ROBOT PROGRAMMING FRAMEWORK IN VIRTUAL ENVIRONMENT

Martin Maťátko
Master Degree Programme (2), FIT BUT
E-mail: xmatat01@stud.fit.vutbr.cz

Supervised by: Jaroslav Rozman
E-mail: rozmanj@fit.vutbr.cz

Abstract: This project deals with a theoretical analysis, design and implementation of a programming framework in a virtual environment. Robots use sensors to gain information about the environment and they subsequently modify this environment as a result of an instruction code. The instructions are entered in a graphic form using a custom visual programming language. The last part is editing and rendering of the virtual world in which the robots are moving. The result of the work is a concept of the aforementioned components and their linking to a functional unit.

Keywords: .Net, visual programming language, VPL, robot, instruction execution

1 ÚVOD

Tato práce se zabývá tvorbou programovacího prostředí s virtuálními roboty. Základem je programovatelný robot. Jeho chování je definované instrukčním kódem, který mu uživatel zadá v grafické podobě prostřednictvím vizuálního programovacího jazyka (anglicky *Visual Programming Language*, zkráceně VPL). Instrukce vykonává virtuální stroj a umožňuje tak interakci robota s okolním světem.

Celá práce si klade za cíl přiblížit oblast robotiky v jednoduché, srozumitelné a interaktivní podobě i běžným uživatelům počítačů. Z toho důvodu je celý svět, včetně robota, virtualizován a výsledek je integrován do jediného programu.

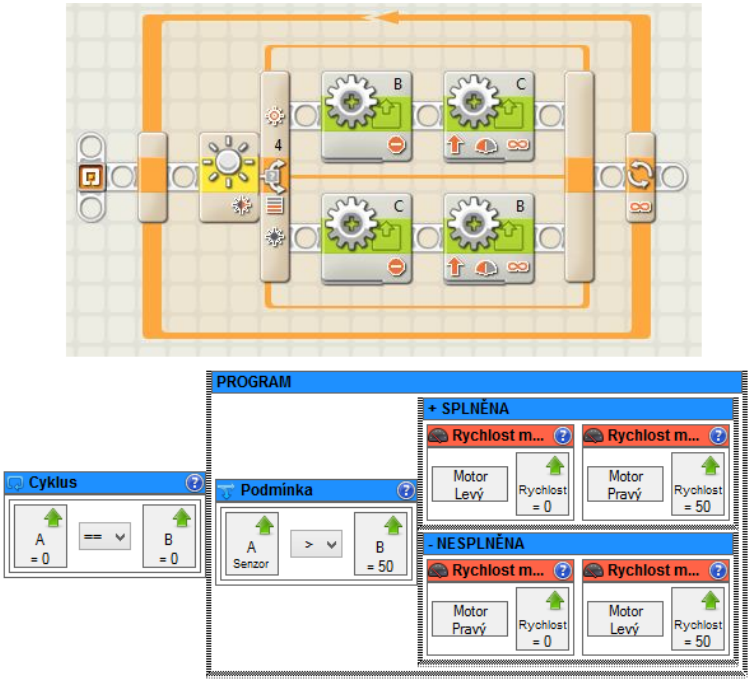
2 VIZUÁLNÍ PROGRAMOVACÍ JAZYK

VPL umožňuje sestavení programu v grafické podobě a je tak alternativou ke standardním textovým programovacím jazykům. Hlavní rozdíl je v tom, že rozhraní vizuálních jazyků už ze své podstaty předem definuje jejich syntaxi i sémantiku a uživatel tak při zadávání instrukčního kódu nemůže udělat chybu vycházející ze zápisu jazyka. Uživatel se tak může více soustředit na samotnou tvorbu rozhodovací logiky robota.

Některé složitější konstrukce mohou být při zápisu pomocí vizuálního jazyka nepřehledné, i proto je projekt doplněn komponentou pro podporu vlastního textového jazyka, podobného jazyku C. Překladač je implementován pomocí funkcionálního jazyka F# na platformě .Net.

Problematikou vizuálních programovacích jazyků se v současnosti zabývají desítky vývojových týmů. Vlastní VPL vychází primárně z projektu Lego Mindstorms. Základním stavebním prvkem je blok, který odpovídá jedné instrukci robota. Programové bloky se do sebe skládají zleva doprava podobně jako puzzle. Jejich vykonávání má stejnou logiku. Vlastní VPL podporuje jediný datový typ, kterým je celé číslo. Pokročilí uživatelé mohou využít podporu pro pseudoparalelní vykonávání instrukcí, včetně technik pro pozastavení vláken a jejich synchronizaci. Základní instrukční bloky, jako je větvení programu nebo cyklus, jsou vestavěné přímo do vizuálního jazyka. Jejich skládáním může uživatel vytvářet vlastní bloky, a to včetně podpory pro rekurzivní zanoření.

Na obrázku 1 je totožný program pro porovnání vytvořený v programu Lego Mindstorms a ve vlastním VPL. Program slouží ke sledování čáry robotem. Na základě vstupní hodnoty optického senzoru je poháněn buď levý, nebo pravý motorický efektor diferenciálního podvozku robota.



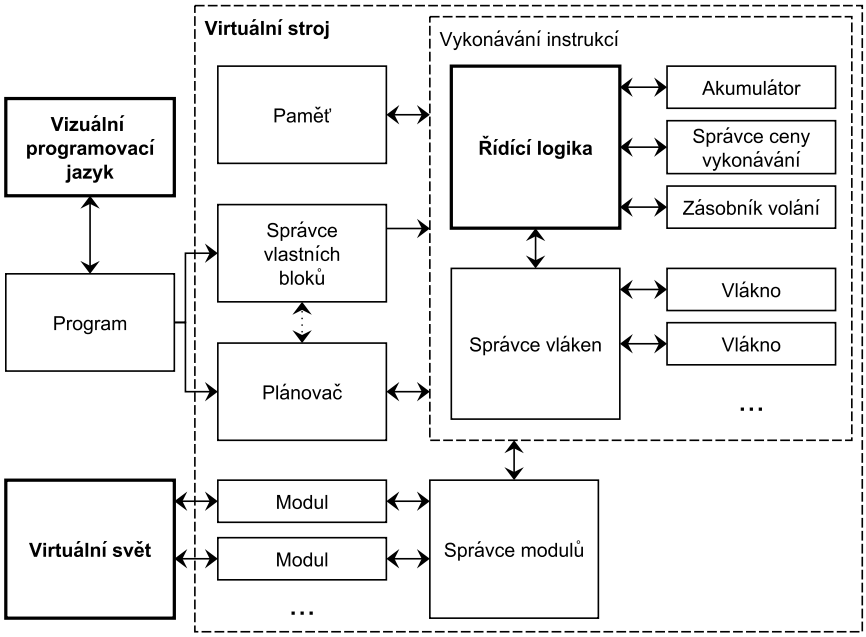
Obrázek 1: Porovnání vizuálních jazyků - nahoře Lego Mindstorms a dole vlastní VPL

3 STRUKTURA A IMPLEMENTACE

Výsledný program se skládá ze tří hlavních částí, kterými jsou *vizuální programovací jazyk*, *virtuální stroj* a *virtuální svět*. Celou strukturu ukazuje obrázek 2. VPL obsahuje definici instrukcí a umožňuje uživateli jejich poskládání do funkčního programu robota. Uživatelem zadaný program je dále plánovačem převeden na interní plán atomických instrukcí. O jejich vykonávání se stará řídící logika. Akumulátor slouží jako dočasný odkládací prostor pro probíhající výpočty. Hodnoty akumulátoru a zásobník volání jsou různé pro každé vlákno a závisí na konkrétním stavu vykonávání zadaného programu. Správce vláken zajišťuje nastavení těchto komponent pro každé vykonávané vlákno. Základním úložným prostorem pro data je paměť robota, která je sdílená pro všechna vlákna. Zásobník volání upravuje viditelnost jednotlivých proměnných v paměti a zajišťuje inicializaci lokálních proměnných při vytváření nových vláken nebo při volání zanořených uživatelských bloků. Správce modulů zprostředkovává komunikaci s jednotlivými moduly a poskytuje jednotné rozhraní pro jejich ovládání. Senzory a efektory jsou jedinými prostředky robota pro vnímání resp. ovlivňování okolního světa. Jejich implementace je řešena právě pomocí modulů. Senzory slouží k získávání informací o okolním světě (např. sonar nebo laser k měření vzdálenosti od překážky), efektory k ovlivňování okolního světa (jedná se primárně o motory zajišťující pohyb robota).

Virtuální svět představuje vysokou abstrakci (model) reálného světa, ve které je možné simulovat např. úlohy sledování čáry a průchod bludištěm. Vykreslování a aktualizace světa probíhá periodicky. Hlavním objektem je samotný robot.

K implementaci byl zvolen jazyk C# na platformě .Net. Vizuální programovací jazyk je realizován pomocí WinForms GDI+. Virtuální svět a jeho animace byly vytvořeny za pomoci *Delta Engine*. Výsledný program je určen primárně pro operační systém Windows. Delta Engine je volně dostupné



Obrázek 2: Blokový diagram výsledného řešení

grafické a fyzikální prostředí určené pro tvorbu programu v jazyce C# a C++. Kombinuje fyzikální prostředí *Farseer Physics* a mnoho knihoven pro vykreslování 2D i 3D objektů (například *OpenTK*, *GLFW*, *SharpDX*, *SlimDX* nebo *XNA*).

4 ZÁVĚR

Výsledkem práce je funkční program, díky kterému se mohou i běžní uživatelé seznámit se základními principy robotiky. Uživatel si může díky virtualizaci prostředí vyzkoušet celý proces tvorby robota, aniž by musel znát konkrétní hardware a software pro realizaci robota.

Celý program byl vytvořen na základě návrhu založeného na rozhraních (anglicky *Interface-based Design*). To znamená, že všechny funkce pracují pouze s rozhraním objektu, ne s jeho konkrétním typem. Rozhraní definuje vlastnosti a metody, kterými musí objekt disponovat. Díky tomu lze následně vytvořit různé třídy, které budou implementovat různá chování při zachování stejného rozhraní. Na místo, kde je vyžadováno určité rozhraní, lze dosadit libovolnou třídu, která toto rozhraní implementuje. To je zajištěno díky principu dědičnosti, protože daná třída je specializací požadovaného rozhraní. Celý koncept tak umožňuje snadnou programovou rozšiřitelnost. Programátor může vytvořit vlastní třídy, které implementují daná rozhraní, dosadit je na místo původních, a tak dosáhnout požadovaného chování.

REFERENCE

[1] Hunt, A.; Thomas, D.; Hargett, M.: Pragmatic Unit Testing in C# with NUnit, Second Edition. Pragmatic Bookshelf, 2007, ISBN 978-0-9776166-7-4.

[2] Siegwart, R.; Nourbakhsh, I. R.: Introduction to Autonomous Mobile Robots. MIT Press, 2004, iSBN 0-262-19502-X.