VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÉ PROSTŘEDÍ PRO SIMULACI ROBOTŮ

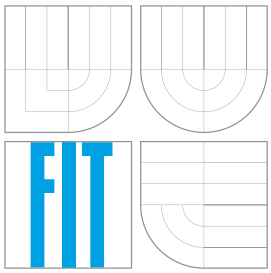
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR BINKO

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÉ PROSTŘEDÍ PRO SIMULACI ROBOTŮ

GRAPHICAL ENVIRONMENT FOR ROBOT SIMULATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR BINKO
AUTHOR

VEDOUCÍ PRÁCE — Ing. FRANTIŠEK ZBOŘIL, Ph.D.
SUPERVISOR

BRNO 2007

Abstrakt

Práce popisuje nástroj Microsoft robotics studio. Jeho logiku a simulační prostředí. Dále hovoří o simulaci a modelování. Popisuje různé vlastnosti simulace, rozdělení a tvorbu modelu. V další části hovoří o robotice a obsahuje popis robota a jeho využití. Nakonec je popsán vytvořený program, který generuje scény pro Microsoft robotics studio.

Klíčová slova

Microsoft robotics studio, model, modelování, robot, robotika, scéna, senzory, simulace, simulační prostředí, stavba robota, systém

Abstract

The work describes the Microsoft Robotics Studio tool, its logic and simulation environment. It also deals with simulation and modeling. It describes various features of simulation and model division and creation. In the next part the work deals with robotics and it contains the description of a robot and its use. Last there is the description of a program I created which generates scenes for the Microsoft Robotics Studio.

Keywords

Microsoft robotics studio, model, modeling, robot, robotics, scene, sensors, simulation, simulation environment, robot composition, system

Citace

Petr Binko: Grafické prostředí pro simulaci robotů, bakalářská práce, Brno, FIT VUT v Brně, 2007

Grafické prostředí pro simulaci robotů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Zbořila, Ph.D.

.....
Petr Binko
12. května 2008

Obsah

Kapitola 1

Úvod

Po celá staletí byla většina lidí odsouzena k namáhavé manuální práci. Není divu, že se člověk snažil svou práci všelijakými vynálezy usnadnit. To vedlo k rozvoji techniky. Tak jak se vyvíjela, objevily se první napodobeniny člověka - androidy švýcarských mistrů Piera a Henry Drozů (18. stol).

Po věku mechaniky k rozvoji robotů přispěla elektrotechnika. Rok 1920 je v robotice zásadním mezníkem, protože český spisovatel Karel Čapek ve svém díle R.U.R poprvé použil slovo robot. Toto slovo se tak stalo nejznámějším českým slovem na světě.

Ve 20 století se začínají objevovat první praktické aplikace, které spadají do oblasti robotiky. Jednalo se především o průmyslové roboty, kteří byli využíváni tam, kde byla práce pro člověka nebezpečná nebo příliš namáhavá. Tak se začalo robotů využívat například v automobilovém průmyslu nebo pro manipulaci s radioaktivními či jedovatými materiály.

Po roce 1980 se roboti zdokonalují pomocí zabudovaných čidel a pomocí počítačového vidění. V roce 1995 se objevuje první chirurgický robot a v roce 1997 je na Marsu vysazen robot Sojourner. Zhruba ve stejné době jsou založeny mezinárodní organizace Federation of International Robot-soccer Association (FIRA) a RoboCup, které organizují soutěže v robotickém fotbale. Cílem těchto organizací je urychlení výzkumu v oblasti robotiky.

Zdá se, že konečným cílem robotiky je sestrojení robota, který by dokázal téměř vše co člověk. K tomuto cíli ovšem vede ještě dlouhá cesta. Nicméně robotický výzkum již dosáhl užitečných výsledků. Například pomůcky pro pohybově postižené nebo vývoj tzv. exeskeletonů, což jsou zařízení, které na sebe člověk obléká, aby několikanásobně zvýšil své fyzické schopnosti, především sílu. K robotickému výzkumu patří také dálkové řízení strojů na principu telepresence. Při takovémto řízení získává člověk, který robota řídí informace v podobě vhodné pro smysly člověka (zrak, hmat, sluch), takže si připadá jako by opravdu byl v prostředí, kde se robot nachází. Takto řízení roboti by mohli být velkým přínosem například pro hasiče nebo záchranáře [6].

Při vývoji každého robota je důležitou součástí jeho simulace. Zjišťování jak bude robot reagovat na různé vlivy prostředí (gravitace, vlhkost, tření různých materiálů). Pro tyto účely, vyvinula společnost Microsoft software, známí jako Microsoft Robotic studio.

Má práce je založena na tomto nástroji tím způsobem, že můj software pro něj bude generovat scény typu robotický fotbal nebo hledání cesty v bludišti. Ačkoli MS Robotic studio umožňuje vlastní tvoření scény a objektů, práce v něm je náročnějšího charakteru. Můj nástroj by měl zajistit, aby si scénu mohl vygenerovat i uživatel, který se nechce zabývat prostředím, ale pouze samotným robotem.

Na začátku své práce bych se rád zmínil o Robotic studiu, co umí, jak pracuje a jaké jsou jeho největší výhody. Dále bych rád napsal něco o simulaci, jaké jsou její principy a

jak simulace probíhá. V další kapitole se budu zabývat robotikou. Jak se k ní přistupuje a jeké typy robotů existují. Také bych rád napsal, co to robot je a z jakých částí se nejčastěji skládá. Na závěr kapitoly bych uvedl, kde se robotů využívá. V poslední kapitole bych napsal něco o svém programu. jak pracuje a jak jsem řešil základní problémy. V závěru bude shrnutí mojí práce a výsledky programu.

Kapitola 2

Microsoft Robotic Studio

2.1 Co je Microsoft Robotic Studio?

Microsoft robotic studio (dále jen MSRS) je postaveno kolem platformy windows. Umožňuje vývojářům poměrně snadno a rychle budovat robotické aplikace. Vývojové prostředí umožňuje vyvíjet software pro mnoho hardwarových zařízení. Od robotů připojených k počítači přímo (za pomoci sériových portů, bluetooth, USB apd.) až k robotům, kteří obsahují procesor uvnitř své architektury.

Programovací model MSRS, může být aplikován na rozličné hardwarové platformy. Toto umožňuje uživateli využívat znalostí, které nabyl na jiných platformách. Programovací rozhraní může být použito pro vývoj aplikací pro roboty, využívající 8, 16 nebo 32-bitový procesor a to jak jednojádrový tak i dvojjádrový.

MSRS umožňuje interakci s roboty pomocí windows nebo rozhraní založeném na webu. Lze vyvíjet aplikace, které umožňují monitorovat a ovládat robota vzdáleně, pomocí webového prohlížeče a posílat mu příkazy, které využívají webové technologie jako HTML formuláře nebo javascript.

Vyvíjené aplikace je možné simulovat ve 3D prostředí, jehož základem je $AGEIA^{TM}$ $PhysX^{TM}$ technologie od společnosti AGEIA technologies. Ta umožňuje simulovat fyziku reálného světa.

Vývojový balíček společnosti Microsoft obsahuje jednoduché, ale výkonné nástroje pro simulaci robotických aplikací. Tyto aplikace lze vyvíjet v mnoha jazycích včetně těch v Microsoft Visual Studiu, jako C# nebo VB.NET. Lze využít i skriptovacích jazyků jako Microsoft Iron Python.

2.2 Základní stavební prvky runtimu MS Robotics Studia

2.2.1 CCR - Concurrency and Coordination runtime

Poskytuje programovací model, založený na zasílání zpráv, který obsahuje silné řídící prvky. Tyto prvky umožňují koordinaci zpráv bez užití manuální správy vláken, zámků nebo semaforů. CCR je odpověď na potřeby aplikací, které jsou zaměřeny na služby. Model ulehčuje správu asynchronních operací a řeší problémy souběžnosti, využití paralelního hardwaru a manipulace s dílčí chybou.

Díky CCR je možné navrhovat aplikace tak, že její jednotlivé programovací části mohou být volně spojeny, to znamená, že tyto části mohou být vyvíjeny odděleně. Tento přístup umožňuje vývojářům nový pohled na návrh aplikace a ulehčuje řešení souběžnosti [4].

CCR je samostatná knihovna .NET přístupná z jakéhokoliv jazyka využívajícího .NET 2.0 CLR (Common Language Runtime).

2.2.2 DSS - Decentralized Software Services

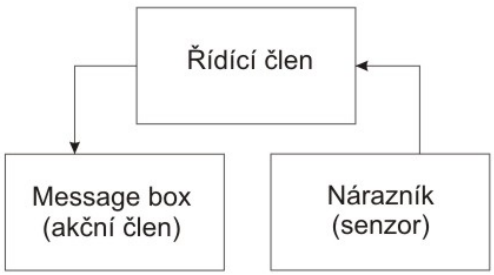
Poskytuje jednoduchý, na služby zaměřený, aplikační model, který kombinuje klíčové aspekty tradiční webové architektury (REST) s trochou webových služeb. Aplikační model definovaný pomocí DSS staví na modelu REST tak, že odkrývá služby jejich stavem a množinou operací nad tímto stavem. Poté rozšiřuje aplikační model poskytnutý službou HTTP o strukturovaná manipulační data, schopnost ohlásit událost a stavbu služby.

DSS je obzvláště vhodné pro tvorbu aplikací, jako složení služeb bez ohledu na to jestli běží ve stejném uzlu nebo na internetu. Výsledkem je vysoce flexibilní, přesto však jednoduchá platforma, pro psaní široké škály aplikací. DSS využívá HTTP a DSSP jako základ pro interakci se službami. DSSP je jednoduchý protokol založený na SOAP (Simple Object Acces Protokol - slouží pro výměnu zpráv, založených na xml, po síti), který poskytuje podporu pro obsluhu strukturovaného stavu a pro událost, vyvolanou jeho změnou. DSSP je využíváno pro manipulaci se službami a z toho důvodu společně s HTTP tvoří jednoduchý, stavem řízený aplikační model.

DSS je postaveno nad CCR a nespoléhá na žádné jiné komponenty v MSRS. Poskytuje hostitelské prostředí pro správní služby a pro sadu služeb infrastruktury, které mohou být využity pro vytvoření, odhalení, záznam, monitorování a správu bezpečnosti služeb [4].

2.3 Aplikační model MS Robotics Studia

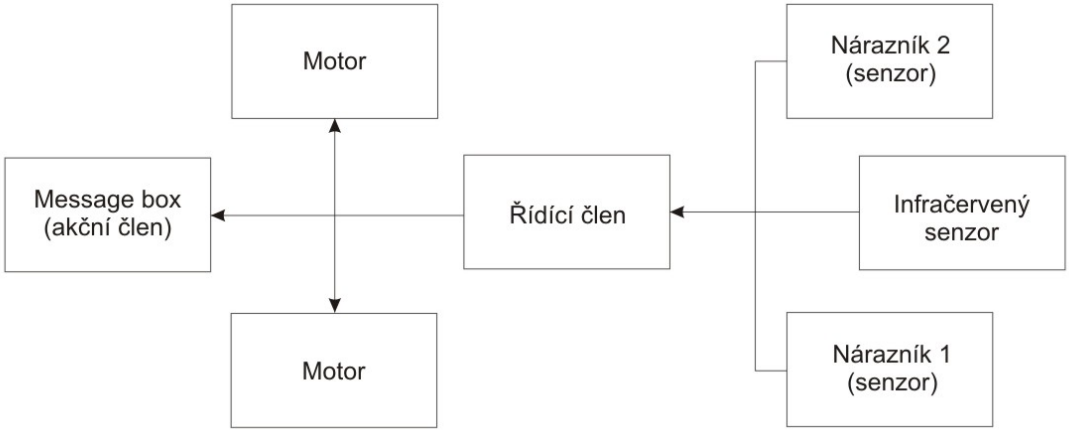
Primárním úkolem robotických aplikací je získat vstup senzorů z různých zdrojů a řídit skupinu akčních členů, kteří budou reagovat na tyto výstupy tak, aby aplikace plnila svojí funkci. Jako příklad takovéto aplikace lze uvést jednoduchý data-flow diagram obsahující nárazník, který zasílá zprávu, pokud narazí. Dále je zde řídící člen, který spojuje tyto dvě věci dohromady.



Obrázek 2.1: Příklad jednoduché aplikace

Tento příklad je jednoduchý, avšak pokud chceme vytvořit složitější operace, je třeba zvýšit počet senzorů, akčních členů a řídících členů, kteří spolu budou komunikovat. V čem je takováto aplikace náročnější si vysvětlíme na dalším složitějším diagramu.

Zatímco každý ze senzorů a akčních členů se podobá prvnímu příkladu, řídící člen nyní musí pracovat s více komponenty. S rostoucí náročností na řízení jednotlivých členů (kteří



Obrázek 2.2: Příklad složitější aplikace

mohou být sami řídícími členy) vzniká několik základních problémů, s kterými musí vývojář počítat.

1. Obsluha senzorů a řízení akčních členů musí být řešena konkurenčně, jinak by mohlo dojít k jejich vyhladovění a senzory by mohly být ignorovaný.

2. Řízení a sestavení jednotlivých částí je kritickým bodem aplikace, zvláště v okamžiku kdy počet senzorů a akčních členů výrazně vzroste.

3. Autonomní a spolupracovní řízení vyžaduje, aby komponenty mohly být distribuovány a zpřístupněny na síti.

Pro řešení těchto základních problémů MSRS využívá DSS (viz. kapitola 2.2.2) a CCR (viz. kapitola 2.2.1). Je samozřejmé, že tento pohled na aplikaci není pouze pro robotiku a že senzory a akční členy nejsou limitovány pouze robotickými komponenty jako motory nebo nárazníky [1].

2.3.1 Slučování webu a webových služeb

Hlavním zaměřením webové architektury je jednoduchost, součinnost a volné spojení. HTTP povyšuje tento model tak, že dovoluje aplikacím využívat jednoduchý, stavový, aplikační model známí jako REST (Representational state transfer). Webové aplikace skrz HTTP demonstrovaly, že tento model pracuje dobře, poskytuje součinnost a je flexibilní natolik aby uspokojil širokou škálu situací. Navzdory úspěchu, jsou však určité aspekty, které neřeší. Konkrétně jsou tři omezení, se kterými se webové aplikace stále potýkají.

1. Žádná podpora pro manipulaci se strukturovanými daty. To znamená, že jediné operace povolené nad zdroji, jsou na jeho úrovni, ale nikoliv v podstruktuře zdrojů. Toto ztěžuje aktualizaci stavu služby, který omezuje její interakci.

2. Žádná podpora ohlašování události. HTTP je dotaz/odpověď protokol a nepodporuje model ohlašování událostí, založený na dodávání dat odběratelům. Přestože jsou

mechanizmy jako RSS (Really Simple Syndication) velmi užitečné, jsou v podstatě založeny na odebírání dat a poskytují malou podporu strukturovaného filtrování.

3. Žádná podpora pro vyjádření typu vztahu mezi jednotlivými komponenty. Jinak řečeno, neexistuje způsob jak vyjádřit, že konkrétní řídící člen by měl být spojen se specifickým senzorem a akčním členem.

MSRS všechny tyto omezení řeší za pomoci DSS (viz. kapitola 2.2.2). Výsledkem je poté mnohem více interaktivní a dynamická aplikace, vytvořená jako kompozice služeb uspořádaných jak v jednom uzlu, tak i napříč sítí.

2.3.2 Manipulace se strukturovanými daty v REST

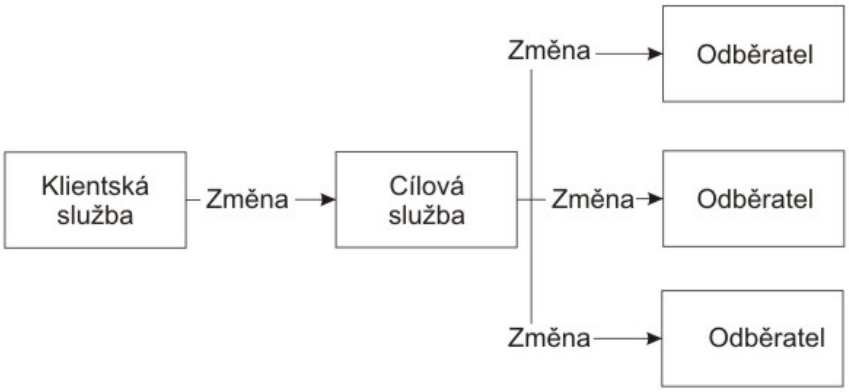
Jelikož je HTTP starší než XML, nesmýšlí nad zdroji jako nad strukturovanými entitami. Výsledkem je, že poskytuje metody, které pracují na úrovni zdroje, jako GET nebo PUT, které zamění celý stav zdroje.

Na druhou stranu webové služby jsou reprezentovány jako XML, které je často popisováno v různých typech systémů včetně XSD, C# nebo Java. Smýšlením o zdroji jako o strukturované entitě, je možné definovat obecné operace, které fungují na částech služby.

MSRS runtime si přisvojil myšlenku, že zdroje mohou být strukturované entity a definuje sadu operací, které umožňují manipulaci se stavem služby (přidání, odebrání, aktualizace). Manipulační stav tohoto stylu není nový, ale kombinací strukturovaného pohledu na webové služby, společně s REST, vzniká flexibilní způsob pro její interakci.

2.3.3 Ohlášení události v REST

Další věc, kterou MSRS přebírá z webových služeb je ohlášení události. Událost je modelována jako něco, co mění stav služby. Například v případě UPDATE, se stav služby změní jako přímý důsledek této operace. Kromě toho, UPDATE sám reprezentuje změnu stavu, takže je přirozené, smýšlet o ohlašování události jednoduše jako o operaci UPDATE.



Obrázek 2.3: Příklad ohlášení události

Tímto vyjádřením ohlášení události, mají odběratelé jednoduchý způsob jak monitorovat poskytovatele, nezávisle na sémantice konkrétní služby. Poskytovatelé zase mohou jednoduše ohlašovat události tak, že veškeré změny stavů zasílají odběratelům.

2.3.4 MSRS model služeb

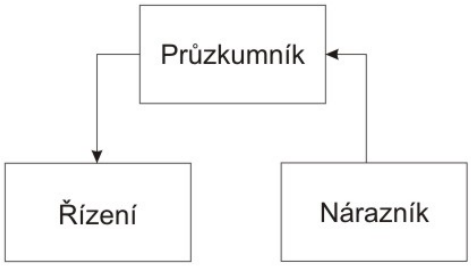
Jak už víme, v runtimu MSRS je aplikace složena z více služeb, kde je každá založena na REST a rozšířena o ohlašování událostí a o práci se strukturovanými daty. Díky vztahu s HTTP, stav každé služby může být monitorován a měněn přímo přes webový prohlížeč nebo skrz jednoduchý protokol DSSP.

1. HTTP umožňuje podporu operací jako GET, PUT, POST nebo DELETE. Tohoto lze využít v uživatelském prostředí, jaké má například webový prohlížeč, za pomoci existující infrastruktury a datových formátů.
2. DSSP přidává podporu pro manipulaci se strukturovanými daty stavu služby a umožňuje dotazy jako UPDATE, INSERT a QUERY. Navíc poskytuje model ohlašování událostí, který je propojen se změnou stavu služby.

2.3.5 Příklad robotické aplikace

Teď, když už víme, jak pracuje aplikační model MSRS a co všechno umožňuje, si uvedeme příklad robotické aplikace, která bude spojena ze třech služeb, které mohou být použity k řízení robota. Služby jsou:

1. Řízení (akční člen), který může hýbat s robotem v různých směrech a různou rychlostí.
2. Nárazník (senzor), který signalizuje kolize s jinými objekty.
3. Průzkumník (řídící člen), který mění rychlost a směr na základě vstupu od nárazníku.



Obrázek 2.4: Příklad jednoduché aplikace

V tomto příkladu je průzkumník odběratelem služby nárazník a dodavatelem pro řízení. Takováto kompozice se nazývá partnerství. Ta reprezentuje vztah, že jedna služba souvisí s druhou. Informace o partnerovi jsou vystaveny jako stav a mohou být zjištěny za pomoci operace LOOKUP, která je součástí DSSP.

2.4 Simulační prostředí MSRS

MSRS obsahuje silný simulační runtime, který uživatelům umožňuje vyvíjet roboty v bohatém virtuálním prostředí s realistickou fyzikou a reprezentačním vykreslením. Základem pro simulační prostředí byly počítačové a konzolové hry, které spoléhají na fotorealistickou vizualizaci, vyspělý fyzikální model a které běží v reálném čase.

2.4.1 Simulační runtime

Simulační runtime MSRS obsahuje tyto základní části:

1. Simulační engine (Simulation Engine Service) - je zodpovědný za vykreslování entit a za běh času pro fyzikální engine. Monitoruje stav celého simulačního světa a představuje přední službu simulace.

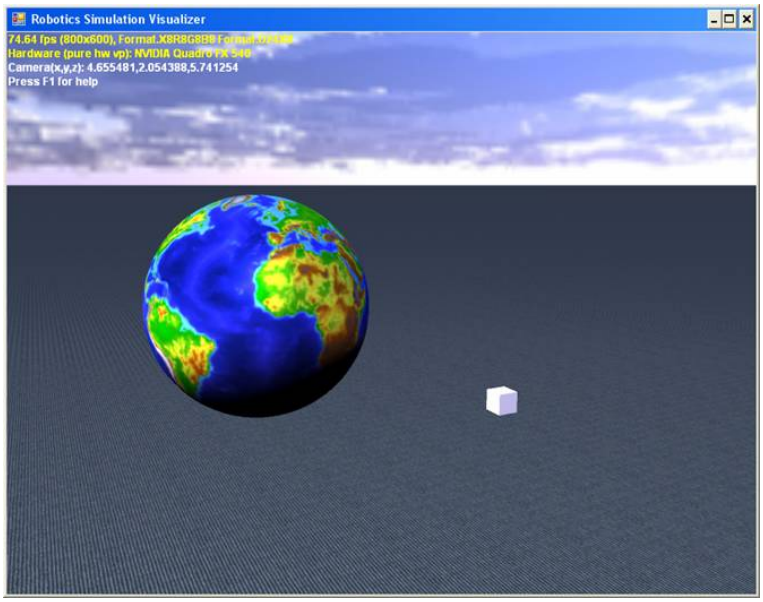
2. Řízený obal fyzikálního enginu (Managed Physics Engine Wrapper) - odděluje uživatele od nízkoúrovňového fyzikálního enginu API a poskytuje stručnější, řízené rozhraní pro simulaci.

3. Původní knihovna fyzikálního enginu (Native Physics Engine Library) - umožňuje hardwarovou akceleraci pomocí AGEIA PhysX Technologie. Ta využívá procesoru, který je dostupný v kartách PhysX.

4. Entita (Entity) - reprezentuje hardwarové a fyzikální objekty v simulaci. Několik entit je předdefinováno v MSRS a umožňuje uživateli vybudovat jednoduché scény.

2.4.2 Simulace v MSRS

Po spuštění nástroje basic simulation environment, který je součástí MSRS, se objeví okno s úvodní scénou.



Obrázek 2.5: Úvodní scéna v simulačním prosředí

Je možné vytvářet nové entity buďto v tomto nástroji nebo za pomoci některých programovacích jazyků, které práci s MSRS podporují, jako C# nebo VB.NET a to jednoduše za pomoci Microsoft Visual Studia. Pro vývoj ve VS, stačí přidat reference na knihovny, které jsou součástí MSRS, do svého projektu (viz. [5]). Je také možné načíst již připravené entity nebo celé scény.

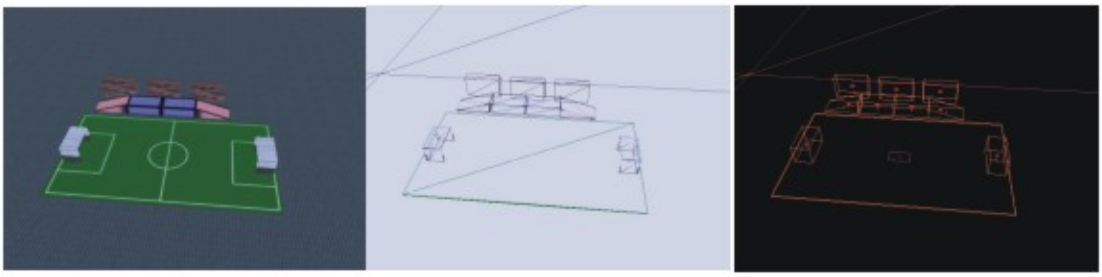
Pro vytvoření nového objektu, stačí vybrat entity → new, což zpustí dialog na vytvoření nové entity. Zde si uživatel může zvolit některý z předem vytvořených typů objektů (TerrainEntity, SingleShapeEntity) a zadat název. Po vytvoření ve scéně přibude objekt a po jeho vybrání se v levém dolním rohu objeví jeho vlastnosti. Základní jsou: pozice (udána třemi hodnotami os souřadné soustavy), rotace (udává natočení objektu v jednotlivých osách) a rozměr. Tyto vlastnosti jsou společné pro všechny základní typy objektů.

V mém programu k bakalářské práci, jsem vytvářel objekty, jejichž tvar je načítán ze souboru s příponou obj. (viz. kapitola 5.1.2). V tom případě, objekt obsahuje vlastnost ConvexMeshShape, v které je možné upravovat pozici těžiště, hustotu, hmotnost, tření materiálu apd. Pokud bych nevyužil načítání tvaru ze souboru obj, musel bych použít některý z předem vytvořených typů, jako BoxShapeEntity (objekt typu kvadr) nebo SphereShapeEntity (objekt typu koule) v kterých jde tyto vlastnosti upravovat také, ale objekt je omezen pouze na zadaný typ (z kvádru nevytvořím kouli apd.).

U těchto vytvořených objektů je nezbytné nastavit hustotu a hmotnost, protože pokud jsou tyto hodnoty nulové, stávají se entity statickými, to znamená, že nemají žádnou fyziku.

Veškeré nastavení vlastností samozřejmě ovlivňuje simulaci. Například při kolizi těžšího objektu s lehčím, jde vidět, jak na sebe objekty reagují, jako by byly v reálném světě.

Lze také upravovat vlastnosti scény. Například v menu physics → settings je možné nastavit gravitaci nebo rychlost toku času ve scéně. Lze také vybrat způsob zobrazení objektů v záložce Render. Je zde na výběr například drátěný nebo fyzikální model.



Obrázek 2.6: Klasické zobrazeni, drátěný model, fyzikální model

Pokud je scéna připravena, stačí v menu Mode, zmáčknou Run a MSRS spustí simulaci. Tu je možné kdykoliv zastavit přepnutím do edit módu.

2.5 Shrnutí kapitoly

V této kapitole jsem chtěl vysvětlit základní prvky a vlastnosti MSRS a také poukázat na věci, které činí tento nástroj velice zajímavým. Zaměřil jsem se na runtime a jeho nejdůležitější části CCR a DSS. Hovořil jsem o tom, co tyto prvky umožňují a v čem jsou výjimečné. Dále jsem psal o aplikačním modelu. Co všechno nabízí a jak řeší některé problémy, které souvisí s jeho funkcí. Nakonec jsem popsal simulační prostředí MSRS a základní úkony, které v něm lze provádět.

Kapitola 3

Simulace

Pokud mám hovořit o simulaci robotů a o simulačních nástrojích, myslím, že by bylo vhodné, v krátké kapitole, pohovořit o tom, co to simulace vlastně je, jak se při ní postupuje a jaké jsou její principy(vycházím přitom z [7]).

3.1 Simulace a základní pojmy

Jestliže studujeme či pozorujeme nějakou skutečnost např. provoz na křižovatce, automobil, pak k této skutečnosti přistupujeme s jistým záměrem např. řízení provozu na křižovatce nebo studium zamořovaní ovzduší výfukovými plyny v jejím okolí. Podle tohoto záměru soustřeďujeme svojí pozornost na některé rysy dané skutečnosti: jsou pro nás důležité jen některé části skutečnosti, některé vlastnosti těchto částí a vztahy mezi nimi. Můžeme říci, že si definujeme na původní skutečnost určitý systém odpovídající danému záměru. Pojem systém je tedy vždy spojen s jistým stupněm abstrakce. Systém by se dal velmi obecně definovat jako soubor prvků, mezi nimiž existují jisté vztahy.

Prvek systému je charakterizován svými vlastnostmi, které mohou být též ve vzájemném vztahu. Předpokládáme, že každá vlastnost může nabývat hodnoty vyjádřitelné nějakým symbolem tak, že různým hodnotám přísluší různé symboly. Některé vlastnosti mají charakter veličiny (např. délka, spotřeba) a jejich hodnota se běžně vyjadřuje číslem. Jiné vlastnosti (např. barva, tvar) pokud jsou důležité, je třeba nějak ohodnotit a jednotlivým hodnotám přiřadit symboly.

Při simulaci robotů se často setkáváme s prvky, u nichž je charakteristické to, že mají vstup a výstup. Na vstup lze přivádět vstupní signál, na výstupu lze pozorovat výstupní signál (např. výstupy různých čidel a senzorů). Chování prvku je určeno závislostí výstupního signálu na vstupním signálu a je ho možno vyjádřit nějakým operátorem transformace (tabulka stavů, tabulka přechodů apod.) Prvky jsou mezi sebou propojovány prostřednictvím vstupů a výstupů a vytváří se tak struktura systému. Nejčastěji předpokládáme, že prvek má jeden výstup, který může být propojen se vstupy jiných prvků i se vstupem vlastním. Prvky, které mají pouze výstup, se nazývají zdroje. Systémy s relací vstup-výstup mezi prvky jsou označovány jako orientované.

V obecnějším případě je vhodné místo vstupů a výstupů hovořit o podnětech a reakcích. Chováním prvků nebo systému označíme způsob jeho reakce na podněty.

Mino prvků, které tvoří vlastní systém, mohou existovat vně systému další prvky, které tvoří okolí systému. Jestliže systém jako takový může existovat pouze v určitém okolí, nazývá se otevřený. Uzavřený systém může existovat v libovolném okolí (jeho chování není

okolím ovlivněno). Jestliže systém reaguje na okolí, děje se tak prostřednictvím vstupů systému a naopak díky výstupům, systém působí na své okolí. Uzavřený systém nemá vstupy ani výstupy.

Stav systému je (abstraktní) veličina vyjadřující minimální množství informace o historii systému, která je nutná k tomu, abychom mohli předpovědět chování systému v budoucnosti. Je patrné, že znalost stavu (spolu se znalostí vstupu) umožňuje určit nejen budoucí hodnoty výstupu ale i stavu systému tj. umožňuje předpovědět chování systému. Proměnné, jejichž hodnoty je nutno znát k určení hodnoty stavu jsou tzv. stavové proměnné. Výběr těchto proměnných není pro daný systém jednoznačný. Systém, jehož stav v okamžiku $t = t_i$ závisí na jeho stavu před okamžikem $t = t_i$ se nazývá dynamický. Statický systém se vyznačuje tím, že jeho stav je neměnný (přičemž ztrácí smysl otázka určení jeho hodnoty).

Deterministický systém je takový, jehož chování v budoucnosti je určeno jednoznačně počátečním stavem v době $t = t_i$ a budoucími hodnotami vstupů (podnětů). U stochastických systémů je na rozdíl od předchozího, chování určeno s jistou pravděpodobností, nikoliv jednoznačně.

Pokud změna stavu systému probíhá spojitě, jedná se o systém spojitý. Systém diskrétní se vyznačuje tím, že v diskrétních časových okamžicích, dochází ke změnám stavu, přičemž mezi libovolnými dvěma sousedními okamžiky se stav systému nemění. Systém se nazývá kombinovaný, jestliže jeho část je diskrétní povahy a zbývající část spojité povahy.

U diskrétních systémů se definuje dále pojem aktivita, jako časový interval, během něhož probíhá jistá akce. Aktivita začíná a končí změnou stavu, tj. událostí. Během aktivity se na prvku vykonávají operace, které připravují změnu jeho stavu a vedou k další události. Posloupnost událostí v čase je proces.

Podle rozlišovací úrovně, kterou příjmem při studiu dané skutečnosti, se může prvek systému stát systémem a naopak systém může být považován za prvek systému nadřazeného. Typy úloh, které se na systému řeší:

1. Řízení: Je známa struktura a chování systému. Hledá se takové působení na vstup systému, pří kterém se dosáhne zadané odezvy (reakce).

2. Analýza: Je známa struktura a má se zjistit chování systému.

3. Identifikace: Experimentálně je možno zjistit chování. Úkolem je odvodit z něj strukturu.

4. Syntéza: Úkolem, je navrhnou takovou strukturu, aby hledaný systém vykazoval požadované chování.

3.2 Model

Při řešení úloh (řízení, analýza, identifikace, syntéza) má velký vliv význam modelovaní a simulace. Chceme-li definovat pojem model, ocitáme se v podobné situaci jako v případě definice pojmu systém. Každý model je totiž systémem. Definovat model má však smysl pouze ve vztahu k originálu, který má být tímto modelem reprezentován. Požadavkem je, aby model a originál byly dva, v jistém smyslu podobné systémy.

Model je účelově definovaný systém, který umožňuje nebo usnadňuje řešení úlohy definované na originálním systému. Snáze se na něm například provádí pozorování a měření, lze s ním provádět experimenty, které s originálem provádět nelze (havarijní situace) apod.

Podobnost dvou systémů, modelu a originálu, tedy posuzujeme podle záměru, s nimž model vytváříme.

Dva různé typy podobností jsou:

1. Podobnost ve struktuře. Dva systémy S1 a S2 jsou si podobné ve struktuře jestliže:

   (a) Každému prvku z S1 lze jednoznačně přiřadit prvek z S2

   (b) Každému vztahu mezi prvky S1 je jednoznačně přiřazen vztah mezi odpovídajícími prvky z S2.

   Pak říkáme, že systém S2 je homomorfní se systémem S1. Pokud uvedené podmínky 1a a 1b platí současně též při záměně S2 za S1, pak jsou S1 a S2 izomorfní.

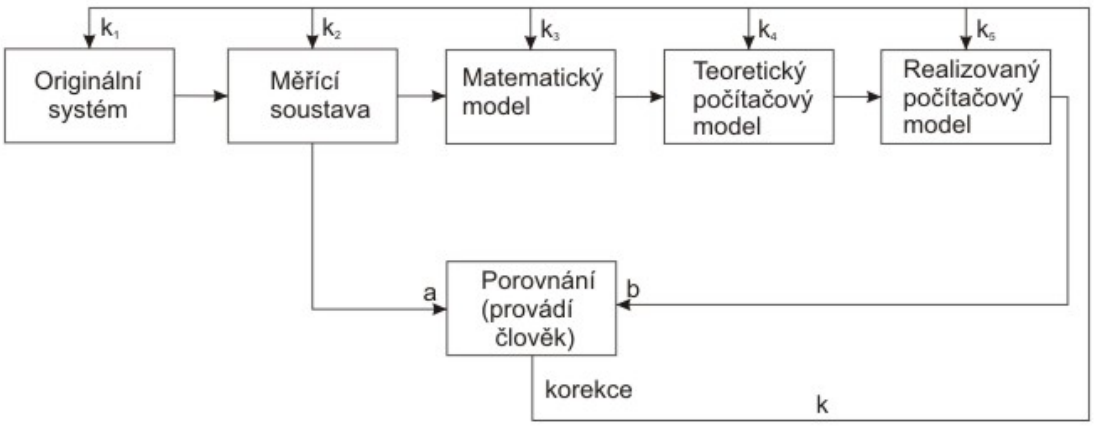
2. Podobnost v chování. V tomto případě se vychází z představy systému jako černé skříňky, kdy se posuzuje hlavně jeho interakce s okolím. Dva systémy S1 a S2 mají stejné chování, jestliže stejné podněty vyvolají u obou systémů stejné reakce (odezvy).

Podobnost chování je běžná zvláště v kybernetice a proto se modely chování někdy nazývají kybernetické (též funkcionální). Je patrné, že dva systémy podobné ve struktuře budou též podobné v chování, ale ne naopak.

Modely lze dělit podle různých hledisek, obdobně jako systémy. Dosti typické je dělení na modely fyzikální a matematické. Je to vlastně dělení podle stupně abstrakce od reálného modelování systému.

Fyzikální modely jsou postaveny podle zákonů fyzikální podobnosti. Typ fyzikálního procesu se u nich zachová (zmenšenina přehrady, zmenšené křídlo letadla).

Matematické modely jsou postaveny podle zákonů matematické podobnosti. Za matematický model můžeme považovat nejen nějaký reálný obvod nebo zařízení, ale též i vlastní matematický popis nebo program. Matematický model tedy je matematický popis nebo program, který může mít formu soustavy diferenciálních rovnic, přenosů, grafů apod. Může popisovat strukturu systému nebo celkové chování.



Obrázek 3.1: Proces modelování

Pokud budeme modelovat na počítači, máme na mysli činnost, která zahrnuje tyto základní etapy:

1. Tvorba matematického modelu
2. Realizace tohoto modelu
3. Hodnocení výsledků, hodnocení modelu, experimentování s modelem.

Pro etapy 2 a 3 se ustaluje termín simulace. Proces modelování a jednotlivé typy simulačních úloh můžeme sledovat na obrázku 3.1.

Originální systém nechť je nějaký konkrétní fyzikální dynamický systém. Blok, označený měřící soustava, představuje činnost, která směřuje k získání informací nutných k formulaci matematického modelu a k hodnocení systému (výstup a). To nemusí být nijak jednoduché, protože se vyžaduje přesná formulace problému, který se má řešit a definování všech měřitelných a nebo pozorovatelných veličin (vstupních, výstupních, stavových). Vychází se přitom obvykle částečně ze znalosti systému, částečně z nějaké hypotézy o něm (která se má právě simulací ověřit) a dále z ujasněného cíle simulace. Navíc je vhodné připomenout, že vhodná volba stavových proměnných je důležitá z hlediska správnosti modelu.

Blok porovnávání porovnává očekávané chování modelu, které je dáno buď přímo měřitelnými výsledky z měřící soustavy nebo je z nich podle nějaké hypotézy odvozeno, se skutečným chováním modelu. Základním elementem v tomto bloku je člověk (řešitel). Člověk vyhodnocuje a srovnává získané výsledky podle kritérií, které si volí v závislosti na typu problému a cíli modelování. Podle výsledků hodnocení pak provádí eventuelní korekce vyznačené na obr. 3.1 jako k_1 až k_5.

Při analýze identifikovaného systému se bude provádět zřejmě korekce k_3 a s tím spojená k_4 a k_5. Vstup zde může sloužit k posouzení správnosti modelu. Při identifikaci se bude rovněž provádět korekce k_3, k_4, k_5 tak, aby signály a a b stále byly dostatečně blízko (ve smyslu zvoleného kritéria).

Při syntéze, kdy je dán model a hledá se originál, přicházejí v úvahu korekce k_1 a k_2.

Pro simulaci je tedy charakteristická zpětná vazba (k) obsahující lidský prvek. Tato vazba vyžaduje pro efektivní průběh simulace bezprostřední komunikaci člověk - počítač, aby bylo možno provádět experimenty s počítačem.

3.3 Zařazení robotické simulace

Pokud se podíváme na předchozí podkapitoly, vyvstává otázka, jak bychom definovaly systém, který reprezentuje nějakou simulaci robota. Podle mého názoru se při simulaci robota narazí téměř na všechny typy systémů. Systém některé části robota, která bude mít propojeny výstupy a vstupy jednotlivých prvků (například výstup dotykových senzorů je vstupem pro řídící prvek a jeho výstup je vstupem pro akční prvek) bude orientovaný.

Pokud budeme simulovat robota, jehož účelem bude průzkum povrchu Marsu, můžeme tento systém nazvat otevřeným, protože závisí na vlivu okolí, zatímco systém robota, který bude vařit různé druhy kafe, může být označen za uzavřený, protože okolí na jeho funkci nemá přímí vliv.

Stav systému nějakého robota by se nejlépe označil vlastností dynamický, protože předchozí stav ovlivňuje stav následující (např. Nemůže dojít ke kolizi s objektem, pokud se předtím robot nezačal pohybovat) a spojitý.

Z pohledu modelu jako reprezentace robota v simulaci, lze vytvářet opět více typů podle toho, jakého robota chceme simulovat.

Pokud budeme modelovat robota, který bude hledat cestu v bludišti, lze využít model, který je se svým originálem podobný pouze v chování (nezáleží na přesné reprezentaci

všech jeho částí). Naopak budeme-li modelovat například robotické rameno, nepůjde jen o chování, ale budeme chtít reprezentovat přesné složení (poměr délek jednotlivých částí ramene, přesné umístění řídících motorů apod.) využijeme tedy model homomorfní.

Simulace robota je tedy velmi pestrá, ale zároveň i poměrně náročná. Naštěstí existuje spoustu nástrojů, které nám simulaci usnadňují, jako například Microsoft Robotics Studio.

3.4 Shrnutí kapitoly

V této kapitole jsem se snažil vysvětlit co to je a jak probíhá simulace a modelování. Popisoval jsem různé druhy systémů a uvedl jsem spojitost s robotickou simulací a napsal jsem, jak probíhá tvorba modelu. I když v simulačních nástrojích jako je MSRS je o tyto věci postaráno automaticky (tím myslím, že uživateli při vytváření entit stačí zadat jeho rozměry a základní fyzikální vlastnosti jako hmotnost, hustotu a o jiné věci které se týkají tvorby modelu, se nemusí starat) a uživatel pracuje pouze s uživatelským rozhraním, myslím, že je dobré vědět co se za simulací skrývá.

Kapitola 4

Robotika

4.1 Přístupy k robotice

Robotika je chápána jako disciplína o vytváření inteligentních strojů integrující několik vědeckých a inženýrských oblastí. Existuje několik základních přístupů k robotice:

1. Teoretická robotika: hledá principy, možnosti a omezení (biologie, psychologie, etologie, matematika, fyzika).
2. Experimentální robotika: ověřuje principy, staví "hračky" (kybernetika, umělá inteligence, inženýrské disciplíny).
3. Průmyslová robotika: navrhuje, staví a používá průmyslové roboty (teorie a instrumentace řízení, elektronika, strojírenské technologie, automatizace a organizace výroby, znalost konkrétní oblasti nasazení robotů).
4. Různá aplikovaná robotika: Navrhuje různé inteligentní stroje pro průmysl i jinam. Např. stoje pro kontrolu kvality ve výrobě (často jsou vybaveny schopností vidět), mobilní roboty se schopností autonomní navigace atd.

4.2 Dělení robotů

Stejně jako lze rozdělit robotiku, je možné rozlišit i několik typů robotů. Základní dvě skupiny jsou [3]:

1. Průmyslový robot: je automatické ústrojí s následujícími schopnostmi.
   (a) Manipulační schopnost
   (b) Automatická činost
   (c) Snadná změna programu
   (d) Univerzálnost
   (e) Zpětné vazby
   (f) Prostorová soustředěnost
2. Kongitivní robot: je kybernetický systém schopný autonomní interakce s reálným prostředím za účelem splnění stanoveného cíle. Má různou míru schopností:

(a) Vnímat a rozpoznávat prostředí (senzorický modul).

(b) Vytvářet a aktualizovat vnitřní reprezentaci prostředí (kognitivní modul, model prostředí).

(c) Řešit nepředvídané události v prostředí (dynamický model prostředí).

(d) Automaticky řešit úlohy na základě modelu prostředí a formulovaného cíle (modul řešení úloh a plánování).

(e) Samostatně vykonávat plány činnosti v prostředí (modul realizátoru plánu, motorický modul).

(f) Aktivně ovlivňovat prostředí manipulací s předměty (efektory).

(g) Komunikace s ostatními činiteli působících v prostředí včetně komunikace s člověkem (komunikační modul).

(h) Vnímat a rozpoznávat situaci vlastní a ostatních činitelů v prostředí pro spolupráci a určení napodobováním (modul chování).

(i) Formulovat vlastní cíle (generátor cílů).

4.3 Co je to robot?

Co je to robot si vysvětlíme na příkladu autonomního robota. Autonomní mobilní robot je inteligentní stroj schopen vykonávat úkoly samostatně, bez lidské pomoci. Nejdůležitější vlastností autonomního robota je jeho schopnost reagovat na změny prostředí. Za tu většinou vděčí svému mozku - počítači, který je zodpovědný, více méně za vše, počínaje zpracováním vstupních údajů, rozhodováním i konečným provedením vybraných akcí.

Vstupní údaje počítači poskytují senzory. Senzor je zařízení, které je schopné měřit nějakou vlastnost prostředí. Příkladem takovéhoto senzoru může být třeba i obyčejný spínač. Tento spínač může robot využít jako nárazník. Měřená vlastnost prostředí je tedy fakt, že se v cestě nachází překážka.

Naopak působit na své prostředí může robot pomocí efektorů. Typickým příkladem efektoru je elektromotor. Po připojení na kolo umožní robotovi pohyb. V závislosti na velikosti robota se můžeme setkat i s dalšími typy efektorů, ať už to jsou spalovací motory či hydraulika.

Z nutných částí zbývá zmínit ještě zdroj energie, kterým je nejčastěji akumulátor, který poskytuje energii senzorům, počítači i většině efektorů. Větší roboti vybaveni spalovacími motory mívají alternátory, které akumulátor průběžně dobíjí, a doba, po kterou je robot schopen operovat autonomně, je tím značně prodloužena. Někteří roboti používají i solární panely (např. roboti vysláni na Mars).

4.3.1 Mozek robota

Robotovým srdcem i mozkem je počítač. Nejčastěji je vybaven tzv. jednočipem neboli mikrokontrolérem, což je malý integrovaný obvod se spoustou pinů. Každý pin může být buď vstupní, nebo výstupní a program běžící v mikrokontroléru pomocí nich může zjišťovat informace o svém okolí nebo naopak něco ovládat.

Jedničky a nuly, se kterými pracuje program, se do reálného světa převádějí jako různé napěťové úrovně. Mikrokontroléry nejčastěji využívají tzv. TTL (Tranzistor To Tranzistor Logic). Ta definuje, že napětí 5V reprezentuje logickou jedničku a napětí 0V reprezentuje

logickou nulu. Stačí tedy, když se vstupní piny chovají jako jednoduché voltmetry a výstupní jako zdroje napětí a čip může komunikovat s okolím.

Jednoduché vstupní piny nám umožňují pouze detekovat stav on/off. Stejně tak jednoduché výstupní piny nám umožní do světa vyslat pouze 0 nebo 5V. Jsou ale situace, kdy by se nám hodilo vědět přesně, jaké napětí je přivedeno na vstupní pin. K tomu slouží tzv. analogové vstupy. Tyto vstupy mají dokonalejší voltmetr (A/D převodník), který je schopen přesněji změřit příslušné napětí. Toto napětí je převedeno na číslo, které je k dispozici programu. Stejně tak můžeme mít D/A výstupy, které jsou schopny na výstupním pinu generovat různá napětí.

Další proměnnou, která hraje klíčovou roli, je čas. Není důležitá pouze aktuální hodnota napětí na jednotlivých pinech, ale i jak se mění v čase. Čas je opět diskrétní. Asi nejvhodnější paralela je s tikáním hodin. Číst hodnotu na pinech je možné pouze vždy, když tiknou. Pokud by tikaly jednou za vteřinu, tak by robot o svém okolí mnoho nezjistil. Běžnější frekvence se blíží spíše několika megahertzům. Při frekvenci 10MHz robot svět pozoruje každých 100us.

U jednodušších robotů, kteří si vystačí s malým výpočetním výkonem, mikrokontrolér stačí. U složitějších strojů jsou jednočipy pouze zprostředkovatelem dat o reálné prostředí pro výkonné počítače, popřípadě pro celou síť počítačů, která robota ovládá.

4.3.2 Senzory

Již víme, že jednočip je schopen měřit napětí na svých vstupech a to ať už velmi hrubě (0/1) nebo jemněji (A/D převodník). Je tedy nutné, aby všechny informace, které má robot využívat, byly převoditelné na napětí. To je primárním úkolem senzorů.

Nejjednodušším druhem senzoru je obyčejné tlačítko, které můžeme využít jako nárazník. Přestože se tlačítko může zdát příliš triviální, je to jeden ze základních a nejspolehlivějších robotických senzorů. Je ovšem ještě velké množství jiných senzorů, které lze použít podle toho, co chceme měřit např. množství odraženého světla - senzory na sledování čáry (CNY70), doba letu ultrazvuku - sonar a měření do vzdálenosti cca 10m, radiový signál od satelitů - navigace pomocí GPS a mnoho dalších.

4.3.3 Efektory

Podobně jako vnímání světa probíhá i robotova interakce s ním. Na pinech konfigurovatelných jako vstupní je mikrokontroler schopen napětí měřit, na pinech výstupních naopak nastavovat. Efektor je pak zařízení, jehož stav můžeme ovládat změnou napětí iniciovanou mikrokontrolerem.

Asi nejčastějším efektorem jsou motory, motorky či serva. Motory i serva se nejčastěji řídí pomocí různě dlouhých pulzů. Délka pulzu u serv určuje polohu a u motorů je to příkon.

Efektor nutně nemusí umožňovat robotovi pohyb. Nejjednodušším takovým zařízením je LED dioda (Light Emitting Diode). Ta může blikat na znamení, že něco funguje/nefunguje, ale může i pouštět televizi, video apod. pokud bliká na správné frekvenci a posílá správné kódy pro jednotlivá zařízení.

4.4 Využití a přínos robotů

Největší využití robotů je v průmyslovém, výzkumném a armádním prostředí. Všechna tyto prostředí mají společnou vlastnost, že se v něm nepohybují osoby nepoučené o možnostech

daného robota. To snižuje nároky na použití robota, který není nucen se vyrovnat s nepředvídatelným chováním lidí.

V průmyslu se mobilní roboti nasazují ve výrobních halách k dopravě materiálu. Můžeme najít automatické sklady, kde mobilní roboti optimalizují umístění jednotlivých položek, automaticky rozvážejí materiály k obráběcím strojům apod. Častým důvodem nasazení mobilních robotů je prostředí nebezpečné nebo nevhodné pro člověka, jako jsou provozy s vysokou hladinou hluku, vysokou teplotou, nebezpečnými plyny apod.

Automobilový průmysl využívá značné množství robotických přístupů při vývoji různých asistenčních zařízení, usnadňující řidiči řízení. GPS navigace, udržování odstupu od předchozího automobilu, sledování krajnice. Všechny tyto zařízení sledují trend vývoje autonomního automobilu. Paradoxně je pro autonomní automobil největší problém člověk. Nepředvídatelné reakce ostatních řidičů a chodců vytvářejí z projektu autonomního automobilu náročnou výzvu.

Armádní prostředí také nahrává mobilním robotům. Bezpilotní letouny schopné plnit úkoly v autonomním nebo teleoperovaném režimu. Mobilní roboti se používají k odstranění námořních i pozemních min, k průzkumům nebezpečných oblastí apod. Mobilní roboty Sherpa se používají k nošení břemen těžko prostupným terénem. Tento kráčející robot sleduje vojáka stejně jako pes. Automatické řízení aut v zásobovací koloně přispívá k bezpečnosti vojáků. Stačí řídit první vůz kolony a ostatní následují autonomě.

Výzkumné prostředí využívá mobilní roboty všude, kam se člověk nemůže dostat. Mobilní roboti prozkoumávají hlubiny oceánů stejně jako hlubiny vesmíru. Dálkově řízené ponorky pomáhají v průzkumu vraků, hlubokomořských příkopů, pomáhají sledovat chování mořských živočichů bez ovlivnění přítomnosti člověka.

Při průzkumu Marsu se také využívalo mobilních robotů. Vzhledem k časovému zpoždění radiového signálu musel být robot vybaven velkou mírou autonomie. Vědci zadávali cíle, které je zajímaly, ale robot sám vybíral nejlepší cestu k dosažení cíle.

Do vědeckého oboru také můžeme zařadit soutěže jako je Grand Challenge nebo robotická kopaná. Takovéto soutěže pomáhají urychlovat vývoj robotické vědy. A nejedná se pouze o takto velké soutěže. Zajímavé jsou i soutěže méně nákladné, zejména v kontextu výuky. Takovouto soutěží je jistě Eurobot. Každoroční změna pravidel umožňuje rovnocenné zapojení nových týmů a dobře definované prostředí zjednodušuje řešení.

Nezapomínejme ani na využití mobilních robotů v záchranných akcích. Teleoperovaní roboti se využívají při zneškodnění nálože. Podobně se využívají roboti při prohledávaní trosek a vyhledávání obětí.

Velký potenciál má také vývoj robotů pro zábavu. Robotický pes Aibo nebo Robotsapien jsou prvními pokusy o roboty pro zábavu. Z takovýchto robotů se můžou vyvinout robotičtí společníci, ošetřovatelky nebo zdravotní sestry. Takoví roboti připravují laickou veřejnost na přítomnost robotů v běžném životě [2].

Kapitola 5

Program

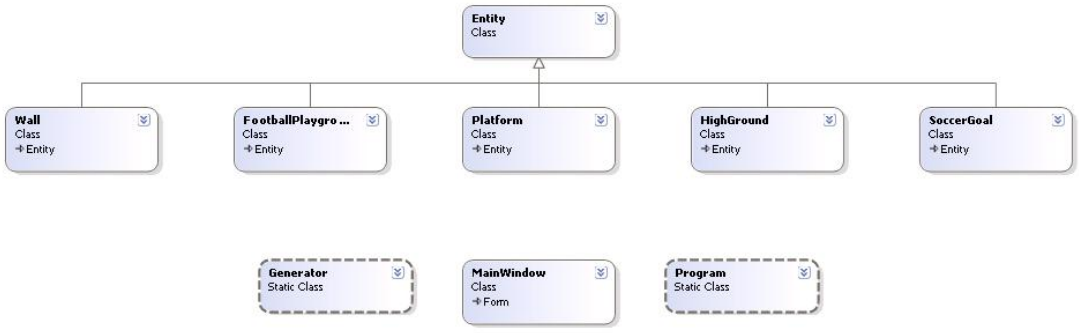
Program by měl být schopen generovat scény pro MSRS. Program by měl obsahovat 2D rastr a paletu několika předem vytvořených objektů, které bude možné rozmístit po scéně. Takto vytvořená scéna by se měla uložit ve formátu, který odpovídá formátu uložené scény v MSRS. Tudíž by mělo být možné tuto scénu v MSRS otevřít. Tento program by měl umožňovat rychlé a jednouché vytvoření scény bez znalosti MSRS.

5.1 Popis řešení a implementace

Funkce programu by se dala rozdělit do čtyř základních kroků:

1. Práce s paletou předem vytvořených objektů a jejich umístění do scény.
2. Nastavení vlastností jednotlivých objektů.
3. Nastavení vlastností scény.
4. Vygenerování scény do xml.

Pro implementace jsem si vybral jazyk C# a pro vývoj jsem využíval Visual Studio 2005 s verzí .Net Framework 2.0



Obrázek 5.1: Diagram tříd programu

5.1.1 Paleta objektů a jejich rozmístění do scény

Každý objekt v paletě je reprezentován samostatnou třídou, která dědí ze třídy Entity (obr. 5.1). Tato rodičovská třída obsahuje proměnné s hodnotami různých vlastností objektů a metody pro drag and drop. Třídy, které z ní dědí, přidají vlastní metody pro rotaci objektu a pro načtení svých textur.

Základní entity, které jsou v programu předem vytvořené: zeď (wall), plošinka (platform), fotbalové hřiště (football playground), fotbalová branka (soccer goal) , vyvýšenina (high ground). Tyto objekty jsou ve scéně reprezentovány instancí třídy PictureBox. Tato třída nabízí snadnou práci s obrázky. Když si uživatel vybere objekt, který chce vložit do scény, změní se mód programu do vkládacího (insert) módu. Po stisknutí levého tlačítka se objekt vloží do scény a společně s tím proběhne několik dalších událostí.

Nejdříve se nastaví souřadnice jednotlivých os objektu. Těm se přiřadí hodnota, která odpovídá levému hornímu rohu obrázku. Je samozřejmé, že měřítko souřadnic v mém editoru se liší od měřítka v MSRS. Toto byl jeden s hlavních problémů, s kterým jsem se musel vypořádat. Nakonec jsem našel správný převodní vztah, který se rovnal jedné padesátině (souřadnice z mého editoru se dělí padesáti, aby se dosáhlo správného měřítka).

Jako další se nastaví rozměry objektu. Jako rozměr používám šířku a výšku obrázku, který znázorňuje objekt. Z toho vyplívá, že velikost objektů je neměnná. Navíc velikost obrázku nemá přímí vliv na velikost objektu v MSRS. Velikost je totiž reprezentována uloženou předlohou v souboru s příponou obj (soubory, které obsahují grafická data o objektu). Pro každý vytvořený objekt sem vytvořil takovouto předlohu v programu 3ds Max7 a to tak, aby velikost odpovídala obrázku. To znamená, že při nastavování hodnoty MeshScale (tedy rozměr) v xml uložené scény, se zadají hodnoty všech os rovny jedné. To znamená, že objekt bude mít velikost rovnu té v souboru obj. Pokud by se MeshScale nastavila na dvojku, velikost by byla dvakrát větší. Tato možnost by nabízela jistou schopnost měnit velikost objektů, ale nesmíme zapomínat, že společně s velikostí výsledného objektu by se musela upravovat také velikost předlohy a samozřejmě souřadnice polohy, které by nyní měly jiný poměr.

Další věc, která nastane při vložení objektu do scény je, že se vytvořená instance vloží do listu, který obsahuje všechny vytvořené objekty. Tímto způsobem se dá snadno zjistit počet entit ve scéně a navíc usnadňuje generování scény do xml, které nyní může probíhat ve smyčce a zapisovat jeden objekt za druhým.

Nyní, když je objekt vložen do scény a nastaví se příslušné atributy, se zpřístupní v pravém horním rohu vlastnosti, které může uživatel ručně nastavit.

5.1.2 Vlastnosti objektů

Jak již víme, základní vlastnosti objektu jako rozměr a souřadnice se nastaví ihned po vložení objektu do scény. Ostatní vlastnosti se dají nastavit ručně a uživatel si je může nastavit sám, jak potřebuje.

Dynamické a statické tření, ovlivňují povrch objektu ve scéně. Toho se dá využít zejména u plošinky nebo například u fotbalového hřiště.

Hmotnost zásadně ovlivňuje výslednou entitu a její fyziku. Tělesa s různou hmotností na sebe reagují jako v reálném světě. Uživatel si musí dávat pozor na nastavení hmotnosti například plošinky, protože pokud by byla lehčí než robot, který na ní má vyjet, je pravděpodobné, že dojde k jejímu posunutí.

Ve svém programu jsem umožnil nastavení jen těchto základních tří vlastností, protože jsou nejčitelnější.

Samozřejmě entity mají ještě spoustu jiných vlastností. Mezi těmi, které využívám je například natočení v osách (ratace), ta ovšem zůstává uživateli skryta.

Dále MSRS umožňuje nastavení celé škály jiných vlastností, které jsem nezpřístupnil uživateli kvůli jednoduchosti. Například lineární a úhlová rychlost, které by se dalo teoreticky využít pro simulaci větru. Další vlastnosti jsou setrvačnost, hustota, objem nebo nastavení polohy těžiště. Všechny tyto vlastnosti by se dali použít v implementačním rozšíření programu.

5.1.3 Vlastnosti scény

Stejně jako objekty, i scéna má své vlastnosti. Ty nejzákladnější se týkají kamery. U té se dá nastavit dohled, rozhled a pozice. Tyto vlastnosti může uživatel nastavit ručně společně se zaměřením kamery. To je udáno souřadnicemi bodu, na který je kamera zaměřena.

Další výraznou vlastností je gravitace. Automaticky je nastavena na hodnotu země, ale do nastavení jsem přidal také hodnoty Měsíce a Marsu. Tato vlastnost má samozřejmě hodně velký vliv na simulaci. Pokud například uživatel zadá nulovou gravitaci, může simulovat stav beztíže apod.

Do základního nastavení scény jsem přidal možnost obsáhnout základní prvky. Nová scéna v MSRS neobsahuje totiž vůbec nic, tedy pouze tmu. Aby bylo možné vygenerovat prázdnou scénu, v tom smyslu, že neobsahuje žádné z předem vytvořených objektů, uživatel si může vybrat, jestli chce do scény vložit zem (ground), slunce (sun) a oblohu (sky).

Slunce slouží jako základní zdroj světla a obloha pouze dotváří vzhled scény. Zemi lze nastavit několik vlastností jako texturu nebo dynamické a statické tření. Tyto prvky jsou nepovinné a uživatel může klidně objekty vkládat do tmy a vzduchoprázdna.

5.1.4 Vygenerování scény do xml

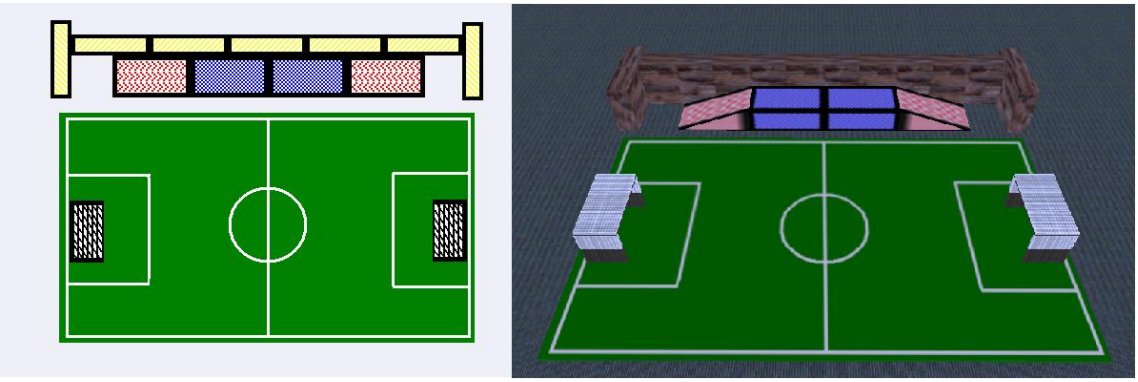
Když uživatel vloží požadované objekty a nastaví jim žádané vlastnosti společně s vlastnostmi scény, zbývá poslední krok a to vygenerovat scénu do xml tak, aby se dala otevřít v MSRS.

Pokud se podíváme na uloženou scénu v MSRS zjistíme, že je to klasické xml, rozděleno do několika základních uzlů. Hlavní uzel je SimulationState, který v sobě obsahuje všechny ostatní uzly. Na začátku je několik tagů, které obsahují hodnoty základních vlastností scény (nastavení kamery apod.) potom následuje uzel, který by měl obsahovat zdroje světla a nakonec uzel SerializedEntities, který obsahuje všechny objekty, které scéna obsahuje.

Po zmáčknutí tlačítka generate v programu, se zavolá funkce třídy Generator. Tato třída si vytvoří novou instanci třídy XmlTextWriter do kterého se vše zapisuje. Nejdříve vytvoří hlavni uzel SimulationState a poté využije proměnných, které obsahují nastavení scény k tomu, aby vytvořila tagy, které obsahují vlastnosti scény.

Poté přichází na řadu hlavní smyčka. V ní se čte z listu vložených objektů prvek po prvku a pro jednotlivé objekty se volá metoda insertEntity. Ta si jako parametr mimo jiné přebírá xml writer a objekt, který se má zapsat. Díky němu má metoda přístup ke všem jeho proměnným tedy k vlastnostem. Jednotlivé vlastnosti jsou v xml reprezentovány jako uzly nebo pouze tagy. Například pozice vytvoří uzel, který obsahuje tagy s hodnotami jednotlivých os naopak název objektu je jen samostatný tag.

Vytvořenou scénu je potřeba nakopírovat do složky MSRS, protože simulační nástroj odjinud scénu neotevře.



Obrázek 5.2: Vzhled scény v editoru / výsledná scéna po převedení do MSRS

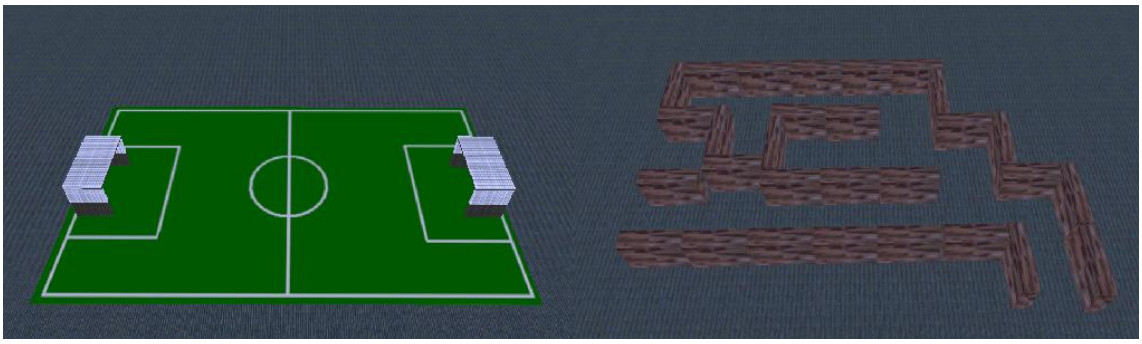
5.2 Zavěr kapitoly

Program plní svůj účel a dělá to, co se od něj očekává, avšak má spoustu nedodělaných a nedotažených věcí, na kterých by bylo potřeba důkladněji zapracovat. Výhodou je že ho mohou používat i uživatelé neznalí MSRS. (viz. o hodnocení programu kapitola 6)

Kapitola 6

Závěr

Úkolem bylo, vytvořit grafický mezieditor, ve kterém bude možné vytvářet scény pro nástroj MSRS a zároveň nějaké vzorové scény vytvořit (robotický fotbal, hledání cesty v bludišti).



Obrázek 6.1: Robotický fotbal / hledání cesty v bludišti

Tohoto cíle jsem dosáhnul (obr. 6.1). Program funguje a lze v něm vytvářet scény pomocí předem vytvořených objektů. Pokud bych měl ovšem hodnotit objektivně svůj výtvor, musel bych říci, že je to první krok nebo také cvičný program. Posloužil mi, abych lépe prostudoval a pochopil tvoření scény v nástroji MSRS a její uložení do xml. Donutil mne také pochopit různé vlastnosti objektů, jejich nastavení a vliv na simulaci (hmotnost, tření, gravitace).

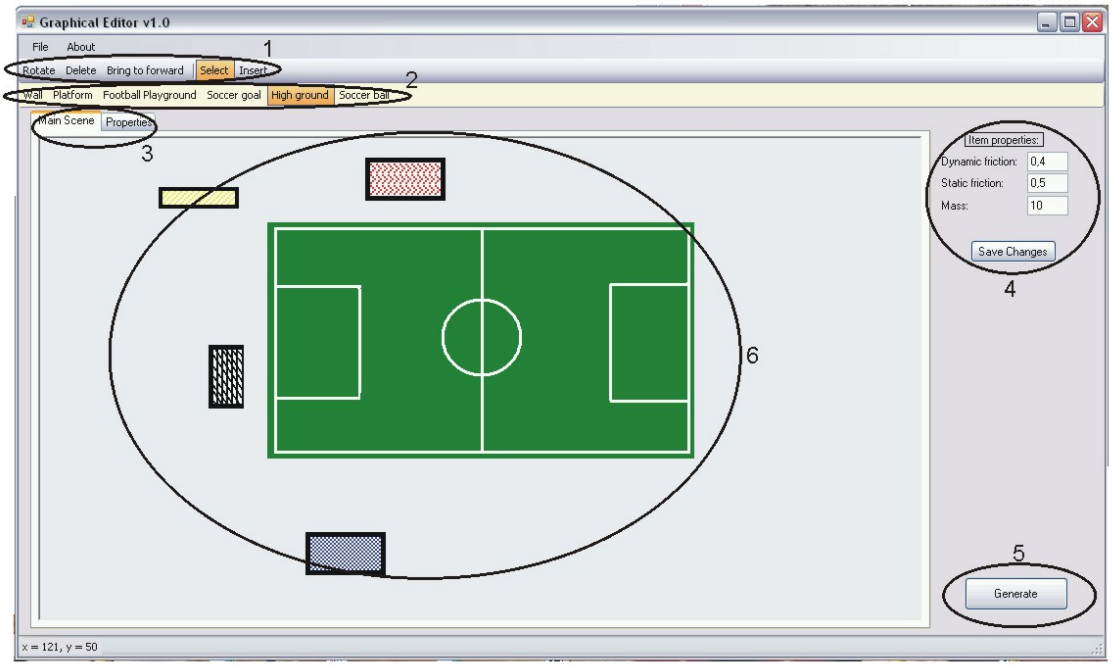
Do kladů bych zařadil rychlé, jednoduché ovládání a přehledné prostředí. Pokud bych měl navrhnout implementační rozšíření pro můj program, provedl bych kompletní předělání do 3D a to z toho důvodu, aby bylo možné pokládat objekty na sebe. Dále bych navrhoval lepší práci s vloženými objekty, jako změnu velikosti nebo změnu barvy.

Závěrem bych chtěl jen dodat, že nástroj MSRS mne velice zaujal a vidím za ním silný prostředek pro simulaci robotů.

Kapitola 7

Přílohy

7.1 Ovládání programu



Obrázek 7.1: Uživatelské prostředí programu

1. Toolbar - Obsahuje několik tlačítek pro práci s objekty. Jsou tu:
   (a) Rotate - při zmáčknutí tohoto tlačítka, se vybraný objekt otočí o devadesát stupňů proti směru hodinových ručiček.
   (b) Delete - Slouží pro mazání označeného objektu.
   (c) Bring to forward - Přenese vybraný objekt do popředí.
   (d) Select - Slouží pro přepnutí do módu vkládání (jinak se dá mezi módy přepínat pomocí kliknutí pravého tlačítka myši do prázdného prostoru scény)

(e) Insert - Slouží pro přepnutí do vkládacího módu.

2. Paleta objektů - Je zde připraveno 5 předtvořených objektů. Zeď (wall), plošinka (platform), fotbalové hřiště (football playground), fotbalová branka (soccer goal) a vyvýšenina (high ground). Po vybrání některého z těchto objektů se program přepne do vkládacího módu a je možné levým kliknutím do scény objekty rozmísťovat.

3. Záložky - Jsou zde dvě záložky:

   (a) Scene - Záložka, která obsahuje hlavní scénu.

   (b) Properties - Tato záložka obsahuje základní nastavení scény, jako nastavení kamery, gravitace apod.

4. Item properties - Po vložení nebo vybrání objektu, se zde zobrazí jeho nejzákladnější vlastnosti (statické a dynamické tření, hmotnost). Tyto vlastnosti lze změnit (přepsáním) a uložením pomocí příslušného tlačítka

5. Tlačítko generate - Po zmáčknutí tohoto tlačítka se otevře save file dialog a uživatel si může vybrat, kam chce scénu uložit. Doporučuje se, scény ukládat do složky MSRS, protože jinak není možné scénu otevřít. (Přípona xml je dopisována automaticky).

6. Scéna - Hlavní scéna do které lze vkládat objekty.

Literatura

[1] Application Model Introduction. [online], [cit. 2008-05-08].
URL <http://msdn.microsoft.com/cs-cz/library/bb466255(en-us).aspx>

[2] Mobilní robotika. [online], [cit. 2008-05-08].
URL <http://www.roznovskastredni.cz/dwnl/pel2007/06/Kosnar.pdf>

[3] Robotika - První přednáška. [online], [cit. 2008-05-08].
URL <http://cmp.felk.cvut.cz/cmp/courses/roblec/robuvod.pdf>

[4] Runtime Introduction. [online], [cit. 2008-05-08].
URL <http://msdn.microsoft.com/cs-cz/library/bb483056(en-us).aspx>

[5] Simulation Tutorial 1 (C#)- Introduction to the Simulation Runtime. [online], [cit. 2008-05-08].
URL <http://msdn2.microsoft.com/en-us/library/bb483077.aspx>

[6] Churý, L.: Robotika I. 2006, [cit. 2008-05-08].
URL
<http://programujte.com/index.php?akce=clanek&cl=2006022101-robotika-i>

[7] Ing. Ivo Brachtl, I. J. K., Ing. Jiří Douša: *Číslicová simulace I.* Vydavatelství ČVUT, Praha, 1979.