Vývojové prostredie Microsoft Robotics Studio

František Duchoň, Martin Štrenger

Abstrakt
Článok oboznamuje čitateľa s vývojovým prostredím Microsoft Robotics Studio. Venuje sa najmä základným pojmom, s ktorými je nutné sa oboznámiť pred začiatkom práce s Microsoft Robotics Studio. Uvedené sú základné výhody a nevýhody používania tohto softvérového nástroja.

Kľúčové slová: modely mobilných robotov, simulácia mobilných robotov, Microsoft Robotics Studio, vizuálny programovací jazyk, manifest

Úvod

Mobilná robotika je v súčasnosti veľmi dynamickým odvetvím, ktoré sa čoraz viac začína uplatňovať nielen vo vedeckých ústavoch, ale aj v bežnej praxi. Ako príklady spomeňme automatické vysávače Trilobite alebo dvojkolesový osobný transportér Segway. Pri konštruovaní mobilných robotov vzniká množstvo úloh, ktoré sa javia ako jednoduché, pri samotnej realizácii a riešení sa však ukazujú ako náročné. Jednou z týchto úloh je aj riadenie robota, čiže plánovanie jeho činnosti v prostredí, ktoré by malo byť čo najviac inteligentné. Riešenie takýchto úloh bolo dôvodom vzniku rôznych simulátorov mobilných robotov, ktoré umožňujú vyskúšať si navrhnuté riešenie na počítači, a tak znížiť opotrebovanie hardvéru, poškodenie samotného robota, čas spojený s testovaním navrhnutého riešenia apod. Tieto simulátory dokážu navrhnuté riešenie prepojiť aj s konkrétnym hardvérom, čo možno považovať za pozitívum. Jedným z takýchto simulátorov je aj voľne dostupné Microsoft Robotics Studio.

Stručný opis Microsoft Robotics Studia

Microsoft Robotics Studio (MRS) je vývojové prostredie určené pre operačné systémy Windows a umožňuje vývojárom jednoduché navrhnutie robotických aplikácií pre rôzne hardvérové platformy. Obsahuje vizuálne vývojové prostredie, ktoré umožňuje jednoduchý návrh a odladenie rôznych robotických aplikácií. MRS využíva realistické 3D modely s licencovaným fyzikálnym modelom AGEIA™ PhysX™ [1], ktorý umožňuje simulácie robotických modelov s fyzikou reálneho sveta. Okrem samotného prostredia, umožňuje používateľovi generovať štandardné služby pre hardvér a softvér, a teda aj komunikáciu s robotmi pomocou webového alebo Windows rozhrania. Programovací model MRS môže byť aplikovaný na rôzny hardvér, pričom umožňuje prechod z jednej platformy na inú. Vývojár si môže zvoliť programovací jazyk, v ktorom chce programovať. MRS dokáže pracovať s jazykmi C#, VB.NET, MS Iron Python a MS Visual Programming Language. Základné pojmy pre Microsoft Robotics Studio:

- **Concurrency and Coordination Runtime (CCR)** – proces pre súbežnosť a koordináciu, zabezpečuje koordináciu správ bez nutnosti manuálne tvoriť vlákna, semafory a zámky. CCR je založené na asynchrónnom prechode správ.
- **Decentralized System Services (DSS)** – služby, ktoré poskytujú hostiteľské prostredie pre ostatné služby, umožňujúce úlohy ako ladenie, záznam, monitorovanie, bezpečnosť, sprístupnenie a dátovú stálosť.
- **Application** – aplikácia je spojenie služieb, ktoré pomocou riadenia môžu byť spriahnuté pre vykonanie želanej úlohy
- **Contract** – kontrakt je skrátený opis služby, ktorý opisuje jej správanie tak, aby ju mohli použiť aj iné služby. Kontrakt je zvyčajne identifikovaný pomocou HTTP URI.
- **Manifest** – manifest je XML dokument, ktorý opisuje súbor služieb, ktoré majú byť použité pri behu DSS. Manifest môže byť použitý na explicitné prepísanie štandardných vlastností služieb ako napríklad aktuálne meno služby alebo s ktorým partnerom alebo službou má spolupracovať. V spojení s vytvorením služby pomocou manifestu, služby môžu byť vytvorené programovo pomocou iných služieb alebo pomocou použitia Control Panelu v internetovom prehliadači. Služba Control Panel je spojenie služieb a je prítomná vždy, keď je spustený DSS proces (DSS node).
- **Orchestration** – v spojení s robotickými aplikáciami predstavuje úlohu zosúladenia vstupov zo senzorického systému a ovládanie súboru akčných členov na základe týchto vstupov. Cieľom je poskytnúť funkčnosť a vystupovať ako nezávislá aplikácia alebo ako časť (komponent) nadradenej úrovne riadenia.
- **Partner** – partner je služba spojená s inými službami so zámerom partnerstva. Partnerstvo je spôsob opisu vzťahu medzi dvomi službami v takom smere, v ktorom je umožnené použitie podriadených členov pri použití neskoršieho zviazania.
- **Proxy** – zástupca, je to vygenerovaná zostava, ktorá odkrýva kontrakt služby tak, aby túto službu mohli využiť aj ostatné služby. Zostava (proxy assembly) obsahuje verejný operačný port a typy stavov vyjadrené službou. Keď služby vytvárajú medzi sebou partnerstvo spoja sa medzi sebou pomocou proxy assembly. Proxy je možné pomocou použitia príkazového riadku príkazom DssProxy. Tento príkaz je automaticky zapojený do procesu kompilácie počas generovania novej služby použitím príkazu DssNewService. Výsledkom spustenia DssProxy je projekt Visual Studia, ktorý je vytvorený v adresári Proxy v priečinku aktuálnej služby.

Hlavný proces (Runtime)

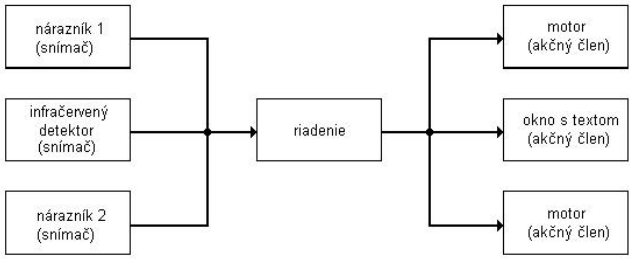
Základným cieľom robotických aplikácií je získať senzorický vstup z rozličných zdrojov a riadiť súbor akčných členov ako odozvu na senzorický vstup tak, aby bol dosiahnutý poža-

dovaný cieľ aplikácie. Ako príklad možno uviesť jednoduchú robotickú aplikáciu zobrazenú na Obr. 1. Obsahuje jednoduchý nárazník – snímač, ktorý dáva správu o tom, či bol stlačený. Ďalej obsahuje okno správy (akčný člen) a blok riadenia (orchestration). Riadenie čaká na vstupný signál zo snímača a na výstupe dáva nejakú textovú správu.

Obr. 1 Príklad jednoduchej aplikácie

Fig. 1 Simple application example

Na predchádzajúcom obrázku je riadenie ešte jednoduchá záležitosť, ale ako aplikácia rastie, riadenie sa stáva čoraz viac komplikovanejšie. Aplikácia môže obsahovať viac snímačov, akčných členov a dokonca aj viac blokov riadenia, ktoré musia vedieť medzi sebou komunikovať a fungovať ako celok, aby mohli riešiť aj komplikovanejšie úlohy. Príklad zložitejšej aplikácie je na Obr. 2.



Obr. 2 Príklad zložitejšej aplikácie

Fig. 2 Complicated application example

Aj keď sa zdá, že snímače a akčné členy sú rovnaké ako v predchádzajúcom príklade, riadenie sa musí v tomto prípade starať až o šesť komponentov. Navyše aplikácie:

- musia spracovať údaje zo snímačov a riadiť akčné členy súbežne, inak sa môže stať, že snímače budú ignorované a akčné členy nedostanú žiadne príkazy.
- v ktorých je počet snímačov a akčných členov vyšší, je riadenie kritická časť a musí byť stabilná
- mať komponenty distribuované a dosiahnuteľné po sieti, aby bolo možné autonómne riadenie a riadenie pre spoluprácu

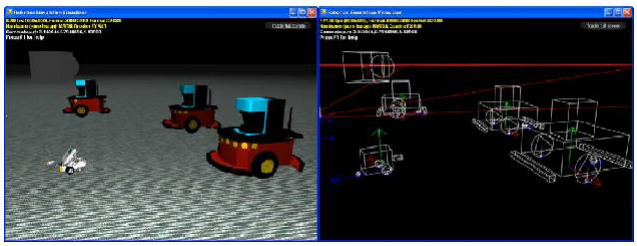
MRS Runtime [2] je navrhnutý tak, aby podporoval širokú škálu robotických aplikácií od jednoduchých pozorovacích senzorických vstupov až po autonómny chod alebo dokonca spoluprácu medzi viacerými autonómnymi robotmi (tzv. multiagentové systémy). MRS môže byť použitý aj pri robotoch priamo pripojených na počítač pomocou sériového portu, USB portu alebo pomocou technológie Bluetooth. Umožňuje spoluprácu aj s robotmi, ktoré majú vlastný počítač alebo so simulovanými robotmi, ktoré môžu byť ovládané pri práci v simulovanom prostredí.

Služby

Služby sú základný stavebný blok pri tvorbe aplikácií MRS. Aplikácia na Obr. 1 môže byť napríklad nahradená tromi službami, teda namiesto každého bloku jedna služba. Keďže služby môžu medzi sebou komunikovať rovnako dobre v rámci jedného uzlu (node) ako aj po sieti, tieto tri služby môžu bežať všetky naraz na jednom počítači alebo každá zvlášť na inom počítači spojenými pomocou internetu. Aby to všetko fungovalo správne, riadenie potrebuje nárazník a okno pre text, aby ich mohlo riadiť. Kvôli rôznym ohraničeniam však služby o sebe „nevedia“ až kým nie sú spustené. Riadenie teda v tomto prípade vie, že musí komunikovať s niečím, čo sa podobá na snímač a okno s textom, ale nevie presne kde sa dané služby nachádzajú až kým nie sú spustené. Táto forma spájania sa nazýva partnerstvo. Partnerstvo je spôsob opisu vzťahu medzi dvomi službami v takom smere, v ktorom je umožnené použitie podriadených členov pri použití neskoršieho zviazania. Kompozícia pomocou partnerstva je cestou k poskytnutiu vyššieho stupňa abstrakcie. MRS Runtime obsahuje súbor služieb, ktoré poskytujú veľa často používaných a potrebných funkcií ako sú monitorovanie, ladenie, riadenie životného cyklu služieb a manažér podpisov (subscription manager).

Simulácia mobilných robotov

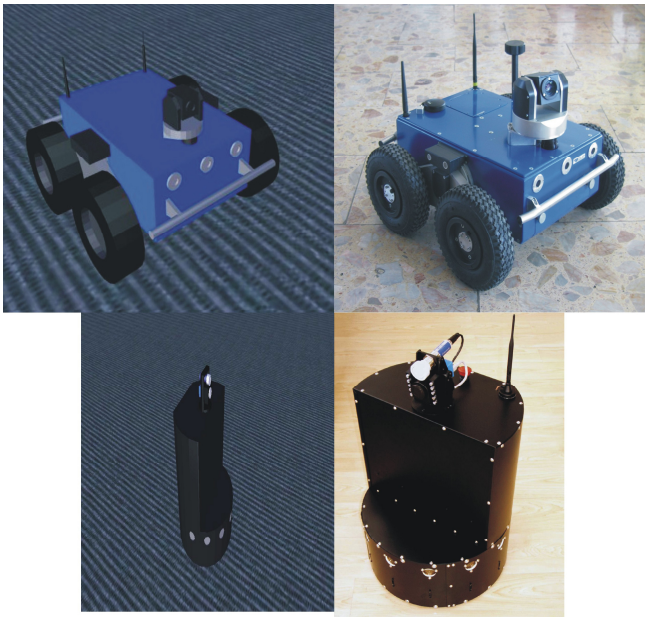
Štandardné robotické platformy ako Lego Mindstorms alebo Fischertechnik [3] spravili robotiku cenovo prístupnú pre širokú verejnosť. Tieto platformy sú vynikajúci štartovací bod pre vzdelávanie a nadšencov robotiky. Ak však potrebujeme zložitejší robot alebo potrebujeme väčší počet robotov, môžeme naraziť na problém vysokej ceny. Rovnako aj odstraňovanie chýb hardvéru nie je najjednoduchšie. Spotrebná elektronika sa stala spoľahlivou, a preto veľa ľudí nemá starosti o to, že sa niečo pokazí. Keď sa však dáva dokopy modulárny robot, treba mať na to dostatok skúseností, času a úsilia pri dolaďovaní robota. Vývoj pokročilého robota tímom ľudí sa stáva čoraz častejším javom, ale výsledkom je zvyčajne len jeden drahý robot. Tieto vplyvy robia skúšanie vecí súbežne s ostatnými ťažšie a navyše s rizikom poškodenia robota.



Obr. 3 Príklad prostredia s robotmi v rôznych režimoch zobrazenia

Fig. 3 Scene with robots in different display mode

Simulácia umožňuje ľudom s počítačom vyvíjať zaujímavé roboty alebo skupiny robotov, pričom jediné obmedzenia sú čas a predstavivosť. Navyše ich simulácia udržiava v medziach podobných reálnym robotom, takže všetko vynaložené úsilie môže byť neskôr aj zrealizované. MRS pristupuje k simulácii postupne v rôznych krokoch. To znamená, že na odladenie simulovaného robota nám postačujú aj len základné a jednoduché vedomosti. Keďže je vloženie simulovaného robota alebo iných objektov do prostredia jednoduché, celé ladenie a hľadanie chýb v simulácii je uľahčené. Fyzikálne modely pre robot a simulačné služby, ktoré ich využívajú, môžu byť vyvíjané súčasne mnohými ľuďmi. Vytvorenú platformu môžu využiť a modifikovať viacerí výskumníci bez obáv o zničenie robota. Ďalšou zaujímavou vlastnosťou simulácie je to, že môže byť použitá súčasne za behu robota ako nástroj na predvídanie alebo učiaci sa modul.



Obr. 4 Príklad využitia simulačného prostredia na modelovanie skutočného robota

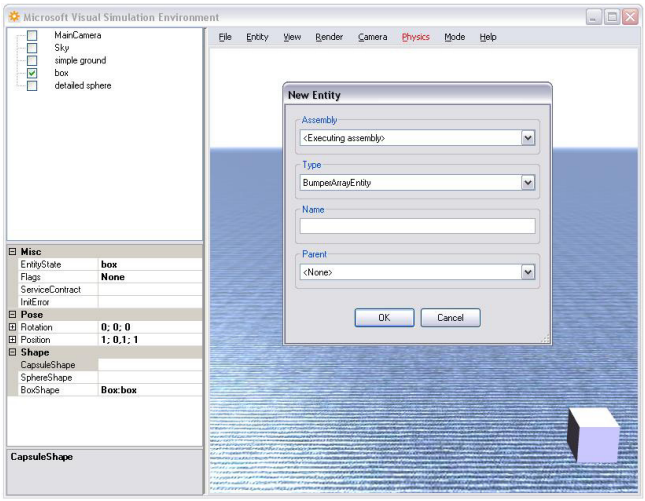
Fig. 4 Simulation scene for the real robot

Pri simulácii sa v podstate snažíme premeniť hardvérový problém na softvérový. Vývoj softvéru a fyzikálneho modelu má však svoje vlastné problémy. V skutočnosti to znamená, že existuje rozsah aplikácií, kde je simulácia presná a rozsah aplikácií, kde použitie reálneho robota je jednoduchšie a nevyhnutné. Je teda zrejmé, že zdokonaľovaním simulácie sa tento rozsah „vhodných“ aplikácií zväčšuje.

Medzi nevýhody simulačného prostredia môžeme zaradiť nedostatok rušivých vplyvov, nepresnosť modelov a potrebu nastavenia parametrov. V skutočnosti musíme s reálnym robotom stráviť veľké množstvo času, ktorý nezávisí od toho, aká dobrá bola simulácia. Je to hlavne preto, lebo simulácia má pred sebou ešte veľa potrebnej práce aby bola použiteľná a dostatočne predstavovala realitu. Reálny svet je nepredvídateľný a komplexný s množstvom rušivých vplyvov, ktoré ovplyvňujú senzory. Tieto vplyvy sa v simulácii nedajú dostatočne predvídať. Tak isto, veľké množstvo udalostí z reálneho sveta stále nie je vysvetlených alebo sú ťažko modelovateľné. To spôsobuje, že nie sme schopní namodelovať všetko dostatočne presne a najmä nie v reálnom čase (napr. modelovanie ultrazvukových snímačov alebo pohyb pri nízkych rýchlostiach). V simulovanom prostredí nie je ťažké dosiahnuť, aby sa robot po prostredí pohyboval a vytváral interakcie s ostatnými objektmi v prostredí. Napriek tomu simulácia stále vyžaduje nesmierne úsilie na nastavenie parametrov objektov tak, aby sa správali ako v reálnom svete. Táto úloha sa v MRS použitím technológie AGEIA PhysX [1] zjednodušila, ale stále nie je automatizovaná. Samotný simulačný proces MRS sa skladá z týchto častí:

- **Simulation Engine Service** – služba zodpovedná za renderovanie objektov (entities) a postup simulačného času pre fyzikálne jadro. Sleduje stav celého simulačného prostredia.
- **Managed Physics Engine Wrapper** – poskytuje stručné, riadené prepojenie pre simuláciu fyziky.
- **Native Physics Engine Library** – umožňuje hardvérovú akceleráciu pomocou AGEIA PhysX SDK, ktorý umožňuje použiť hardvérovú akceleráciu použitím AGEIA PhysX procesora.
- **Entities** – reprezentujú hardvérové a fyzikálne objekty v simulačnom prostredí. MRS už obsahuje mnoho vopred definovaných objektov. Používatelia ich môžu jednoducho a rýchlo zostaviť a tým vytvoriť simulované robotické platformy v rôznych virtuálnych prostrediach.

Dva najčastejšie využívané simulačné softvérové komponenty týkajúce sa fyzikálneho modelu sú objekt (entity) a služba (service). Objekt je softvérový komponent, ktorý je prepojený s fyzikálnym jadrom a s renderovacím jadrom. Služba (Service) používa tie isté typy a operácie ako služba, ktorú simuluje. Objekty simulovaného prostredia je možné vytvoriť viacerými spôsobmi. Jeden spôsob je editovať simulované prostredie priamo v spustenom simulačnom procese (Obr. 5), druhý spôsob spočíva v programovom riešení simulovaných objektov, ktoré možno vytvoriť pomocou jazyka C# v Microsoft Visual Studiu .

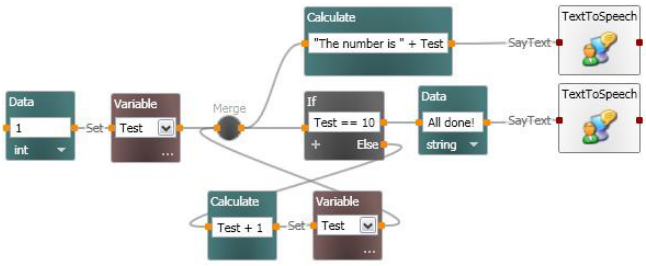


Obr. 5 Príklad tvorenia objektov priamo počas behu simulačného prostredia

Fig. 5 Creation of objects in simulation scene example

Vizuálny programovací jazyk

Silným nástrojom MRS je vizuálny programovací jazyk [2] [4] (Visual Programming Language – VPL). VPL je grafické prostredie na tvorenie aplikácií. Toto prostredie je založené na toku dát (namiesto toku príkazov ako je to pri klasickom programovaní). Tok dát môžeme prirovnať k výrobnej linke, kde sa s dátami vykoná daná operácia až keď dorazia do daného bloku (istý typ udalostného sytému). Jazyk VPL je mimoriadne vhodný na programovanie rôznych súbežných alebo rozptýlených výrobných prostredí. Navyše VPL je nenáročný na vedomosti a preto je vhodný hlavne pre začiatočníkov, ktorí ovládajú aspoň základy týkajúce sa premenných a logiky. VPL však nie je limitovaný iba pre začiatočníkov. Kompozičná povaha tohto programovacieho jazyka môže vyhovovať aj skúseným programátorom pre rýchlu a prehľadnú tvorbu kódu. Aj keď je toolbox tohto jazyka prispôsobený na vývoj robotických aplikácií, podstata architektúry nie je limitovaná na programovanie robotov a môže byť využitá aj pre iné aplikácie. Vďaka týmto vlastnostiam môže VPL vyhovovať širokému spektru ľudí od študentov cez nadšencov až po web developerov a profesionálnych programátorov. Tok dát vo VPL je zložený z postupnosti aktivít, ktoré sú reprezentované ako bloky so vstupmi a výstupmi. Spojenie rôznych vstupov a výstupov sa vykonáva cez čiary, ktoré posielajú dáta z jednej aktivity do druhej (Obr. 6).

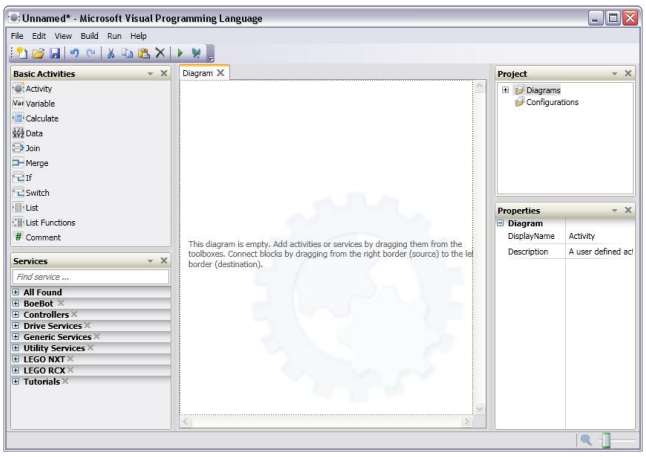


Obr. 6 Ukážka programu vo VPL

Fig. 6 Program in VPL example

Po spustení vizuálneho programovacieho jazyka sa zobrazí okno (Obr. 7) obsahujúce niekoľko menu, toolboxov a okno diagramu so záložkami. Toolboxy zobrazujú obsah aktuálneho súboru projektu a aktivít, ktoré možno použiť pri tvorení nového programu – diagramu. Začatie tvorby diagramu je jednoduché. Stačí chytiť požadovaný blok aktivity alebo blok služby z toolboxu a presunúť ho do oblasti diagramu (princíp Drag&Drop). Pospájanie blokov funguje na rovnakom princípe. Ak chceme pridať novú aktiviu stačí dvojklik na jej názov v toolboxe. Ak podržíme kurzor myši nad nejakou položkou v toolboxe, zobrazí sa nám krátky text obsahujúci opis ako má byť daná aktivita použitá. Všetky aktivity je možné zmazať, kopírovať, vystrihnúť alebo prilepiť zo schránky použitím klasických klávesových skratiek, pomocou príkazov v Edit menu alebo pomocou kontextového menu, ktoré sa zobrazí pri kliknutí pravým tlačidlom myši na aktivitu. Po spustení vizuálneho programovacieho jazyka sa zobrazia na obrazovke štyri základné panely:

- **Basic Activities** – toolbox obsahujúci bloky so základnými operáciami pre kontrolu toku dát a vytváranie dát a premenných. Okrem týchto obsahuje ešte aj blok pre komentár (Comment), ktorý je možné umiestniť ľubovoľne v okne diagramu a vpísať komentár.
- **Services** – tento toolbox zobrazuje služby, ktoré sú kompatibilné s VPL. Ak z neho vyberieme službu, ktorá už v diagrame existuje, MRS sa spýta, či chceme vytvoriť novú službu alebo chceme vytvoriť odkaz na službu, ktorá už v diagrame existuje. Ako príklad možno uviesť situáciu, keď chceme použiť rovnakú službu pre súbor snímačov alebo motorov s inou konfiguráciou nastavení. Ak chceme vyhľadať nejakú konkrétnu službu, stačí napísať časť jej mena do políčka nad zoznamom služieb. Tento filter prehľadáva všetky služby a zobrazí len tie, ktorých názov alebo opis obsahujú danú časť. Do filtra je možné zadať aj viac slov oddelených medzerou, pričom filter vyhľadá služby obsahujúce prvé, druhé alebo obidve slová. Ak potrebujeme vyhľadať len názvy obsahujúce obidve slová, musíme podmienku napísať v tvare *"text1+ text2"*. Ak podmienku napíšeme v tvare *"text1 - text2"*, filter vyhľadá služby, ktoré obsahujú v názve alebo opise prvé slovo, ale neobsahujú druhé slovo. Ak si chceme výsledky vyhľadávania uložiť, stačí kliknúť na malý štvorček vedľa opisu *All Found*. Kliknutím na tento symbol sa vytvorí v toolboxe nová sekcia obsahujúca výsledky hľadania. Ak chceme zvoliť iný názov, musíme do políčka filtra napísať za všetky podmienky *"=meno"*. Takto si môžeme vytvoriť vlastné skupiny služieb.
- **Project** – toto okno zobrazuje diagramy a konfigurácie súborov pre služby zahrnuté v danom projekte. Ak chceme niektorý diagram zatvoriť, musíme kliknúť na symbol *X* v záložke diagramu. Na opätovné zobrazenie diagramu stačí dvojklik na názov požadovaného diagramu. Pre zmenu názvu diagramu musíme kliknúť na meno diagramu v okne *Project* a následne zmeniť meno diagramu v okne *Properties*. Pre pridanie nového diagramu potrebujeme vyvolať kontextové menu kliknutím pravého tlačidla myši na položku *Diagrams* v okne *Project*. V kontextovom menu máme následne možnosť pridať ďalší diagram kliknutím na *Add Diagram*. Pre odstránenie diagramu z projektu musíme vyvolať kontextové menu kliknutím pravého tlačidla na meno požadovaného diagramu a vybrať položku *Delete*.
- **Properties** – toto okno zobrazuje vlastnosti pre aktuálne označenú položku, čo nám umožňuje upraviť nastavenia danej služby alebo aktivity.

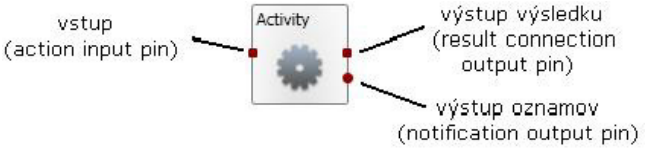


Obr. 7 Úvodná obrazovka po spustení VPL

Fig. 7 VPL entry screen

Základné bloky používané vo VPL sú:

- **Activity** – blok aktivity nám umožňuje vytvoriť vlastné aktivity, ktoré môžu obsahovať vlastný súbor dataflow diagramov. Tieto vytvorené diagramy možno použiť ako samostatné bloky v iných diagramoch. Aktivity môžu byť taktiež skompilované ako služby a byť použité s ostatnými službami. Aktivity môžu predstavovať hotové služby, funkcie, riadenie toku dát alebo iné časti kódu. Aktivity sa môžu skladať taktiež z iných aktivít. Takáto modulárnosť umožňuje použiť už skôr vytvorenú aktivitu ako stavebný blok. Z toho vyplýva, že aplikácia vytvorená vo VPL je taktiež aktivitou. Aktivity sú spojené medzi sebou cez konektory (piny). Konektor na ľavej strane bloku predstavuje port pre prichádzajúce správy (vstupy) a konektory na pravej strane predstavujú porty pre odchádzajúce správy (výstupy). Blok dostáva správy pomocou vstupného konektora. Tieto vstupné konektory predstavujú spojenie s vnútornou funkciou, ktorá sa nazýva akcia alebo obslužný program (handler). Blok aktivity spracuje dáta prichádzajúcej správy hneď, ako prijme relevantnú prichádzajúcu správu. Ak chceme vstupné dáta poslať ešte inej aktivite, musíme ich skopírovať a pripojiť na výstupný konektor (všetky dáta poslané aktivite aktivita spotrebuje). Bloky môžu mať aj viacero vstupných konektorov a každý z nich má vlastnú sadu výstupných konektorov. Výstupné konektory sa rozdeľujú do dvoch typov (Obr. 8):
  - výstup výsledku (result, response output) – konektor sa označuje malým štvorcom. Používa sa v situáciách, keď je výstupná správa posielaná ako výsledok nejakej špecifickej prijatej správy. Tento typ výstupu môže posielať iba jednu správu (odozva na vstupnú správu).
  - výstup oznamov (notification output) – konektor sa označuje malým krúžkom a môže posielať informácie vychádzajúce zo vstupnej správy, častejšie však spúšťa správu ako zmenu svojho vnútorného stavu. Tento výstup môže generovať správu aj niekoľkokrát bez ohľadu na počet vstupných správ. Používa sa teda pre posielanie dát správy bez opakovanej požiadavky na stav aktivity.



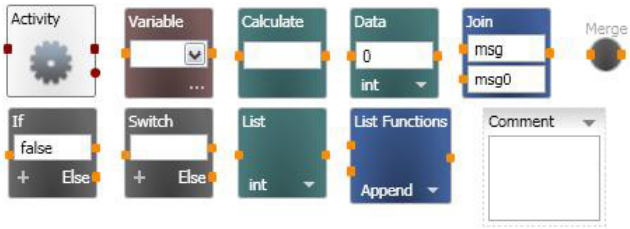
Obr. 8 Typy vstupov a výstupov

Fig. 8 Input/output types

- **Variable** – aktivita umožňujúca vytváranie premenných, načítanie alebo nastavenie ich hodnoty. Výber premennej možno uskutočniť jej výberom z vysúvacieho zoznamu. Ak ešte nie je vytvorená žiadna premenná alebo chceme vytvoriť novú premennú, vyberieme zo zoznamu položku *Define Variables* alebo tú istú položku z menu *Edit.* Táto akcia nám otvorí nové okno, kde môžeme pridať premennú a upravovať typy a mená všetkých premenných. Mená premenných sú citlivé na veľké a malé písmená a musia začínať vždy písmenom. Obsahovať môžu len alfanumerické znaky. Činnosti bloku Variable podporujú *GetValue* spojenie na zistenie hodnoty premennej a *SetValue* spojenie pre nastavenie hodnoty premennej.
- **Calculate** – táto aktivita poskytuje základné aritmetické operácie so zadanými výrazmi, ktoré môžu obsahovať číselné hodnoty, hodnotu správy a jej dátových členov alebo vopred zadané hodnoty poskytnuté inými službami z diagramu. Po kliknutí do políčka pre text v bloku *Calculate* sa zobrazí zoznam zahŕňajúci hodnoty prichádzajúcej správy, dátových členov a rovnako aj vopred definované hodnoty poskytnuté ostatnými službami. Pre číselné dáta môžeme použiť nasledovné operácie:
  - Sčítanie (**+**)
  - Odčítanie (**-**)
  - Násobenie (*)
  - Delenie (**/**)
  - Modulo (**%**)

  Operátor plus (+) možno použiť na spojenie textových reťazcov rovnako ako aj na skombinovanie textu a číselných dát, napríklad: *"Výsledok je "*+ *x*. Na presnejšie určenie postupnosti výpočtu je možné použiť okrúhle zátvorky. Ako logické operátory môžeme použiť operácie:
  - And (&&)
  - Or (**|** **|**)
  - Not (!)
- **Data** – tento blok aktivity sa používa na dodanie jednoduchých dát pre inú aktivitu alebo službu. Pre definovanie požadovaného typu dát je potrebné vybrať daný typ zo zoznamu pod textovým poľom a do tohto textového poľa zadať požadovanú hodnotu.
- **Join** – aktivita pre spojenie dvoch alebo viacerých tokov dát. Významne sa odlišuje od aktivity *Merge* tým, že táto aktivita spája dáta správ zo všetkých vstupných pripojení a posiela tieto dáta ďalej až vtedy, keď prijme všetky správy zo všetkých vstupných pripojení. Text, ktorý sa píše do textových polí bloku predstavuje názvy lokálnych premenných, ktoré reprezentujú správu. Premenné možno používať priamo alebo použitím bodky (dot notation), napríklad *"student.meno"*.
- **Merge** – táto aktivita slúži na jednoduché zlúčenie dvoch a viacerých tokov dát do jedného. Neexistuje žiadna podmienka alebo závislosť na type prepúšťaných správ. Úlohou tejto aktivity je len poslať správu do ďalšieho bloku.
- **If** – aktivita poskytuje možnosť výberu výstupu prichádzajúcej správy na základe zadanej podmienky. Ak je podmienka splnená, na prvý výstupný kontakt sa pošle prichádzajúca správa aj so svojimi dátami. Ak nie je splnená, použije sa výstup typu *Else*. Výraz v podmienke môže využívať nasledovné operátory:
  - je rovné **= alebo ==**
  - nerovná sa **!= alebo <>**
  - menšie ako **<**
  - väčšie ako **>**
  - menšie alebo rovné ako **<=**
  - väčšie alebo rovné ako **>=**

  Je možné použiť operátory ako pri aktivite *Calculate*, potom sa bude vyhodnocovať výsledok celého výrazu.
- **Switch** – aktivita slúži na určenie trasy správy podľa toho, či prichádzajúca správa spĺňa podmienky dané výrazom v textovom poli. Ak chceme pridať ďalšiu podmienku, stlačením znaku **+** sa pridá ďalšie textové pole. Ak potrebujeme odstrániť niektorú podmienku, stačí kliknúť na symbol *X* v danom riadku.
- **List** – aktivita sa používa na vytvorenie prázdneho zoznamu dátových položiek. Na vytvorenie zoznamu potrebujeme vybrať typ položky z vysúvacieho menu. Pre pridanie údajov do zoznamu je potrebné použiť blok *List Functions*. Ak chceme zoznam použiť aj niekde inde v diagrame, môžeme to urobiť pomocou vytvorenia premennej typu *List*, použitím bloku *Variable.*
- **List Functions** – aktivita, ktorá nám umožňuje modifikáciu už existujúceho zoznamu. Použitím vysúvacieho menu bloku možno zvoliť funkciu, ktorú chceme aplikovať na zoznam.
- **Comment** – tento blok nám umožňuje pridať do diagramu blok textu. Nepodporuje žiadne vstupy ani výstupy, a preto ho možno umiestniť v diagrame ľubovoľne. Želaný text stačí napísať do textového poľa.

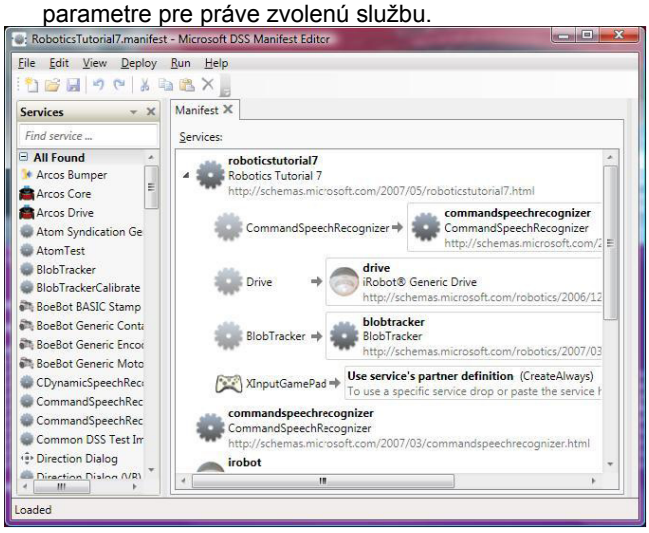


Obr. 9 Bloky jednotlivých základných aktivít
Fig. 9 Basic activities blocks

DSS Manifest Editor

DSS Manifest Editor (DSSME) [2] je aplikácia, ktorá slúži na tvorbu a editáciu manifestov. Manifesty sú XML súbory, ktoré obsahujú zoznam služieb a ich konfiguráciu. Používajú sa na spúšťanie služieb. Okno editora obsahuje menu pre vytváranie, ukladanie, načítanie manifestov a na upravovanie položiek z manifestu. Projekt, ktorý vytvoríme pomocou editora sa skladá z manifestu a z pripojených konfiguračných súbrov. DSSME navyše obsahuje panel nástrojov pre rýchly prístup k najbežnejším príkazom. Okrem toho obsahuje ešte ďalšie tri panely s nástrojmi:

- **Service toolbox window** – zobrazuje dostupné služby, ktoré môžeme zahrnúť do manifestu.
- **Project toolbox window** – zobrazuje súbor manifestu a pripojené konfiguračné súbory.
- **Properties toolbox window** – zobrazuje konfiguračné parametre pre práve zvolenú službu.

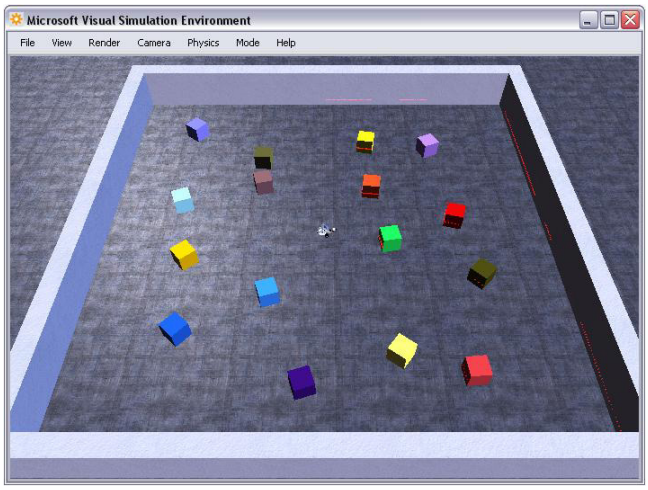


Obr. 10 Manifest tvorený pomocou DSS Manifest Editora

Fig. 10 Manifest made by DSS Manifest Editor

Záver

Ako vyplýva z predchádzajúcich kapitol, MRS je vývojové prostredie určené na simuláciu robotických systémov. Nielenže je silným CAD nástrojom, ale uplatňuje sa aj v praxi. Dôkazom sú mnohé vytvorené projekty vedeckými pracovníkmi a tímami [5], ktoré sú dostupné na internete. V MRS sa teda dajú simulovať všetky špecifické skupiny robotiky: mobilné roboty, robotické manipulátory a aj kráčajúce mobilné roboty. Výhodou je teda možnosť vyvíjať inteligentné algoritmy správania sa bez znižovania životnosti hardvéru. MRS umožňuje aj vyvíjanie viacerých druhov algoritmov paralelne, čo môže byť veľkou výhodou napríklad v prípade, keď máme hardvér len po jednom kuse. MRS môže poslúžiť aj na pedagogické účely (kinematické štruktúry, riadenie, inteligentné algoritmy atď.). Napriek niektorým nevýhodám a chybám sa toto štúdio stáva silným nástrojom pri vývoji robotických systémov. Čím viac užívateľov toto štúdio bude používať, tým silnejším a dokonalejším nástrojom sa môže stať. Navyše, veľkou výhodou oproti iným simulačným nástrojom je možnosť toto štúdio voľne stiahnuť, existencia veľkého množstva už vyrobených modelov robotov a hardvérova nenáročnosť.



Obr. 11 Prostredie s robotom vytvorené pomocou programu v jazyku C#

Fig. 11 Environment with robot made by program in C#

Poďakovanie

Publikácia vznikla za podpory agentúry VEGA v projekte VEGA 1/3120/06.

Literatúra

[1] AGEIA Technologies: Advanced Gaming Physics Defining The New Reality In PC Hardware, March 2006, http://www.ageia.com/pdf/wp_advanced_gaming_physics.pdf

[2] MRS User Guide, http://msdn2.microsoft.com/en-us/library/bb881626.aspx

[3] Microsoft Robotics Studio Now Available to Provide Common Development Platform, http://www.robocup-us.org/Media/2006-MSRS-release.pdf

[4] Microsoft Robotics Studio Community, http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=383&SiteID=1

[5] LINGYUN H., CHANGIJU Z., BI W., TIANWU Y.: Humanoid Online Locomotion Generation based on Markov Chain with Microsoft Robotics Studio

Abstract

This paper contents basic information about Microsoft Robotics Studio. It is oriented on basic knowledge needed for using of this studio. There are also mentioned advantages and disadvantages of this software tool.

Ing. František Duchoň, Bc. Martin Štrenger

Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
frantisek.duchon@stuba.sk, mastr@ynet.sk