq 10$:

$$Left, Right = Y \pm \frac{X}{3.0} \pm \frac{R_Z}{3.0} \tag{6.1}$$

$|\mathbf{Y}| \leq 10$:

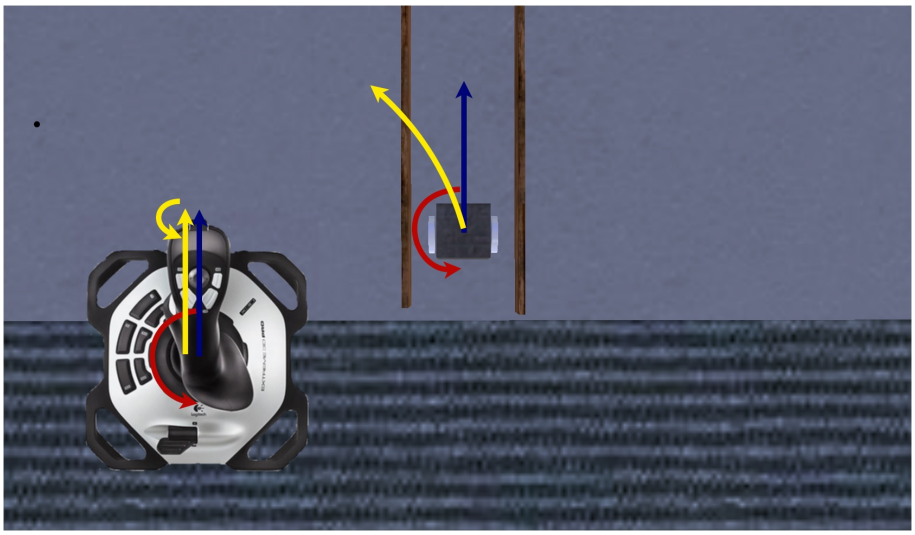
$$Left, Right = 0 \pm \frac{R_Z}{2.0} \pm \frac{X}{2.0} \tag{6.2}$$

Left ..... výkon levého motoru
Right ... výkon pravého motoru
Y ..... výchylka joysticku v ose Y
X ..... výchylka joysticku v ose X
Rz ..... výchylka rotace joysticku

Pokud není osa Y joysticku vychýlena dostatečně, uplatňuje se oblast necitlivosti. Oblast necitlivosti je používaná při ovládání pákovými ovladači, kde je vyžadována přesnost ovládání. Při malém vychýlení osy Y simulovaný robot na tuto osu nebude reagovat, jelikož se může jednat o chybu způsobenou například špatnou kalibrací ovladače.

6.2.3 Integrální řízení s deaktivovanou osou X

Jedná se o klasické řízení, kdy je rychlost robota ovládána osou Y a pohyb do stran osou R_Z s upravenou citlivostí.



Obr. 6.3: Pouze Rz řízeni

Ovládání znázorňuje obrázek 6.3. Řízení je obdobné jako v 6.2.2 s tím rozdílem, že je vyřazena osa X. Při naklonění joysticku v kladném směru osy Y (modrá šipka) začne robot akcelerovat v přímém směru. Při současném vychýlení osy Y a osy R_Z (žlutá šipka) začne robot opisovat kruh, kde je jeho velikost určena rychlostí robota a měrou vychýlení osy R_Z. Při vychýlení joysticku v ose R_Z (červená šipka) začne robot rotovat ve směru ve kterém byl joystick vychýlen.

$|Y| \geq 100$:

$$Left, Right = Y \pm \frac{R_Z}{3.0} \tag{6.3}$$

$|\mathbf{Y}| \leq 100$:

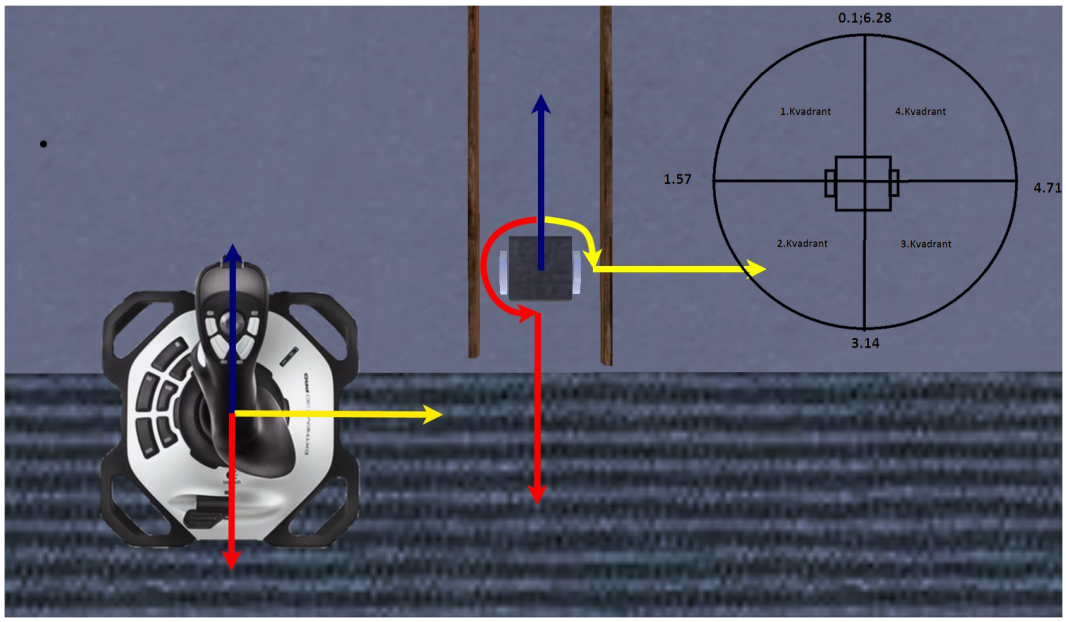
$$Left, Right = 0 \pm \frac{R_Z}{2.0} \tag{6.4}$$

Left ..... výkon levého motoru
Right ... výkon pravého motoru
Y ..... výchylka joysticku v ose Yu
$\mathbf{R_Z}$ ..... výchylka rotace joysticku

Vzhledem k návykům některých uživatelů byl vytvořen tento režim řízení, který odstraňuje jednu z os joysticku. Nepoužívaná osa vnášela nepřesnosti do řízení.

6.2.4 Ortogonální řízení

Jedná se o řízení kde je náklonem joysticku řízen přímo směr kterým se robot pohybuje.



Obr. 6.4: Ortogonální řízení

Ovládání znázorňuje obrázek 6.4. Při naklonění joysticku v kladném směru osy X (žlutá šipka) se robot nejprve natočí tímto směrem a pokud je vyhodnoceno, že robot je správně natočen, začne pomalu zrychlovat vpřed. Obdobně se robot chová při naklonění do všech čtyř os. Otáčky kol robota jsou závisle na směru, kam se má robot otočit a z jakého kvadrantu. Rovnice 6.5 až 6.7 popisují výkon přidělovaný kolům robota při pohybu vpřed (modrá šipka) v závislosti na kvadrantu, do kterého je robot natočen. Obdobné rovnice platí i při pohybu do ostatních směrů.

Y $\geq$ 500, 1.kvadrant:

$$Left, Right = \pm \left(\frac{975}{\frac{3.14}{2}}(-0.01 + \Theta) + 25 \right) \tag{6.5}$$

Y $\geq$ 500, 2.kvadrant:

$$Left, Right = \pm 1000 \tag{6.6}$$

Y $\geq$ 500, 3.kvadrant:

$$Left, Right = \pm 1000 \tag{6.7}$$

Y $\geq$ 500, 4.kvadrant:

$$Left, Right = \pm \left(\frac{975}{\frac{3.14}{2}}(6.26 - \Theta) + 25 \right) \tag{6.8}$$

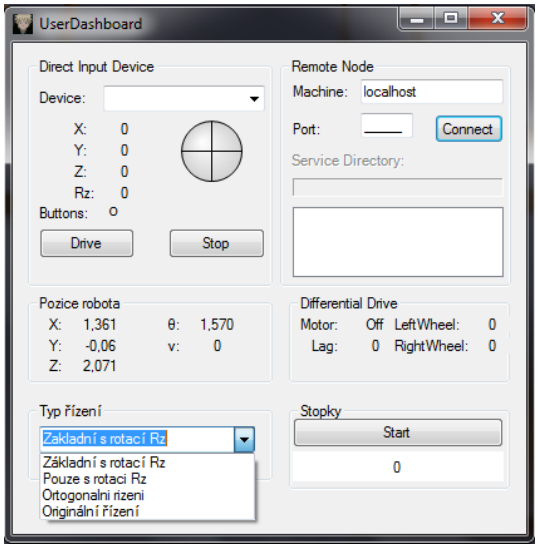
Left ..... výkon levého motoru
Right ... výkon pravého motoru
Θ ..... úhel natočení robota v radiánech

Pokud dojde ke kolizi robota, není možné robota otočit. Kvůli ošetření kolize byla do řízení přidaná reverzace. Reverzace při malé výchylce joysticku ve směru couvání neotočí robotem nýbrž jenom otočí chod kol.

7 DOSAŽENÉ VÝSLEDKY

7.1 Testování

Testování spočívá v projetí bludiště (obr. 5.1) s různými typy řízení (viz obr.7.1). U každé testované osoby je zaznamenáván čas potřebný k projetí bludiště, počet kolizí robota se stěnami bludiště a zkušenost s joystickem. Po kliknutí na tlačítko start (viz obr.7.1) je program připraven měřit čas. Časomíra se spustí automaticky při jakémkoliv pohybu joysticku a ukončuje jí uživatelem sepnutím prvního tlačítka joysticku.



Obr. 7.1: User Dashboard

7.2 Výsledky testování

V Tabulce 7.1 jsou shrnuty výsledky testování. Z výsledků je patrné, že při použití jakéhokoliv specializovaného ovládání se výrazně sníží počet kolizí a zkrátí čas potřebný k projetí bludiště. Při použití Integrálního řízení bylo dosaženo nejkratších časů. Integrální řízení bylo preferováno osobami, které již měli zkušenost s pákovými ovladači. Pokud bylo integrální řízení použito osobami, které nemají zkušenost s pákovými ovladači, docházelo k častým kolizím a tím pádem i zhoršení času průjezdu. Při použití ortogonálního řízení trvalo projetí dráhy sice delší čas oproti integrálnímu řízeni, ale docházelo k výrazně menšímu počtu kolizí. Toto řízení bylo preferováno nováčky, kteří neměli zkušenosti s pákovými ovladači.

Ortogonální řízení sice v průměru nedosahuje nejlepšího času průjezdu dráhy, ovšem při srovnání směrodatných odchylek je patrné, že dosahuje nejmenších odchy-

lek. Při srovnání ortogonálního řízeni s originálním řízením je zřejmé, že průměrný čas projetí dráhy je o 28 s delší u originálního řízení a toto řízení má také o 23 s větší směrodatnou odchylku. Při stejném srovnání integrálního řízení s originální je průměrný čas projetí dráhy o 32 s delší u originálního řízení a také jeho směrodatná odchylka je 9,03 s větší. Rozdíl průměrných časů projetí dráhy mezi integrálním řízením s deaktivovanou osou X a originálního řízení je o 29 s delší v případě originálního řízení a jeho směrodatná odchylka je také o 18,32 s větší.

Při porovnání počtu kolizí u jednotlivých řízení nejlepších výsledků dosahuje ortogonální řízení s průměrem 0,64 kolize na jedno projetí dráhy se směrodatnou odchylkou 0,77. Podobné výsledky dosahovaly integrální řízení a integrální řízení s deaktivovanou osou X kde průměrný počet kolizí byl 3,5 a 3,89, ale rozdíl jejich směrodatných odchylek byl 4,5 a 2,08 čili dvojnásobný. Nejhorších výsledků v počtu kolizí bylo dosahováno originálním řízením kde průměrný počet kolizí byl 18,36 se směrodatnou odchylkou 9,55.

Tab. 7.1: Test

Osoba č.	Zkušenosti	Čas průjezdu [s]				Počet kolizí			
		Univerzální interpret	Specializovaný interpret			Univerzální interpret	Specializovaný interpret		
			Integrální	Pouze Rz	Ortogonální		Integrální	Pouze Rz	Ortogonální
1	Zkušený	44	32	———	48	7	3	—	0
2	Nezkušený	58	62	———	53	15	9	—	0
3	Nezkušený	117	———	50	47	34	—	4	1
4	Nezkušený	105	67	57	52	33	6	4	1
5	Zkušený	66	25	34	42	23	1	3	0
6	Zkušený	61	25	43	40	17	0	2	0
7	Zkušený	51	35	54	48	4	1	6	2
8	———	65	36	61	51	10	0	5	1
9	Zkušený	69	31	33	43	19	1	2	0
10	Zkušený	62	40	37	52	13	0	1	2
11	Nezkušený	135	84	54	54	27	14	8	0
Průměr		75,81	43,66	47,06	48,15	18,36	3,50	3,89	0,64
Směrodatná odchylka		28,16	19,13	9,84	4,82	9,59	4,50	2,08	0,77

8 ZÁVĚR

Cílem této bakalářské práce bylo důkladně se seznámit s prostředím RDS a navrhnout v něm specializovaný interpret joysticku pro ovládání jednoosého diferenciálně řízeného robota.

V první kapitole byl vypracován přehled robotických simulátorů, ve kterých by bylo možné testovat různé interprety ovládání jednoosého diferenciálně řízeného robota. Při řešení semestrální práce byl vybrán program Robotics Developer Studio společnosti Microsoft. K tomuto programu byl vypracován manuál popisující tvorbu simulační scény a objektů v simulaci.

Během řešení této práce bylo navrhnuta testovací dráha reprezentující bludiště. Toto bludiště bylo využíváno pro návrh a testování jednotlivých specializovaných interpretů.

V práci byly navrženy tři specializované interprety joysticku (Integrální, Integrální bez osy X, ortogonální řízení). Integrální řízeni bylo navrženo na základě originálního řízení s přidáním rotační osy R_Z a s upravenou citlivosti jednotlivých os. Také bylo navrženo Integrální řízení bez osy X. Ortogonální řízení bylo speciálně navrženo pro úlohu průjezdu bludiště. Jedná se o řízení kdy směr naklonění joysticku určuje přímo jeden ze čtyř možných směrů pohybu robota v bludišti.

Při testování průjezdu bludiště s použitím jednotlivých interpretů byl sledován čas potřebný k průjezdu dráhy a počet kolizí se stěnami bludiště. Z tabulky 7.1 plyne, že nejstabilnějších časů průjezdu bylo dosaženo s ortogonálním řízením, které také vyniká nejmenším počtem kolizí při obsluze jakýmkoliv uživatelem (zkušeným nebo začátečníkem). Nejlepšího času bylo dosaženo integrálním řízením, ale toto řízení dosahovalo dobrých výsledků pouze při použití osobami, které již měli nějaké zkušenosti s pákovými ovladači.

Souhrnně lze říct, že při použití ortogonálního řízení je dosaženo nejmenšího počtu kolizí s časy, které měly malou směrodatnou odchylku, tudíž je pro tuto úlohu nejvhodnější.

LITERATURA

[1] Anykode: *anyKode Marilou*. [online], [cit. 2011-5-25].
Dostupné z WWW: <http://www.anykode.com/index.php>.

[2] Erben, Vojtěch: *Adaptabilní interpret joysticku pro RDS*. Bakalářská práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2011, vedoucí práce byl Ing. Petr Honzík Ph.D.

[3] Kuparowitz, Tomáš: *Inteligentní řízení robotického fotbalisty*. Bakalářská práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2011, vedoucí práce byl Ing. Petr Honzík Ph.D.

[4] Microsoft: *Microsoft Robotics Developer Studio*. 2008, [online], [cit. 2011-5-25].
Dostupné z WWW: <http://www.microsoft.com/robotics/>.

[5] Microsoft: *Simulating The World With Microsoft Robotics Studio*. 2011, [online], [cit. 2011-5-25].
Dostupné z WWW: <http://msdn.microsoft.com/en-us/magazine/cc546547.aspx>.

[6] Sourceforge.net: *Simbat Project Home*. [online], [cit. 2011-5-25].
Dostupné z WWW: <http://simbad.sourceforge.net/>.

[7] Sourceforge.net: *The Player Project — Download The Player Project software for free at SourceForge.net*. [online], [cit. 2011-5-25].
Dostupné z WWW: <http://sourceforge.net/projects/playerstage/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

MRDS Microsoft Robotic Developer Studio

RDS Robotic Developer Studio

VSE Visual Simulation Environment

VPL Visual Programming Language