Universitat
Autònoma
de Barcelona

ANÀLISI DEL MICROSOFT ROBOTICS STUDIO

Memòria del Projecte Fi de Carrera
d'Enginyeria Informàtica
realitzat per
Carlos López de Toro
i dirigit per
Lluís Ribas Xirgo
Bellaterra, 18 de Setembre de 2008

El sotasignat, Lluís Ribas Xirgo,
Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon la present memòria ha estat realitzat sota la seva direcció per en Carlos López de Toro.

I per a que consti firma la present.

Signat: Lluís Ribas Xirgo

Bellaterra, 18 de Setembre de 2008

Agraïments

Amb aquestes paraules voldria agrair a tothom l'ajuda i el suport que m'ha donat, d'una manera o una altre, fent possible que aquest projecte arribés a ser una realitat.

Dono les gràcies a la meva família, especialment a la meva mare, i als meus amics, que sempre m'han donat ànims, i sempre s'han interessat pel procés. Les dono també a Laura especialment, que ha estat qui més m'ha animat, estant allà en tot moment, oferint-me la seva ajuda constantment i preocupant-se tant com jo mateix pel bon fi d'aquest projecte.

Per últim voldria donar les gràcies al meu director de projecte, en Lluís Ribas, que gairebé sempre ha estat disponible per a guiar-me i corregir-me, tot i ser vacances.

Signat: Carlos López de Toro

Taula de continguts

1 Introducció

Cada cop són més presents en el nostre món els sistemes robòtics, i Microsoft ens ofereix una eina de programació i simulació de robots. En aquest primer capítol es plantegen les motivacions que han portat a realitzar aquest projecte fi de carrera, els seus objectius, sub-objectius i tasques així com una planificació inicial del projecte en relació a aquestes tasques i un anàlisi de possibles riscos que impedeixin la seva consecució.

1.1 Motivació

En l'actualitat l'esser humà viu en un entorn en el que, cada cop més, l'aplicació de la robòtica és més important. Existeixen molts robots amb els que treballem diàriament, aquests robots ens faciliten la vida: realitzen tasques repetitives, difícils o fins i tot impossibles per a humans. Esdevé un tema molt important el desenvolupament de software de control per aquests.

Al mercat existeixen diverses possibilitats per a realitzar aquesta tasca de desenvolupament i, tot i que Microsoft no és la primera empresa que s'hi posa, el fet que una companyia com aquesta ofereixi un entorn de desenvolupament, com el Microsoft Robotics Studio, val la pena tenir-ho en compte.

D'altra banda, per poder fer servir el Microsoft Robotics Studio és convenient invertir moltes hores en el seu aprenentatge: Microsoft ja ofereix força material per poder-ho fer, però és interessant contemplar la possibilitat de generar nou material, un material que pugui fer-se servir en un curs d'un semestre que no hagi de ser especialment dedicat a la robòtica.

Per tant, es pretén desenvolupar una mena de guia d'ús simple per a l'eina que permeti a usuaris novells poder treballar-hi per fer coses relativament simples però sense haver d'estar instruïts especialment en la robòtica.

1.2 Objectiu principal

La idea principal d'aquest projecte és la de fer una anàlisi complerta del programa Microsoft Robotics Studio (MRS). Aquest software, a grans trets, és un entorn de programació (essent de Microsoft, basat en Windows) per al control i simulació de robots.

És un software que va sortir al mercat a finals de 2006 i, encara que cada cop més, el material educatiu que n'existeix no és molt, si més no, té carències, gairebé tot és proporcionat per Microsoft, pel que seria interessant millorar-lo.

L'objectiu general i resumit d'aquest projecte és el d'elaborar un entorn docent complert accessible mitjançant web per a l'aprenentatge i comprensió de l'aplicació en totes les seves parcel·les, compost per anàlisis dels diferents apartats, *tutorials*,

pràctiques guiades, errors freqüents, exemples, casos d'ús, etc. Aquest entorn ha de millorar o completar el que es pot trobar actualment a Internet, i permetre a l'usuari assolir els coneixements necessaris de forma gradual per a utilitzar l'MRS.

En aquest projecte concret es dissenyarà un *"walk-through",* és a dir, una guia pas a pas d'aprenentatge sobre l'aplicació, que inclogui aquests elements per a la inicialització a l'aplicació, representant per a l'usuari una **guia d'introducció** a l'entorn.

1.2.1 Subobjectius

Partint de l'objectiu principal del projecte, els subobjectius del mateix són:

- Un informe crític de Microsoft Robotics Studio on s'analitzin les seves característiques i que contingui una comparativa amb les eines similars més importants així com un anàlisi del material docent que n'existeix
- Una guia d'ús de l'eina que en permeti la docència i que incorpori una sèrie d'exercicis guiats que facilitin a l'usuari l'assoliment dels coneixements

Aquest serà el resultat que quedarà a la conclusió del desenvolupament d'aquest projecte.

1.3 Tasques

En aquest apartat es portarà a terme la descripció i planificació de les tasques. La idea principal és la de dividir el projecte en dos blocs:

- Estudi teòric: Estudiar tot el necessari per a desenvolupar aquest projecte i entendre què farà falta en cada moment.
- Desenvolupament guia: Desenvolupar tot el *walk-through* resultat de l'estudi teòric.

Aquestes línies són molt generals, per tant, això ens porta a plantejar un seguit de punts en que podem dividir el desenvolupament del projecte, de la següent forma:

- Definició d'objectius i planificació
- Estudi de l'estat de l'art de software de control i simulació de robots
- Analitzar i aprofundir en l'entorn MRS
- Estudi de l'estat de l'art del material educatiu sobre MRS
- Anàlisi de les mancances i necessitats del material educatiu
- Elaboració de una guia d'introducció a l'aplicació
- Creació d'una senzilla web per a presentar i fer més accessible la guia

Aquests punts representen uns objectius globals poc acotats, que al llarg del desenvolupament del projecte poden veure's modificats, donada l'evolució del mateix.

A partir dels punts establerts definirem les tasques a realitzar afegint, a més, imprevistos, tasques de suport (buscar informació per a redactar la memòria, consultar

altres projectes, reunions i correus amb el tutor del projecte, instal·lació de software, etc), elaboració d'annexos i l'informe previ, que és una tasca ja realitzada.

Aquest projecte consta d'una forta base teòrica, fet pel qual el propi assoliment d'objectius deriva en la memòria, és a dir, s'anirà realitzant la memòria a mida que es van portant a terme les tasques i objectius, i no al final després d'haver assolit els objectius.

Les tasques comprenen, com a part de les mateixes, els requeriments per a poder desenvolupar-les (documentació, lectures, etc) i la pròpia confecció de la part corresponent a la memòria.

1. Informe previ
2. Definició d'objectius i planificació
    a. Documentació sobre l'elaboració de memòries i lectura d'algunes
    b. Elaborar capítol inicial
3. Estudi de l'estat de l'art de software de control i simulació de robots
    a. Estudi de les diferents alternatives a MRS per al control i simulació de robots
    b. Elaborar un informe sobre elles
4. Analitzar i aprofundir en l'entorn MRS
    a. Documentació sobre MRS: possibilitats, requeriments, etc
    b. Seguir les diferents guies oficials i l'ajuda que incorpora l'aplicació
    c. Cercar a Internet altre ajuda
    d. Fer proves
    e. Realitzar un anàlisi de MRS
5. Estudi de l'estat de l'art del material educatiu sobre MRS
    a. Cercar a Internet
    b. Fer un estudi d'aquesta informació
6. Anàlisi de les mancances i necessitats del material educatiu
    a. Trobar els punts febles de la informació que n'existeix o està malament explicada
    b. Fer un informe
7. Desenvolupament de una guia d'introducció a l'aplicació
    a. Definir les necessitats de la guia: tant existents com nous
    b. Definir les parts de la guia
    c. Elaborar-la
8. Creació d'una senzilla web per a presentar i fer més accessible la guia
    a. Buscar una manera eficient de portar la guia a la web
    b. Disseny i elaboració de la pàgina web
9. Tasques de suport
    a. Reunions
    b. Correus
    c. Instal·lació software
10. Finalitzar memòria
    a. Introducció i Conclusions
11. Elaboració d'annexos
12. Imprevistos

S'han englobat les tasques en les parts principals de les mateixes, tractant de no arribar a entrar massa en detall.

Tot seguit es mostra una planificació inicial per a seguir a l'hora de desenvolupar el projecte. Aquest projecte es desenvoluparà al llarg de l'estiu de 2008 amb l'objectiu de presentar-lo al Setembre, pel que existeix una limitació temporal prou estricta.

Tasca	Descripció de l'activitat	Duració
1	Informe previ	30
2	Definició d'objectius i planificació	16
3	Estudi de l'estat de l'art de software de control i simulació de robots	30
4	Analitzar i aprofundir en l'entorn MRS	48
5	Estudi de l'estat de l'art del material educatiu sobre MRS	30
6	Anàlisi de les mancances i necessitats del material educatiu	20
7	Elaboració de una guia d'introducció a l'aplicació	60
8	Creació d'una senzilla web	20
9	Tasques de suport	30
10	Finalitzar memòria	8
11	Elaboració d'annexos	10
12	Imprevistos	10

Com s'ha dit anteriorment, l'elaboració de la memòria no és una tasca independent en sí mateix, si no que s'anirà confeccionant a mida que s'avanci el procés, per això no existeix una tasca concreta que en faci referència.

Seguint aquesta estimació la durada prevista del projecte és de 312 hores, comptant que la tasca 1 *(Informe Previ)* ja es va realitzar, ens deixa 282 hores, el qual entra dins de les, aproximadament, 280 hores de las que es disposa, comptant 7 setmanes a raó de 40 hores setmanals.

Potser sembla una mica per sota de les hores que s'estima ha de durar un projecte però és el temps del que es disposa i s'espera sigui suficient per a acomplir els objectius.

1.4 Anàlisi de costos

En el procés de desenvolupament del projecte existeixen diversos costos: hardware, software, temps, altres.

Els costos econòmics del projecte no són elevats, havent de comptar només el cost del **hardware** necessari, del que ja es disposava prèviament. El PC utilitzat té les següents característiques:

- Memòria RAM: 2 GB
- Processador: Core 2 Duo 1,86Ghz
- Disc Dur: 150Mb instal·lació i 2GB lliures
- Unitat de DVD-ROM
- Monitor SVGA

A més es contempla la possibilitat d'utilitzar un controlador de XBox360[1] per a fer proves amb l'MRS, del que ja es disposava i l'aplicació permet utilitzar-ho.

A nivell de recursos **software** tot ha estat gratuït, ja que l'entorn MRS no suposa cap càrrec en el cas d'una utilització no-comercial (al capítol d'anàlisi de MRS es parlarà de les llicencies existents).

El cost, en **hores de treball**, estarà al voltant de les 280h, comptant que es disposa d'unes 7 setmanes a raó de 40 hores setmanals, perquè s'ha de realitzar gairebé tot el projecte en els mesos d'estiu, amb la fita del 15 de Setembre. Tot i això només és una estimació de temps, que pot veure's alterat.

Les **hores del supervisor** del projecte, en Lluís Ribas-Xirgo, són més difícils d'estimar. Comptant que li dediqui 4 hores setmanals tenim la xifra de 28 hores.

Altres costos són els de transport i connexió a Internet.

1.4.1 Beneficis

Els beneficis del projecte, però, no seran econòmics, si no d'aprenentatge propi en l'elaboració i investigació del projecte. Evidentment, també per a tractar d'aconseguir els 15 crèdits corresponents al projecte final de carrera de l'Enginyeria Superior en Informàtica.

1.5 Planificació de les tasques

La planificació feta a priori del projecte és la que es mostrarà a continuació. S'ha de notar que aquest és un apartat realitzat abans que el major gruix del projecte pel que només és una planificació teòrica, la qual pot veure's modificada durant l'evolució del projecte.

Dita planificació s'ha realitzat amb Microsoft Project (**Figura 1-1**).

[1] Xbox360: Videoconsola de sobretaula produïda per Microsoft

Id	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1	Informe previ	30 horas	jue 20/03/0E	mar 25/03/0E	Carlos
2	**Definició d'objectius i planificació**	**2 días**	**mié 23/07/08**	**jue 24/07/08**	**Carlos**
3	Documentació sobre l'elaboració de n	8 horas	mié 23/07/0E	mié 23/07/0E	Carlos
4	Elaborar capítol inicial	8 horas	jue 24/07/0E	jue 24/07/0E	Carlos
5	**Estudi de l'estat de l'art de software**	**3,75 días**	**vie 25/07/08**	**mié 30/07/08**	**Carlos**
6	Estudi de les diferents alternatives a	12 horas	vie 25/07/0E	lun 28/07/0E	Carlos
7	Elaborar un informe sobre elles	18 horas	lun 28/07/0E	mié 30/07/0E	Carlos
8	**Analitzar i aprofundir en l'entorn MR!**	**6 días**	**mié 30/07/08**	**jue 07/08/08**	**Carlos**
9	Documentació sobre MRS: possibilitat	16 horas	mié 30/07/0E	vie 01/08/0E	Carlos
10	Seguir les diferents guies oficials i l'a	8 horas	vie 01/08/0E	lun 04/08/0E	Carlos
11	Cercar a Internet altre ajuda	8 horas	lun 04/08/0E	mar 05/08/0E	Carlos
12	Fer proves	8 horas	mar 05/08/0E	mié 06/08/0E	Carlos
13	Realitzar un anàlisi de MRS	8 horas	mié 06/08/0E	jue 07/08/0E	Carlos
14	**Estudi de l'estat de l'art del material**	**3,75 días**	**jue 07/08/08**	**mié 13/08/08**	Carlos
15	Cercar a Internet	20 horas	jue 07/08/0E	mar 12/08/0E	Carlos
16	Fer un estudi d'aquesta informació	10 horas	mar 12/08/0E	mié 13/08/0E	Carlos
17	**Anàlisi de les mancances i necessita**	**2,5 días**	**mié 13/08/08**	**vie 15/08/08**	Carlos
18	Trobar els punts febles de la informa	12 horas	mié 13/08/0E	jue 14/08/0E	Carlos
19	Fer un informe	8 horas	vie 15/08/0E	vie 15/08/0E	Carlos
20	**Elaboració de una guia d'introducció**	**7,5 días**	**lun 18/08/08**	**mié 27/08/08**	Carlos
21	Definir les necessitats de la guia: tant	4 horas	lun 18/08/0E	lun 18/08/0E	Carlos
22	Definir les parts de la guia	4 horas	lun 18/08/0E	lun 18/08/0E	Carlos
23	Elaborar-la	52 horas	mar 19/08/0E	mié 27/08/0E	Carlos
24	**Creació d'una senzilla web per a pre**	**2,5 días**	**mié 27/08/08**	**vie 29/08/08**	Carlos
25	Buscar una manera eficient de portar	4 horas	mié 27/08/0E	mié 27/08/0E	Carlos
26	Disseny i elaboració de la pàgina wel	16 horas	jue 28/08/0E	vie 29/08/0E	Carlos
27	**Tasques de suport**	**3,75 días**	**lun 01/09/08**	**jue 04/09/08**	Carlos
28	Reunions	18 horas	lun 01/09/0E	mié 03/09/0E	Carlos
29	Correus	10 horas	mié 03/09/0E	jue 04/09/0E	Carlos
30	Instal•lació software	2 horas	jue 04/09/0E	jue 04/09/0E	Carlos
31	**Finalitzar memòria**	**1,25 días**	**jue 04/09/08**	**vie 05/09/08**	Carlos
32	Introducció i Conclusions	10 horas	jue 04/09/0E	vie 05/09/0E	Carlos
33	Elaboració d'annexos	10 horas	lun 08/09/0E	mar 09/09/0E	Carlos
34	Imprevistos	10 horas	mar 09/09/0E	mié 10/09/0E	Carlos
35	Suport supervisor	28 horas	lun 28/07/0E	lun 08/09/0E	Lluís

Figura 1-1

Tal com observem, s'ha estimat el *suport del supervisor*, en Lluís Ribas-Xirgo com les 28 hores comptades abans, de forma que dura tot el projecte treballant 4 hores a la setmana.

El diagrama de Gantt (**Figura 1-2)** derivat de dita planificació és el següent:

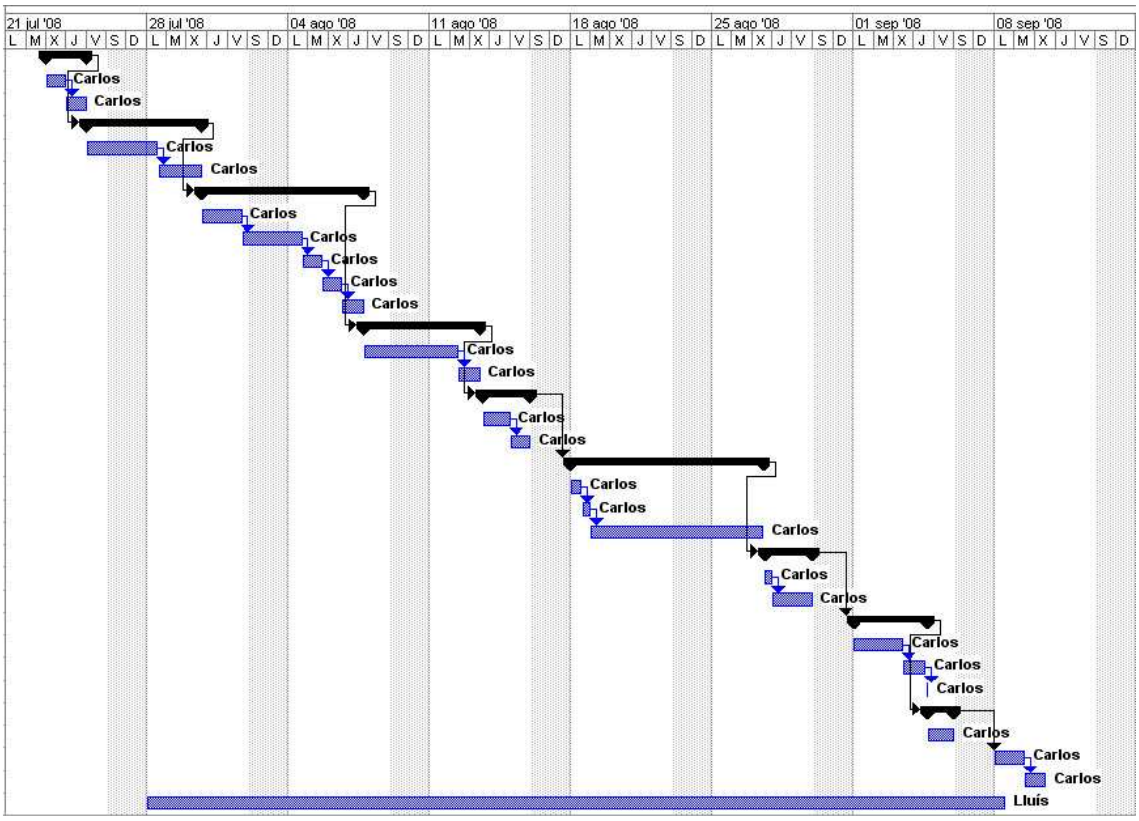

Figura 1-2

Al diagrama de Gantt no es mostra la tasca 1*(Informe previ)* ja que es va realitzar al Març i desvirtua el gràfic, el qual mostra des del 21 de Juliol al 14 de Setembre.

Cal destacar que la idea principal és la de treballar 8 hores diàries de dilluns a divendres. Aquest és, de fet, el *calendari de treball* que ens proporciona el Microsoft *Project* per defecte.

1.6 Riscos

La determinació de les carències del context educatiu del MRS és complicada per la constant aparició de nou material al voltant del programa i per la manca d'experiència docent de l'autor del treball.

L'elaboració de material docent complementari pot veure's compromès pel limitat nombre d'hores que s'hi pot dedicar en el marc d'un projecte de fi de carrera.

1.7 Alternatives

Les alternatives possibles en el desenvolupament d'aquest projectes es formen en dues vies:

- Software utilitzat: Les alternatives de software en aquest projecte no són aplicables ja que l'orientació del mateix no ofereix alternativa en aquest sentit, ja que ha de ser Microsoft Robotics Studio. En tot cas, es pot contemplar la comparació amb altres eines, com per exemple Player/Stage/Gazeebo, per a realitzar un anàlisi objectiu de MRS. Els programes més importants seran analitzats en el següent capítol de la memòria: *Anàlisi de l'estat de l'art*.
- Tipus de guia: L'altre punt del projecte que presenta alternatives és el tipus i presentació de la guia, els punts que contindrà, etc.

2 Estat de l'art

En aquest capítol es portarà a terme un anàlisi de l'estat de l'art de la robòtica en general i concretament de la programació de robots. Es farà una petita explicació dels robots més utilitzats que hi ha i quines són les seves característiques i posteriorment s'analitzarà el software de programació i control de robots més important que existeix: URBI, Player/Stage/Gazego, Webots i Microsoft Robotics Studio.

Aquest projecte està dedicat a l'anàlisi de Microsoft Robotics Studio, és per això que aquest s'explicarà amb més detall, prou detall com per a que el lector comprengui la seva estructura i funcionament.

Finalment es realitzarà una comparativa objectiva entre el software analitzat segons les seves característiques.

2.1 Robòtica

La robòtica (en angles, "robotics", que dóna nom al Microsoft **Robotics** Studio) és la ciència i tecnologia dels robots, així com el seu disseny, fabricació i aplicació.

Un robot (paraula que procedeix del txec "robota"-treball forçat-) és una màquina o aparell electrònic programable que té la capacitat de manipular objectes i realitzar operacions que abans eren possible només mitjançant persones.

La estructura d'un robot normalment és mecànica i se sol anomenar "cadena cinemàtica". Aquesta cadena està composa de diversos elements (Figura 2-1):

- Enllaços. Els seus "ossos"
- Actuadors. Els seus "músculs"
- Juntes. Les "articulacions". Poden tenir un o més graus de llibertat[2] (en la Figura 2-1 cada una només té un grau de llibertat)

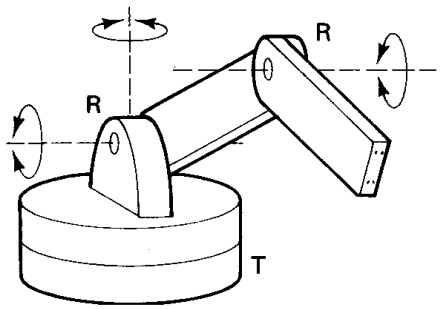

Figura 2-1. Exemple d'un braç robòtic amb 3 graus de llibertat

[2] Grau de llibertat: Mínim número de coordenades necessàries e independents per a determinar completament la posició de totes les parts d'un sistema en un instant

Un robot ha de tenir la majoria (si no totes) les següents característiques:

- Artificial
- Interactua amb l'entorn
- Té alguna habilitat de prendre decisions basades en l'entorn
- És programable
- Es mou amb un o més eixos de rotació
- Fa moviments amb destresa i coordinació

2.1.1 Història

El terme "robot" es va utilitzar per primer cop el 1920 pel txec Karel Capek en la seva obra "R.U.R."(Rossum Universal Robots). A la seva obra, però, el terme s'aplicava a humans artificials, de fet, la paraula "robot" se sol associar a humans mecànics. El terme "androide" pot fer referència a un d'aquests essers, però el terme cyborg és una criatura que combina parts orgàniques i mecàniques.

El primer en introduir el terme "robòtica" com la disciplina científica encarregada de construir i programar robots va ser Isaac Asimov[3] el 1942 en una sèrie de relats. A més, aquest autor va plantejar que les accions que desenvolupa un robot han de ser dirigides per una sèrie de normes morals, les que s'anomenen "les tres lleis de la robòtica":

1. Un robot no pot fer mal a un ésser humà, o per inacció, permetre que un pateixi mal
2. Un robot ha de fer cas les ordres d'un ésser humà, a no ser que entrin en conflicte amb la primera llei
3. Un robot ha de protegir la seva pròpia existència, sempre que no entri en conflicte amb la primera o segona llei

D'altra banda, des de la generalització de la utilització de la tecnologia en processos de producció amb la Revolució Industrial es va tractar de construir dispositius automàtics que ajudessin o substitueixin l'home.

D'aquests dispositius van destacar els "*Jaquemarts*"(Figura 2-2-2), ninots de dues o més posicions que fan cops a campanes accionats per mecanismes de rellotgeria asiàtics.

[3] Isaac Asimov (1920-1992): escriptor i bioquímic nord americà nascut a Rússia que va recollir gran èxit amb obres de ciència-ficció, historia i divulgació científica.

Figura 2-2. Fotografia de Jaquemarts reals

També es van fer servir robots amb una sola roda per a navegació i planejar rutes. Més tard es van començar a dissenyar robots amb potes per a que poguessin caminar. Aquests robots presenten una gran adaptabilitat a tot tipus de superfícies. Amb més de 4 potes són estables, el que fa que treballar amb ells sigui senzill. Molt recentment s'estan aconseguint robots de dues cames (Figura 2-3).

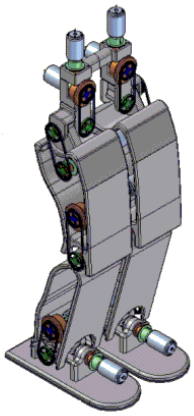

Figura 2-3. Exemple de cames robòtiques

Al 2002 *Honda* i *Sony* van començar a vendre robots com a mascotes, robots amb forma de gos o de serp es troben en una fase de producció molt amplia. L'exemple més famós és el robot *Aibo* de Sony (Figura 2-4).

Figura 2-4. Robot Aibo al costat d'un gos real

2.1.2 Actualitat

A l'actualitat s'utilitzen els robots, en gran part, per a realitzar tasques brutes, perilloses, difícils o repetitives pels humans. Aquestes característiques normalment deriven en un robot industrial utilitzat en línies de producció (Figura 2-5). Altres aplicacions són les de neteja de residus tòxics, exploració espacial, mineria... En general són tasques que un robot realitza millor que un humà, ja que aporten velocitat, precisió, productivitat i robustesa.

Figura 2-5. Robot manipulant material a una planta industrial

El principal mercat on s'utilitzen robots actualment és en la manufactura, on els robots articulats (semblants al braç humà) són molt útils. Es fan servir en aplicacions com soldar, pintar o carregar maquinària. El 1995 funcionaven uns 700.000 robots en el mon industrialitzat. Però, més de 500.000 s'empraven només al Japó, 120.000 en Europa Occidental i uns 60.000 a Estats Units.

La industria de la automoció (Figura 2-6) és una de les que més utilitza aquest tipus de robots, on es substitueix el treball humà en moltes de les tasques que desenvolupen des de fa més de 30 anys. *General Motors* utilitza al voltant de 16.000 robots.

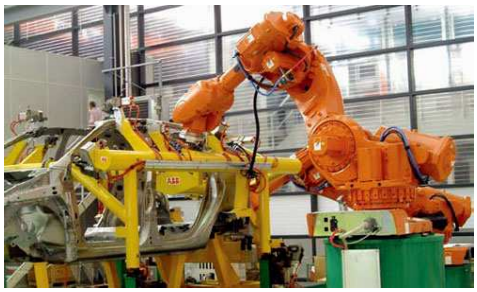

Figura 2-6. Robot en una cadena de muntatge

Però no tot són tasques d'aquesta mena, també s'utilitzen els robots per a realitzar tasques menys rudimentàries, concretament en el camp de la medicina (Figura 2-7) .

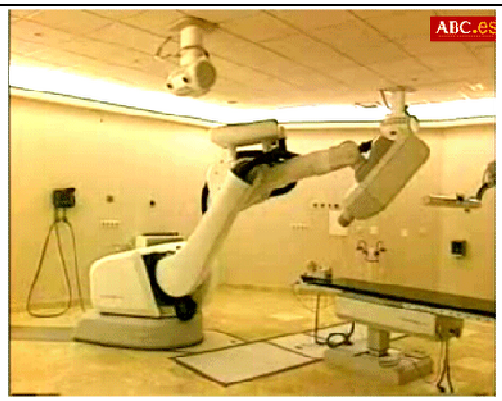

Figura 2-7. Robot emprat en medicina

2.1.3 Efemèrides

- El 1954 es va enregistrar per primer cop una patent d'un robot al Regne Unit. *George C. Devol* va patentar el primer robot als Estats Units el 1961
- El 1956, dos amics es van trobar a una festa i van estar parlant de les novel·les d'en Asimov i la possibilitat real de construir robots. Aquests dos amics eren *George C. Devol* i *Joseph F. Engelberger*, futurs fundadors de *Unimation* i pares de la robòtica moderna.
- Aquest mateix any, al *Dartmouth College* es mostrà el *Logic Theorist*, una màquina de intel·ligència artificial capaç d'elaborar i comprovar proposicions lògiques punt a punt
- El primer robot *Tralfa* es va instal·lar en 1964 en una factoria noruega, durant un període d'absència de ma d'obra, per pintar plats. Dos anys més tard , la ciutat industrial de Byrne (Noruega) comptava amb tota una flota d'aquests "robots pintors".
- La robòtica ha contribuït de forma essencial en la conquista de l'espai. En 1966, la nau robotitzada *Surveyor* aterrava a la Lluna. El 1970, un vehicle rus anomenat *Lunakhod* recorria la superfície prenent mostres. La nau espacial *Viking* aterrava el 1976 en Mart. El 1999, la sonda *Mars Pathfinder* en prenia mostres
- La primera companyia en emprar visió artificial per la aplicació industrial va ser *General Motors* el 1970 al Canadà.
- El primer país que va tenir una institució dedicada als robots va ser Japó, que instaurà el 1971 la *Japanese Industrial Robot Association (JIRA)*
- La primera publicació periòdica a nivell internacional sobre robots s'anomenava *The Industrial Robot* i va aparèixer per primer cop el 1973.
- En 1974 es va fundar la *Robotics Industries Association (RIA)*
- En 1977 es va fundar la *BRA* (*British Robotics Association)*
- El 1984 la empresa *Robot Defense Systems* introdueix el *Prowler* (*Programmable Robot Observer with Local Enemy Response*). Aquest seria el primer robot amb fins militars.
- El 1977, el computador *Deep Blue* de *IBM* va guanyar a una partida a escacs al campió mundial, *Gary Kasparov,* fet que va marcar un abans i un després en el desenvolupament de la intel·ligència artificial.

2.2 Robots

Abans d'analitzar els diferents entorns de control i programació de robots es parlarà d'alguns dels robots programables més utilitzats, ja que trobo adient saber més d'ells abans de llegir els anàlisis de les diferents eines, en que es parla de la utilització d'algun d'aquests robots.

Alguns dels més importants són els que es comentaran a continuació:

2.2.1 Aibo

Aquest és, possiblement, el robot més famós que existeix actualment. *Aibo* (amic en japonès) és un robot mascota fabricat per *Sony* fins 2006. Aquesta "joguina" és una de les més sofisticades que es troben al mercat. Com es pot veure a la imatge (Figura 2-8) *Aibo* té aparença de gos. Disposa de diversos motors (parts mòbils, com les cames o el cap), càmera, sensors que eviten que es xoqui, cua que funciona d'antena, així com és sensible al tacte (carícies).

Figura 2-8. Robot Aibo

Aibo a més està programat per a reaccionar amb "emocions": és capaç d'interpretar els gestos del seu amo, reclama atenció movent-se quan no se li presta, i a més aprèn.

Aibo no és tant car com altres robots (uns 2500$), fel pel qual se l'ha emprat molt en recerca de IA.

Disposa de un lector de targeta *Memory Stick*, lloc on s'instal·la el software que el controla. El software és el *Mind3* controla *l'Aibo* quan està en mode autònom, però també pot rebre ordres del PC o altres dispositius. Mind3 anirà creant la personalitat de *l'Aibo* a mida que passi el temps, segons les experiències i sensacions que tingui. D'aquesta forma, cada *Aibo* es comportarà de forma diferent.

Les parts mòbils de *l'Aibo* són:

- Cap – 3 graus de llibertat
- Boca – 1 grau de llibertat
- Potes – 3 graus de llibertat (x4)
- Orelles – 1 grau de llibertat (x2)
- Cua – 2 graus de llibertat

Total: 20 graus de llibertat de moviment.

A aquestes propietats cal afegir multitud de sensors:

- Càmera de 350.000 píxels
- Micròfons
- Infrarojos de distància (x2)
- Acceleració
- Vibració
- Sensors de tacte al cap, llom, part inferior de la boca i potes

És un robot que té múltiples possibilitats d'operació.

2.2.2 LEGO Mindstorms

Mindstorms (**Figura 2-9**) és un kit de robòtica desenvolupat per *Lego* que inclou els elements basics de la robòtica, com la unió de peces i la programació d'accions. Va ser comercialitzat per primer cop el 1998. Existeix una versió educativa que s'anomena *Lego Mindstorms for Schools*.

Figura 2-9. Lego Mindstorms NXT amb sensors i actuadors connectats

Es pot utilitzar per a construir un model de sistema integrat amb parts electromecàniques controlades per computador. Pràcticament tot pot ser controlat amb les peces com a la vida real, tant elevadors com robots industrials.

Lego Mindstorms va ser fruit de la col·laboració entre *Lego* i *l'Institut Tecnològic de Massachussetts (MIT)*. Lego es va interessar en la forma d'aprendre dels nens i van arribar a un acord amb el MIT en que finançarien les investigacions del grup d'epistemologia i a canvi obtindrien noves idees que després podrien utilitzar als seus productes sense haver de pagar drets d'autor.

Lego va decidir que el mercat que escolliria pel seu producte seria dels nens de entre 10 i 14 anys, i aquesta elecció va definir el disseny i les diferencies entre el desenvolupament del *MIT*, anomenat *Programmable Brick* (maó programable) i el bloc *RCX*, dissenyat i desenvolupat per *Lego* de forma independent i des de zero, ja que el sistema que Lego volia produir havia de ser assequible i robust, ja que es vendria a gran escala, i el sistema del *MIT* s'utilitzaria per investigacions i en laboratoris.

Havia de ser simple, però incloïa motors, sensors, microcontroladors, ports d'entrada, sortida es podia programar, etc.

Figura 2-10. Pack Lego Mindstorms

Finalment va sortir a la venta a un preu d'uns 200$, incloent 717 components, entre ells el bloc *RCX*. Va tenir prou èxit, se'n van vendre 80.000 unitats en tres mesos i a més la comunitat de la robòtica ho va acollir amb gran interès. Aquest interès imprevist del públic adult va fer que les ventes es triplicaren.

Els blocs programables van anar evolucionant fins la versió definitiva del *NXT*, el 1998. Les coses no anaven molt be a *Lego* però finalment el 2006 va treure a la venta la versió ***Mindstorms NXT*** *(*Figura 2-11*)*.

Figura 2-11. Pack Lego Mindstorms NXT

Alguns dels entorns de programació més famosos per a *Lego Mindstorms* són:

- *BrickOS* o (*LegOS*)
- *LejOS*
- *Not Quite C*

2.2.2.1 Bloc RCX

El bloc *RCX* (Figura 2-12) és la part central de *Lego Mindstorms*, aquí és on es troba tota la part lògica i electrònica que permet la majoria d'accions del robot, podent emmagatzemar fins a 5 programes que es poden carregar a la seva memòria interna, i guardant també allà el firmware dels diferents dispositius que es puguin connectar.

Figura 2-12. Bloc RCX

RCX té una desavantatge, i és que no pot paral·lelitzar processos, ja que la velocitat es veu reduïda i impedeix el correcte funcionament. Disposa d'una pantalla LCD i té entrades i sortides per a connectar els sensors i actuadors, a més d'un port d'infrarojos per a comunicar-se amb el PC.

2.2.2.2 Bloc NXT

El bloc *NXT* és la versió millorada del bloc *RCX*, el qual es considera precursor dels blocs programables de Lego.

Degut a la comercialització dels blocs programables, *Lego* va vendré la generació *NXT* en dues versions: *Retail Version* i *Education Base Set*. La versió *Educational* disposava de bateries recarregables i carregador, però no incloïa el software, que s'havia de comprar segons el tipus de llicència: Personal, Sala de classes, Lloc.

Figura 2-13. Robot controlat amb el bloc NXT

Lego, a més, va crear diversos kits per a desenvolupadors segons les característiques dels seus programes:

- *Software Developer* Kit *(SDK)*
- *Hardware Developer* Kit *(HDK)*
- Bluetooth *Developer* Kit *(BDK)*

El microcontrolador del que disposa el bloc *NXT* és més potent que el del bloc *RCX*, millorant la capacitat d'execució de programes i processos.

Les comunicacions també han estat millorades, tenint una interfície USB 2.0 i Bluetooth.

2.2.3 Nao

Nao és un robot programable amb una aparença d'esser humà, desenvolupat per l'empresa francesa *Aldebaran Robotics*.

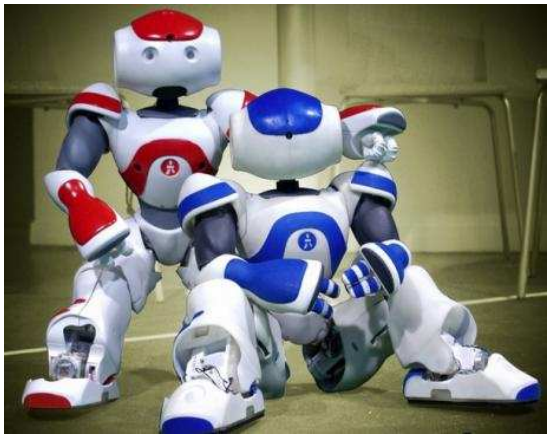

Figura 2-14. Dues versions del robot Nao

Aquest robot va ser presentat al públic l'any 2006. *Nao* va reemplaçar a la *Robocup* el 2007 a *l'Aibo* de *Sony*. Existeixen diverses versions, actualment s'està desenvolupant la versió 6, en que els graus de llibertat total van des de 21 a 25.

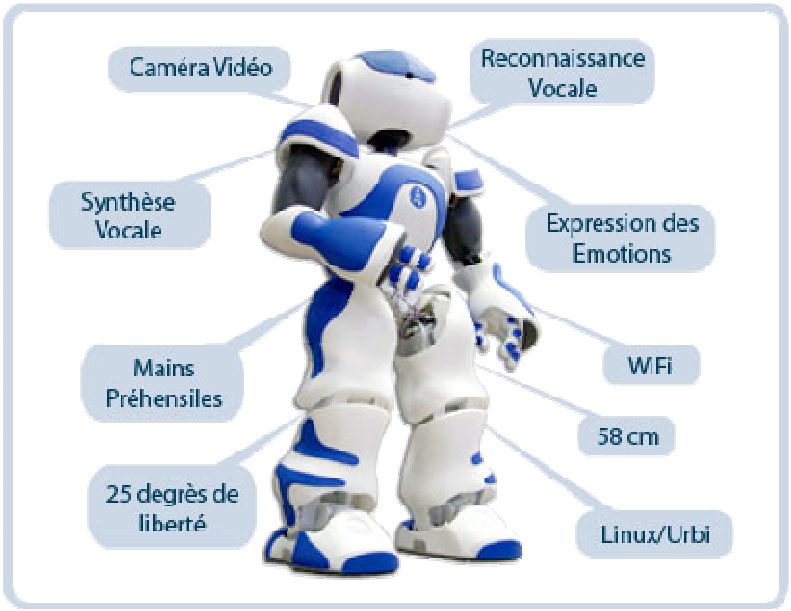

Figura 2-15. Característiques robot Nao

Nao per defecte està programat amb URBI. Pot ser controlat a través del PC gràcies a la comunicació wifi de la que disposa. S'adapta bé a un entorn domèstic, on se li poden assignar tasques.

Nao és un robot molt similar a *l'Aibo* de *Sony* i encara que la seva aparença no sigui exactament la mateixa en quant a funcionalitats i possibilitats estan molt a prop els dos robots.

2.2.4 Khepera, Hemisson i E-Puck

Aquests robots tenen un funcionament similar: són robots petits, rodons i plans que porten rodes, un motor per moure-les i inclouen diversos sensors i/o càmeres.

Els tres robots els fabrica i comercialitza l'empresa *K-Team.*

- *Hemisson* (Figura 2-16) és un robot dissenyat per l'educació: equipat amb diversos sensors i un controlador programable, és un robot capaç d'evitar obstacles, detectar la intensitat de la llum ambient i seguir una línia al terra.

Figura 2-16. Robot Hemisson

- *Khepera* (Figura 2-17) és un minirobot, encarat més a la investigació robòtica: la versió 3 d'aquest robot és el resultat de 10 anys d'investigació en robòtica en miniatura. Disposa de múltiples sensors, de rang curt, rang llarg i pack de bateria substituïble. Inclou un sistema millor de moviment, essent capaç de moure's a una taula o a un laboratori. És la solució perfecta per a experiments amb colònies de robots.

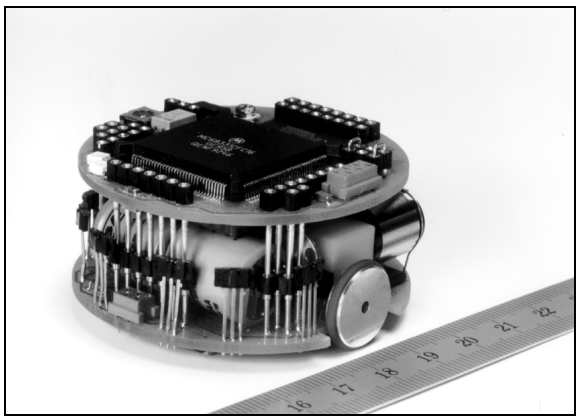

Figura 2-17. Robot Khepera

- *E-Puck* (Figura 2-18) és un robot dissenyat per educació a alt nivell: va ser dissenyat *al Institut Federal de Tecnologia de Laussanne (Suïssa)* amb propòsits

educacionals i actualment també es comercialitza. Disposa de molts sensors i motor, com els seus companys, però a més tant el hardware com el software són lliures, donant accés a baix nivell al sistema i ampliant les seves possibilitats il·limitadament.

Figura 2-18. Robot E-Puck

2.2.5 Roomba

Roomba (Figura 2-19) és un robot aspiradora fabricat i distribuït per l'empresa *iRobot* amb el nom de *Robotic Florrvac*. Roomba es va llençar per primer cop el 2002, posteriorment van sortir actualitzacions i noves versions.

Figura 2-19. Robot iRobot Roomba

Roomba té simuladors elèctrics que enregistren qualsevol tipus de brossa i a més té una alarma que són quan té algun problema. Aquest robot, a més, es carrega sol quan no té prou energia: *Roomba* torna a la seva base sol.

2.3 Entorns de programació

En aquest apartat s'analitzarà el software de robòtica existent més important que hi ha al mercat i es compararà amb el Microsoft Robotics Studio.

2.3.1 URBI

Figura 2-20. Logotip Gostai i URBI

URBI (Universal Real-time Behavior Interface) és un llenguatge de programació utilitzat per al control de robots. Aquest software pretén ser una alternativa a MRS. Funciona en Windows, Mac i GNU/Linux. El desenvolupa la empresa francesa *Gostai*.

El SDK[4] és de codi obert i està publicat sota llicencia GNU GPL[5] i és independent del robot emprat. Actualment es pot provar en simuladors i en alguns robots reals com *Lego Mindstorms* o robots *Roomba*.

És un llenguatge el qual el seu nucli és de baix nivell, dissenyat per treballar amb motors i sensors, encara que permet el desenvolupament de comandes complexes d'alt nivell. Al seu disseny s'ha cuidat la simplicitat, pel que no existeixen arquitectures complexes, aconseguint en pocs minuts realitzar fàcilment programes de control complexes. També permet la interconnexió amb altres llenguatges de programació, així com C++, Java, Phyton, Matlab, etc.

La raó per la que *Gostai* es va introduir en aquest món va ser la necessitat de la industria de tenir un software potent preparat per a afrontar reptes de Intel·ligència Artificial i programació de robots autònoms. *Gostai* entén que els robots no són gaire compatibles entre ells i que les plataformes existents (en 2005) no estan realment adaptades a la situació.

Les quatre característiques que pensen ha de tenir una plataforma universal són:

- **Flexibilitat**: que funcioni per qualsevol robot, sistema operatiu o llenguatge de programació

[4] SDK: Software Development Kit: conjunt de eines de desenvolupament per a crear aplicacions per a un sistema concret.
[5] GPL: Llicència pública general

- **Modularitat**: integrar una arquitectura modular, on els components puguin estar integrats en la plataforma o ser externs i estar en un ordinador remot.
- **Energia**: la robòtica és un domini complet, pel que es necessita una abstracció potent del paral·lelisme i una programació basada en events, ja que es requereix una solució pels investigadors experts en Intel·ligència Artificial.
- **Simplicitat**: ha de ser una plataforma fàcil d'utilitzar: això estimularà la creativitat i beneficiarà la robòtica en general.

URBI ha estat desenvolupat durant quatre anys per universitats líders en Intel·ligència Artificial i és la única plataforma que compleix els quatre requeriments.

Els llenguatges han d'oferir paral·lelisme i programació basada en events, però, la forma en que ho implementaven era amb "*loops*" síncrons propis amb cues corresponents. Això no era suficient, els programadors necessitaven un nivell d'abstracció més alt.

Aquesta fou la justificació de crear URBI: integrant paral·lelisme i events en el nucli semàntic del llenguatge. A més URBI proporciona una potent arquitectura de components integrada en el model d'objecte del llenguatge.

La tecnologia que aporta URBI són noves característiques en termes de paral·lelisme, programació basada en events i gestió d'objectes distribuïts amb *UObject*. URBI utilitza la interfície "*libURBI*", sota llicència lliure *GPL* i permet molts llenguatges de programació (*C++, Java*...).

Gostai ofereix una plana web (Figura 2-21) senzilla però prou complerta i molt intuïtiva, resultant bastant còmode la navegació, amb explicacions, tutorials en format pdf i fins i tot vídeos introductoris. Però, no es valorarà ja que no es considera determinant en la comparació entre els diferents entorns.

Figura 2-21. Pàgina d'inici de Gostai

2.3.1.1 Tecnologia URBI

Arquitectura UObject

Permet importar objectes escrits en *C++* i ficar-los a URBI per a utilitzar-los com a objectes normals dins el llenguatge. S'anomenen "*UObjects*" .

Figura 2-22. Logotip UObject

També es pot executar un *UObject* com a objecte remot, un executable autònom en Windows, Linux o SO Mac. No fa falta canviar res per a passar de mode enllaçat integrat a remot.

Paral·lelisme i events

La robòtica requereix paral·lelisme i URBI ho integra junt amb programació basada en events al nucli de la seva semàntica, el que és una novetat.

Les comandes "*whenever*" o "*at*" comencen codi quan es produeixen events. El símbol "*&*" s'utilitza en comptes del punt i coma habitual: A&B significa que A i B s'executen en paral·lel (Figura 2-23).

```
whenever (ball.visible)
{
  headPan.val  += camera.xfov * ball.x
  &
  headTilt.val += camera.yfov * ball.y;
}
```

Figura 2-23. Exemple paral·lelisme

Característiques avançades del llenguatge

Com a llenguatge realment paral·lel, URBI proporciona abstraccions potents:

Es pot configurar una variable per a que sobrepassi un valor en un temps (**time**) donat, o a una velocitat (**speed**) donada, o a una **oscil·lació** sinusoïdal (Figura 2-24).

```
  neck = 10 time:450ms
& leg = -45 speed:7.5
& tail = 14 sin:4s ampli:45;
```

Figura 2-24. Exemple configuració variables

Tota variable té un mode combinat (blend), el qual especifica com s'han de portar les assignacions simultànies conflictives (Figura 2-25).

```
x->blend = add;
x = 1 & x = 3;
//now x equals 4
```

Figura 2-25. Exemple "blend"

Qualsevol fracció de codi pot esser prefixada amb una **etiqueta**. Després es pot parar, congelar, descongelar aquest codi des de qualsevol lloc utilitzant l'etiqueta (**Figura 2-26**).

```
mytag: { some code };
stop mytag;
freeze/unfreeze mytag;
...
```

Figura 2-26. Exemple etiquetes

2.3.1.2 Interfície

Actualment s'està desenvolupant un "*Studio*" per al sistema URBI, aquest és un conjunt de tres mòduls:

- *URBILive*: editor gràfic
- *URBIMove*: editor de animacions
- *URBILab*: plataforma de control universal per a interactuar amb els robots de forma gràfica integrant una consola per a enviar comandes als robots

De moment només existeix la beta de *URBILab*, versió 1.6 (Figura 2-27).

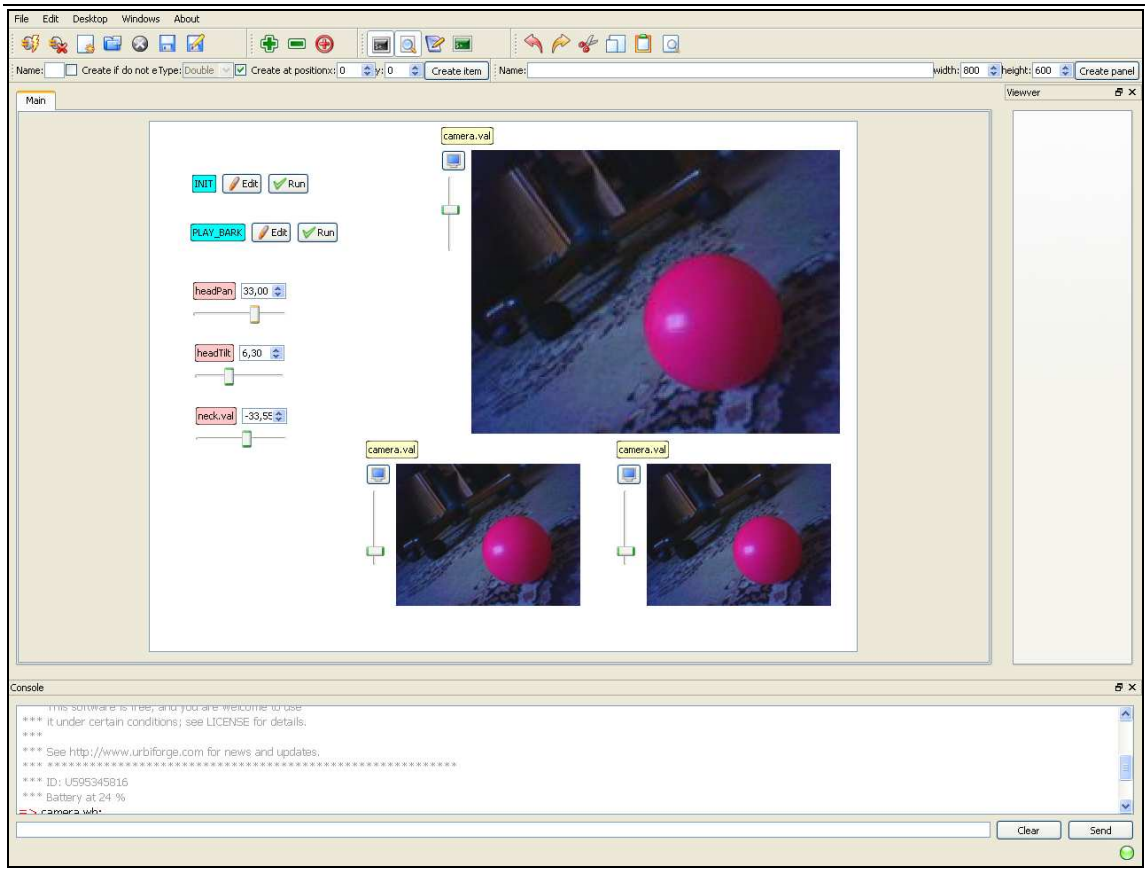

Figura 2-27. Pantalla de URBI interactuant amb la càmera d'un robot

Tal com es pot observar a la part inferior de la figura, conté una consola (Figura 2-28) per a introduir comandes interactives:

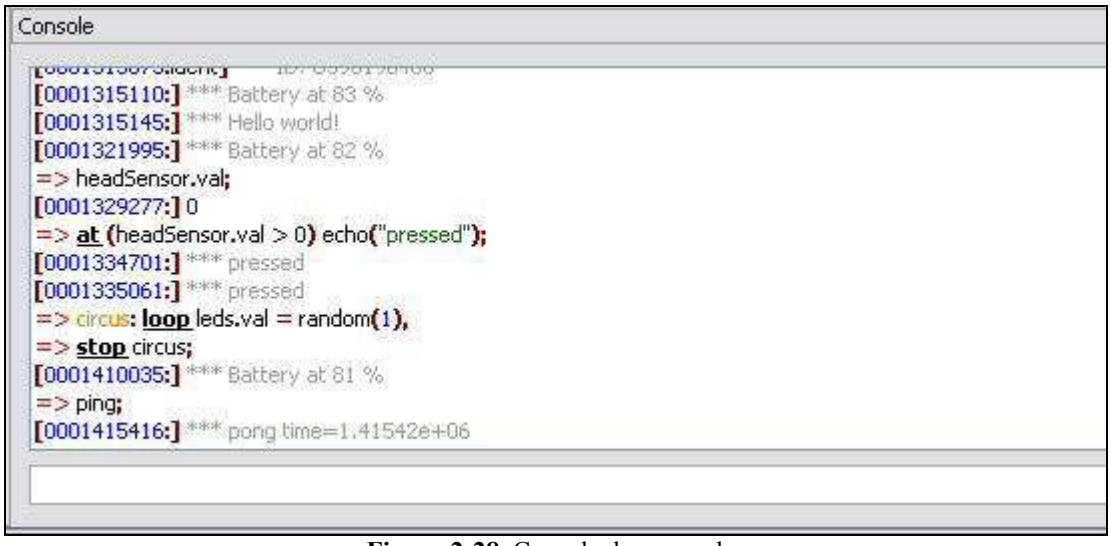

Figura 2-28. Consola de comandes

Les comandes executades reben resposta i es pot observar a la pròpia consola.

A més d'això, *urbiLab* permet crear "*widgets*"(Figura 2-29) per a monitoritzar variables del robot, així com definir botons associats al robot per a crear controladors a la mida (Figura 2-30) :

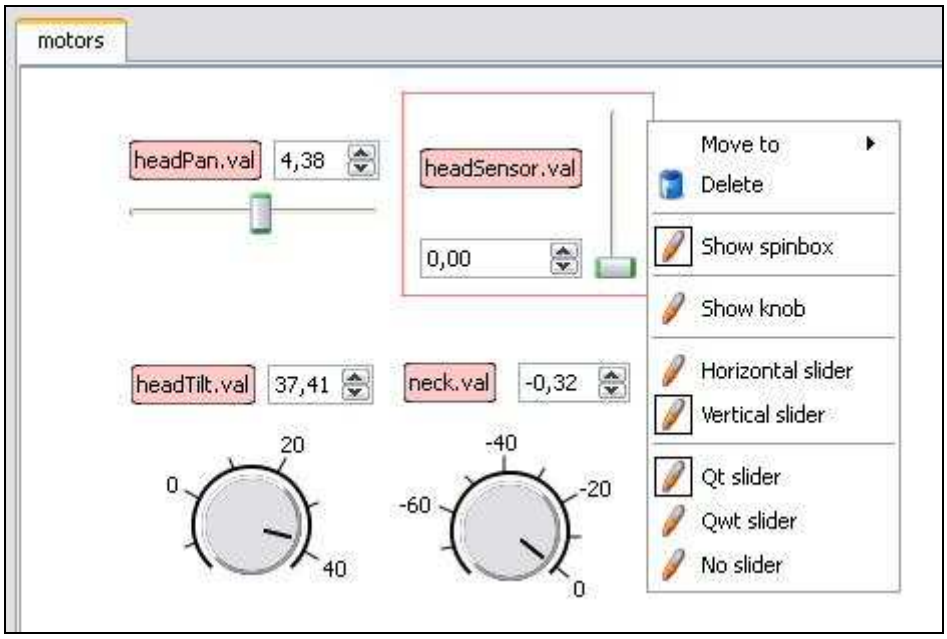

Figura 2-29. Exemple de widget monitoritzant variables

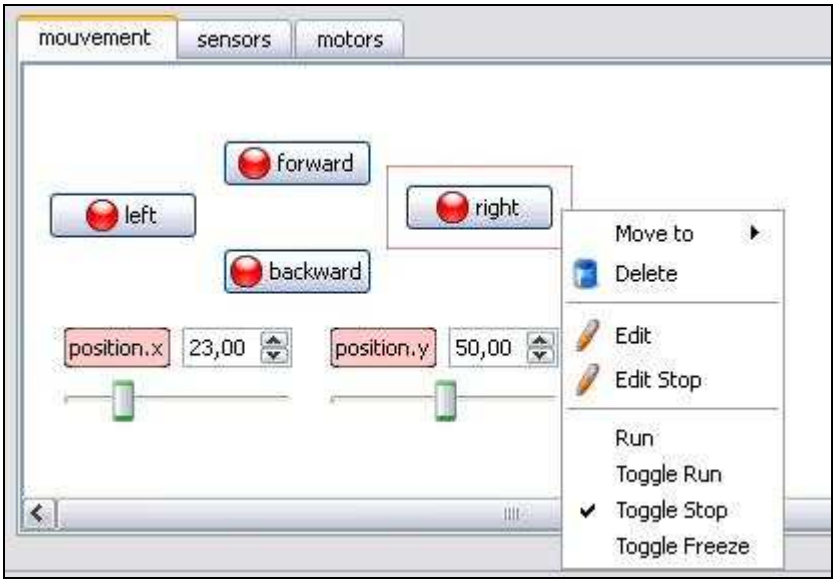

Figura 2-30. Exemple de botons associats a tasques

2.3.2 Player/Stage/Gazebo

El projecte Player/Stage/Gazebo va néixer amb la idea de potenciar la recerca en sistemes de robots i sensors. Consta dels tres programes que li donen nom:

- Player: interfície de gestió de robots
- Stage: simulador per múltiples robots
- Gazebo: simulador 3D per múltiples robots

El Player és un dels sistemes més utilitzats. En simulació, Stage i Gazebo s'utilitzen molt, també. Està distribuït sota llicència GNU[6], i tot el projecte Player/Stage és lliure d'utilitzar, distribuir o modificar. El *Projecte Playe*r funciona a Linux**,** Solaris**,** BSD[7] **i** Mac. Player va ser desenvolupat per un grup internacional de recerca en robòtica. Permeten extensions i estan subjectes a una actualització constant.

2.3.2.1 Player

Player és un servidor en xarxa pel control de robots. Amb el robot funcionant, Player proporciona una interfície simple i neta dels sensors i actuadors del robot mitjançant la xarxa IP. El client parla amb el Player mitjançant un socket TCP, llegint les dades dels sensors i transmetent comandes als actuadors, a més de configurar els dispositius "on the fly".

Suporta una gran varietat de hardware, i la seva arquitectura modular el fa fàcil per a afegir nou hardware amb una comunitat d'usuaris i desenvolupadors que hi contribueixin.

Està dissenyat per a ser independent entre llenguatge i plataforma. El programa del client pot estar instal·lat en qualsevol màquina que tingui connexió al robot, podent estar escrit en qualsevol llenguatge que suporti TCP. Actualment existeixen clients en C++, TCL, Java i Python. El Player no suposa com s'estructurarà el programa de control del robot, i en comparació amb altres interfícies de robots, es molt més minimalista. És l'usuari el que ha d'escriure el programa client. També pot utilitzar el client TCL per a controlar el robot interactivament.

El Player permet a molts dispositius presentar la mateixa interfície. Això permet a diversos robots ser controlats de la mateixa forma (sempre que siguin adequats, és clar). Té una avantatge aquest fet, i és que, combinat amb el simulador Stage, el software per a robots simulats funciona sobre robots reals normalment.

El Player suporta també qualsevol nombre de clients virtuals. Qualsevol client pot connectar-se a qualsevol robot i llegir i escriure en ell. Això permet utilitzar clients diferents sobre el mateix robot, per exemple un que el controli, altre que monitoritzi algunes característiques, altre que mostri un sensor concret... les peticions *al vol* sobre el dispositiu permeten als clients guanyar accés als sensors quan ho requereixen.

[6] GNU: Llicència pública general
[7] BSD: Sistema Operatiu derivat de UNIX

El comportament del servidor també permet configurar-se *al vol*, canviant la velocitat de la transferència de dades dels sensors i altres característiques.

2.3.2.2 Stage

Stage és un simulador 2D per a múltiples robots. Simula una població de robots mòbils, sensors i objectes en un entorn de dues dimensions (Figura 2-31).

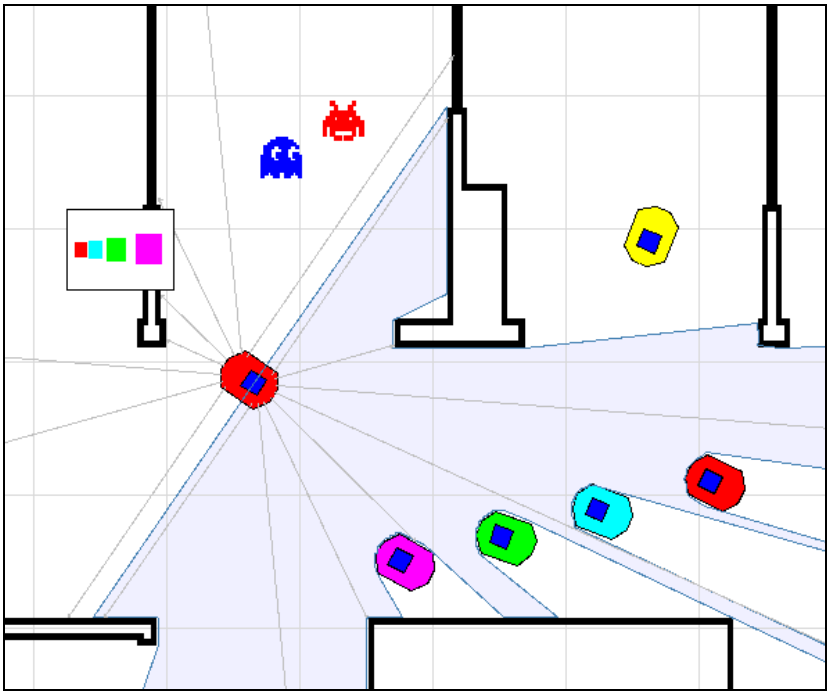

Figura 2-31. Pantalla de l'Stage

Està dissenyat per a donar suport a la recerca de sistemes autònoms multi-agent, és per això que proporciona molts models simples i computacionalment barats per a emular dispositius amb gran fidelitat.

Stage s'utilitza normalment com a mòdul plugin per a Player, proporcionant-li poblacions de dispositius virtuals. Els usuaris escriuen els clients per al servidor. Normalment els clients no poden distingir entre màquina real i el simulat equivalent en Stage, ja que gairebé no se'ls ha de modificar. Stage disposa de diversos sensors i actuadors: sonars, làsers, visió, etc.

Figura 2-32. Pantalla del Stage

El Stage també es pot utilitzar com a llibreria de C per a proporcionar simulació de robots dins els propis programes, quan no es vol utilitzar el Player, o per personalitzar els models de simulació.

2.3.2.3 Gazebo

Gazebo és un simulador 3D de múltiples robots amb dinàmica (Figura 2-33). És un simulador per a entorns oberts. És capaç de fer el mateix que Stage però en tres dimensions. Genera tant reaccions lògiques dels sensors com interacció física entre objectes realista.

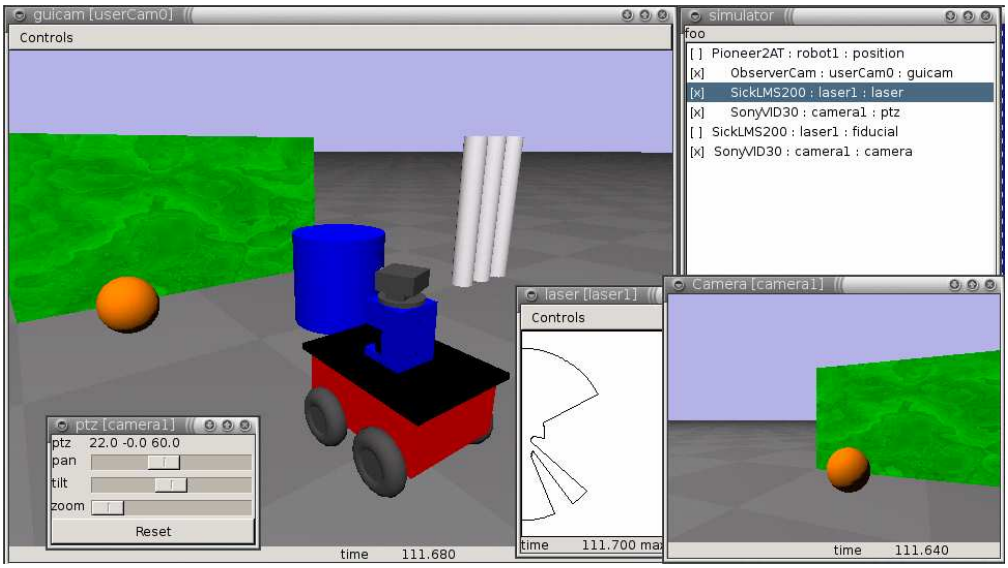

Figura 2-33. Pantalla del simulador que ofereix Gazebo

Té les següents característiques:

- Simulació de sensors estàndards de robots, incloent sonar, làsers, GPS, càmeres (Figura 2-33), etc.
- Models dels robots més utilitzats.
- Simulació realista de objectes rígids: els robots poden interactuar amb els objectes
- Compatible amb Player: tant els robots com els sensors poden ser controlats des del Player

- Operacions autònomes: els programes (externs a Player) poden també interactuar amb Gazebo mitjançant la llibreria *libgazebo*.
- Model estèreo de càmera
- La interfície gràfica completament reescrita: més dispositius poden ser controlats directament mitjançant la interfície gràfica
- Models personalitzats: els usuaris poden desenvolupar robots o sensors propis i carregar aquests models dinàmicament
- Pells (*skins*): els models geomètrics simples (Figura 2-34) es poden millorar amb skins (Figura 2-35)

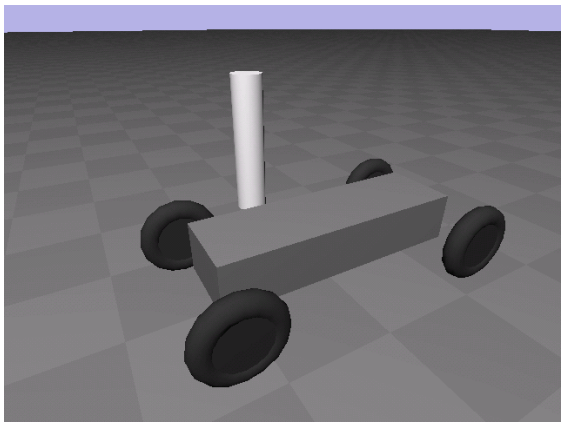

Figura 2-34. Simulador simple

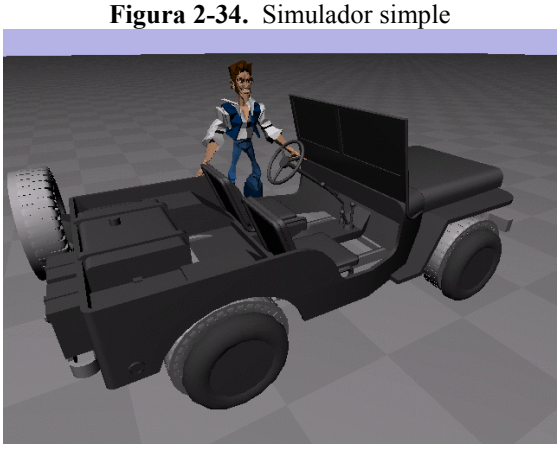

Figura 2-35. Simulador amb skins

El projecte Player/Stage té dos simuladors multi-robot: Stage i Gazebo. Des de que els dos són compatibles amb el Player, els programes escrits per un simulador funcionen amb l'altre sense haver de modificar res o molt poc.

La diferencia entre els dos simuladors és que Stage és més indicat per a simular poblacions grans amb poca fidelitat i Gazebo al reves, poblacions petites però amb molta fidelitat. És, per tant, que els dos simuladors són complementaris, no s'exclouen entre ells, i l'usuari pot emprar un o l'altre segons les seves necessitats.

2.3.3 Webots

Webots és un software desenvolupat per *Cyberbotics* per a programar i simular robots mòbils que s'utilitza a moltes universitats i centres de recerca al voltant del món. Porta 8 anys de desenvolupament i manutenció.

Figura 2-36. Logotip Cyberbotics

Webots inclou llibreries que permeten transferir els programes de control a molts robots reals que estan al mercat.

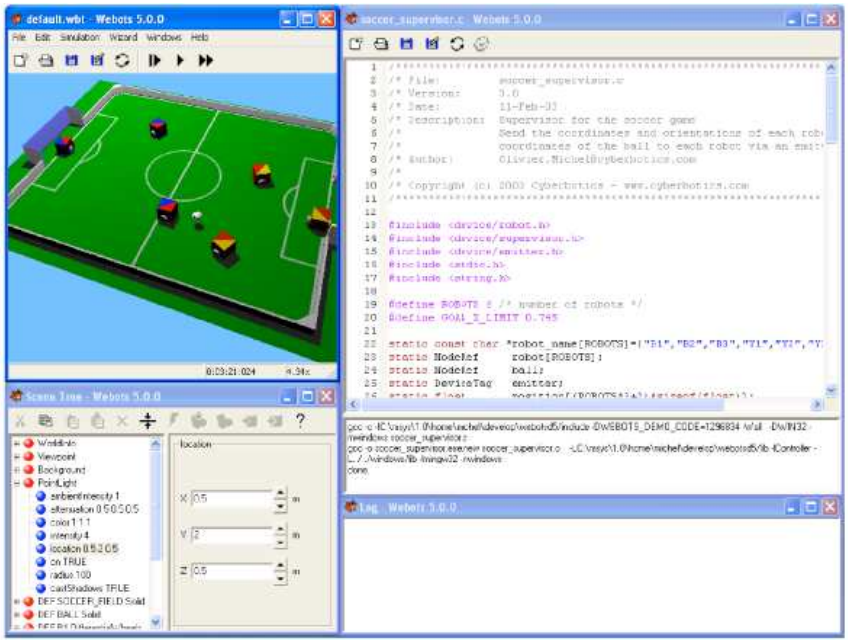

Figura 2-37. Interfície de Webots

Una altre propietat és que permet definir una configuració completa de robòtica mòbil, fins i tot si diversos robots diferents comparteixen el mateix entorn. A cada objecte se li poden definir propietats, com capa, color, textura, fricció... A cada robot se li poden equipar un gran nombre de sensors i actuadors. Aquests robots es poden programar utilitzant l'entorn de desenvolupament que es desitgi, es poden simular i opcionalment transmetre els resultats al robot real.

Webots presenta les següents característiques:

- Els models permeten simular qualsevol robot, incloent els que van amb rodes, cames o voladors
- Inclou una llibreria completa de sensors (distància, rang, llum, receptors...) i actuadors (diferencials, servos, LEDs, etc...)
- Utilitza *OpenGL*[8] i un editor per a crear robots i un món en 3D
- Permet programar els robots en *C, C++ i Java*, o des de software d'alguna third-party[9] (MatLab, LabView, etc)
- Utilitza la llibreria ODE (*Open Dynamics Engine*) per a la simulació de la física
- Permet transmetre els controladors a robots mòbils reals, tals com: *e-puck, Aibo, LEGO Mindstorms, Kephera, Hemisson* i els robots personalitzats propis.
- Permet crear vídeos en format AVI o MPEG
- Inclou molts exemples amb codi per a robots comercials
- Permet simular entorns multi-agent
- Interactivitat: quan s'està executant la simulació es poden moure objectes, rotar la càmera, etc.

[8] OpenGL: estàndard que defineix una API per a escriure aplicacions que produeixin gràfics en 2D i 3D
[9] Third-party: es coneixen així les empreses que desenvolupen software per a diverses plataformes

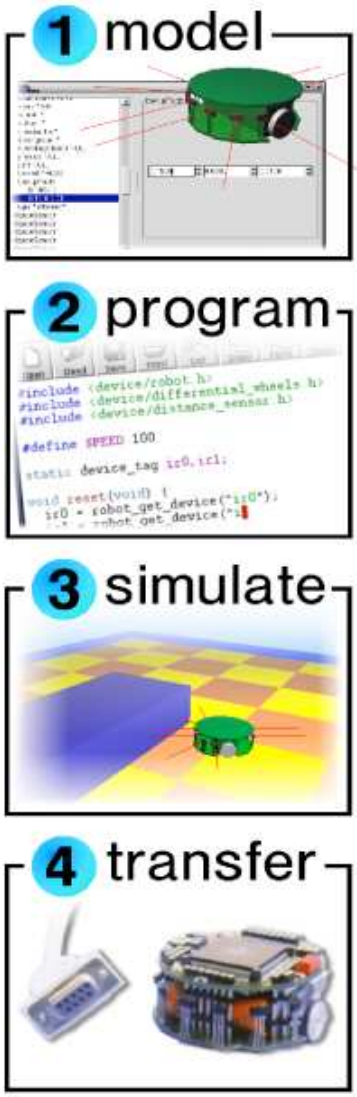

Figura 2-38. Seqüència de creació d'aplicacions

El sistema de simulació que utilitza fa servir temps visual, fent possible executar les simulacions molt més ràpid que el que faria un robot real, fins a 300 vegades. A més té l'opció d'executar pas a pas, per a fer més fàcil l'estudi en detall del que fa el robot.

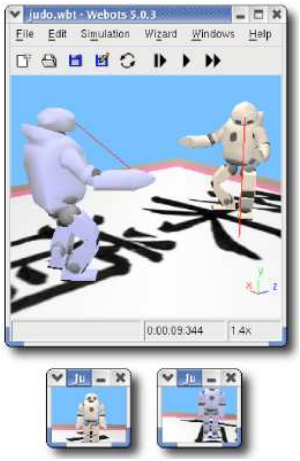

Figura 2-39. Simulador de Judo i càmeres als dos robots

2.3.3.1 Interfície de programació

Aquest codi (Figura 2-40) representa el codi de control d'un robot. El robot en qüestió és un robot amb rodes controlades per un diferencial que té un sensor *IR* (infrarojos) que mira cap endavant. El robot para de moure's quan detecta un objecte massa a prop i torna a moure's quan l'objecte desapareix.

```
#include <robot.h>
#include <differential_wheels.h>
#include <distance_sensor.h>

static DeviceTag ir;

static void my_robot_reset() {
  ir = robot_get_device("ir");
}

static void my_robot_run(int ms) {
  if (distance_sensor_get_value(ir)>100)
   differential_wheels_set_speed(0,0);
  else
   differential_wheels_set_speed(10,10);
  return 64; /* 64 ms simulation step */
}

void main() {
  robot_live(my_robot_reset);
  robot_run(my_robot_run);
}
```

Figura 2-40. Exemple de programació

Existeix una interfície de programació en Java també, així com la possibilitat de utilitzar software de tercers.

També té una funció que s'anomena "*Supervisor Capability*" (capacitat de supervisor) que permet escriure un programa responsable de supervisar l'experiment. Aquest programa és capaç de moure objectes dinàmicament, enviar missatges als robots, enregistrar les trajectòries, etc.

La *Supervisor Capability* es pot utilitzar en simulacions computacionalment cares, on hi ha un gran nombre de robots amb configuracions diferents i els paràmetres han de ser avaluats, com evolució genètica, xarxes neuronals, etc.

2.3.3.2 Llicències



Webots pot ser instal·lat tant en Linux com Windows o Mac, però no és software gratuït, existeixen diverses llicències (Figura 2-41):

- PRO: més potent per a recerca
- EDU: menys potent però més barata per a ús educacional

Webots™	PRO	EDU
Supervisor capability	✔	
Custom physics programming	✔	
Fast simulation mode	✔	
Multiple-platform: Linux / Mac / Windows	✔	✔
Floating license option	✔	✔
Transfer to real robots	✔	✔
Packaged or electronic version	✔	✔
1 year Premier service included	✔	✔

Figura 2-41. Diferències entre llicències

Els preus depenen de molts factors, com el nombre de llicències comprades, trobant-se a la plana web tot un formulari amb aquestes opcions de configuració, a més d'un PDF on indica tots els preus, on està l'opció de ús acadèmic (tant en PRO com EDU). Una altre possibilitat són les "llicències flotants", això permet utilitzar a qualsevol ordinador de la xarxa Webots de forma simultània però sense superar el nombre de llicències flotants disponibles.

Webots ve amb documentació (dos manuals impresos a color), un CD i un USB per a transportar el programa, en els que s'explica tant com s'instal·la tot com els passos necessaris per a desenvolupar aplicacions.

2.3.4 Microsoft Robotics Studio

Microsoft Robotics Studio és un entorn basat en Windows per a que desenvolupadors aficionats, acadèmics o comercials puguin crear aplicacions robòtiques per a gran varietat de plataformes de hardware. MRS té un entorn d'execució lleuger, inclou una rutina orientada a serveis, un conjunt de eines de simulació, així com tutorials i codi d'exemple per a iniciar-se.

És una plataforma de desenvolupament extrem a extrem, permet als programadors crear serveis per una gran varietat de hardware robòtic.
Les característiques més importants de Microsoft Robotics Studio són:

2.3.4.1 Programació visual

Una funció important de MRS és Visual Programming Language*(*VPL*)* (Figura 2-42), que permet a qualsevol crear i provar les aplicacions robòtiques molt fàcilment. És tan fàcil com arrossegar i deixar anar blocs que representen serveis i connectar-los entre ells. També es poden utilitzar una col·lecció de blocs connectats com a un de sol i utilitzar-ho en qualsevol part del programa.

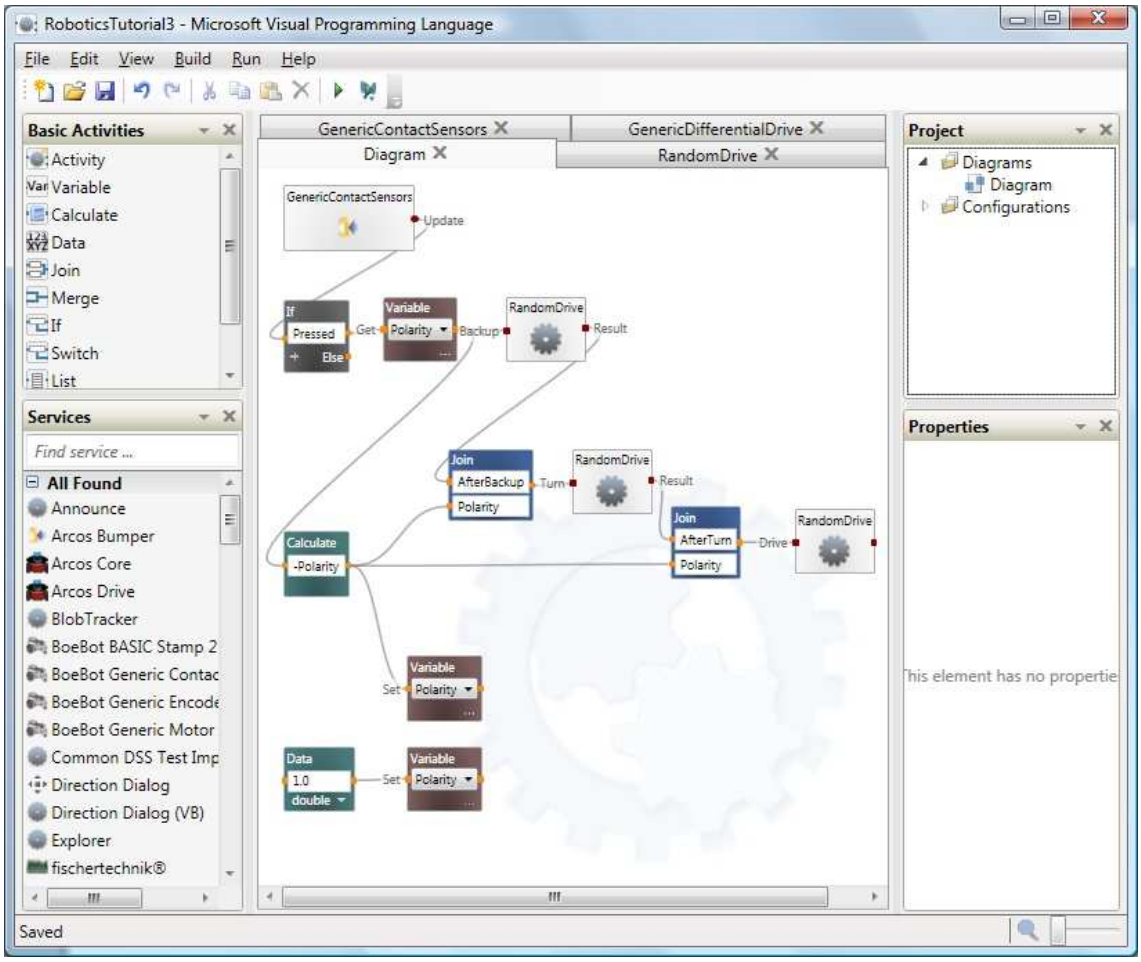

Figura 2-42. Interfície d'usuari de Visual Programming Language

VPL és un entorn de programació dissenyat per a una programació basada en flux de dades gràfic en comptes del flux de control que trobem a la programació convencional. En comptes d'una sèrie de comandes imperatives executades seqüencialment, un programa en flux de dades és com una sèrie de treballadors a una línia de muntatge, on cadascun té la seva tasca quan el material arriba. Com a resultat VPL està dissenyat per a programar una varietat d'escenaris de processament concurrents o distribuïts.

L'objectiu de VPL és que els programadors inexperts entenguin fàcilment els conceptes com variables i lògica. De totes formes VPL no està limitat a novells, la natura del llenguatge de programació pot significar als programadors experts una forma ràpida de fer prototips o desenvolupar codi. Cal afegir que mentre el seu disseny superior és ideal per a programar aplicacions per a robots, l'arquitectura subjacent no està limitada a això. Com a resultat VPL té una gran audiència de públic de més àmbits.

La programació en flux de dades de MRS consisteix en una seqüència d'activitats connectades representades com a blocs amb entrades i sortides que es poden connectar a altres activitats (Figura 2-43).

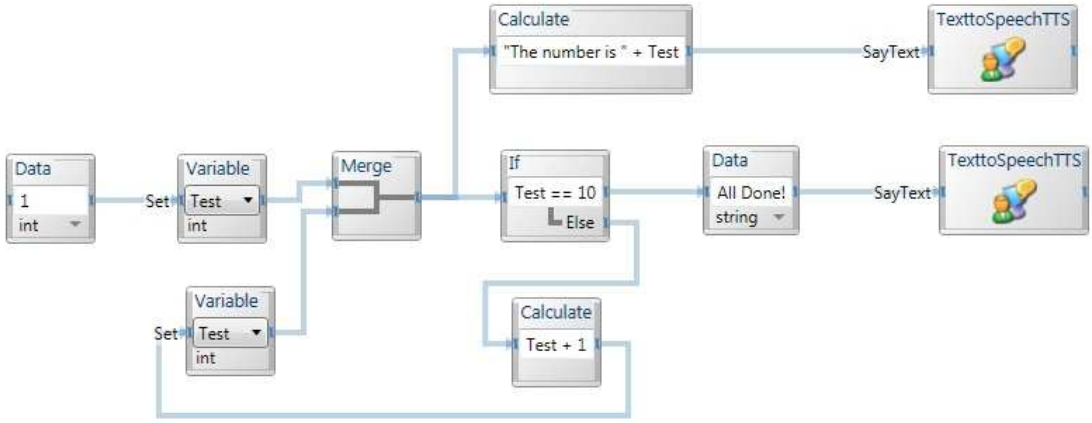

Figura 2-43. Exemple de diagrama de flux de dades de Microsoft Robotics Studio

2.3.4.2 Simular aplicacions en 3D

Microsoft Robotics Studio pretén que la robòtica arribi a un rang ampli de gent i que acceleri el seu desenvolupament i adopció. Una part molt important en aquest esforç és la simulació. Els videojocs són el punt del que es pretén enfocar: són visualitzacions foto-realistes amb simulació avançada de la física funcionant en temps real.

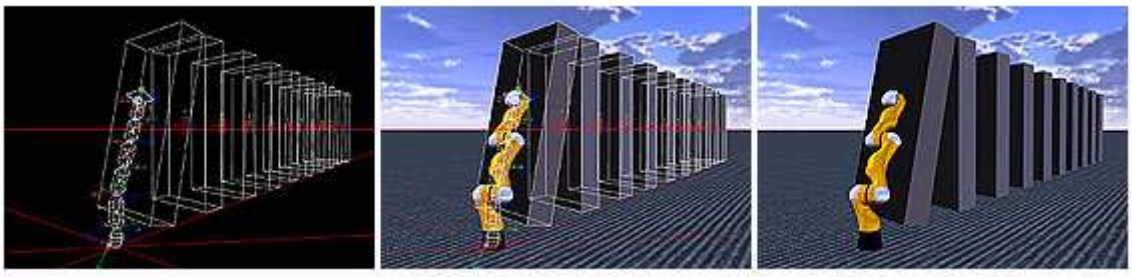

Figura 2-44. Simulació 3D

S'ha definit la rutina de simulació per a que es pugui utilitzar en un gran rang d'escenaris amb altes demandes de fidelitat, visualització i escala. Al mateix temps, un usuari novell sense nocions de programació pot utilitzar la simulació, programant aplicacions en un entorn similar als videojocs.

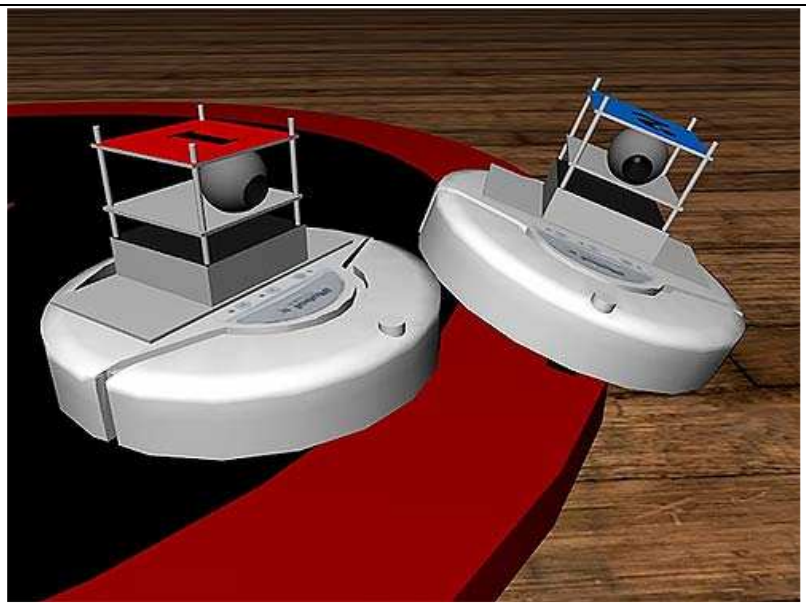

Figura 2-45. Nivell de detall molt elevat en simulació

MRS permet simular aplicacions robòtiques en uns models realistes en 3D (Figura 2-44). La eina de simulació de MRS inclou *AGEIA™ PhysX™ Technology* de *AGEIA Technologies Inc*., una empresa pionera en físiques accelerades per hardware, permetent crear simulacions en el món real per a models de robots. Les simulacions *PhysX* es poden accelerar amb el hardware *AGEIA*.

El motor de renderització[10] està basat en Microsoft XNA *Framework*[11].

2.3.4.3 Interacció amb robots mitjançant interfície basada en web

MRS permet crear aplicacions que proporcionen a l'usuari la possibilitat de fer un seguiment o controlar un robot remotament utilitzant un explorador web i transmetent comandes mitjançant les tecnologies web, com html o Javascript, així com muntar càmeres als robots i controlant-los en entorns remots.

Entorn d'execució lleuger orientat a serveis

El desenvolupador pot accedir fàcilment als sensors i actuadors del robot, gràcies a una llibreria d'implementació de concurrència desenvolupada en .NET . La comunicació està basada en missatges, permetent la comunicació entre mòduls.

2.3.4.4 Arquitectura de Microsoft Robotics Studio

La tasca principal de les aplicacions robòtiques és la de consumir senyals d'entrada de sensors des d'una sèrie de fonts i organitzar una sèrie d'actuadors per a respondre a les entrades de forma que es segueixi el propòsit de l'aplicació.

[10] Renderització: generar una imatge des de un model
[11] XNA Framework: API desenvolupada per Microsoft per al desenvolupament de videojocs per PC i Xbox360

La següent figura mostra un diagrama de flux de dades d'una aplicació robòtica que conté un sensor (bumper) que avisa quan ha estat tocat, un actuador (message box) que controla la pantalla, i un orquestrador que connecta i organitza el sistema:

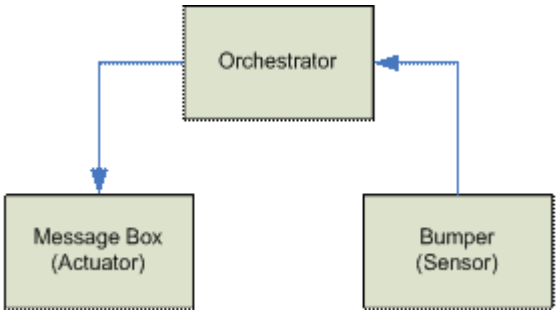

Figura 2-46. Exemple arquitectura

La base de l'arquitectura de Microsoft Robotics Studio són les rutines: **Concurrència i Coordinació (CCR)** i **Serveis de Software Descentralitzat (DSS)**.

Concurrència i Coordinació (CCR)

La *Concurrency and Coordination Runtime* (Rutina de Concurrència i Coordinació - CCR) fa simple la programació asíncrona. Fa simple escriure programes per gestionar entrades asíncrones de múltiples sensors de robots i sortides a motors i actuadors.

CCR proveeix un model de programació molt concurrent, orientat a missatges amb unes primitives d'orquestració que permeten la coordinació de missatges sense haver-ho de fer de forma manual amb semàfors, *threads*, etc. CCR direcciona la necessitat d'aplicacions orientades a servei proveint un model de programació que faciliti la gestió d'operacions asíncrones, interactuant amb la concurrència, aprofitant el hardware paral·lel i gestionant errors parcials

CCR és apropiat per a un model d'aplicació que separa components en trossos que es poden comunicar únicament mitjançant missatges.

Això permet a l'usuari dissenyar les seves aplicacions de forma que els seus mòduls de software o components es puguin connectar fàcilment. Es poden desenvolupar independentment sense haver de tenir en compte els altres components.

Aquest enfocament canvia la forma de pensar del programador des del principi del procés de disseny i facilita interactuar amb concurrència i errors de forma consistent.

Serveis de Software Descentralitzat (DSS)

El model d'aplicació de *Decentralized Software Services* (Serveis de Software Descentralitzats - DSS) fa simple accedir i respondre a l'estat d'un robot utilitzant un navegador web (Figura 2-47) o una aplicació basada en Windows.

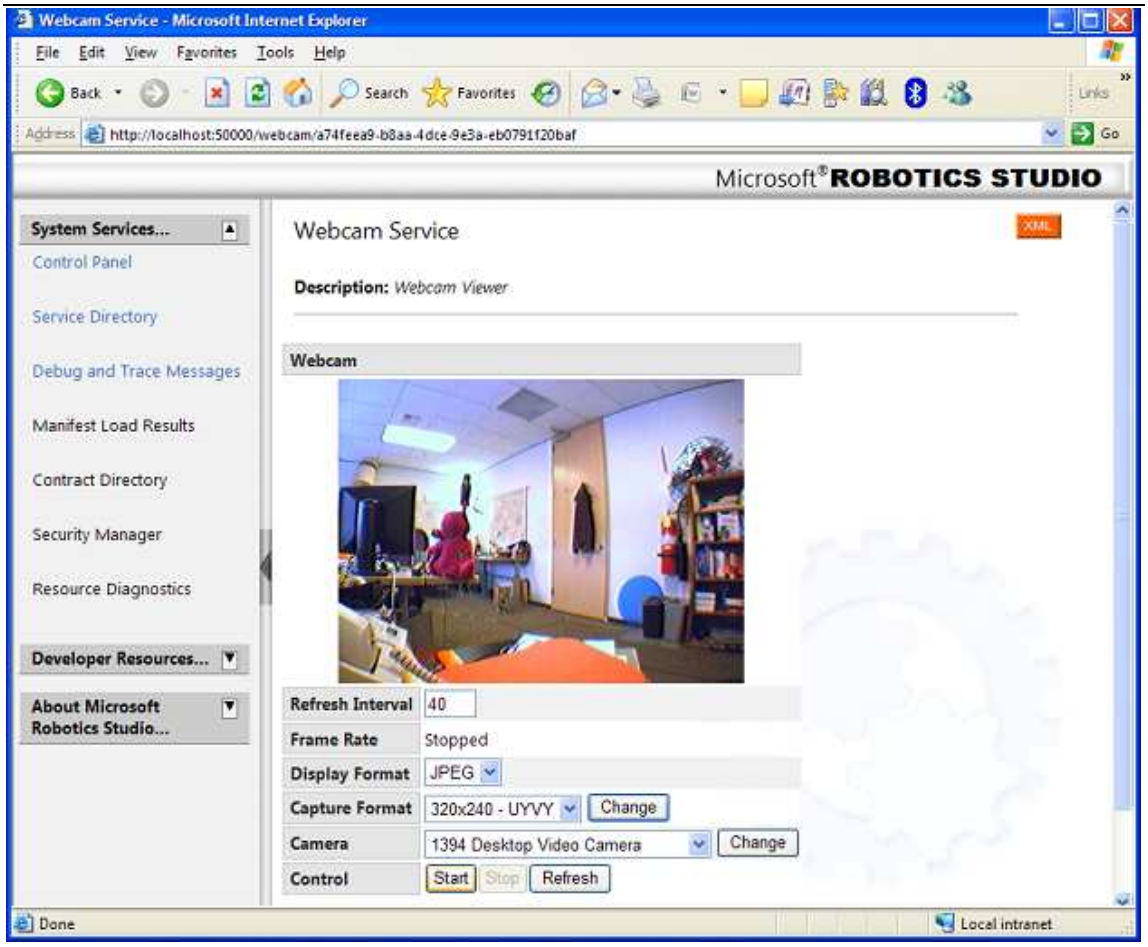

Figura 2-47. Accedint a l'estat d'un robot des del navegador web

DSS proveeix un model d'aplicació lleuger, orientat a servei que combina termes de l'arquitectura basada en web tradicional (anomenada *REST*) amb part de l'arquitectura de serveis web (*Web Services architecture*). El model d'aplicació definit per DSS s'ha construït sobre el model *REST* controlant els serveis a través del seu estat juntament amb un conjunt d'operacions sobre aquest estat que amplien el model d'aplicació de *HTTP* afegint manipulació estructurada de dades, notificació de events i composició de serveis.

L'objectiu principal de DSS és promoure la simplicitat, interoperabilitat i fàcil connexió. Això ho fa ideal per a crear aplicacions com a composicions de serveis a pesar de que aquests serveis estiguin funcionant sense el mateix node o a través de xarxa. El resultat és una plataforma flexible però simple per escriure un ampli conjunt d'aplicacions.

La rutina DSS està implementada a més alt nivell que CCR i no depèn de cap altre component de Microsoft Robotics Studio. Proveeix un entorn d'emmagatzemament per a serveis de gestió i un conjunt de serveis d'infraestructura que es poden usar per a creació, descobriment, proves, supervisió i seguretat de serveis.

2.3.4.5 Reutilitzar serveis modulars

Un **servei** és el bloc bàsic per a construir aplicacions utilitzant **Microsoft Robotics Studio** i el seu component clau del model d'aplicació DSS. Els serveis es poden utilitzar per representar, per exemple:

- Components Hardware: sensors, actuadors
- Components Software: interfície d'usuari, emmagatzemament, ect
- Agregacions: sensors de fusió, etc.

Els serveis s'executen dins el context d'un node DSS. Un node DSS és un entorn hosting que proveeix suport als serveis per a esser creats o gestionats fins que s'eliminen o el node DSS para. Els serveis són inherents a la xarxa i es poden comunicar amb els demès d'una forma uniforme encara que no estiguin executant-se dins el mateix node DSS.

El model de servei DSS s'ha dissenyat per a facilitar reutilització de serveis fent-los fàcils d'utilitzar i de combinar entre ells.

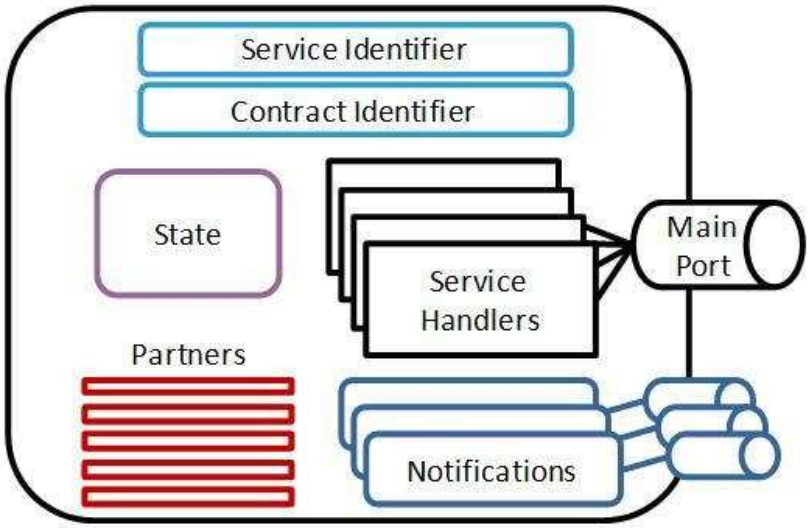

Figura 2-48. Model de serveis DSS

Tots els serveis DSS disposen d'una sèrie de components comuns, els principals són:

- **Service Identifier (Identificador de Servei)**: proporciona identitat al servei, donant lloc a que altres serveis puguin comunicar-se amb ell o permetent fins i tot al navegador web accedir-hi.
- **Contract Identifier (Identificador de Contracte)**: identificació única de la funcionalitat que implementa el servei, de forma que altres serveis ho puguin reutilitzar
- **Service State (Estat del Servei)**: estat del servei en qualsevol moment de l'execució. Qualsevol informació que s'hagi de recuperar, modificar o supervisar en un servei DSS ha de formar part del seu estat.
- **Service Partners (Serveis Associats)**: serveis especials que interactuen amb d'altres i s'encarreguen de que funcionin correctament. Es declaren amb l'atribut "*Partner*".
- **Main Port (Port Principal)**: És un port CCR on arriben els missatges d'altres serveis. Aquest port és un membre privat de la classe *Service* i s'identifica amb l'atribut *ServicePort*.
- **Service Handlers (Administradors de Servei):** registre de totes les operacions DSS definides sobre el port principal. Es defineixen amb l'atribut *ServiceHandlers*.

D'aquesta forma un robot no és una entitat en sí mateixa dins de Microsoft Robotics Studio, si no que és un conjunt de serveis, cadascun d'ells representant algun element del mateix, com pot ser un servei per a representar les seves rodes, un altre per a representar un sensor com pot ser una càmera, etc. Només es fan servir els serveis requerits. Això es veu clarament a Visual Programming Language, on cada instància de servei fa referència a una funció.

MRS permet construir funcions d'alt nivell fent servir components simples, proveint de reusabilitat els mòduls de codi així com d'una millor fiabilitat i possibilitat de substitució. Per exemple es pot integrar un servei de sensor de baix nivell a un servei de navegació.

Plataforma escalable i extensible

El model de programació de MRS es pot aplicar a una gran varietat de plataformes de hardware, donant la possibilitat als usuaris de transferir les seves habilitats entre plataformes. Les interfícies de programació es poden utilitzar per a desenvolupar aplicacions per a robots utilitzant processadors de 8,16 o 32 bits, tant d'un o més nuclis.

2.3.4.6 Fàcilment extensible la funcionalitat de MRS

Les third parties poden estendre la funcionalitat de MRS proveint llibreries i serveis addicionals. Els comerciants de hardware o software poden fer els seus productes fàcilment compatibles amb MRS.

2.3.4.7 Suporta aplicacions mixtes

Els escenaris connectats remotament permeten comunicar-se des d'un PC a un robot mitjançant un port sèrie, Bluetooth*, 802.11 o RF*. Els programes es poden executar nativament en robots basats en PC fent servir Microsoft Windows, permetent operacions totalment autònomes.

2.3.4.8 Llenguatges

MRS permet que les aplicacions es desenvolupin en una gran varietat de llenguatges, incloent aquells que hi ha a Microsoft Visual Studio i Microsoft Visual Studio Express (C# i VB.NET), així com llenguatges com Microsoft Iron Python. També permet llenguatges de third parties que suportin l'arquitectura basada en serveis del MRS.

2.3.4.9 Versions

Actualment existeixen dues versions:

- **Microsoft Robotics Studio *1.5***: és la versió que s'utilitzarà per a portar a terme aquest projecte. És una versió final, evolució de la versió 1.0 sortida al mercat el 2006.
- ***Community Technical Preview July***: aquesta versió és una primera mostra de la nova evolució del programa. Es pot descarregar, però no és una versió final, és una versió pre-beta.

Microsoft Robotics Studio pot instal·lar-se tant en Windows XP com en Windows Vista.

2.3.4.10 Llicència

Existeixen dos tipus de llicència:

No Comercial

Llicència per a una utilització no comercial. Es defineix "no comercial" com a utilitzar el software per a ús personal i no per a fer negocis o generar beneficis. És una llicència indicada per a estudiants, professors, recerca acadèmica o gent que s'interessi com a hobby.

El software es pot descarregar gratuïtament. Les companyies o gent que pretén obtenir beneficis pot utilitzar aquest software per un període de temps limitat per a avaluar-ho.

La llicència no comercial només permet la utilització de les rutines de components en un nombre limitat de robots o PCs. El codi de mostra que inclou el software pot ser utilitzat, modificat i distribuït sempre dins els termes de la llicència.

Comercial

Aquesta llicència està indicada si es desitja crear o distribuir aplicacions comercials o basar operacions de negocis en Microsoft Robotics Studio. En aquests casos s'ha de comprar la llicència a la botiga online. Una llicència comercial costa 399$ (per Espanya costa 254.95€).

La llicència comercial substitueix la no comercial i dóna dret a alguns beneficis, com una llicència per a distribuir els components de rutina amb el software creat. Cada llicència comercial permet distribuir 200 còpies d'aquests components.

Si es disposava d'una llicència per Microsoft Robotics Studio 1.0, Microsoft considera la versió 1.5 com una actualització de la 1.0 respecte la versió 1.0 comercial adquirida prèviament. Microsoft permet que en cas de no haver distribuït les 200 rutines per la versió 1.0 es puguin distribuir les restants amb la versió 1.5, amb els mateixos termes de la llicència 1.0. No es poden sobrepassar les 200.
El codi de mostra que s'inclou es pot modificar i distribuir d'acord als termes de llicència.

2.3.5 Comparativa

En aquest punt es portarà a terme una comparació objectiva entre Microsoft Robotics Studio i la resta de programes analitzats prèviament: URBI, Player/Stage/Gazebo i Webots. Es recordaran les principals característiques de cada un dels programes.

Primer s'analitzaran els tipus de llicències d'adquisició i les plataformes d'execució disponibles ja que són paràmetres fàcilment mesurables i després s'analitzarà un conjunt de característiques tècniques de més difícil comparació.

2.3.5.1 Llicència d'adquisició

Per a començar parlarem de les llicències amb les que es pot descarregar i utilitzar cada programa. En aquest aspecte MRS queda en un terme mitjà:

URBI	Llicència gratuïta per a descarregar-lo i utilitzar-lo	●
Player/Stage/Gazebo	Llicència lliure per a descarregar-lo, utilitzar-lo i modificar-lo	●
Webots	Llicència de pagament	●
Microsoft Robotics Studio	Llicència gratuïta per a propòsits no comercials i de pagament per a comercials / límit de distribucions	●

Taula 1

Tal com observem a la Taula 1, Player/Stage/Gazebo és totalment gratuït, a més de modificable, pel que representa la millor opció en aquest aspecte, especialment si es té interès en millorar-lo.

Microsoft Robotics Studio presenta una llicència lliure, que és la que s'utilitza en l'elaboració d'aquest projecte, indicada per a ús educacional, pel qual és adequat i una de pagament. Suposa un lleu desavantatge el fet que per a ús comercial s'hagi de comprar una llicència.

2.3.5.2 Plataformes d'execució

A continuació s'esmentaran les plataformes en que es poden executar els programes:

URBI	Windows, Linux i Mac	●
Player/Stage/Gazebo	Linux i Mac	●
Webots	Windows, Linux i Mac	●
Microsoft Robotics Studio	Windows	●

Taula 2

URBI i Webots són els programes més compatibles. Microsoft Robotics Studio només funciona en Windows, pel que és més limitat, i suposa un desavantatge.

En el desenvolupament d'aquest projecte s'utilitza la plataforma Windows XP, pel que no suposa cap problema aquesta limitació.

2.3.5.3 Dades tècniques

Les dades que s'analitzaran en aquest punt són:

- **Compatibilitat amb robots**: quants robots comercials suporten l'ús del programa
- **Facilitat**: com de fàcil i intuïtiu és el programa per treballar-hi (especialment per a usuaris novells)
- **Possibilitats:** en aquesta característica es mesuraran les possibilitats del programa
- **Simulació:** primer, si té la possibilitat, i després quant de potent és

PROGRAMA	COMPATIBILITAT	FACILITAT	POSSIBILITATS	SIMULACIÓ
URBI	Baixa	Alta	Mitja/Alta	-
Player/Stage/Gazebo	Alta	Baixa	Alta	Mitja/Alta
Webots	Mitja/Baixa	Mitja	Mitja/Alta	Alta
Microsoft Robotics Studio	Alta	Mitja/Alta	Alta	Alta

Taula 3

En aquests termes, més centrats en el que és el propi funcionament del programa és Microsoft Robotics Studio el que destaca per sobre dels demés, amb una valoració alta en gairebé totes les parcel·les.

Els programes més **compatibles** amb robots existents al mercat són el Player/Stage/Gazebo i el Microsoft Robotics Studio, que tenen gran varietat de possibilitats, a part de robots personalitzats. URBI i Webots es poden utilitzar en un rang petit (3-6) de robots, a part dels personalitzats. Però, la personalització segons el robot és molt alta.

La **facilitat** és un aspecte en que URBI surt guanyant. Una de les característiques de les que fa gala URBI és, precisament, la seva simplicitat per tal de millorar la creativitat dels usuaris, amb un funcionament molt simple que no fa difícil aprendre a controlar els robots. Microsoft Robotics Studio és també fàcil d'utilitzar per a iniciar-se, gràcies sobre tot a la programació visual per a gent que no té nocions de programació. És, però, una plataforma molt més complexa, també indicada per a usuaris experts, ja que moltes tasques no són fàcils de realitzar.

Les **possibilitats** de Player/Stage/Gazebo i Microsoft Robotics Studio són les més elevades. Player/Stage/Gazebo és obert totalment i permet fer modificacions al programa deixant a la comunitat que l'ajudi a millorar. És un programa en constant evolució i amb moltes extensions possibles, pel que resulta molt potent. Microsoft Robotics Studio és un programa molt complet, amb molts mòduls i també amb moltes possibilitats.

La **simulació**, com es pot veure, estan gairebé tots els programes molt igualats. URBI no en disposa d'un entorn de simulació. Actualment s'està treballant en URBI Studio, similar al programa de Microsoft, però de moment no ha aparegut al mercat. Respecte als altres, Webots disposa d'un mòdul de simulació molt potent, amb possibilitat de supervisar i de simular molt més ràpid. No hi ha gaire robots compatibles, pel que el seu modelat 3D està molt aconseguit. Microsoft Robotics Studio té un simulador realment potent, a part de tenir moltes possibilitats el motor basat en *PhysX* el fa un simulador molt realista, amb molts detalls. Player/Stage/Gazebo és també un bon simulador però menys potent que el de Microsoft, encara que igualment funcional..

2.3.5.4 Conclusions

Un cop analitzats aquests termes es poden extreure les següents conclusions:

- URBI és una gran alternativa als altres programes si no es pretén tenir el màxim de prestacions i es vol un sistema instal·lable a qualsevol plataforma. Cal esmentar que és gratuït i que s'està treballant en millorar-lo, pel que en un futur potser és encara millor alternativa. Per exemple inclourà programació visual, molt similar a Microsoft Robotics Studio.
- Player/Stage/Gazebo es presenta com la gran alternativa a Microsoft Robotics Studio. És, de fet, l'altre aplicació més utilitzada en l'àmbit de la robòtica. És un sistema molt potent i amb una comunitat molt gran darrera que li proporciona un suport i una capacitat de millora molt gran. Té, però, la desavantatge de no ser compatible amb Windows, i això li resta molt mercat, així com tampoc és una aplicació fàcil d'utilitzar.
- Webots és, potser, el més fluix, ja que no és gratuït i no és millor que els altres. Pot suposar una bona aplicació a empreses que es dediquin a explotar algun dels robots que suporta, ja que està limitat en aquest aspecte i en facilita la seva utilització.

- Microsoft Robotics Studio és tant bo o millor que els altres en molts aspectes, però pitjor en algun altre. Resulta una molt bona opció, la millor segurament en entorn Windows, ja que manté un bon equilibri entre possibilitats i dificultat d'utilització. Presenta algunes avantatges i desavantatges respecte els altres programes:
  - **Avantatges**: és el que disposa de millor simulador, és molt complet, molt potent i té infinites possibilitats, a més de compatible amb robots i fàcil per a iniciar-se. També la seva llicència gratuïta és molt interessant per a començar i aprendre.
  - **Desavantatges**: presenta un gran inconvenient que és la plataforma on s'ha d'instal·lar: Windows. Aquesta característica li resta molt mercat i en limita el nombre d'usuaris, així com que en la llicència comercial per la que s'ha de pagar, que ja suposa un desavantatge per sí mateix, la limitació en el nombre de components de rutines distribuïdes és de només 200, fet que a les empreses que tinguin intenció de fer negoci els pot resultar incòmode.

Referències

[1] Merriam Webster, "Definició Robotics", Merriam-Webster Online Dictionary, 2008 , http://mw1.merriam-webster.com/dictionary/Robotics, (accés 29 Juliol 2008)

[2] Real Academia Española, "Definició Robot", Real Academia Española. Diccionario usual, 2008 , http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&TIPO_BUS=3&LEMA=robot, (accés 29 Juliol 2008)

[3] Wikipedia, "Definició Robot", Wikipedia, 25 Juliol 2008, the online enciclopedia, http://en.wikipedia.org/wiki/Robot, (accés 29 Juliol 2008)

[4] Wikipedia, "Definició Aibo", Wikipedia, 23 Juliol 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Aibo, (accés 4 Agost 2008)

[5] Wikipedia, "Definició Lego Mindstorms", Wikipedia, 10 Maig 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Lego_Mindstorms, (accés 4 Agost 2008)

[6] Wikipedia, "Definició Nao robot", Wikipedia, 2 Agost 2008, the online enciclopedia, http://en.wikipedia.org/wiki/Nao_(robot), (accés 4 Agost 2008)

[7] Wikipedia, "Definició iRobot Roomba", Wikipedia, 4 Agost 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Roomba, (accés 6 Agost 2008)

[8] Wikipedia, "Definició URBI", Wikipedia, 5 Juny 2008, the online enciclopedia, http://es.wikipedia.org/wiki/URBI, (accés 30 Juliol 2008)

[9] Aldebaran Robotics, "Nao-About", Aldebaran Robotics, Setembre 2007, http://www.aldebaran-robotics.com/eng/Nao.php, (accés 4 Agost 2008)

[10] K-Team Corporation, "Robot Hemisson, robot Kephera i robot E-Puck", K-Team Products, 2008, http://www.k-team.com/kteam/home.php?rub=0&site=1&version=EN, (accés 29 Juliol 2008)

[11] iRobot Corporation, "iRobot Roomba Family", iRobot, 2008, http://www.irobot.com/, (accés 6 Agost 2008)

[12] Gostai, "URBI", Gostai Robotics, 2008, http://www.gostai.com, (Accés 30 Juliol 2008)

[13] Múltiples, "Player-Stage-Gazebo", Player Project, 28 Març 2008, http://playerstage.sourceforge.net, (Accés 31 Juliol 2008)

[14] Cyberbotics Ltd, "Webots 5", Cyberbotics Professional Mobile Robot Simulation, Març 2008, http://www.cyberbotics.com/products/webots/index.html, (accés 1 Agost 2008)

[15] Microsoft, "Documentació Microsoft Robotics Studio", Microsoft Developer Network, 2008, http://msdn.microsoft.com/es-es/library/bb483024(en-us).aspx, (accés 4 Agost 2008)

[16] Múltiples, "Player-Stage-Gazebo", Player Project, 28 Març 2008, http://playerstage.sourceforge.net, (accés 31 Juliol 2008)

[17] Sony Entertainment Robot Europe, "Aibo", Sony Aibo Europe, 2006, http://support.sony-europe.com/aibo/index.asp, (accés 4 Agost 2008)

[18] The LEGO Group, "Lego Mindstorms NXT", Lego Mindstorms NXT Home, 2008, http://mindstorms.lego.com/eng/Salzburg_dest/default.aspx, (accés 4 Agost 2008)

3 Revisió del material docent de MRS

En aquest capítol el que es farà serà treballar amb el material docent que existeix a Internet i el que inclou el propi Microsoft Robotics Studio. D'una banda aquest treball servirà per a la meva pròpia formació en la utilització de l'aplicació i d'altre banda per a realitzar un estudi sobre la qualitat del material docent que hi ha i trobar quines són les mancances que presenta aquest material i què seria interessant aportar, millorar o complementar.

Cal destacar que aquest projecte s'ha desenvolupat sobre la versió **Microsoft Robotics Studio 1.5**, que és la versió final que hi ha en aquest moment. Hi ha la versió *CTP*[12] *July*, la qual conté novetats, però no és una versió final, i no disposa de la popularitat en el mercat de la versió 1.5.

3.1 Documentació

Quan s'instal·la Microsoft Robotics Studio, a la seva carpeta (tant a la instal·lació com al menú inici) es troba un arxiu de documentació (*.chm*). És un arxiu HTML Compilat:

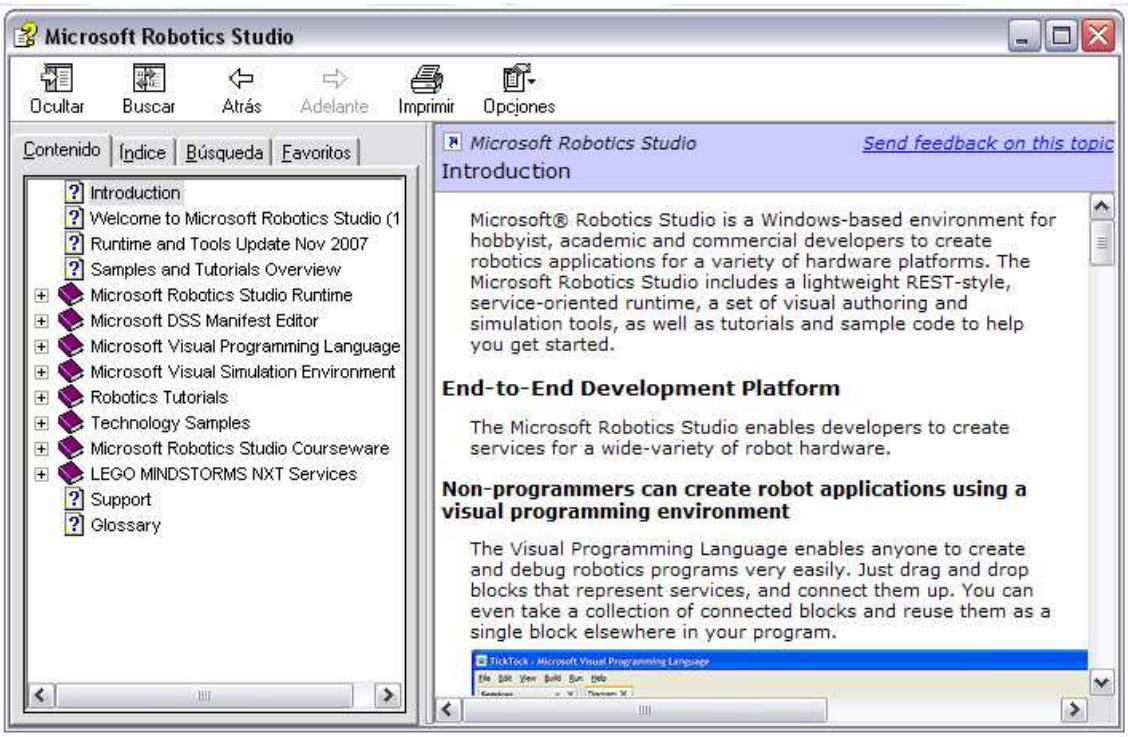

Figura 3-1. Arxiu de documentació

[12] *CTP: Community Technical Preview: versió per a mostrar a la comunitat*

Aquest fitxer és una guia on es pot accedir fàcilment a totes les seves parts, tal com es mostra a la Figura 3-1 a la part de l'esquerra mentre que a la part de la dreta es veu el seu contingut.

Conté molta ajuda: va des de una introducció al programa a diversos *tutorials* (en tots els àmbits dels que disposa el Microsoft Robotics Studio*)* passant per guies d'usuari. Tota aquesta informació es troba exactament reproduïda també a la pàgina web de MSDN[13].

La documentació existent l'ha creat directament l'equip de desenvolupament, no ha passat per un equip d'edició. Això ho fa Microsoft quan una documentació està més "madura". Tant al fitxer com a la web la informació està tan sols en anglès. La traducció de la mateixa al castellà depèn de la organització local de Microsoft Espanya, i sembla que de moment no tenen intenció de fer-ho.

Malgrat que Microsoft Robotics Studio permet programar en diversos llenguatges (C#, C++, Visual Basic .NET, Python i Visual Programming Language) i operar amb alguns robots reals, cal destacar que en aquest projecte, la guia resultant del treball realitzat es centrarà tant en el Visual Programming Language com en la simulació, sense requerir cap robot real.

S'ha decidit això ja que es pretén que sigui una guia d'introducció per gent sense gaires nocions de robòtica ni de programació, essent VPL a part, un entorn de programació agradable i intuïtiu que anima a l'usuari a "jugar", a diferència dels entorns que es basen purament en codi. Aquest entorn de programació visual és, també, l'element que diferencia Microsoft Robotics Studio de les altres aplicacions amb objectius similars.

Un altre fet que ha ajudat a prendre aquesta decisió és que per a programar en aquests llenguatges es requereix d'un altre entorn de programació, principalment Visual Studio (algunes de les versions disponibles, o programes dins la *suite*) i això implica utilitzar més programes que no pas el Microsoft Robotics Studio tan sols, allunyant una mica la idea de programar amb ell, precisament del que es tracta.

La documentació que inclou el Microsoft Robotics Studio inclou diversos *tutorials*, com ja s'ha esmentat prèviament i molts d'ells tenen requisits tant de hardware com de software. Alguns estan fets per a utilitzar amb robots concrets: *Roomba, Lego NXT*, etc. Si es disposa d'aquests robots són *tutorials* molt interessants, ja que permeten provar moltes coses, però si no es disposa de cap i simplement es pretén aprendre a utilitzar una mica l'aplicació simulant els robots, no ho són. Els requisits de software fan referència a entorns de programació, normalment a Visual Studio, com s'ha dit anteriorment.

Tot això ens deixa pocs *tutorials* sense cap més requisit que tenir Microsoft Robotics Studio instal·lat. Tot i així, a continuació s'analitzarà el contingut dels diferents apartats de la documentació molt per sobre, posant especial interès en els *tutorials*.

[13] *MSDN (*Microsoft *Developer Network): http://msdn.*Microsoft*.com/es-es/robotics/default(en-us).aspx*

El primer apartat: ***Microsoft Robotics Studio Runtime***.

Figura 3-2. Contingut de l'apartat MRS Runtime

Aquest és un apartat extens i complex en el que s'explica amb detall com funciona tota la rutina de Microsoft Robotics Studio: **CCR (*Concurrency and Coordination Runtime*) i DSS (*Decentralized Software Services*)**. Aquestes rutines (explicades amb més detall al Capítol 2) són la base del funcionament de l'entorn Microsoft Robotics Studio, dels programes que es desenvolupen amb ell, pel que és un apartat que s'explica tot amb molt de detall i amb un alt nivell de complexitat.

La ***Concurrency and Coordination Runtime*** *(Rutina de Concurrència i Coordinació* - CCR) fa simple la programació asíncrona. Fa simple escriure programes per gestionar entrades asíncrones de múltiples sensors de robots i sortides a motors i actuadors.

El model d'aplicació de ***Decentralized Software Services*** (*Serveis de Software Descentralitzats* - DSS) fa simple accedir i respondre a l'estat d'un robot utilitzant un navegador web (Figura 2-47) o una aplicació basada en Windows.

A la Figura 3-2 es pot veure que conté *tutorials* de DSS. Aquests són en C# i es divideixen en dos blocs: *Service* i *Hosting*. En el primer bloc s'ensenya a l'usuari algunes característiques del model d'aplicació orientat a serveis de Microsoft Robotics Studio i en el segon se'l proveeix d'uns tutorials d'introducció a com un node DSS pot fer de *host* amb altres aplicacions. Un node DSS és un entorn de *hosting* que disposa de suport per a serveis que poden ser creats i manipulats fins que s'esborrin o es pari el node DSS.

Els serveis són blocs bàsics que s'utilitzen per construir aplicacions. Tots els serveis estan basats en el model d'aplicació DSS. Un robot es pot entendre com un conjunt de serveis, cadascun d'ells associat a una funcionalitat del robot.

El següent apartat que trobem és ***Microsoft DSS Manifest Editor***:

Figura 3-3. DSS *Manifest Editor*

L'editor de manifests és una aplicació dissenyada per a permetre a l'usuari tant crear com editar manifests. Els manifests són arxius XML que contenen una llista de serveis i la seva configuració.

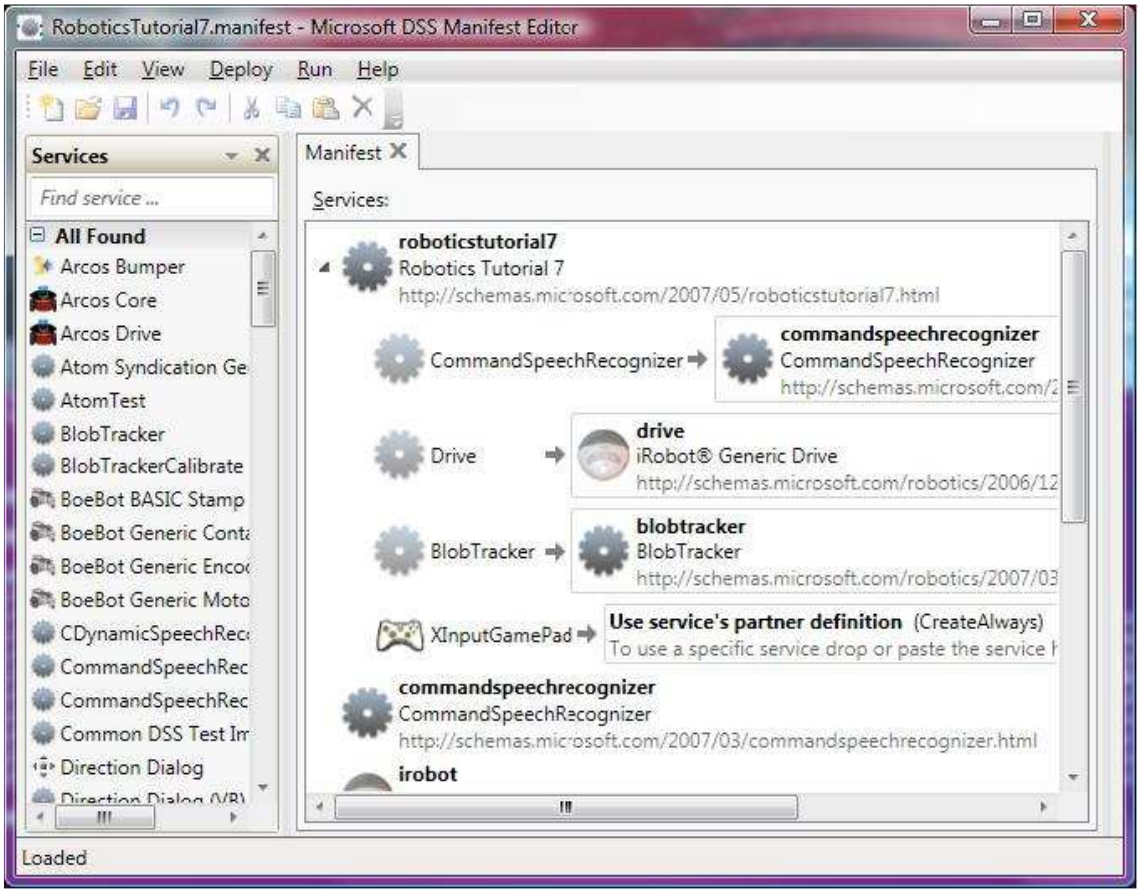

Figura 3-4. Pantalla del DSS *Manifest Editor*

Aquest apartat explica com funciona l'editor de manifests bàsicament, sense entrar en conceptes teòrics.

Seguirem amb la secció dedicada a Visual Programming Language, que és el que més ens interessa, com a mínim inicialment. Aquest apartat inclou 4 punts:

- *Getting Started*: una breu introducció a VPL explicant les seves possibilitats
- *User Guide*: explicacions de com realitzar algunes operacions simples: crear un diagrama, configurar activitats, connectar-les, generació de codi...
- *Reference*: menús de l'aplicació, tipus de variables i explicació de les *Activities* bàsiques de les que disposa el programa
- VPL *Introductory Tutorials*: 4 *tutorials* sobre VPL que serveixen d'introducció

L'últim apartat és el que ens interessa més:

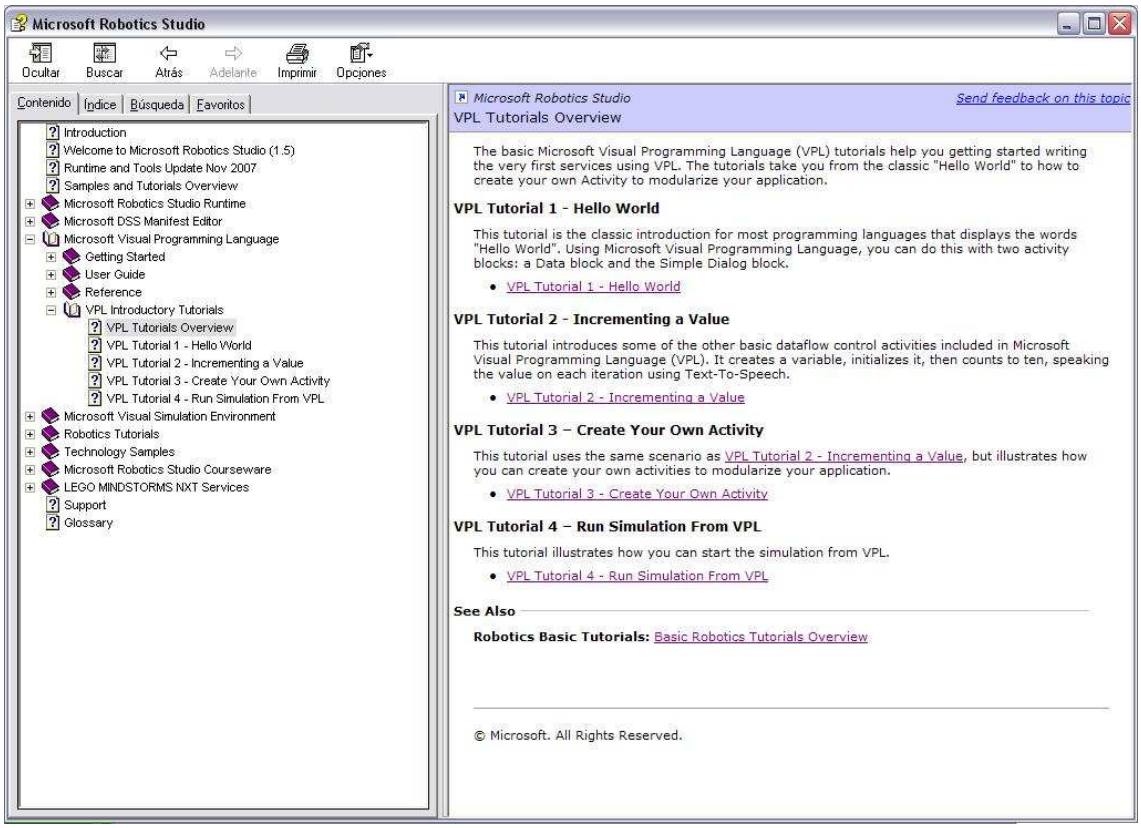

Figura 3-5. Pantalla VPL Introductory Tutorials

Els *tutorials* que es poden veure a la Figura 3-5 són 4, els quals són molt bàsics. Efectivament però, són tutorials molt útils per a iniciar-se, ja que no acompleixen tasques molt complexes:

- desenvolupar un "*Hello World*"
- incrementar un valor
- crear una activitat
- executar una simulació

Són bons *tutorials*, amb un apartat de prerequisits (hardware i software), una mica incòmodes de seguir en alguns punts però ben il·lustrats amb moltes imatges, el que ajuda al seu seguiment. Són, com diu el seu propi nom (*Introductory*), molt bàsics, i no s'arriba a realitzar cap tasca de cert nivell.

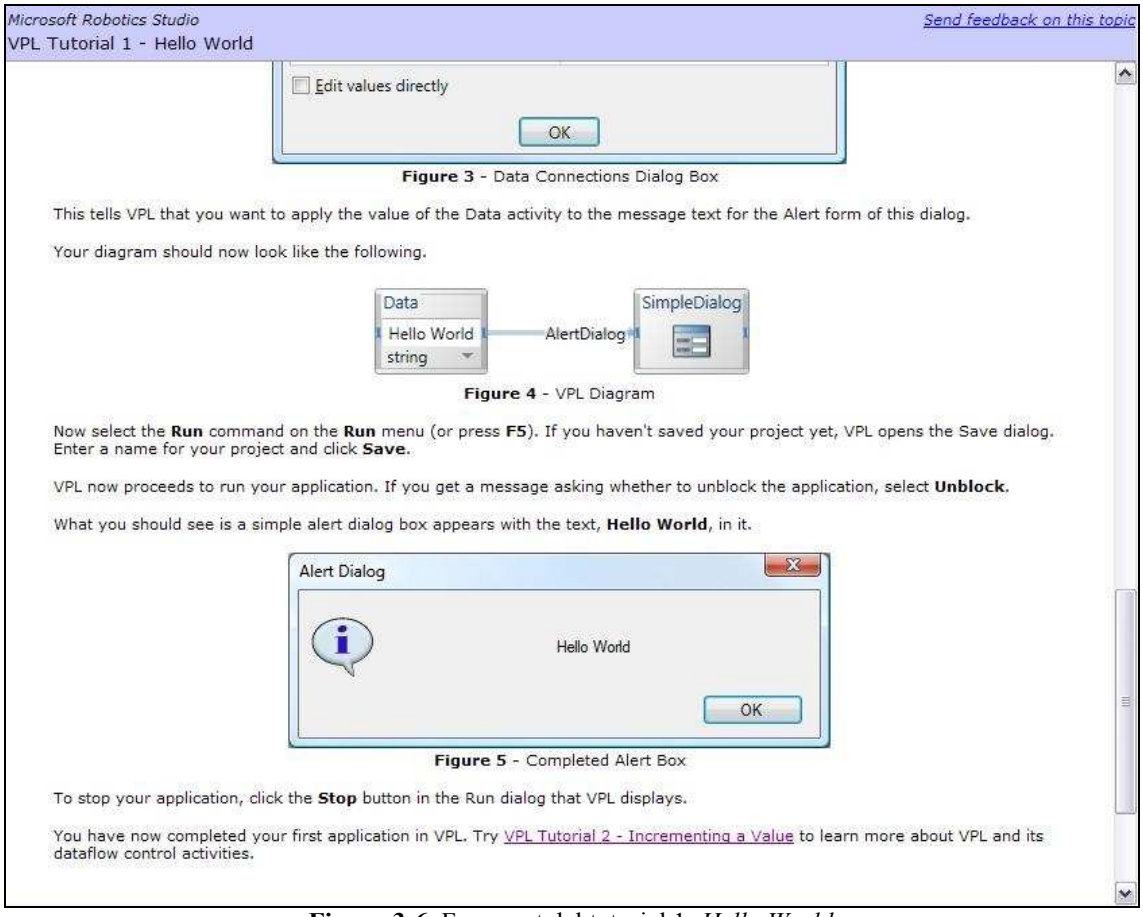

Figura 3-6. Fragment del tutorial 1: *Hello World*

Cal destacar que tots els *tutorials* que es troben en aquesta documentació estan ja realitzats i es poden trobar els fitxers a la carpeta on és instal·lat el Microsoft Robotics Studio de forma que es poden executar directament, per exemple si l'usuari no desitja realitzar el tutorial i tan sols vol veure el resultat, o si troba algun problema i no pot avançar.

El *VPL Tutorial 4* tracta de fer anar una simulació. Fins aquest no ens trobem el primer problema, més enllà de que les imatges del programa que ens presenten pertanyen a versions anteriors i no quadren completament, donant lloc a alguna petita confusió.

Per començar, la Rutina de Simulació necessita una targeta gràfica que suporti *DirectX9* amb "*shader model*" 2.0 o superior. Són simulacions a alt nivell i amb molt detall, i és per això que requereix una targeta gràfica de cert nivell.

Tot i acomplir els requisits i haver realitzat una instal·lació correcta de Microsoft Robotics Studio, de vegades no funciona la simulació. Això és degut a que a l'hora d'instal·lar l'aplicació s'instal·la XNA *2.0*[14], però l'entorn de simulació (anomenat *Simulation Environment*) requereix XNA 1.0 i, malgrat que la versió 2.0 és superior, s'ha d'instal·lar la versió 1.0 per a poder executar les simulacions (ambdues versions conviuen instal·lades al sistema). És un error comú, i no és fàcil de trobar, però, és tan fàcil de solucionar com això, tot i que és prou recriminable a Microsoft.

[14] *XNA: API desenvolupada per* Microsoft *principalment pel desenvolupament de videojocs*

És un punt negatiu d'aquest tutorial de la documentació, ja que en cap moment s'indica que pot ocórrer aquest error ni com solucionar-ho, tan sols indica el requisit de la targeta gràfica. També, però, és un problema de la pròpia instal·lació, més enllà de la documentació.

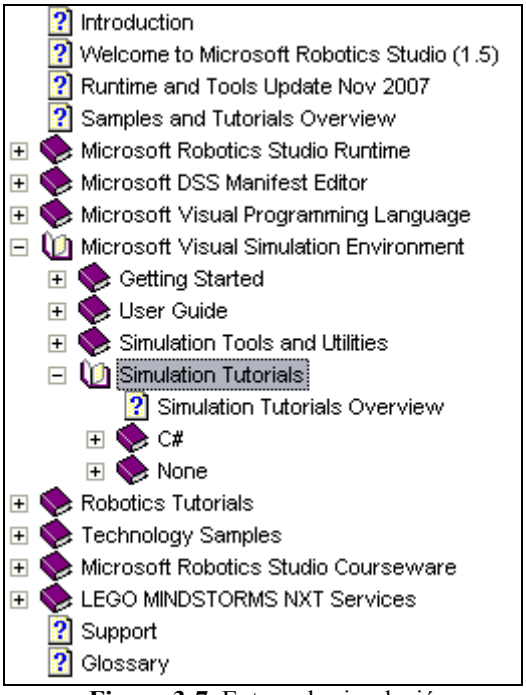

Figura 3-7. Entorn de simulació

Més endavant, en els següents apartats, trobem *tutorials* generals, per a diversos llenguatges. En primer lloc està ***Visual Simulation Environment*** (Figura 3-7), apartat que incorpora diversos tutorials per C# (per a programar l'entorn, tant per a editar el món com per afegir robots i objectes) i dos de, directament, el simulador. En aquests últims es pot veure, a grans trets, com funciona el simulador i quins paràmetres es poden configurar.

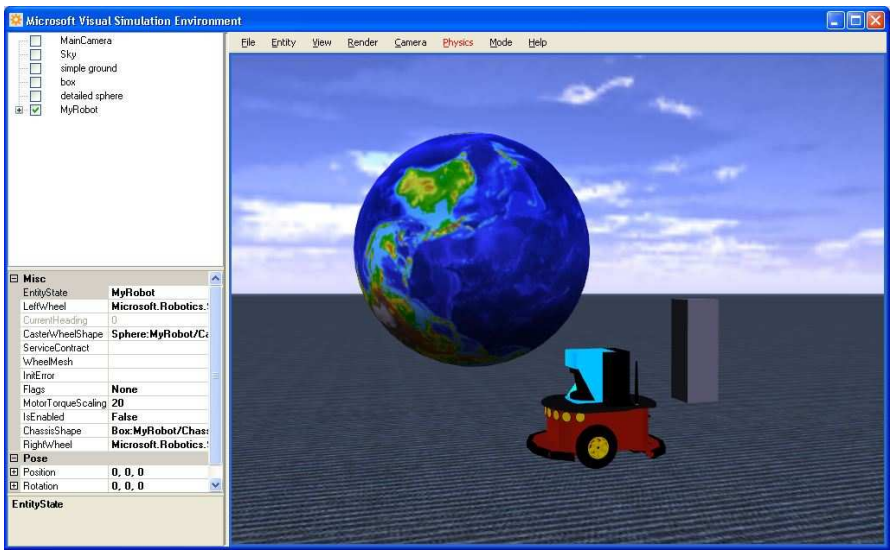

Figura 3-8. Pantalla de l'entorn de simulació

Prosseguint amb la documentació, posteriorment es troba l'apartat ***Robotics Tutorials***. Aquest és un apartat amb diversos tipus de tutorials, però, molt centrats en el treball directe amb robots reals. El primer lloc on trobem VPL és al *Robotics Tutorials/Basic Robotics Tutorials/VPL*. La idea d'aquest apartat és la d'introduir a l'usuari en els primers serveis per als seus robots. Inclou diversos *tutorials*:

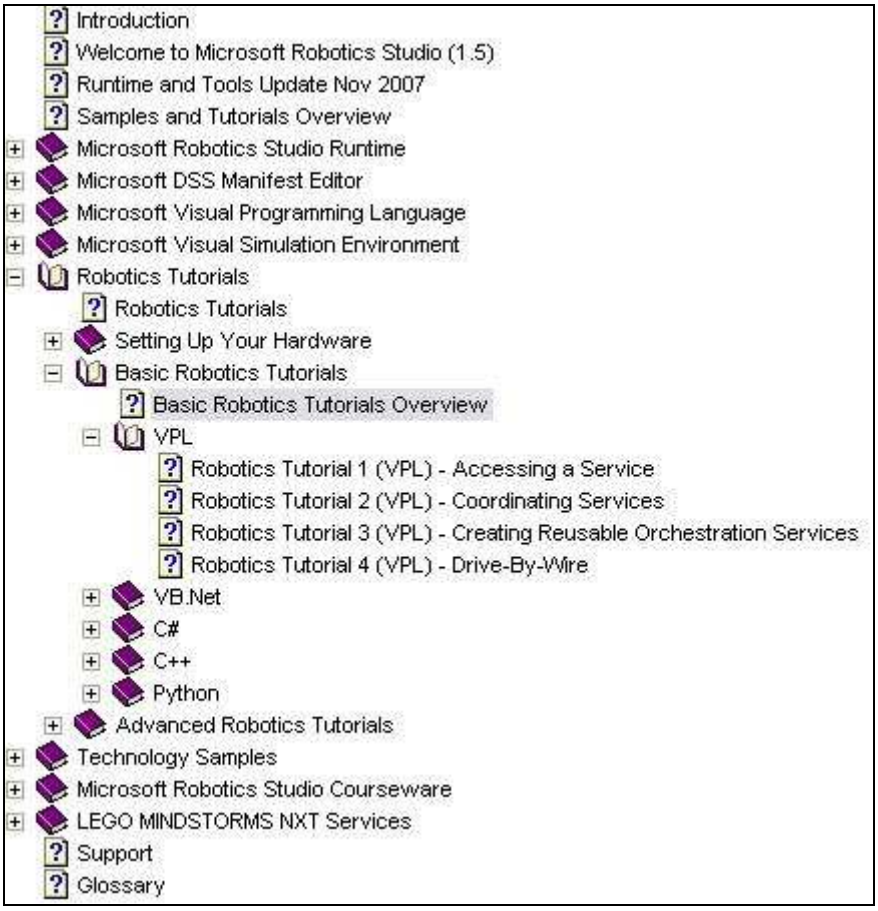

Figura 3-9. Secció *Basic Robotics Tutorials*

En aquests *tutorials* es tracta de fer aproximadament el mateix amb els diferents llenguatges. És un apartat interessant perquè es poden veure els mateixos resultats però variant el llenguatge de programació.

Es requereix d'un robot, pel que no resulta tant útil per aprendre si no en disposem de cap, ja que no es pot simular (o no explica com). Es treballa amb els sensors que incorpora el robot, ja sigui imprimint un missatge a pantalla o fent que el propi robot reaccioni.

Aquests tutorials es basen en els serveis. Els serveis són un element molt important en l'arquitectura de MRS, poden representar diverses funcions:

- Una aplicació escrita amb MRS és un servei (component software)
- Sensors i actuadors (component hardware)
- El codi que permet comunicar amb un robot és un servei

- Els contractes genèrics són serveis. Permeten fer servir parts concretes de robots, com sensors o càmeres. Aquests serveis són els mateixos pels diferents robots, per tant es pot utilitzar el mateix programa per diversos robots
- Altres comportaments. Es pot utilitzar el mateix servei en diferents programes

Els serveis s'executen dintre del context d'un node DSS. En els *tutorials* d'aquesta secció s'accedeix a un servei, es coordinen entre ells, es reutilitzen (recordem que es pot escriure un servei d'acord amb una especificació hardware i després reutilitzar-lo amb una varietat de hardware de robots) i es controlen.

El següent apartat és ***Technology Samples***, el qual està molt dedicat a mostrar a l'usuari una introducció de com interaccionar amb algunes tecnologies i robots.

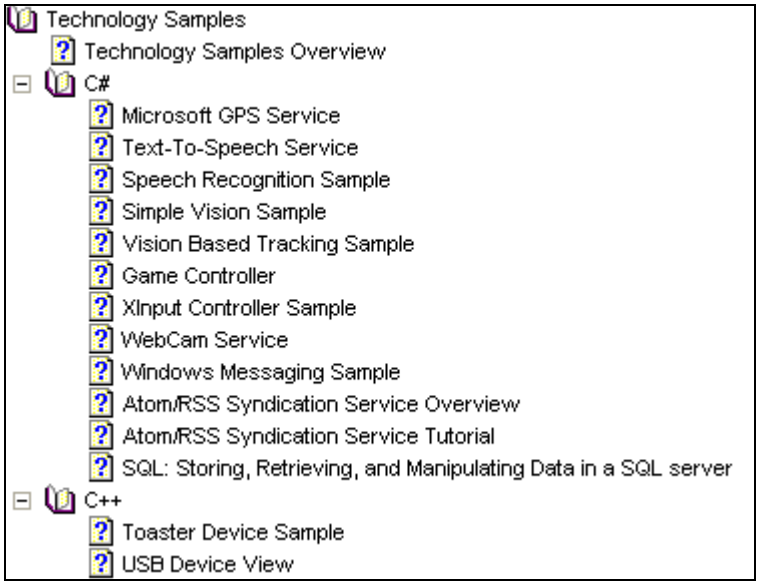

Figura 3-10. Apartat *Technology Samples*

Tracta de tecnologies com GPS, Webcam, controladors, etc. Els *tutorials* que conté són nomes per C# (la gran majoria) i C++, essent indicat per a usuaris més avançats o que vulguin treballar amb aquestes tecnologies.

A continuació ens trobem l'apartat ***Microsoft Robotics Studio Courseware***. Aquest apartat el forma un conjunt de "pràctiques" que, tal com suggereix, es poden completar individualment o fer-les part d'un curs de robòtica.

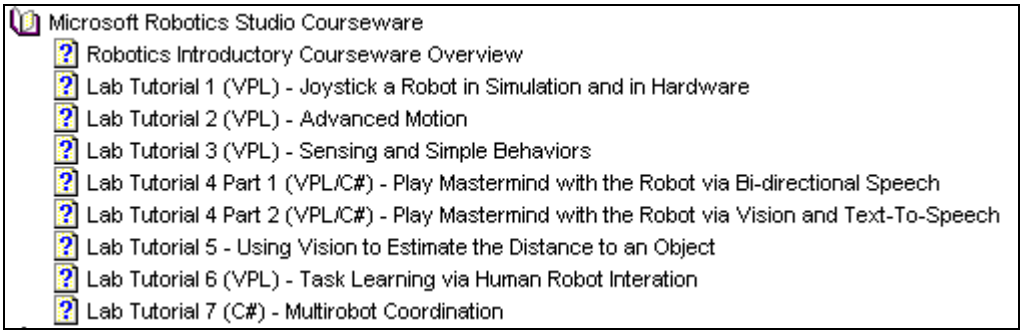

Figura 3-11. Microsoft Robotics Studio Courseware

És, precisament, un curs d'introducció, pel que tracta conceptes bàsics de la robòtica com controlar sensors, actuadors, rutines o interactuar amb l'entorn. Aquests tutorials arriben més enllà que els d'introducció a VPL. Tal com es pot veure a la Figura 3-11. **Microsoft Robotics Studio Courseware**, tot el que explica ho fa amb VPL i C#, de manera que, malgrat que pretén ser introductori, requereix certs coneixements de programació, donant a les sessions una profunditat major.

Els primers *tutorials*, que són de VPL tan sols, introdueixen alguns conceptes nous que no s'han vist als *tutorials* d'altres apartats, com l'accés a variables i el control de sensors. Concretament, el primer d'ells és un senzill tutorial en el que es veu com controlar un robot a l'entorn de simulació amb un controlador (si no es disposa d'un físic s'utilitza un virtual).

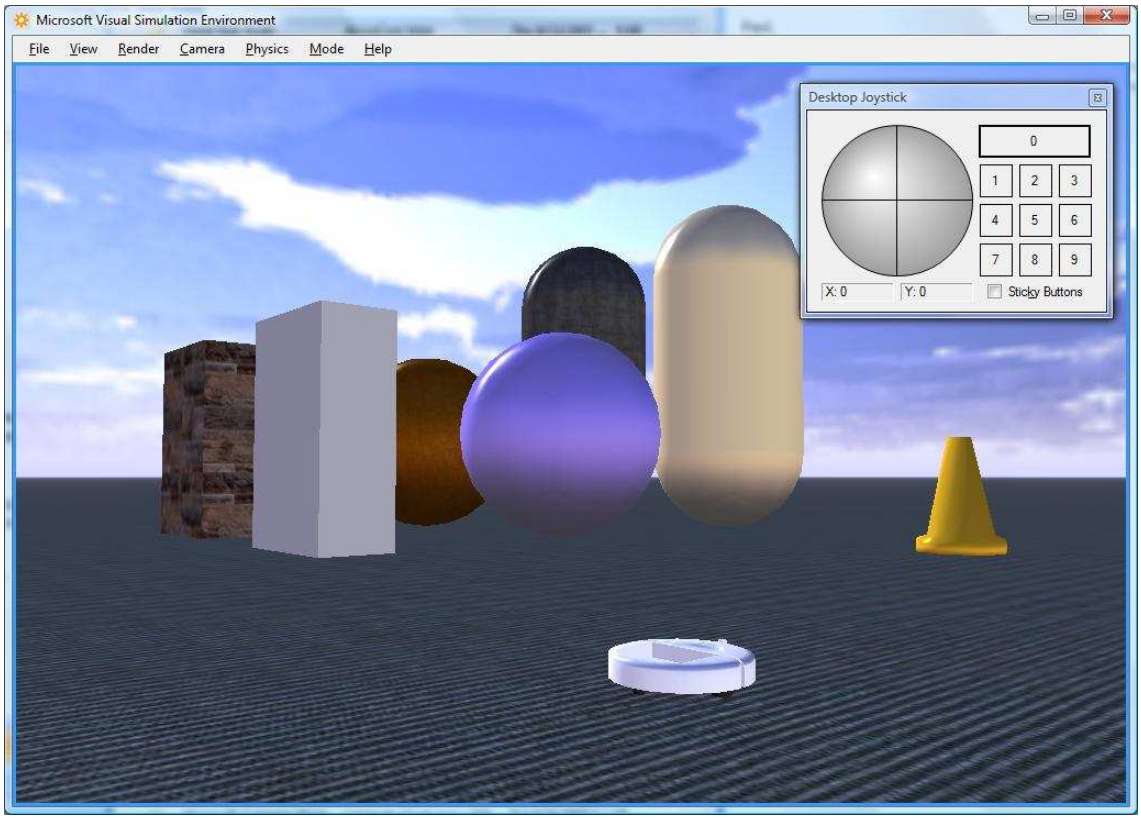

Figura 3-12. Robot Roomba simulat a l'entorn de simulació i controlat amb joystick

El segon tutorial també tracta de controlar el *iRobot Create*, però en aquest cas de forma automàtica (simplement el moviment, en forma de semicercles). Alguns dels *tutorials* es poden realitzar simulant (el primer i segon), però la majoria requereixen d'un robot (el *iRobot Create*) i/o d'altres sensors, com càmeres (no tothom en disposa d'una), micròfons o altaveus (gairebé tothom en té).

Un petit error trobat és que en un moment al *tutorial* suggereix muntar un circuit (col·locació i connexió d'activitats i serveis), però posteriorment indica que es desfaci i es faci d'una altra manera per a fer-ho més simple (amb menys activitats). L'error ve perquè la primera figura no funciona, la connexió entre activitats és incompatible i el propi VPL ho indica quan es realitza la connexió. No afecta al desenvolupament del

tutorial, ja que la forma correcta d'implementar-ho és la segona, però presenta un punt de confusió a l'usuari, que si no segueix el tutorial no s'adona que no s'ha de fer així.

Un altre lleuger error és que es requereix haver instal·lat l'actualització de mostres per a tenir tots els serveis, cosa que en cap moment s'informa a l'usuari.

En el *tutorial* 3 es tracta un tema molt interessant: el comportament del robot per a arribar a un objectiu. Existeixen tres tipus de comportaments possibles:

- Reactiu: el robot escull què fer en resposta a un canvi en l'entorn
- Deliberatiu: el robot pensa en les raons sobre les possibles accions que realitzi abans de decidir quina fa
- Híbrid: combina els dos

En aquest *tutorial* es combinen diversos comportaments reactius ja que no impliquen la utilització de memòria, el que implica que un comportament reactiu no pot ser diferent en el cas del mateix estat. Molts comportaments reactius es poden combinar en un sistema complex.

Un exemple són els comportaments *go-to-goal* que dirigeixen el robot cap a l'objectiu i *avoid-obstacle-behavior* que fan que el robot esquivi els objectes. Combinar aquests dos comportaments permeten al robot arribar a l'objectiu sense xocar amb els objectes (Figura 3-13).

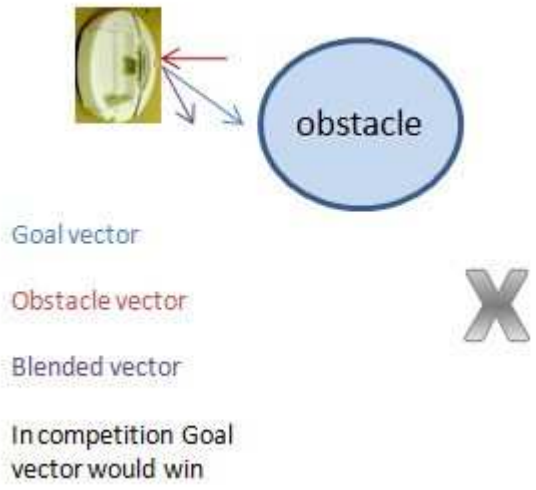

Figura 3-13. Combinació de vectors de comportament

En la figura anterior es poden veure uns vectors que simbolitzen força i direcció. Combinar les forces dels obstacles i les de l'objectiu dóna lloc al vector resultant (*blended vector*).

Un altre problema que sorgeix al utilitzar aquest *tutorial* és que la mostra ja realitzada del tutorial que es pot trobar a la carpeta del programa no funciona correctament. Si anem a la carpeta trobem la versió amb robot i la simulada, i si executem la simulada ens trobem que el robot es mou en forma recta fins que xoca amb una paret, sense evitar els obstacles.

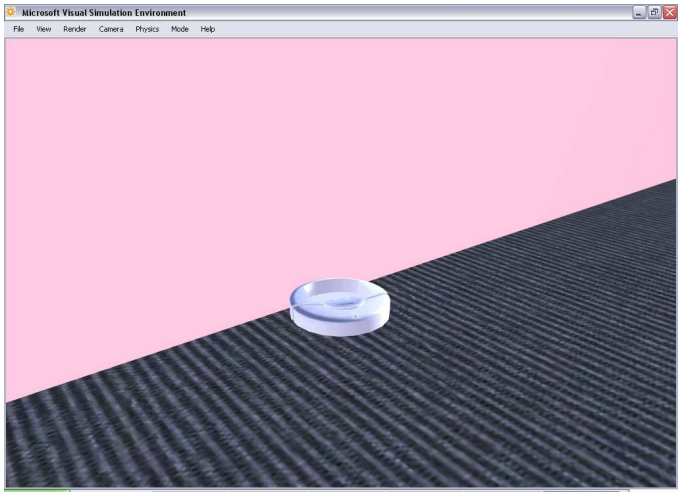

Figura 3-14. Robot xocant

Els següents tutorials requereixen, a més de VPL, treballar amb C#. En el número 4 mostra com fer reconeixement de veu i passar text a veu fent un joc (*Mastermind*) i fent jugar al robot amb l'usuari.

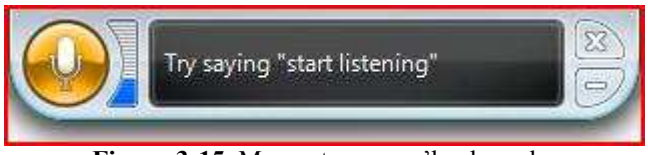

Figura 3-15. Moment en que s'ha de parlar

El següent tutorial és una introducció al reconeixement de imatge amb càmera, els serveis bàsics que ofereix, com segmentació de colors i detecció de *blobs*[15]. També jugant al *Mastermind*, el robot serà el moderador i el jugador li ensenyarà papers de colors als qual el robot respondrà.

[15] *Blob: objectes bàsics en imatges*

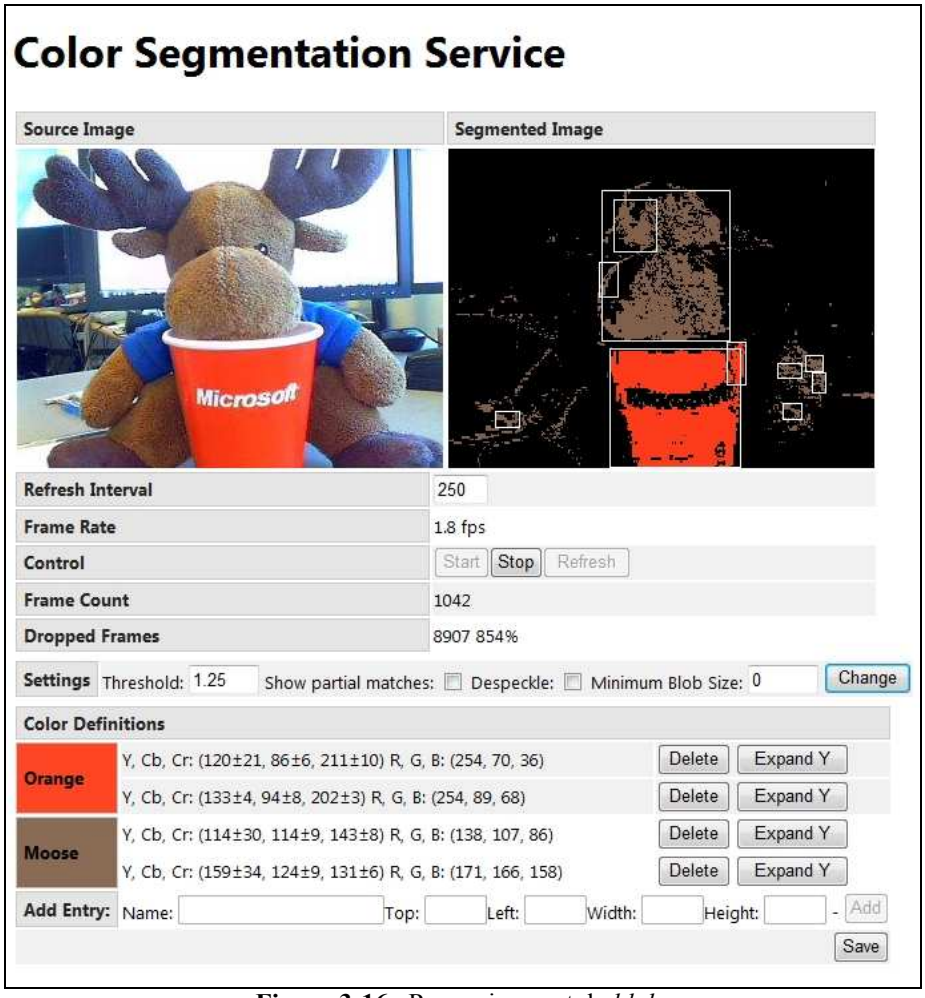

Figura 3-16. Reconeixement de *blobs*

La següent sessió de laboratori ensenya a estimar la distància entre el robot i un objecte.

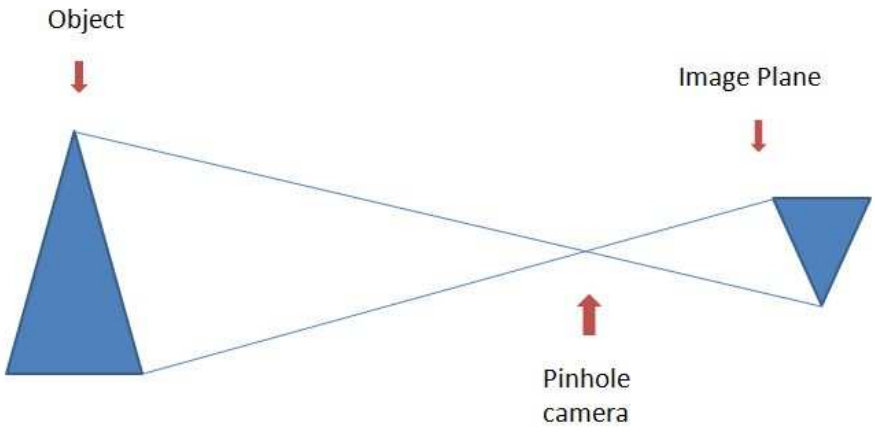

Figura 3-17. Distància entre el robot i l'objecte

En el següent (número 6) es torna al reconeixement de veu i s'ensenya a l'usuari a controlar al robot mitjançant comandes de veu. Aquest és l'últim *tutorial* de VPL del *Courseware*, així que inclou tasques més avançades, com per exemple el procés de com VPL decideix quin flux de control és exclusiu (quins processos s'executaran concurrentment i quins no). En aquest procés participa la rutina CCR.

Finalment l'última secció del curs (en codi C#) tracta de coordinar multitud de robots (comunicant-se entre ells), mentre s'examina una estratègia de coordinació descentralitzada i centralitzada. S'aprèn a construir serveis des de zero també. En aquest tutorial s'utilitzen dos robots.

L'últim apartat de la documentació és un dedicat a ***Lego Mindstorms NXT***:

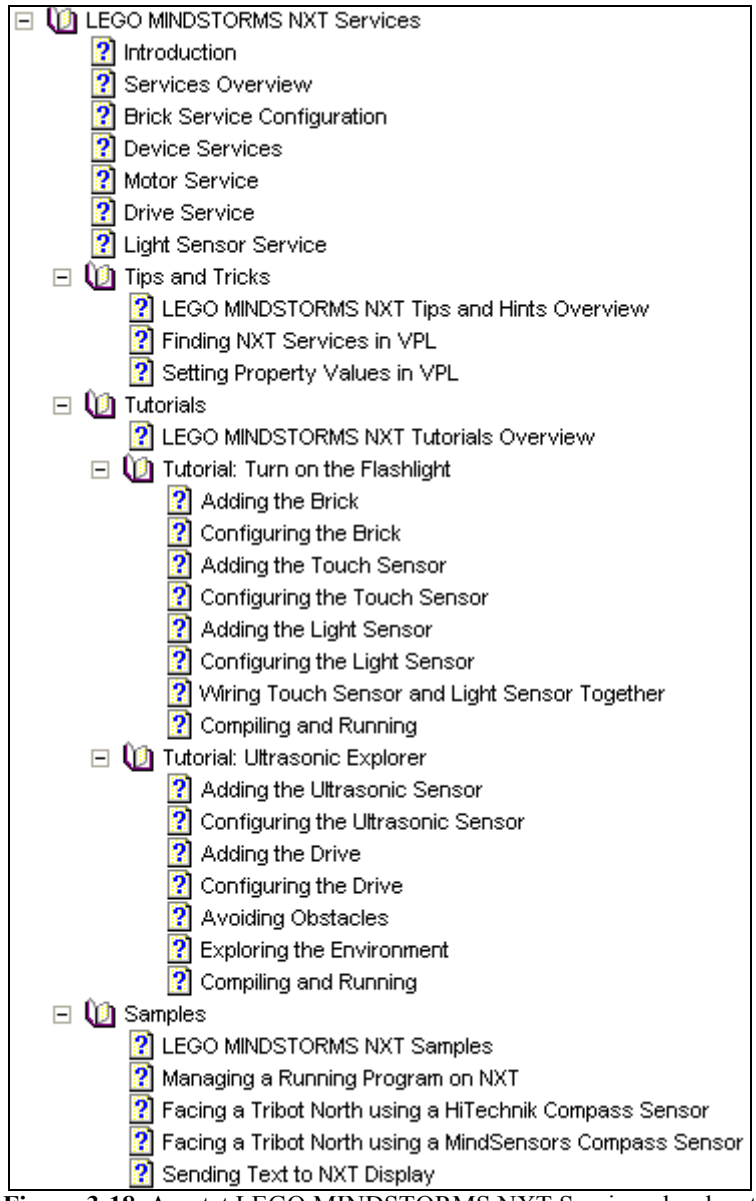

Figura 3-18. Apartat LEGO MINDSTORMS NXT Services desplegat

Aquest és un apartat dedicat exclusivament al kit de desenvolupament de *Lego*, i tal com es pot veure a la Figura 3-18 és prou complert. Inclou molta informació sobre les possibilitats d'utilitzar el kit de *Lego*, diferents elements que es poden fer servir, els paràmetres de configuració de que disposen, etc. Cal destacar que tota la documentació d'aquest apartat és per Visual Programming Language.

Inclou també dos *tutorials* senzills, explicant pas a pas els passos a realitzar així com diversos exemples ja realitzats que es troben a la carpeta on és instal·lat el MRS i es poden provar directament.

3.2 Altre material docent

A Internet es poden trobar multitud de planes web que fan referència a Microsoft Robotics Studio (Figura 3-19).

Figura 3-19. Cerca "Microsoft robotics studio" a Google

Algunes d'elles presenten material docent, tot i això, la gran majoria no han elaborat un material autònom, si no que ofereixen enllaços als *tutorials* de la web de MSDN, els mateixos tutorials que es poden trobar a la documentació que inclou MRS.

En aquest apartat es veurà més detalladament què és el que hi ha a Internet per a aprendre a utilitzar Microsoft Robotics Studio. Tot s'ha trobat a cercant a Google, i tal com es veurà a continuació, el material trobat no sol formar part d'un material docent per a algun curs.

S'analitzarà el material que es trobi, tant articles com petites guies o pàgines web, sempre que es considerin importants. Articles que siguin còpia d'altres ja comentats o poc rellevants no es tindran en compte.

3.2.1 Controlar LEGO NXT amb Wiimote

Aquest és un dels *tutorials* més referenciats i copiats que es troben a la xarxa (sense parlar, evidentment, dels de Microsoft). Realitzat per Alberto Bietti al seu *blog*[16], explica, tal com diu el seu nom, com controlar un robot *LEGO Mindstorms NXT* amb un *Nintendo Wiimote*[17] (Figura 3-20).

[16] http://alandtech.blogspot.com/2007/11/lego-nxt-wiimote-with-MSRS-tutorial.html
[17] *Nintendo Wiimote: controlador de Nintendo per a la consola Wii. Incorpora sensors de moviment.*

Figura 3-20. Nintendo Wiimote

Aquest és un controlador inalàmbric amb sensors de moviment i diversos botons que s'utilitza a la consola *Nintendo Wii*. Aquest tutorial s'ha realitzat per a controlar el robot de LEGO però serveix per a controlar qualsevol robot diferencial que permeti Microsoft Robotics Studio, com el *iRobot Create*. Cal destacar que és un tutorial per MRS 1.5 i en anglès.

La estructura d'aquest tutorial és una mica diferent de les que s'han vist a la documentació inclosa:

- **Crear un servei:** fa una petita introducció al concepte bàsic de servei i explica com crear-ne un (projecte de nou servei DSS)
- **Afegir referències:** descarrega la llibreria de Wiimote (per MRS 1.0 i la converteix i compila a 1.5), afegeix la referència del servei que havia creat anteriorment
- **Subscriure's al Wiimote:** per a rebre les notificacions (acceleració) des del Wiimote és necessari subscriure's. Per a comunicar dos serveis s'ha d'afegir un *Partner Service*, definir els ports (necessaris per a rebre missatges) i subscriure'ls al servei.
- **Enviar peticions als motors:** quan canvia l'estat del Wiimote tan sols s'ha d'enviar la petició al motor mitjançant el *Partner*
- **Fer funcionar el servei:** abans de poder executar s'ha d'especificar quin manifest s'utilitza, és a dir, amb quin robot es vol utilitzar.

Finalment es mostra un vídeo del sistema funcionant i es disposa del codi per descarregar. Aquest projecte està realitzat en C# i excepte la simulació (que es pot executar mitjançant Visual Programming Language o a la consola) i l'assignació de referències tot es fa per codi i a la consola.

És un tutorial que no és introductiu, gran part del mateix és en codi (el qual requereix certs coneixements) i dóna molts conceptes per coneguts, però si es disposa dels elements no és difícil de seguir i realitzar (sempre, és clar, que no sorgeixi algun problema).

3.2.2 Programant amb Microsoft Robotics Studio

El títol original és "*Programando con Microsoft Robotics Studio*" (és un article en castellà), realitzat per Alejandro Martínez (Mèxic) és un dels deu millors articles sobre tecnologia dels *Student Partners*[18] de Mèxic.

En aquest article[19], de tan sols 3 pàgines, es fa una petita introducció a Microsoft Robotics Studio i posteriorment hi ha una guia per a fer una petita aplicació amb un robot *Fischertechnik (*similar a *LEGO Mindstorms*).

Figura 3-21. Robot *fischertechnik*

Aquest robot disposa d'un sensor de llum, llavors quan detecta que s'ha interromput el flux de llum l'aplicació ho rebrà i avisarà l'usuari. Aquest exemple és molt simple, adequat per a iniciació, tot en Visual Programming Language i no suposa cap problema la seva realització. En tot cas, és necessari el robot *fischertechnik.*

Però, no es pot considerar una guia, és tan sols un article, en tot cas podria ser part d'una, ja que explica un exemple molt senzill.

3.2.3 Microsoft Robotics Studio i LEGO Mindstorms NXT

És un altre article[20], escrit per Brian Peek en anglès, relacionat amb *LEGO Mindstorms NXT*. En aquest cas es controla amb un controlador de XBox360 mitjançant Bluetooth[21] i es programa també amb C# i Visual Programming Language.

[18] *Student Partners*: cada any Microsoft premia i reconeix els estudiants més talentosos del món i els fa "student partner"

[19] http://www.aquora.net/studentpartners/docs/Alejando%20Martinez%20-%20Microsoft%20Robotics%20Studio.pdf

[20] http://blogs.msdn.com/coding4fun/archive/2007/07/16/3902344.aspx

[21] Bluetooth: especificació de comunicació inalàmbrica

Aquest article s'ha dissenyat per a fer-lo servir amb el robot *TriBot (*Figura 3-22*)*, que és una construcció que es pot fer amb *LEGO Mindstorms NXT* per la qual incorpora instruccions de muntatge el propi kit. Realment es pot fer amb qualsevol robot amb rodes (o fins i tot un de simulat), però com aquest és un robot molt popular és el que l'autor considera que serà més apropiat.

Figura 3-22. TriBot

Ha afegit, a més, el sensor del botó de prémer al davant, per a que si el robot xoca frontalment, es pari.

- La guia comença explicant com realitzar la connexió Bluetooth. Primer la connexió amb el pc, simple, i després una línia de comanda que, des de la consola, connecti amb el NXT.

- A continuació inclou tot el codi tant en C# com en Visual Basic. Ho va fent per parts, explicant cada fragment de codi en els dos llenguatges

- En el següent punt, l'execució, explica com s'ha de fer per executar.

- L'últim punt tracta de fer el mateix que en els punts anteriors però des de Visual Programming Language tot (Figura 3-23), proporcionant a l'usuari la imatge del circuit final i explicant com s'han de realitzar les connexions

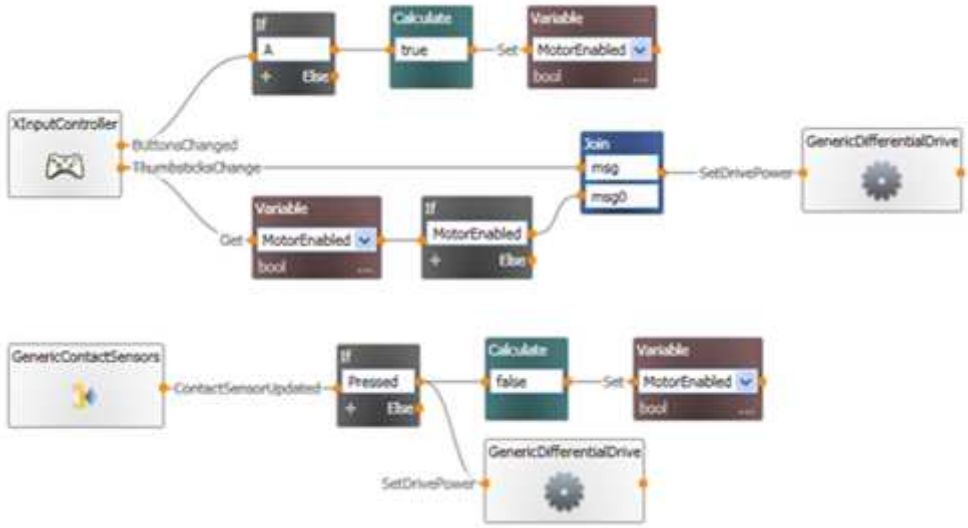

Figura 3-23. El mateix programa en VPL

Aquest *tutorial* té la particularitat que realitza el mateix en diversos llenguatges, entre ells VPL, el que és molt interessant. Encara que només sigui un article, podria fer-se servir, en part, en un tutorial més extens ja que és adequat per a un usuari expert amb la codificació C# o Visual Basic però també per a un usuari més novell amb la programació en Visual Programming Language.

3.2.4 Blog JSantillan

És un *blog*[22], escrit per Jefferson Felipe Santillán Paredes, en el qual el seu autor ha escrit diversos articles sobre Microsoft Robotics Studio[23] (entre d'altres aplicacions).

Si cerquem articles sobre aquest programa trobem 3, els quals tenen una idea molt introductòria:

- VPL Introducció: petita introducció a la aplicació, similar a la pròpia de la documentació de Microsoft
- VPL - Accés a un servei: explica de forma molt senzilla i amb l'ajuda d'un robot amb sensors com accedir a un servei (en aquest cas, el sensor) i mostrar un avís
- VPL - Crear bucles: com crear un bucle (Figura 3-24), pràcticament igual que als tutorials introductius de la documentació inclosa

[22] Blog: pàgina web que periòdicament va recollint articles del seu autor
[23] http://es.wordpress.com/tag/microsoft-robotics-studio/

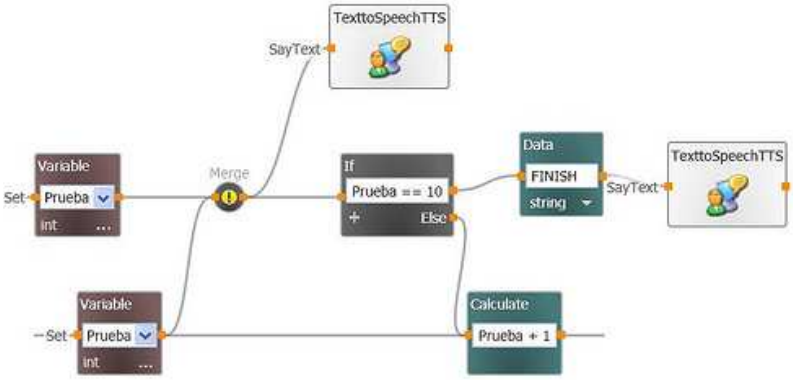

Figura 3-24. Bucle en VPL

Són tan sols articles escrits per un usuari, no molt ben explicats (certa falta d'ordre) i poc suport d'imatges (especialment l'accés a un servei), però interessants i molt introductius.

3.2.5 Conscious-Robots

Conscious-Robots[24] és una plana web (feta per espanyols però amb molt contingut en anglès) dedicada a la robòtica i especialment a la intel·ligència artificial (consciència dels robots) i amb una secció dedicada a Microsoft Robotics Studio.

Segons explica la pàgina, han escollit Microsoft Robotics Studio com la millor plataforma per a desenvolupar nous sistemes de control de robots autònoms, i per això la web disposa d'informació, serveis, exemples i recursos per la programació de robots *Pioneer 3 DX* (la majoria de software i informació de que disposa està orientada a aquest robot exclusivament) utilitzant MRS.

Figura 3-25. Pioneer 3-DX

[24] www.conscious-robots.com

Conscious-Robots està orientada tant a estudiants com a científics. Disposa d'una secció de **descàrregues** (el mateix contingut que a la secció **serveis**) en la que es poden trobar diversos projectes ja fets, els quals es poden descarregar (el projecte MRS amb codi font) i amb una explicació del que fan.

No hi ha gaires, però són projectes interessants:

- Sonar simulat
- Simular el servei para-xocs del *Pioneer 3DX*
- Simular un laberint
- Servei de simular un sonar
- Aplicació de simular un sonar (agrupa alguns projectes)
- Panell de control
- Logs concurrents
- Servei sonar per a ARCOS

Tot són projectes realitzats pels administradors de la web, pel que es poden considerar autònoms (encara que no tot està realitzat per ells: el laberint està basat en altre projecte). Inclou una secció de recursos que té documentació o programes, en aquest cas tan sols per *Aibo* i *i-Cybie*.

Figura 3-26. Robot i-Cybie

Un altre apartat és el de **notícies**. Tan sols són les noticies relacionades amb l'aplicació

Disposa, a més, d'un **fòrum**. És també en anglès, encara que té una petita secció en castellà. La majoria d'apartats són dedicats a la intel·ligència artificial i a la consciència, i un apartat és dedicat a Microsoft Robotics Studio. En castellà hi ha un apartat dedicat a consciència i altre a MRS. Curiosament el fòrum espanyol té més activitat que no pas el fòrum en anglès (referint-nos a la secció MRS).

Un altre apartat interessant per a la comunitat castellanoparlant és el de la **documentació en castellà**. En aquesta secció han traduït al castellà una petita part de la informació que inclou Microsoft Robotics Studio. Hi ha un total de 6 articles que poden servir per a representar una introducció al programa sense haver de llegir en anglès. Són traduccions no oficials realitzades pels administradors de la web.

Finalment s'analitzarà la secció **"*How to*".** Aquesta secció conté tant alguns articles per aprendre a realitzar certes accions com introduccions més generals i novells. Cal destacar que alguns dels articles tan sols són l'agrupació dels enllaços als documents de MSDN. Els demés articles són accions molt concretes com crear un "*assembly"* que contingui un contracte DSS genèric, veure els contractes d'un "*assembly*" o crear un "*PortSet"* amb més de 20 ports.

Com es pot veure, són articles molt específics, per a realitzar tasques molt concretes. *Conscious-Robots* és, en conclusió, una plana web molt interessant, que presenta una comunitat relacionada amb la robòtica en general i MRS en concret i pot servir com a introducció a algú que vulgui saber una mica de l'aplicació, o en alguna ocasió requereixi ajuda d'altres usuaris, especialment si en busca en castellà.

Però, cal esmentar que és una plana web poc ordenada i el material que ofereix no és molt adient per a emprar en docència, ja que o és molt específic o massa introductori, a més que no aporta material docent autònom si no que fa referència al material que es troba a la web de MSDN en els apartats que defineix per a aprendre. En definitiva, no és millor que el que ens ofereix Microsoft, però és la millor alternativa a nivell de comunitat, i a més en castellà.

3.2.6 MSRS Code Page

Aquesta és una altre plana web[25] dedicada a Microsoft Robotics Studio. L'autor és Trevor Taylor, consultor independent en Educació Robòtica en Austràlia, qui el mes de Juny ha publicat un llibre en el que és coautor sobre desenvolupar per Microsoft Robotics Studio, titulat "*Professional Microsoft Robotics Developer Studio*", el qual es considera una de les primeres guies en l'aplicació.

La plana és senzilla, sense cap tipus d'interfície més enllà que uns pocs enllaços al diferent software que ha desenvolupat i que anuncia, es troba sempre sota desenvolupament.

També fa referència a la web del seu llibre, la qual disposa de codi d'exemple.

El que ell categoritza com a "software" és, en realitat, una sèrie de parts de la pàgina on, no sols disposa del codi font de les aplicacions que ha realitzat, si no que explica amb prou detall com s'instal·len, s'executen i funcionen.

Tot el material que té fa referència al seu *Maze Simulator* (simulador de laberint), en el qual, per cert, es basa el projecte del simulador de laberint de *Conscious-Robots*, i el qual, a la vegada, està basat en altre projecte, de *Ben Axel,* però millorant-lo.

[25] http://www.soft-tech.com.au/MSRS/

Aquesta aplicació permet convertir imatges (*bitmaps*) en entorns de simulació on cada píxel és un cub:

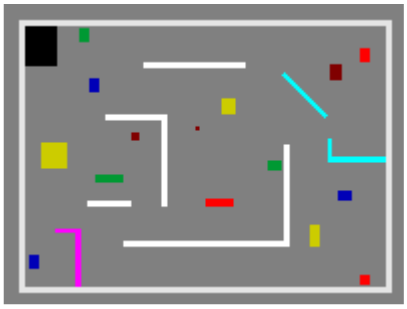

Figura 3-27. Imatge original

El *Maze Simulator* crea un món de blocs simples en el simulador de Microsoft Robotics Studio utilitzant la imatge (Figura 3-1) com a plantilla. Aquesta plantilla defineix tant mida com color dels objectes.

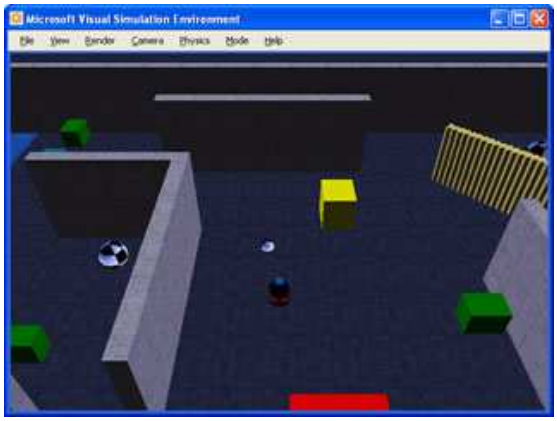

Figura 3-28. Laberint generat

A més d'això ha realitzat una modificació del "*Dashboard*" que es mostra així:

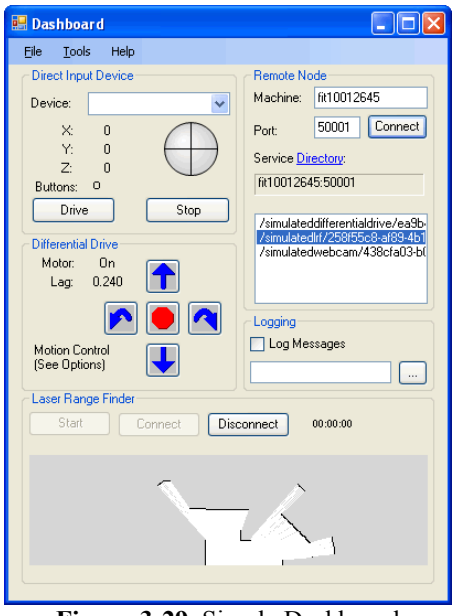

Figura 3-29. Simple Dashboard

En aquesta modificació, d'una banda, permet guardar paràmetres per a properes execucions (millorant la interfície original), i integra control del robot, suport per a una vista del làser, possibilitat d'obrir una altre finestra amb la visió de la càmera integrada al robot (Figura 3-30), etc...

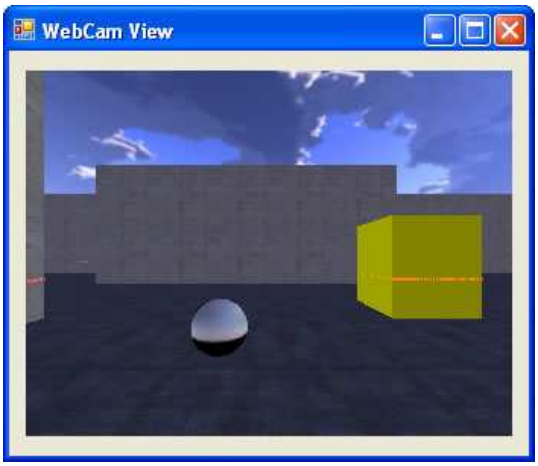

Figura 3-30. Imatge de la càmera integrada

Diferencial simulat (*Simulated Differential Drive*): quan ell ho va realitzar la versió existent era la 1.0, la qual no disposava de gaires opcions. Ell ha implementat altres funcions per aquesta versió mentre es desenvolupava la versió 1.5, les quals posteriorment Microsoft ha inclòs a la versió 1.5. El seu diferencial és el que utilitza a les seves aplicacions, malgrat que ja incorpora aquestes millores l'aplicació, però tenen menor nombre d'errors, segons la seva explicació.

Partint de la base del que ja s'ha esmentat que ha desenvolupat l'autor: *Maze Simulator*, *Dashboard* i Diferencial Simulat, ha elaborat dos articles: Introducció a robòtica autònoma i programa *ExplorerSim*.

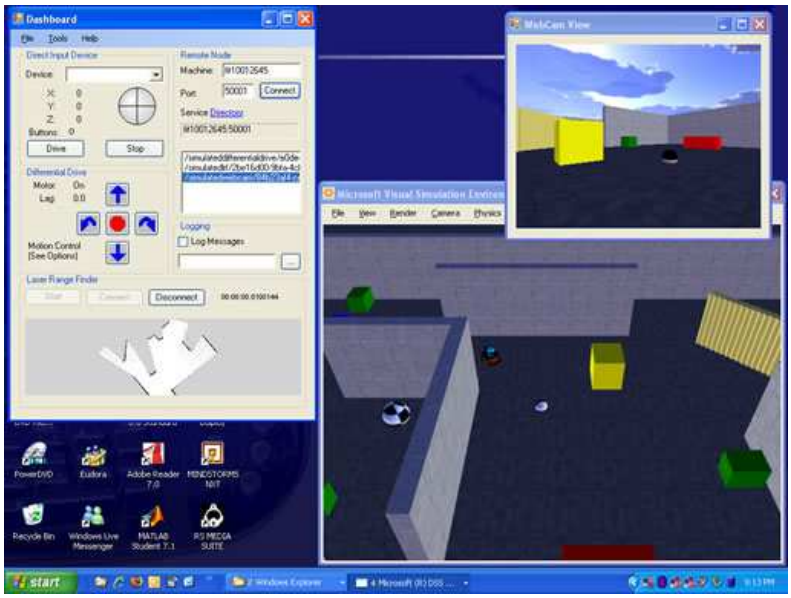

Figura 3-31. Entorn de simulació

En el primer, utilitzant el software que ha desenvolupat l'autor explica com muntar un robot que funcioni de forma autònoma, molt senzill i amb alguns errors, però per sí sol i totalment simulat, un aspecte molt interessant, perquè permet a l'usuari aprendre i "jugar". El programa el que fa és controlar un robot que es mogui pel laberint i pugui ser controlat amb el *Dashboard*. L'article consta de documentació per a fer-ho funcionar i un *tutorial* explicatiu.

I finalment està *l'ExplorerSim*. És una modificació de l'explorador de mostra de Microsoft. En aquesta aplicació es mostra com fer un robot que vagi explorant i construint un mapa gràcies al làser detector de rang de que disposa. De nou, tot és simulat i no està provat en robots reals.

Per a fer-ho servir també utilitza els projectes que ha creat prèviament. A mida que es va movent va dibuixant el mapa del que "veu" amb el làser:

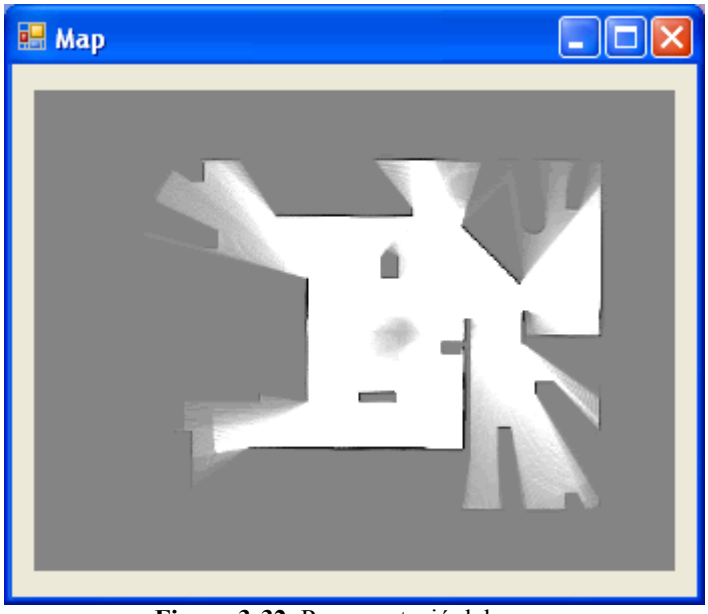

Figura 3-32. Representació del mapa

El treball realitzat a aquesta plana és molt interessant, sobre tot perquè és totalment simulat, la qual cosa incrementa les possibilitats. És, potser, un material massa autònom com per a incloure'l en un material docent introductiu, a més de requerir un cert nivell de programació, però es pot utilitzar a nivells més alts, ja que són aplicacions que no requereixen robots reals i molt interessants.

3.2.7 Llibres

Realment existeixen pocs llibres sobre Microsoft Robotics Studio, tan sols dos dedicats exclusivament. Es farà una petita ressenya del que inclou el llibre i es comentarà el que opinen els usuaris. Microsoft Robotics Studio és una aplicació amb no gaire temps al mercat (un parell d'anys) i no hi ha massa material sobre ella, de fet els llibres que s'analitzaran a continuació són de recent publicació (durant l'any 2008):

Sara Morgan, *Programming Microsoft Robotics Studio*, Microsoft Press, 304 pàgines

Aquest llibre va ser publicat el 15 de Març. L'autora és una MVP[26] de Microsoft a l'àrea de Servidor de Comunicacions.

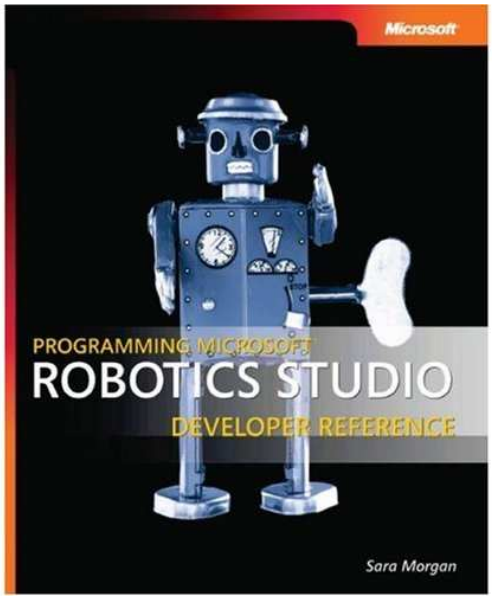

Figura 3-33. Programming Microsoft Robotics Studio

Segons indica la seva descripció és un llibre que servirà com a guia per a utilitzar serveis relacionats amb programar robots, donarà referències a simulacions, navegació i control remot, així com a vídeo i parla, i donarà codi de mostra en Visual Programming Language i C#.

Però, segons opinions d'usuaris, és un llibre molt superficial i fa molta referència als tutorials que ofereix Microsoft, sense aportar nova informació. A més té una manca d'imatges i esquemes que no ajuden a la seva comprensió. D'altra banda és un llibre que és fàcil de llegir i està ben estructurat.

Kyle Jones, Trevor Taylor, *Professional Microsoft Robotics Studio*, Wrox, 864 pàgines

Aquest llibre, publicat el 19 de Maig d'aquest mateix any, és un llibre molt complert, co-escrit pel fundador arquitecte de MRS, és un llibre ple d'exemples que amb aquesta guia pretén fer que els lectors escriguin aplicacions utilitzant tres tipus de robots bàsics: un robot de seguretat interna, un robot competidor i un braç jugador d'escacs.

[26] Microsoft MVP: Premi que atorga Microsoft als seus líders més actius

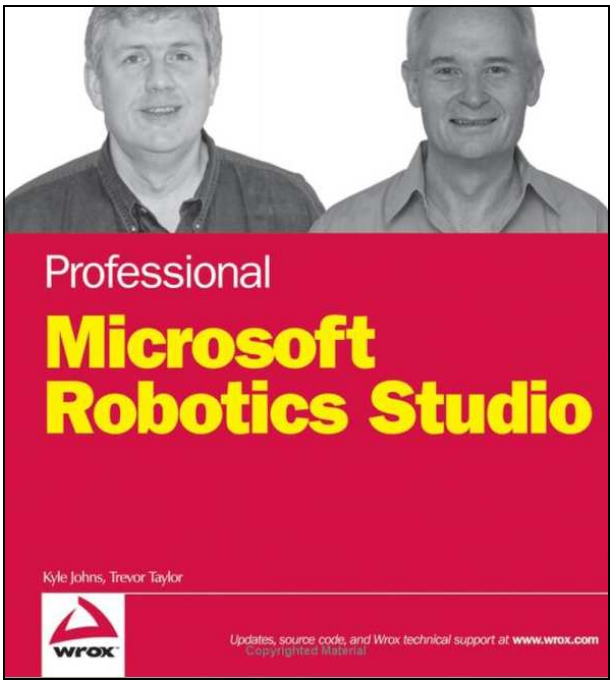

Figura 3-34. Professional Microsoft Robotics Studio

El llibre està compost per quatre blocs:

- Fonaments bàsics de Microsoft Robotics Studio
- Simulacions
- Visual Programming Language
- Hardware robòtic

No cal dir que el llibre ha estat publicat en anglès. Les referències al llibre són bones, indicant que té molts exemples i és un bon llibre de referència per a començar a utilitzar l'aplicació, a part de fàcil d'entendre i de seguir.

A més disposa de pàgina web[27] on es pot descarregar codi de mostra i hi ha informació.

És un llibre indicat per a millorar la documentació de Microsoft.

Nota: es pot trobar el llibre també amb el nom: ***Professional Microsoft Robotics Developer Studio***.

[27] http://www.promrds.com/

3.3 Mancances i necessitats del material educatiu

Arribats a aquest punt es poden treure algunes conclusions: el material que inclou MRS conté informació explicativa i a més algunes guies d'usuari per ajudar-lo a introduir-se en l'aplicació. És un material que resulta útil i, si es cerca a Internet, és el que té més pes.

Disposa de prou informació com per introduir l'usuari en tots els camps de l'aplicació, però, resulta una guia amb certa falta d'organització i detall. Microsoft té una forma de crear documentació, en un primer pas són els propis desenvolupadors els que ho realitzen, gent que no es dedica a això però que tenen els coneixements. Posteriorment, un cop ha passat un temps un equip de documentació la refà: revisa i organitza tota la informació de forma que el lector en pugui treure el màxim profit possible. En el cas de Microsoft Robotics Studio no ha passat això encara (es desconeix si passarà aviat o si tan sols passarà), i la documentació que inclou l'aplicació és la que els desenvolupadors del programa han realitzat.

Això ens dóna una bona informació, assegurant que podrem trobar uns nivells alts de detall i especificació així com de coneixements impresos en aquesta documentació però no tenen per què estar ben escrits. Probablement ho han fet el millor que han pogut, però no és un equip acostumat a aquesta tasca, amb això no es vol dir que sigui una documentació dolenta, és prou bona, millor que molta altre documentació sobre altres programes, i potser és més que suficient per molta gent, però és millorable.

Penso que en alguns punts li falta una mica d'organització i detall per a començar a utilitzar Microsoft Robotics Studio, així com una guia d'introducció més extensa i complerta, i a més, totalment simulada, ja que el hardware de robòtica no és molt extens, i si ens estem dirigint a un públic novell, o en uns cursos de robòtica inicials, que el que vol fer l'usuari és iniciar-se en l'aplicació, el més normal és que no disposi d'aquest hardware de forma inicial. La majoria de guies que inclou que fan servir robots reals ho fan perquè no es pot utilitzar simulats ja que interaccionen amb una càmera real o amb un micròfon i no presenten una alternativa simulable on es pugui utilitzar l'aplicació creada.

A més, trobo que no s'explota molt el Visual Programming Language, el qual és la gran novetat que presenta aquest software, i resulta molt atractiu pels usuaris gràcies a la seva interfície visual, ja que els convida a "jugar" amb l'aplicació i pot resultar dinàmic i estimulant.

Penso també que hi ha molt poc detall sobre els serveis, no el concepte de servei, si no els que hi ha disponibles i com utilitzar-los, de fet, no hi ha una documentació sobre això, el qual fa difícil utilitzar-los i són la base en la comunicació amb els robots.

Com ja s'ha esmentat anteriorment, existeix molta informació sobre Microsoft Robotics Studio a Internet. De fet, realitzant cerques a Internet, podem trobar moltes referències, les quals, en un alt percentatge inclouen seccions de tutorials, guies, "how to", etc.

Molts d'ells no tenen material autònom, si no que realment són enllaços a la pròpia documentació allotjada a la web *Microsoft Developer Network (*MSDN*).*

A la secció 3.2 s'ha analitzat part del material existent a Internet, s'ha analitzat precisament el material més autònom, el que no era el mateix que el de Microsoft. Realment, el material docent que existeix, el que ajuda a l'usuari a aprendre és molt escàs. No existeixen millors alternatives al que ens proposa Microsoft. La major part del que es pot trobar són projectes concrets, els quals ha realitat algú i ha compartit com ho ha fet, però en cap cas han intentat ser manuals de referència en l'aprenentatge de l'aplicació.

En alguns casos aquestes guies utilitzen robots o altres elements hardware. Per a algunes funcions concretes es podria simular el robot (per exemple controlar les rodes d'un robot és fàcilment simulable), havent d'adaptar el codi prèviament. De tota manera precisament algunes guies el que pretenen és demostrar que es pot interactuar entre uns controladors y un ordinador i bluetooth o wifi i un robot, per exemple amb sensors que amb la ma es puguin activar per a veure resultats, cosa difícil simulant.

No es pretén dir que a Internet no hi ha informació, això no és cert, n'existeix, i molta, hi ha més comunitats a part de la de MSDN, però, més minoritàries i menys completes. El que es pretén dir és que, excepte als llibres, els quals, per cert, són escassos i, de fet, molt nous (segon trimestre d'aquest any 2008), no hi ha una documentació que pugui competir amb la de Microsoft, la qual és millorable, al menys, en uns objectius introductoris.

3.4 Proposta de guia pas a pas

La guia que es realitzarà intentarà que l'usuari assoleixi uns coneixements mínims en Microsoft Robotics Studio*,* especialment en l'eina Visual Programming Language i, un cop havent-la realitzat, sigui capaç de desenvolupar aplicacions senzilles sense cap més ajuda. Com es diu al principi del Capítol 1, es realitzarà amb Microsoft Robotics Studio 1.5, ja que la versió de Juliol no és una versió final, a part que la versió 1.5 està molt més integrada a la comunitat robòtica, i gairebé tota la informació existent és per aquesta versió o l'anterior 1.0, donat que la versió de Juliol és molt nova.

Es pretén que sigui una guia similar a les propostes per Microsoft a la seva documentació però més organitzada i més fàcil de seguir, a part de preveure els problemes que pugui tenir l'usuari en la seva realització. Un cop decidit en termes generals què serà la guia, el primer que s'ha de fer és analitzar on es podrà utilitzar i a qui estarà dedicada.

La proposta de guia que es realitzarà s'haurà de poder fer servir com a material educatiu en, per exemple, una primera sessió de pràctiques de robòtica pensades per a Microsoft Robotics Studio*,* per tant, no podrà aprofundir massa. L'usuari podria ser tant un estudiant que està realitzant el curs com un aficionat a la robòtica i a la programació, però, amb coneixements mínims de programació, i un coneixement general sobre robòtica, el qual es completarà amb la realització de la guia.

Haurà de ser un document no gaire llarg ni pesat, ja que s'ha de poder realitzar en un parell d'hores. Estarà format per dues parts: una primera part serà una breu documentació que formarà l'usuari en el que farà i en l'aplicació que es desenvoluparà, i una segona part que serà una guia pas a pas de l'aplicació. Tot estarà acompanyat per il·lustracions que facilitaran el seguiment i permetran a l'usuari saber en tot moment on es troba i què està realitzant en aquell moment.

Serà una guia molt estructurada i poc carregada, ja que l'objectiu no és realitzar una tasca molt complexa i que l'usuari vagi tan sols seguint una sèrie de punts d'una guia, si no que es pretén que l'usuari assoleixi uns coneixements a mida que va completant l'exercici i que li serveixin per, més endavant, poder realitzar treballs ell tot sol.

El document es presentarà en format electrònic per a facilitar la seva distribució i adjunt en aquesta memòria.

Referències

[1] Microsoft Corporation, "Microsoft Robotics Studio Developer Center", Microsoft Developer Network, 1 Agost 2008 , http://msdn.microsoft.com/es-es/robotics/default(en-us).aspx, (accés 9 Agost 2008)

[2] Amazon.com, Inc. or its affiliates, "Professional Microsoft Robotics Developer Studio (Wrox Programmer to Programmer) (Paperback)", Amazon.com/Books, 7 Juliol 2008, http://www.amazon.com/Professional-Microsoft-Robotics-Developer-Programmer/dp/0470141077/ref=pd_bxgy_b_img_b, (accés 18 Agost 2008)

[3] Amazon.com, Inc. or its affiliates, "Programming Microsoft® Robotics Studio (Paperback)", Amazon.com/Books, 18 Juny 2008, http://www.amazon.com/Programming-Microsoft%C2%AE-Robotics-Studio-Morgan/dp/0735624321, (accés 19 Agost 2008)

[4] Sara Morgan, "An introduction to Programming Robots with Microsoft Robotics Studio", DevX.com, 9 Octobre 2006, http://www.devx.com/dotnet/Article/32729, (accés 12 Agost 2008)

[5] Google, "Fòrums de debat Microsoft Robotics Studio", Google Grupos, 1 Agost 2008, http://groups.google.com/group/microsoft-robotics-studio/topics, (accés 12 Agost 2008)

[6] Mohammed Hossam, "Microsoft Robotics Studio August CTP is out", Bashmohandes, 8 Agost 2008, http://spellcoder.com/blogs/bashmohandes/archive/2006/08/05/257.aspx, (accés 5 agost 2006)

[7] George Gardner, "Bill Gates, Robots & Microsoft Robotics Studio", TECH.BLORGE, 7 Gener 2007, http://tech.blorge.com/Structure:%20/2007/01/07/bill-gates-robots-microsoft-robotics-studio/, (accés 8 Agost 2008)

[8] Brian Peek, "Microsoft Robotics Studio and Lego Mindstorms NXT", Coding4Fun, 16 Juliol 2007, http://blogs.msdn.com/coding4fun/archive/2007/07/16/3902344.aspx, (accés 14 Agost 2008)

[9] Conscious-Robots, "Microsoft Robotics Studio", Conscious-Robots, 25 Abril 2008, http://www.conscious-robots.com/es/robotics-studio/, (accés 12 Agost 2008)
[10] Orlando Hernández, "Microsoft Robotics Studio", Cyber-Trends, 25 Setembre 2007, http://cyber-trends.blogspot.com/2007/09/microsoft-robotics-studio.html, (accés 17 Agost 2008)
[11] Edilson García Chiñas, "Microsoft Robotics Studio Simulation", DelfinesNETCell, 9 Juliol 2007, http://delfinesnetcells.spaces.live.com/Blog/cns!A47E67F9F2526003!909.entry?wa=wsignin1.0, (accés 14 Agost 2008)
[12] Wrox, "Professional Microsoft Robotics Developer Studio", Wrox: Programmer to programmer, Maig 2008, http://www.wrox.com/WileyCDA/WroxTitle/Professional-Microsoft-Robotics-Developer-Studio.productCd-0470141077,descCd-DOWNLOAD.html , (accés 14 Agost 2008)
[13] Microsoft Corporation, "MSDN Forums » Microsoft Robotics » Microsoft Robotics – Community", MSDN, 12 Agost 2008, http://forums.microsoft.com/msdn/ShowForum.aspx?ForumID=1423&SiteID=1 , (accés 14 Agost 2008)
[14] Jefferson Felipe Santillán, "Introducción a Microsoft Robotics Studio", The Code, 9 Juny 2008, http://jsantillan.wordpress.com/2008/06/09/introduccion-a-microsoft-robotics-studio/, (accés 18 Agost 2008)
[15] Ken Berry, "Microsoft Enters Robotics – An educator perspective", Robot, 2008, http://www.botmag.com/issue3/microsoft_technical_preview_3.shtml, (accés 15 Agost 2008)
[16] Edilson García Chiñas, "Microsoft Robotics Studio (1.5) CTP Sep 2007", Tecnologias Microsoft, 1 Setembre 2007, http://mredison.wordpress.com/2007/09/01/microsoft-robotics-studio-15-ctp-sep-2007/, (accés 15 Agost 2008)
[17] Ankur Mittal, "Microsoft Robotics Studio (1.5) Refresh", MS Windows Vista Compatible Software, 27 Desembre 2007, http://www.windowsvistaplace.com/microsoft-robotics-studio-15-refresh/microsoft, (accés 18 Agost 2008)
[18] Ken Cheung, "Microsoft Robotics Studio October Community Tech Preview (CTP)", Eda Blog, Octubre 2006, http://edablog.com/2006/10/08/microsoft-robotics-studio-october-community-tech-preview-ctp/, (accés 13 Agost 2008)
[19] Natural News, "Microsoft Robotics Studio Provides Common Ground for Robotics Innovation (press release)", Robotics, 22 Juny 2006, http://www.naturalnews.com/019454.html, (accés 11 Agost 2008)
[20] Jorge Saavedra, "Microsoft Robotics Studio v1.5 liberado", El mundo informático, 18 Juliol 2007, http://jorgesaavedra.wordpress.com/2007/07/18/microsoft-robotics-studio-v15-liberado/, (accés 18 Agost 2008)
[21] Trevor Taylor, "MSRS Code Page", MSRS Code Page, 4 Maig 2008, http://www.soft-tech.com.au/MSRS/, (accés 20 Agost 2008)
[22] Kyle Johns i Trevor Taylor, "Professional Microsoft Robotics Developer Studio", 12 Agost 2008, http://electronicdesign.com/Articles/ArticleID/19568/19568.html, (accés 21 agost 2008)

[23] Kyle Johns i Trevor Taylor, "Professional Microsoft Robotics Developer Studio", Maig 2008, http://www.wrox.com/WileyCDA/WroxTitle/Professional-Microsoft-Robotics-Developer-Studio.productCd-0470141077.html , (accés 21 agost 2008)

[24] Alejandro Martínez, "Programando con Microsoft Robotics Studio", Scribd, 30 Juliol 2007, http://www.aquora.net/studentpartners/docs/Alejando%20Martinez%20-%20Microsoft%20Robotics%20Studio.pdf (accés 15 Agost 2008)

[25] RoboRealm, "RoboRealm setup for Microsoft Robotics Studio (MSRS)", RoboRealm, desconegut, http://www.roborealm.com/help/MSRS.php, (accés 12 Agost 2008)

[26] Ben Axel, "Robotics Studio Maze Simulator", Channel 9 (MSDN) 24 Juny 2007, http://channel9.msdn.com/playground/Sandbox/220836-Robotics-Studio-Maze-Simulator/ ,(accés 18 Agost 2008)

[27] Matt Mondok, "Roll your own Robot Jones with Microsoft's robotics kit", Ars Technika, 13 Desembre 2006, http://arstechnica.com/journals/microsoft.ars/2006/12/13/6280, (accés 16 Agost 2008)

[28] USC, "Introduction::iRobot Create/Roomba and Microsoft Robotics Studio", USC, Desconegut, http://roboticsprimer.sourceforge.net/workbook/Introduction::iRobot_Create/Roomba_and_Microsoft_Robotics_Studio#Running_the_Simulator , (accés 13 Agost 2008)

[29] Steven Smith, "Microsoft Robotics Studio Released", Steven Smith Blog, 13 Desembre 2006, http://aspadvice.com/blogs/ssmith/archive/2006/12/13/Microsoft-Robotics-Studio-Released.aspx , (accés 13 Agost 2008)

[30] Harplogic, "Surveyor SRV-1 Setup for Microsoft Robotics Studio (MSRS)", 11 Juny 2008, http://www.surveyor.com/MSRS.html ,(accés 13 Agost 2008)

[31] Alberto Bietti, "Lego NXT + wiimote with MSRS tutorial", Alberto Bietti's projects blog, 17 Novembre 2007, http://alandtech.blogspot.com/2007/11/lego-nxt-wiimote-with-msrs-tutorial.html, (accés 18 Agost 2008)

4 Realització de la guia pas a pas

En aquest capítol s'ha realitzat la guia docent de Microsoft Robotics Studio. Aquesta guia s'ha elaborat paral·lelament al capítol i es pot trobar als annexes de la memòria en diversos formats.

Primerament es descriurà breument l'estructura en que està escrita i dividida la guia. Posteriorment s'explicarà la presentació de la guia, que és en format escrit i digital accessible via web.

Finalment es contemplaran alternatives i vies de continuació del material docent exposat.

4.1 Guia realitzada

Primer de tot cal destacar que la guia s'ha realitzat **íntegrament en anglès**. S'ha decidit fer així, encara que suposés un cost d'elaboració una mica més elevat, perquè és el llenguatge més comú i en tecnologia és el que més gent utilitza, i això facilita i promou la distribució de la guia realitzada, de forma que arribarà a més gent arreu del món.

La guia que s'ha desenvolupat inclou dos apartats ben diferenciats:

- Un primer apartat d'introducció que conté una **introducció al sistema** per a que l'usuari sàpiga els termes bàsics de Visual Programming Language
- Una guia d'autoaprenentatge en forma de **pràctica guiada**

La primera part és una introducció al sistema, amb tota una sèrie de descripcions i de imatges per a permetre a l'usuari entendre el sistema amb el que treballarà així com una descripció de l'arquitectura de Microsoft Robotics Studio. Tot i que és prou detallat s'ha intentat no donar massa informació a l'usuari, ja que es pretén que serveixi com a guia introductòria tan sols, amb els objectius de que l'usuari acabi sabent utilitzar l'aplicació i entenent com funciona.

La guia d'autoaprenentatge és una guia que es pot realitzar en poques hores i en la que s'introdueix a l'usuari en la utilització de l'aplicació creant diversos programes senzills però en els quals es realitzen diverses tasques. Conté dues parts principals i es realitza tota sota Visual Programming Language sense cap necessitat d'un robot real, utilitzant l'entorn de simulació.

4.1.1 Programació bàsica

En aquesta part es faran servir les activitats bàsiques que permet utilitzar l'aplicació com *If, Merge, Switch,* etc. per a desenvolupar aplicacions simples similars a la programació convencional. Són la base en la programació de qualsevol robot ja que són la unitat més petita amb la que es pot comptar. És tan sols una pràctica, la qual conté diversos casos.

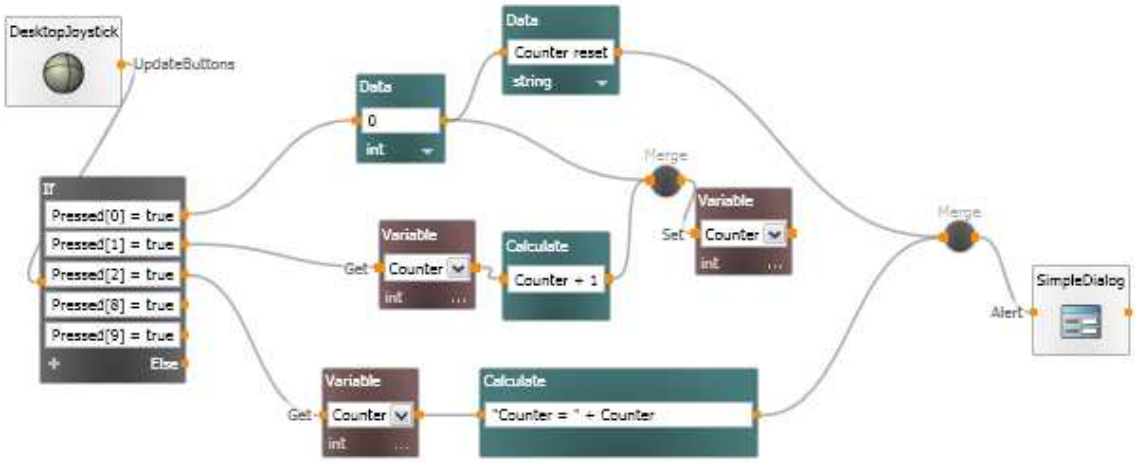

Figura 4-1. Part de la pràctica de programació bàsica

Això es una part de l'aplicació realitzada, la qual és un controlador a l'escriptori que disposa de diversos botons, dels quals alguns tenen un comportament associat a la seva pulsació.

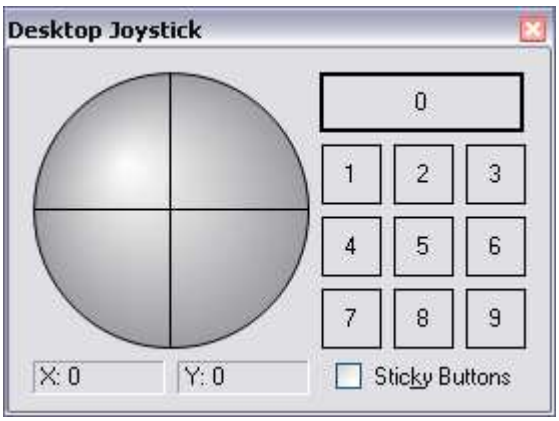

Figura 4-2. Controlador d'escriptori

En aquesta pràctica es fa una impressió per pantalla, una sortida per altaveus (Microsoft Robotics Studio disposa d'un servei de parla des de text) i s'interactua amb una variable sencera incrementant-la i mostrant el seu valor per pantalla.

4.1.2 Programació d'un robot:

Aquí es programarà i simularà un robot, de fet, tres robots. Es realitzaran tres pràctiques independents en les quals a cada una d'elles es farà que el robot (s'utilitzaran dos models diferents) faci alguna cosa (gairebé tot relacionat amb el moviment), ja que es vol que l'usuari vegi diverses possibilitats del programa sense massa nivell de dificultat ni entrant massa en detall. Les pràctiques són:

- Controlar i simular el robot
- Moure el robot automàticament
- Interactuar amb sensors i actuadors del robot

A la **primera** pràctica s'utilitza, de la mateixa manera que a la pràctica de programació bàsica, un controlador d'escriptori. En aquest cas l'escollit és més simple, ja que només s'empraran tres botons per: avançar, retrocedir i parar. S'assignarà a cada un dels botons la funció escollida i es simularà el robot, així com es farà una petita explicació de l'entorn de simulació i com moure's per ell.

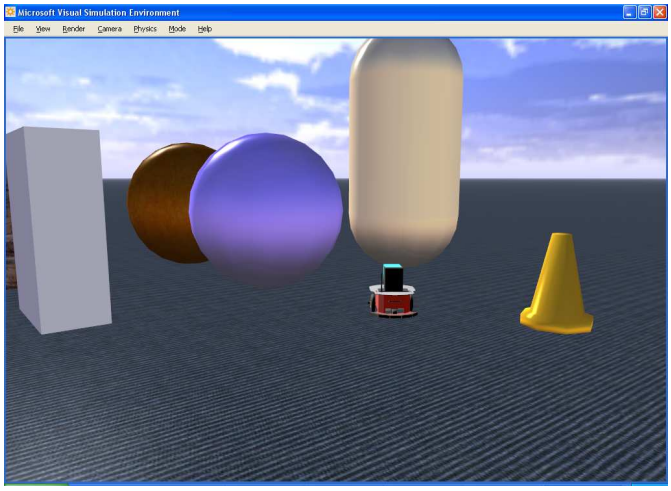

Figura 4-3. Entorn de simulació

A la **segona** pràctica s'introduirà el temporitzador. En aquest cas es programaran diversos moviments: avançar, retrocedir i avançar girant.

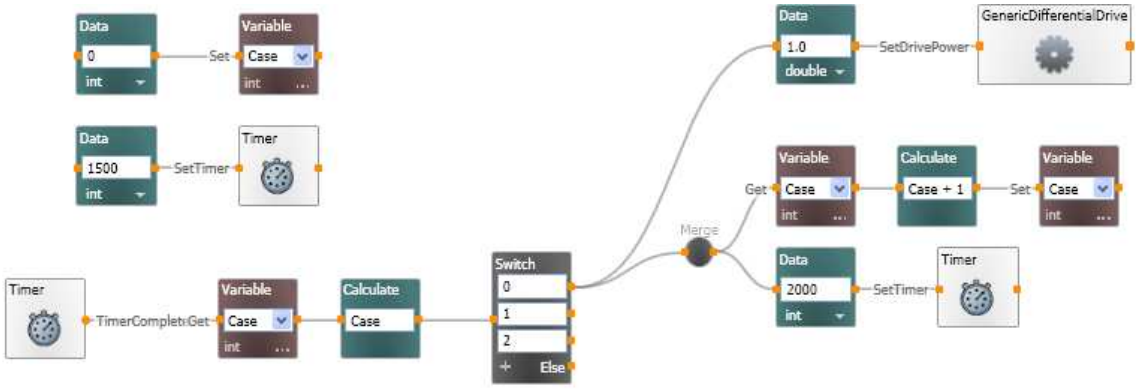

Figura 4-4. Part del diagrama

Aquests moviments aniran realitzant-se un darrere l'altre cada cop que es consumeixi el temps definit, fent que el robot es mogui per l'entorn de forma autònoma, podent xocar amb els objectes que hi trobi i llençar-los al terra, així com fins i tot tombar-se al xocar, veient com funciona la física a l'entorn de simulació.

A la **tercera** i última pràctica s'ha desenvolupat un sistema senzill en el que es detecten parets i objectes. Aquí s'utilitza altre robot, el qual disposa de sensors. Aquest robot avançarà cap endavant sempre que pugui .Quan els sensors detecten alguna cosa s'actualitzen, llavors, el nostre sistema cada cop que detecti una actualització farà el robot girar i avançar en una altre direcció.

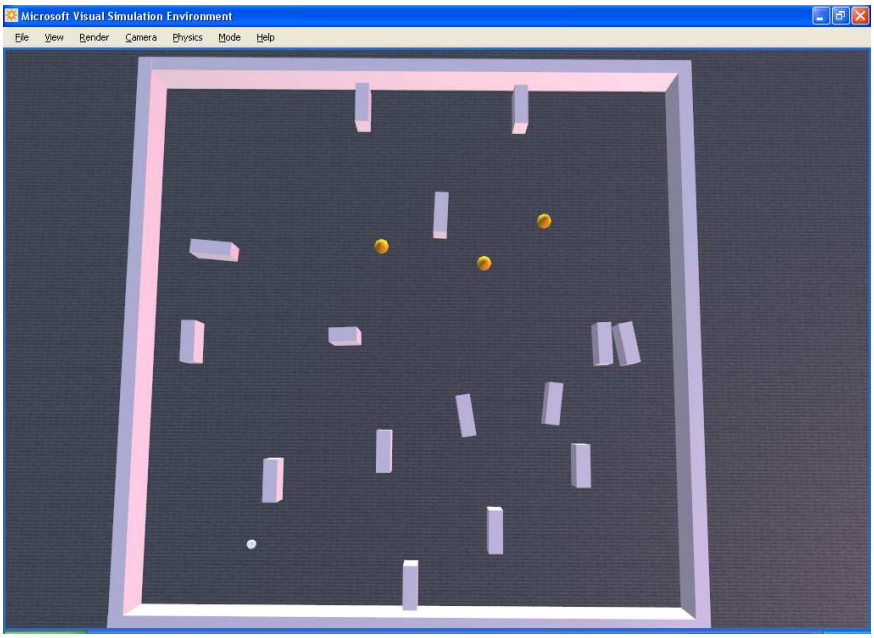

Figura 4-5. Entorn de simulació amb parets

S'utilitza el robot *iRobot Create*, el qual en aquest entorn de simulació permet fer proves molt interessants. El robot, a més, disposa d'un reproductor de música. Bé, és tan sols un reproductor de "*bips*" de l'ordinador, però és un actuador del robot. Aquest reproductor s'activarà cada cop que el robot giri, similar al funcionament d'un camió quan va marxa enrere.

4.2 Presentació de la guia

La guia es presentarà en dos formats i com a annex al projecte:

- **Paper/format electrònic**
- **Pàgina web**

El format **paper** serà el mateix que el d'aquesta memòria i es trobarà al final de la mateixa, com a **annex**. D'aquesta forma el lector podrà llegir-la de forma fàcil (encara que és més recomanada la lectura a la pàgina web). També es presentarà com un fitxer independent en format PDF que es trobarà al CD d'annexos, per a que sigui més fàcil el seu transport i distribució independent de la memòria.

Però, la forma en la que s'ha ideat la guia és la de **pàgina web.** S'ha realitzat una molt simple pàgina web que conté la guia i la distribueix en els diversos apartats que la composen de forma que és fàcil moure's dins la mateixa còmodament, donada la longitud de la mateixa, molt més còmodament que no pas en format paper.

A més, les diferents pràctiques que s'han realitzat estan disponibles per a descarregar, l'usuari les pot descarregar i utilitzar en cas que no li surti alguna part de la guia o simplement si vol veure el resultat abans de fer tot.

Es podrà accedir a la pàgina web al servidor de la UAB Shades, a través de :

http://shades.uab.es/MSRS/

La pàgina presenta la següent forma:

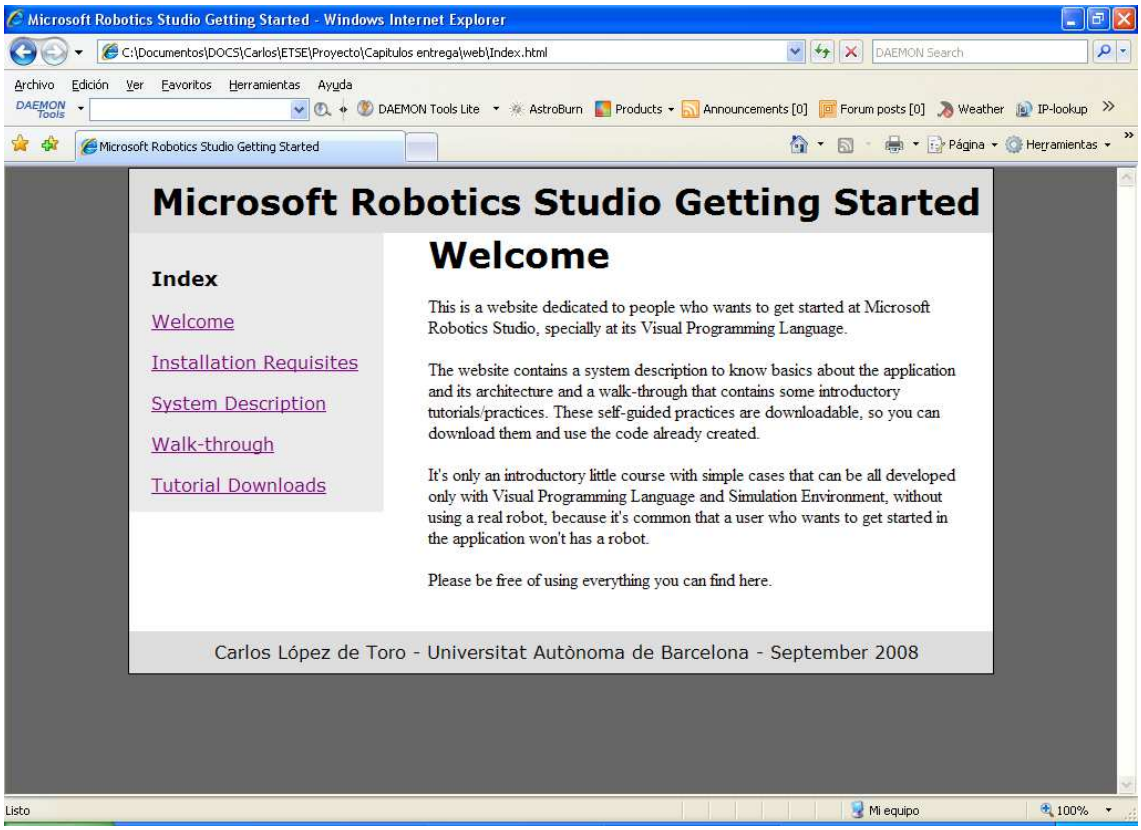

Figura 4-6. Pàgina web de la guia

Aquesta és la pàgina d'introducció, a la que s'accedeix a l'entrar (index.html), en la qual es poden veure les característiques principals de la plana (visibles a totes les parts):

- **Títol**: Microsoft Robotics Studio *Getting Started*
- **Menú esquerra**: conté un índex amb les diferents parts de la guia: requisits d'instal·lació, descripció del sistema, *walk-through* i descàrrega de *tutorials*. Des de qualsevol part de la pàgina es pot accedir a qualsevol altre amb un simple click
- **Peu de pàgina**: les meves dades així com la data de creació de la web
- **Contingut**: simplement el contingut de cada apartat

Aquestes són característiques comuns a totes les parts de la web. Si anem a Walk-through trobarem els diversos tutorials fàcilment accessibles així com botons per a retornar al inici.

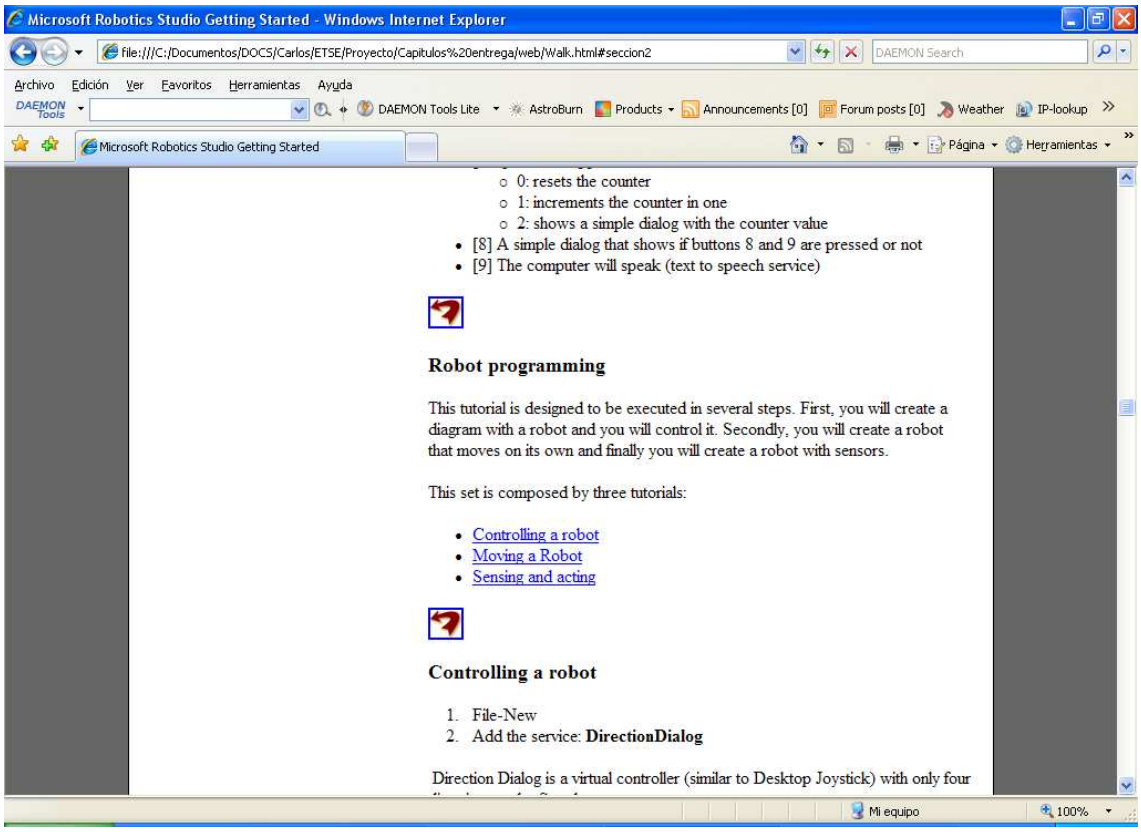

Figura 4-7. Walk-through

És una plana web molt simple amb l'únic objectiu de presentar la informació de forma fàcil i còmode pel lector, així que s'ha realitzat amb un disseny senzill i minimalista, però amb un mínim d'atractiu.

L'apartat *Tutorial Downloads* conté en format comprimit *Zip* els quatre tutorials realitzats, de forma que es pot descarregar qualsevol d'ells.

4.3 Alternatives en la elaboració de la guia

Penso que la millor manera de presentar aquesta guia és en anglès i en format de plana web, així que no contemplarem alternatives en aquest sentit. Sí que ho farem en el seu contingut.

En la guia realitzada en aquest projecte s'ha donat gran importància a Visual Programming Language. S'ha fet això perquè es considera la gran eina que presenta Microsoft Robotics Studio però no és tot el que ofereix aquest programa. Tenint en compte que l'objectiu de la guia és la d'introduir a l'usuari en l'aplicació amb poques hores, es podria haver suprimit part de la programació robòtica i haver treballat altres aspectes, com la creació de serveis o la programació amb *Visual Studio*, allunyant-nos de la programació visual.

De la mateixa forma es podrien haver canviat els tutorials realitzats, o haver-los realitzat per a algun robot real, però ja s'ha esmentat anteriorment que no es volia fer així ja que no tothom disposa d'un robot real per a fer proves. Els tutorials realitzats a més s'han

escollit perquè es consideren les parts més importants, els conceptes bàsics de l'aplicació. És bàsic realitzar una introducció a l'aplicació així com considero bàsic aprendre a controlar un robot i veure l'accés als seus sensors i actuadors.

Tot això es podria haver canviat per tutorials més extensos aprofundint més en la configuració d'un robot concret per a que realitzi tasques més complexes, però com ja s'ha comentat, la idea era la de presentar pràctiques senzilles independents per a introduir l'usuari en la realització de tasques bàsiques.

Microsoft Robotics Studio és un programa molt complet i una profunda revisió del qual requeriria moltes hores, no una pràctica, si no un curs complet.

Deixant de banda la realització d'una guia, si que es podria haver fet una altra cosa amb Microsoft Robotics Studio. Una alternativa que es va tenir en compte a l'hora d'iniciar el projecte va esser la de muntar un robot, o una colònia de robots i simular-ho.

Això ens hagués donat potser més coneixements, però potser no és tan adequat per a utilitzar com a material docent (que era una de les característiques de la proposta de projecte), si no que seria més un projecte final de carrera independent, en el qual no hi hagués altre objectiu que la realització del mateix així com l'assoliment dels coneixements derivats.

4.4 Possibles continuacions

Com s'ha comentat anteriorment Microsoft Robotics Studio és un programa molt complet. Tan sols Visual Programming Language presenta infinites possibilitats, les quals, per cert, no estan molt documentades.

Penso que la guia realitzada es podria fer servir com a primera part d'un programa complet de pràctiques, el qual en fases més avançades podria incloure:

- Edició d'entorns de simulació
- Simulació de múltiples robots
- Programació real d'un robot
- Editar manifests
- Creació de serveis des de VPL
- Accés remot a un servei
- Programar en codi un servei
- Etc

Una cosa que m'hagués agradat incloure a la guia realitzada era la d'editar entorns, però amb tan poques hores no hi ha temps de fer-ho.

Seria molt **important**, dins un programa de pràctiques, la programació d'un robot real. Per a una universitat invertir uns centenars d'euros en un o diversos kits LEGO Mindstorms NXT no hauria de ser molt complicat (de fet, no se si la UAB en disposa), i això fomentaria l' interès de l'estudiant ja que les pràctiques adquiririen un dinamisme que no presenten moltes d'elles. A més Visual Programming Language és un entorn agradable per l'usuari, el que també ajudaria.

Figura 4-8. Kit de Lego Mindstorms

Disposar d'aquest robot permetria portar a terme alguns dels punts esmentats com a part del programa, ja que es podria associar un servei al mateix i accedir-hi des de qualsevol ordinador de per, exemple, un laboratori de la *ETSE*. De totes formes ara mateix crec que no hi ha cap assignatura on es poguessin aplicar aquestes pràctiques, ja sigui per contingut o per nombre d'hores.

Un altre punt important seria el de programar fora de Visual Programming Language, però disposant de poques hores, potser és més important dedicar-s'hi a aquest entorn, el qual de per sí és molt complex i requereix de moltes hores.

Referències

[1] Alicia Fornés, "Pràctiques planificació de sistemes", CVC - Planificació de sistemes, 2008, http://www.cvc.uab.es/shared/teach/a25001/Practiques/c25001pract.htm, (accés 24 Agost 2008))

[2] UAB, "Intel·ligència Artificial", CVC - Intel·ligència Artificial, 9 Juliol 2008, http://www.cvc.uab.es/shared/teach/a25001/Practiques/c25001pract.htm, (accés 24 Agost 2008))

[3] dEIC, "Xarxes de Computadors 2", dEIC – Xarxes, 18 Agost 2008, http://www.deic.uab.es/docencia/viewprog.php?idioma=0&codias=24998-0#material, (accés 24 Agost 2008)

[4] Alberto Bietti, "Lego NXT + wiimote with MSRS tutorial", Alberto Bietti's projects blog, 17 Novembre 2007, http://alandtech.blogspot.com/2007/11/lego-nxt-wiimote-with-msrs-tutorial.html, (accés 26 Agost 2008)

[5] Jefferson Felipe Santillán, "Introducción a Microsoft Robotics Studio", The Code, 9 Juny 2008, http://jsantillan.wordpress.com/2008/06/09/introduccion-a-microsoft-robotics-studio/, (accés 26 Agost 2008)

[6] Trevor Taylor, "MSRS Code Page", MSRS Code Page, 4 Maig 2008, http://www.soft-tech.com.au/MSRS/, (accés 27 Agost 2008)

[7] Microsoft Corporation, "Microsoft Robotics Studio Developer Center", Microsoft Developer Network, 1 Agost 2008 , http://msdn.microsoft.com/es-es/robotics/default(en-us).aspx, (accés 9 Agost 2008)

5 Conclusions

En aquest capítol es farà un balanç dels objectius aconseguits en funció dels proposats i com ha anat l'evolució del treball, així com vies de continuació.

5.1 Objectius i feina realitzada

El projecte "*Anàlisi de* Microsoft Robotics Studio" tenia com a objectius realitzar un informe crític de Microsoft Robotics Studio i del material docent que existia així com realitzar una guia d'ús de l'eina que en facilités la docència.

Al finalitzar aquest projecte es pot afirmar que els objectius establerts al inici del mateix s'han assolit i, tenint en compte l'ajustat temps del que es disposava, és un èxit. Dins d'aquest temps l'evolució del treball ha estat bona i la previsió temporal feta al inici del projecte s'ha acomplert gairebé sempre en els períodes establerts.

En aquest projecte es va planificar un objectiu general al principi: elaborar un entorn docent accessible via web per a l'aprenentatge de Microsoft Robotics Studio i que aquest entorn millorés o complementés el material existent actualment que es pogués trobar a Internet. El projecte consistiria concretament en realitzar un *"walk-through"* de l'aplicació, és a dir, una guia pas a pas que servís com a introducció a l'eina.

D'aquest objectiu general es van extreure dos grans subobjectius a realitzar en aquest projecte: elaborar un informe crític i objectiu tant de Microsoft Robotics Studio com del material docent existent on s'analitzi i compari amb altres eines similars; i elaborar una guia d'ús de l'eina accessible via web.

Un cop finalitzat el mateix es pot dir que s'han assolit els objectius establerts, com ja s'ha esmentat anteriorment, de forma satisfactòria. El resultat es pot trobar a aquesta memòria i als annexos adjunts

El limitat nombre d'hores de que es disposava va limitar la forma de treball, però tot i així, s'ha realitzat tota la feina planificada dins, aproximadament, dels límits establerts en la planificació inicial.

5.2 Anàlisi dels resultats

En aquest apartat es farà balanç dels resultats obtinguts durant el projecte i es donaran les conclusions a las que s'ha arribat un cop finalitzat el mateix.

5.2.1 Microsoft Robotics Studio

S'ha pogut comprovar com Microsoft Robotics Studio és un dels millors entorns de programació i simulació de robots i sistemes multirobot que hi ha avui en dia tant per l'aspecte de la seva facilitat d'ús com per les seves possibilitats, incloure simulador, etc. A més s'està treballant en una nova versió amb correcció d'errors i algunes millores.

Però, té alguns defectes respecte els altres programes similars. Primer de tot, presenta dues versions: no comercial i comercial. La versió no comercial és indicada per a ús educacional, però només permet la utilització de les rutines de components (CCR i DSS) en un nombre limitat de robots o PC's, però la comercial només en permet distribuir 200 d'aquestes rutines de components, és a dir: també té limitacions. En cas d'ús educacional no presenta altres limitacions d'ús.

En definitiva, els defectes de l'aplicació són: només es pot utilitzar en plataforma Windows (el que li resta molta quota de mercat) i que hi ha limitació d'ús de les rutines de components; i en cas de llicència comercial, a més: que no és gratuït i que té un cost addicional distribuir més de 200 còpies de rutines, qualitats que algunes altres eines milloren.

5.2.2 Material docent

Del material docent existent es poden extreure altres conclusions. D'una banda cal dir que l'aplicació porta uns 2 anys al mercat des de les primeres versions (actualment 1.5 és la versió final). Això no és gaire temps, i tot i que existeix bastant material, el material docent autònom que existeix no és gaire, i no és millor que el propi material de Microsoft, de fet se sol fer referència a aquest, del que es pot dir que és el millor.

Cal destacar que aquest material està realitzat directament pels creadors de l'aplicació, no per un equip d'edició, pel que, tot i que és un bon material, no és perfecte. Els últims mesos han sortit a la venda els primers llibres realment bons dedicats exclusivament a l'aplicació, encara que no els he llegit, però sembla que són la gran alternativa, o millor dit, complement, al material proporcionat per Microsoft.

5.2.2.1 Elaboració de nou material docent

Com a alternativa al material existent i tractant de millorar-ho en alguns aspectes s'ha planificat i desenvolupat un entorn docent d'introducció a l'aplicació. La idea ha estat realitzar una guia que serveixi per a introduir a l'usuari en l'aplicació, concretament en Visual Programming Language, el qual es considera la base de Microsoft Robotics Studio. La forma d'aquesta ha estat un petit manual d'usuari i requisits d'instal·lació i unes pràctiques guiades totalment documentades però tractant sempre de, mitjançant el mètode més senzill, mostrar a l'usuari els conceptes bàsics.

La idea es que aquestes pràctiques es puguin portar a terme en unes poques hores de forma simple i l'usuari, un cop treballat tot el conjunt, hagi assolit uns coneixements

sobre l'aplicació que li permetin realitzar senzilles aplicacions tot sol un cop hagi acabat.

A més de presentar la guia com a document escrit s'ha realitzat una senzilla pàgina web que la conté i en facilita la seva lectura, així com posa a disposició de l'usuari la descàrrega de les pràctiques guiades ja realitzades. Aquesta plana s'ha escrit en anglès per a facilitar l'accés i distribució de la guia ja que es pensa que és el millor idioma per aquest tipus de document. La direcció és: **http://shades.uab.es/MSRS/**

Sobre la possibilitat d'utilitzar el controlador de XBox360 per a realitzar proves o fins i tot incloure'l a la guia, s'ha desestimat aquesta opció. D'una banda perquè pot representar alguns problemes la seva instal·lació i configuració, i d'altra perquè s'ha decidit que tot sigui simulat, de forma que qualsevol persona amb un ordinador i el programa instal·lat pugui tenir accés.

Amb l'elaboració d'aquest projecte he guanyat experiència i coneixements, en general sobre el món de la robòtica, en el qual no hi havia treballat mai. D'aquest món puc dir que en les primeres fases del projecte en que vaig començar a investigar sobre l'estat de l'art ho vaig passar molt bé, va despertar el meu interès i vaig aprendre força sobre la programació de robots. Després he assolit coneixements en concret sobre Microsoft Robotics Studio, per descomptat.

En definitiva, el fet d'haver investigat sobre robòtica, un camp relativament jove i en constant evolució, haver treballat amb Microsoft Robotics Studio, també una aplicació bastant nova i amb creixent popularitat, i haver-ne desenvolupat una guia, penso que ha estat una experiència satisfactòria tant personalment com per als meus estudis.

5.3 Possibles ampliacions

Tal com es diu al Capítol 4, existeixen algunes vies d'ampliació del projecte. La més immediata és la de realitzar més tutorials sobre l'aplicació per anar introduint nous conceptes, així com utilitzant altres característiques dels robots disponibles. Això significaria ampliar la web presentada amb més contingut: tutorials, més manuals d'usuari, etc...i seria un projecte viable.

La ampliació més interessant, però, és la de fer que la guia formi part d'un programa complet de pràctiques sobre l'aplicació. Aquest programa podria fer-se servir en alguna assignatura relacionada amb la robòtica o en un curs dedicat a la mateixa directament. Hauria de ser un programa molt més complet que no pas els tutorials realitzats en aquest projecte, i abastar uns coneixements molt més amplis, ja que considero que aquí només es tracta una petita part de la totalitat de l'aplicació.

En la guia realitzada s'ha treballat amb Visual Programming Language concretament, el qual és una part (molt important, tot s'ha de dir) de Microsoft Robotics Studio, per això seria interessant ampliar la guia i treballar amb les demés aplicacions que presenta el programa de Microsoft.

Tal com s'ha comentat al Capítol 4, alguns possibles temes tractats a la guia són:

- Edició d'entorns de simulació
- Simulació de múltiples robots
- Programació real d'un robot
- Editar manifests
- Creació de serveis des de VPL
- Accés remot a un servei
- Programar en codi un servei
- Etc

És especialment interessant treballar amb un robot real, perquè encara que simular té moltes possibilitats fer-ho amb un de real augmenta totes les sensacions i experiències a l'hora d'aprendre a utilitzar l'aplicació i dins d'un curs seria possible que una part del mateix fos el treball amb robots reals.

Una altre possible ampliació és, a la plana web, afegir un fòrum. D'aquesta manera es podria crear una comunitat en torn a Microsoft Robotics Studio, on tots els participants poguessin aportar idees, informació, cursos o exposar els seus dubtes i trobar de forma fàcil ajuda en altres usuaris.

Referències

En aquest punt es trobaran totes les referències del projecte ordenades per autor:

[1] Alberto Bietti, "Lego NXT + wiimote with MSRS tutorial", Alberto Bietti's projects blog, 17 Novembre 2007, http://alandtech.blogspot.com/2007/11/lego-nxt-wiimote-with-msrs-tutorial.html, (accés 18 i 26 Agost 2008)

[2] Aldebaran Robotics, "Nao-About", Aldebaran Robotics, Setembre 2007, http://www.aldebaran-robotics.com/eng/Nao.php, (accés 4 Agost 2008)

[3] Alejandro Martínez, "Programando con Microsoft Robotics Studio", Scribd, 30 Juliol 2007, http://www.aquora.net/studentpartners/docs/Alejando%20Martinez%20-%20Microsoft%20Robotics%20Studio.pdf (accés 15 Agost 2008)

[4] Alicia Fornés, "Pràctiques planificació de sistemes", CVC - Planificació de sistemes, 2008, http://www.cvc.uab.es/shared/teach/a25001/Practiques/c25001pract.htm, (accés 24 Agost 2008))

[5] Amazon.com, Inc. or its affiliates, "Professional Microsoft Robotics Developer Studio (Wrox Programmer to Programmer) (Paperback)", Amazon.com/Books, 7 Juliol 2008, http://www.amazon.com/Professional-Microsoft-Robotics-Developer-Programmer/dp/0470141077/ref=pd_bxgy_b_img_b, (accés 18 Agost 2008)

[6] Ankur Mittal, "Microsoft Robotics Studio (1.5) Refresh", MS Windows Vista Compatible Software, 27 Desembre 2007, http://www.windowsvistaplace.com/microsoft-robotics-studio-15-refresh/microsoft, (accés 18 Agost 2008)

[7] Ben Axel, "Robotics Studio Maze Simulator", Channel 9 (MSDN) 24 Juny 2007, http://channel9.msdn.com/playground/Sandbox/220836-Robotics-Studio-Maze-Simulator/ ,(accés 18 Agost 2008)

[8] Brian Peek, "Microsoft Robotics Studio and Lego Mindstorms NXT", Coding4Fun, 16 Juliol 2007, http://blogs.msdn.com/coding4fun/archive/2007/07/16/3902344.aspx, (accés 14 Agost 2008)

[9] Conscious-Robots, "Microsoft Robotics Studio", Conscious-Robots, 25 Abril 2008, http://www.conscious-robots.com/es/robotics-studio/, (accés 12 Agost 2008)

[10] Cyberbotics Ltd, "Webots 5", Cyberbotics Professional Mobile Robot Simulation, Març 2008, http://www.cyberbotics.com/products/webots/index.html, (accés 1 Agost 2008)

[11] dEIC, "Xarxes de Computadors 2", dEIC – Xarxes, 18 Agost 2008, http://www.deic.uab.es/docencia/viewprog.php?idioma=0&codias=24998-0#material, (accés 24 Agost 2008)

[12] Edilson García Chiñas, "Microsoft Robotics Studio (1.5) CTP Sep 2007", Tecnologias Microsoft, 1 Setembre 2007, http://mredison.wordpress.com/2007/09/01/microsoft-robotics-studio-15-ctp-sep-2007/, (accés 15 Agost 2008)

[13] George Gardner, "Bill Gates, Robots & Microsoft Robotics Studio", TECH.BLORGE, 7 Gener 2007, http://tech.blorge.com/Structure:%20/2007/01/07/bill-gates-robots-microsoft-robotics-studio/, (accés 8 Agost 2008)

[14] Google, "Fòrums de debat Microsoft Robotics Studio", Google Grupos, 1 Agost 2008, http://groups.google.com/group/microsoft-robotics-studio/topics, (accés 12 Agost 2008)

[15] Gostai, "URBI", Gostai Robotics, 2008, http://www.gostai.com, (Accés 30 Juliol 2008)

[16] Harplogic, "Surveyor SRV-1 Setup for Microsoft Robotics Studio (MSRS)", 11 Juny 2008, http://www.surveyor.com/MSRS.html ,(accés 13 Agost 2008)

[17] iRobot Corporation, "iRobot Roomba Family", iRobot, 2008, http://www.irobot.com/, (accés 6 Agost 2008)

[18] Jefferson Felipe Santillán, "Introducción a Microsoft Robotics Studio", The Code, 9 Juny 2008, http://jsantillan.wordpress.com/2008/06/09/introduccion-a-microsoft-robotics-studio/, (accés 18 i 26 Agost 2008)

[19] Jorge Saavedra, "Microsoft Robotics Studio v1.5 liberado", El mundo informático, 18 Juliol 2007, http://jorgesaavedra.wordpress.com/2007/07/18/microsoft-robotics-studio-v15-liberado/, (accés 18 Agost 2008)

[20] Ken Berry, "Microsoft Enters Robotics – An educator perspective", Robot, 2008, http://www.botmag.com/issue3/microsoft_technical_preview_3.shtml, (accés 15 Agost 2008)

[21] Ken Cheung, "Microsoft Robotics Studio October Community Tech Preview (CTP)", Eda Blog, Octubre 2006, http://edablog.com/2006/10/08/microsoft-robotics-studio-october-community-tech-preview-ctp/, (accés 13 Agost 2008)

[22] K-Team Corporation, "Robot Hemisson, robot Kephera i robot E-Puck", K-Team Products, 2008, http://www.k-team.com/kteam/home.php?rub=0&site=1&version=EN, (accés 29 Juliol 2008)

[23] Kyle Johns i Trevor Taylor, "Professional Microsoft Robotics Developer Studio", 12 Agost 2008, http://electronicdesign.com/Articles/ArticleID/19568/19568.html, (accés 21 agost 2008)

[24] Kyle Johns i Trevor Taylor, "Professional Microsoft Robotics Developer Studio", Maig 2008, http://www.wrox.com/WileyCDA/WroxTitle/Professional-Microsoft-Robotics-Developer-Studio.productCd-0470141077.html , (accés 21 agost 2008)

[25] Matt Mondok, "Roll your own Robot Jones with Microsoft's robotics kit", Ars Technika, 13 Desembre 2006, http://arstechnica.com/journals/microsoft.ars/2006/12/13/6280, (accés 16 Agost 2008)

[26] Merriam Webster, "Definició Robotics", Merriam-Webster Online Dictionary, 2008 , http://mw1.merriam-webster.com/dictionary/Robotics, (accés 29 Juliol 2008)

[27] Microsoft Corporation, "Microsoft Robotics Studio Developer Center", Microsoft Developer Network, 1 Agost 2008 , http://msdn.microsoft.com/es-es/robotics/default(en-us).aspx, (accés 9 Agost 2008)

[28] Microsoft, "Documentació Microsoft Robotics Studio", Microsoft Developer Network, 2008, http://msdn.microsoft.com/es-es/library/bb483024(en-us).aspx, (accés 4 Agost 2008)

[29] Mohammed Hossam, "Microsoft Robotics Studio August CTP is out", Bashmohandes, 8 Agost 2008, http://spellcoder.com/blogs/bashmohandes/archive/2006/08/05/257.aspx, (accés 5 agost 2006)

[30] Múltiples, "Player-Stage-Gazebo", Player Project, 28 Març 2008, http://playerstage.sourceforge.net, (Accés 31 Juliol 2008)

[31] Natural News, "Microsoft Robotics Studio Provides Common Ground for Robotics Innovation (press release)", Robotics, 22 Juny 2006, http://www.naturalnews.com/019454.html, (accés 11 Agost 2008)

[32] Orlando Hernández, "Microsoft Robotics Studio", Cyber-Trends, 25 Setembre 2007, http://cyber-trends.blogspot.com/2007/09/microsoft-robotics-studio.html, (accés 17 Agost 2008)

[33] Real Academia Española, "Definició Robot", Real Academia Española. Diccionario usual, 2008 , http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&TIPO_BUS=3&LEMA=robot, (accés 29 Juliol 2008)

[34] RoboRealm, "RoboRealm setup for Microsoft Robotics Studio (MSRS)", RoboRealm, desconegut, http://www.roborealm.com/help/MSRS.php, (accés 12 Agost 2008)

[35] Sara Morgan, "An introduction to Programming Robots with Microsoft Robotics Studio", DevX.com, 9 Octobre 2006, http://www.devx.com/dotnet/Article/32729, (accés 12 Agost 2008)

[36] Sony Entertainment Robot Europe, "Aibo", Sony Aibo Europe, 2006, http://support.sony-europe.com/aibo/index.asp, (accés 4 Agost 2008)

[37] Steven Smith, "Microsoft Robotics Studio Released", Steven Smith Blog, 13 Desembre 2006, http://aspadvice.com/blogs/ssmith/archive/2006/12/13/Microsoft-Robotics-Studio-Released.aspx , (accés 13 Agost 2008)

[38] The LEGO Group, "Lego Mindstorms NXT", Lego Mindstorms NXT Home, 2008, http://mindstorms.lego.com/eng/Salzburg_dest/default.aspx, (accés 4 Agost 2008)

[39] Trevor Taylor, "MSRS Code Page", MSRS Code Page, 4 Maig 2008, http://www.soft-tech.com.au/MSRS/, (accés 20 i 27 Agost 2008)

[40] UAB, "Intel·ligència Artificial", CVC - Intel·ligència Artificial, 9 Juliol 2008, http://www.cvc.uab.es/shared/teach/a25001/Practiques/c25001pract.htm, (accés 24 Agost 2008))

[41] USC, "Introduction::iRobot Create/Roomba and Microsoft Robotics Studio", USC, Desconegut, http://roboticsprimer.sourceforge.net/workbook/Introduction::iRobot_Create/Roomba_and_Microsoft_Robotics_Studio#Running_the_Simulator , (accés 13 Agost 2008)

[42] Wikipedia, "Definició Aibo", Wikipedia, 23 Juliol 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Aibo, (accés 4 Agost 2008)

[43] Wikipedia, "Definició iRobot Roomba", Wikipedia, 4 Agost 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Roomba, (accés 6 Agost 2008)

[44] Wikipedia, "Definició Lego Mindstorms", Wikipedia, 10 Maig 2008, the online enciclopedia, http://es.wikipedia.org/wiki/Lego_Mindstorms, (accés 4 Agost 2008)

[45] Wikipedia, "Definició Nao robot", Wikipedia, 2 Agost 2008, the online enciclopedia, http://en.wikipedia.org/wiki/Nao_(robot), (accés 4 Agost 2008)

[46] Wikipedia, "Definició Robot", Wikipedia, 25 Juliol 2008, the online enciclopedia, http://en.wikipedia.org/wiki/Robot, (accés 29 Juliol 2008)

[47] Wikipedia, "Definició URBI", Wikipedia, 5 Juny 2008, the online enciclopedia, http://es.wikipedia.org/wiki/URBI, (accés 30 Juliol 2008)

[48] Wrox, "Professional Microsoft Robotics Developer Studio", Wrox: Programmer to programmer, Maig 2008, http://www.wrox.com/WileyCDA/WroxTitle/Professional-Microsoft-Robotics-Developer-Studio.productCd-0470141077,descCd-DOWNLOAD.html , (accés 14 Agost 2008)

Annexes

En els annexes s'adjunten una sèrie de documents importants per a que el lector pugui comprendre en la seva totalitat el projecte realitzat així com seguir la guia elaborada i fer proves amb Microsoft Robotics Studio.

I. Instal·lació Microsoft Robotics Studio

Durant tot el projecte, tant a la recerca com a la elaboració de la guia s'ha utilitzat la versió **1.5 Refresh** de Microsoft Robotics Studio. En el moment d'escriure això existeix una *Community Technical Preview* de la versió 2008 del programa, és una versió més nova però no és final. Per aquesta raó i per la popularitat de la versió 1.5 (els fitxers entre una i altre versió no sempre són compatibles) s'ha decidit fer servir aquesta.

Primer de tot s'ha d'instal·lar Microsoft Robotics Studio. L'aplicació només està disponible en anglès, i la versió **Microsoft Robotics Studio 1.5 Refresh** es pot descarregar directament des de la següent direcció, a la plana web de Microsoft:

- http://www.microsoft.com/downloads/details.aspx?FamilyId=73092FF6-E37B-45C6-8E5E-C23D5D632B1E&displaylang=en

És un instal·lador de 87Mb instal·lable a ordinadors amb sistema operatiu: Windows CE; Windows Server 2003 R2 (32-Bit x86); Windows Server 2003 R2 x64 editions; Windows Vista; Windows XP; Windows XP 64-bit.

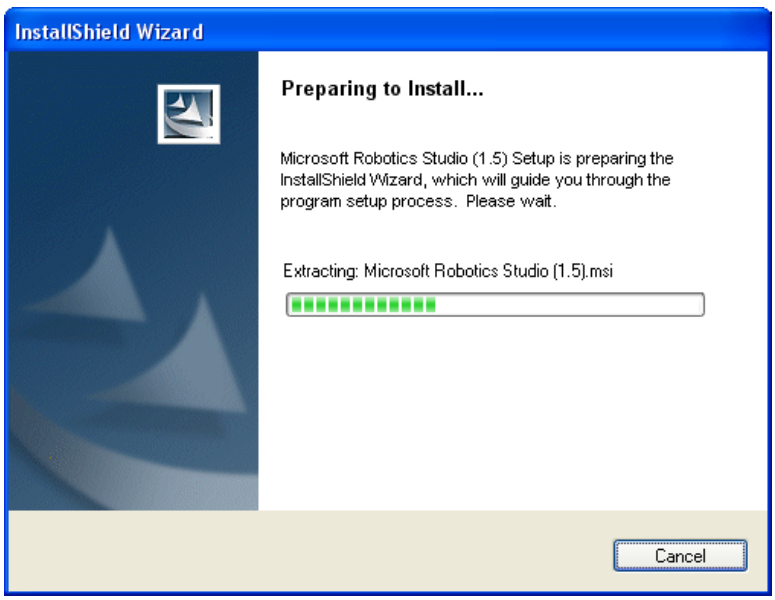

Figura 1. Instal·lació Microsoft Robotics Studio

Un cop s'inicia el procés l'aplicació informa a l'usuari de tot el software addicional necessari i dona la possibilitat d'instal·lar-ho, com són: la plataforma .NET 3.0, Microsoft XNA 2.0 o Ageia Physics Engine. Per defecte el programa s'instal·la a la unitat C a la ruta: *C:\Microsoft Robotics Studio (1.5)*

Després de la instal·lació del paquet inicial, s'ha de tenir en compte que la versió de Microsoft XNA inclosa al paquet és la 2.0 i el simulador requereix la 1.0, així que aquesta versió s'ha d'instal·lar també, abans o després de la instal·lació de l'aplicació. La versió 1.0 es pot descarregar des de la següent direcció:

- http://www.microsoft.com/downloads/details.aspx?FamilyId=A7DA4763-6807-4BD5-8D18-18C60C437F93&displaylang=en

En tercer lloc, per a la versió 1.5 Refresh hi ha dos paquets d'actualització, amb els quals instal·lats s'ha fet tot el treball, així que recomano la seva instal·lació un cop instal·lat tot l'anterior. Els paquets són:

- *Runtime and Tools Update for Microsoft Robotics Studio (1.5):* http://www.microsoft.com/downloads/details.aspx?FamilyId=2F747403-7D94-43F0-836B-84247FEE66C6&displaylang=en
- *Samples Update for Microsoft Robotics Studio (1.5):* http://www.microsoft.com/downloads/details.aspx?FamilyId=7EEB9B70-0E86-4E3E-92AF-6148BDA34B7C&displaylang=en

Si després d'haver realitzat tots els passos sorgeix algun problema, o durant l'execució de l'aplicació, Microsoft disposa d'un portal anomenat MSDN (*Microsoft Developer Network*), el qual inclou fòrums en els que es pot trobar ajuda de tota mena, tant a nivell pràctic com tècnic. Jo mateix m'he trobat amb alguns problemes que m'han solucionat als fòrums. En aquesta direcció es poden realitzar cerques al fòrum:

- http://forums.microsoft.com/qawizard/ask.aspx?siteid=1

Un cop instal·lada l'aplicació es podrà accedir als seus elements des de una nova carpeta a: *Inicio / Programas / Microsoft Robotics Studio (1.5)/*

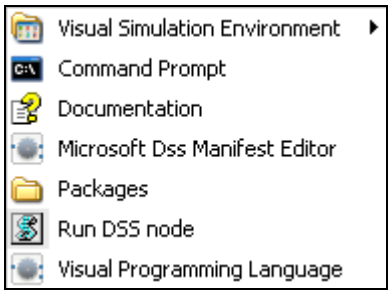

Figura 2. Elements de Microsoft Robotics Studio

II. Manual d'usuari

En aquest annex s'explicarà quins són els components de Microsoft Robotics Studio i com es fa servir l'aplicació Visual Programming Language.

Per començar, quan s'instal·la Microsoft Robotics Studio es crea una nova entrada al Menú Inici. Accedint-hi trobem el següent contingut:

- **Visual Simulation Environment:** l'entorn de simulació de l'aplicació. Des d'aquí es pot executar directament per a una sèrie de robots.
- **Command Prompt:** això executa una consola de comandes des de la qual es poden executar una sèrie d'aplicacions, com per exemple arrancar un servei.
- **Documentació:** molt important, és la documentació oficial de MRS. És una arxiu que conté una sèrie d'informació com guies d'usuari i tutorials sobre les aplicacions que conté MRS. Molt interessant per a resoldre dubtes o aprendre sobre l'aplicació.
- **Microsoft DSS Manifest Editor:** és un programa que permet editar els manifests des de una interfície gràfica.
- **Packages:** tots els paquets d'actualització que s'hagin instal·lat es poden trobar aquí.
- **Run Dss Node:** executa a l'ordinador un node DSS com a host.
- **Visual Programming Language:** és una aplicació que permet programar tot tipus de programes i robots des de una interfície gràfica.

Visual Programming Language – Conceptes Bàsics

El VPL està dirigit a programadors amb coneixements bàsics de conceptes com variables i lògica. No obstant això, no està limitat a usuaris novells.

Un flux de dades Microsoft Visual Programmig Language consisteix en una seqüència d'activitats representades com blocs amb entrades i sortides, els quals poden connectar-se a altres **blocs d'activitat**.

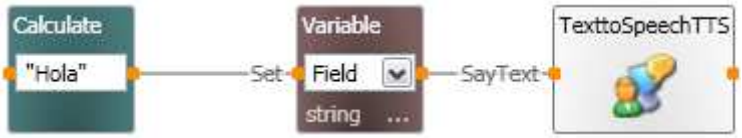

Figura 1. Diagrama de bloc amb activitats

Les activitats poden representar serveis pre-construïts, control de flux de dades, funcions o altres mòduls de codi. També poden incloure composicions d'altres activitats. Això fa possible compondre activitats i reusar la composició com un bloc de construcció. En aquest sentit, una aplicación construïda en VPL és una activitat per se.

Figura 2. Activitat composta

Els blocs d'activitat inclouen típicament el nom de l'activitat i llindars que representen els punts de connexió. Un bloc d'activitat pot incloure gràfics per il·lustrar el seu propòsit, així com elemts d'interfície que poden permetre l'usuari entrar valors, assignacions o transformacions de les dades utilitzades en una activitat.

Una activitat rep missatges contenint dades a través dels seus pins de connexió d'entrada. Els pins d'entrada d'una activitat són punts de connexió amb les seves funcions internes predefinides, conegudes com accions o manipuladors (els quals poden ser funcions proveïdes per un servei o fluxes d'informació niats). Un flux de dades s'executa **d'esquerra a dreta**.

Figura 3. Activitat rebent un missatge a través dels seus punts de connexió.

Una activitat pot tenir múltiples pins de connexió d'entrada, cadascun dels cuals amb el seu propi joc de pins de conexió de sortida. Els pins de connexió de sortida poden ser de dos tipus: una sortida de *resultat* o una sortida de *notificació* (de vegades també se'n parla com un esdeveniment o publicació de sortida). Les sortides de resultat es mostren com pins de connexió rectangulars, mentre que les de notificació tenen pins de connexió arrodonits.

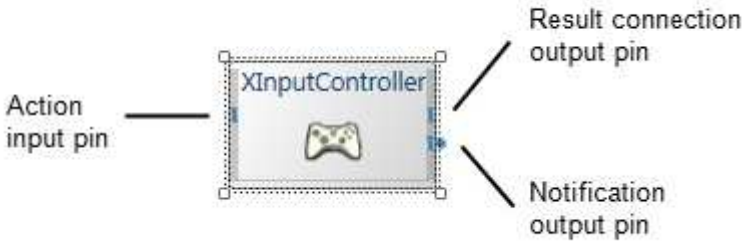

Figura 4. Pins de connexió

Un pin de sortida de resultat s'utilitza en situacions en què un missatge sortint és enviat com el resultat del missatge d'una acció entrant específica. Els pins de notificació poden enviar dades resultant d'un missatge d'entrada, però típicament mostren un missatge degut a un canvi en el seu estat intern.

Visual Programming Language – Utilització

Per connectar dos blocs d'activitat, cal arrossegar des del pin de connexió d'una activitat fins el pin d'una altra. Si existeixen diverses formes (entrades i sortides) de connectar les activitats, apareixerà una caixa de diàleg de connexions que permetrà definir quines entrades i sortides es volen conectar.

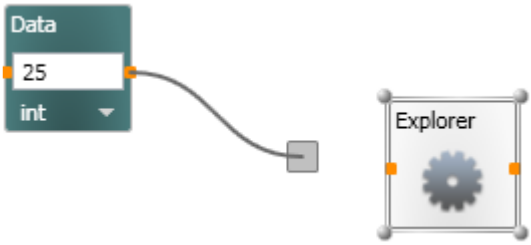

Figura 5. Connexió de dos blocs

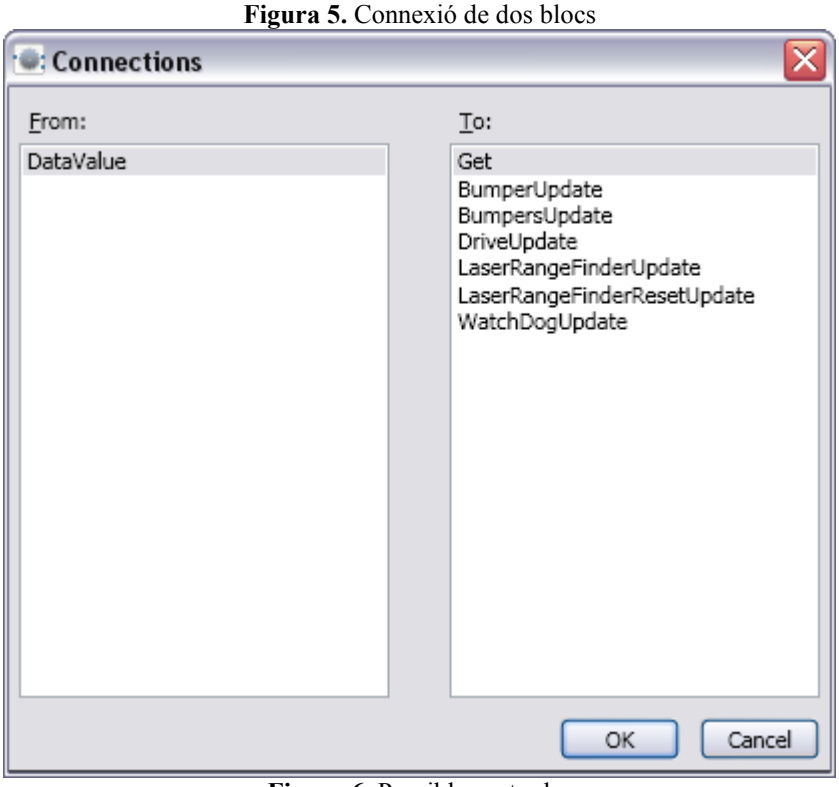

Figura 6. Possibles entrades

De vegades, les connexions entre activitats requereixen aparellar les dades que seran transmeses entre les connexions. Quan això succeeix, apareix un diàleg *Data connections* i mostra les opcions de l'emissor del missatge en la part esquerra i les del receptor en la part dreta. Les opcions d'informació entrant poden incloure els valors per defecte del tipus de dades (0 per tipus numèrics, fals per booleans i nul per tota la resta), el valor de les dades o valor de sub-dades (membres).

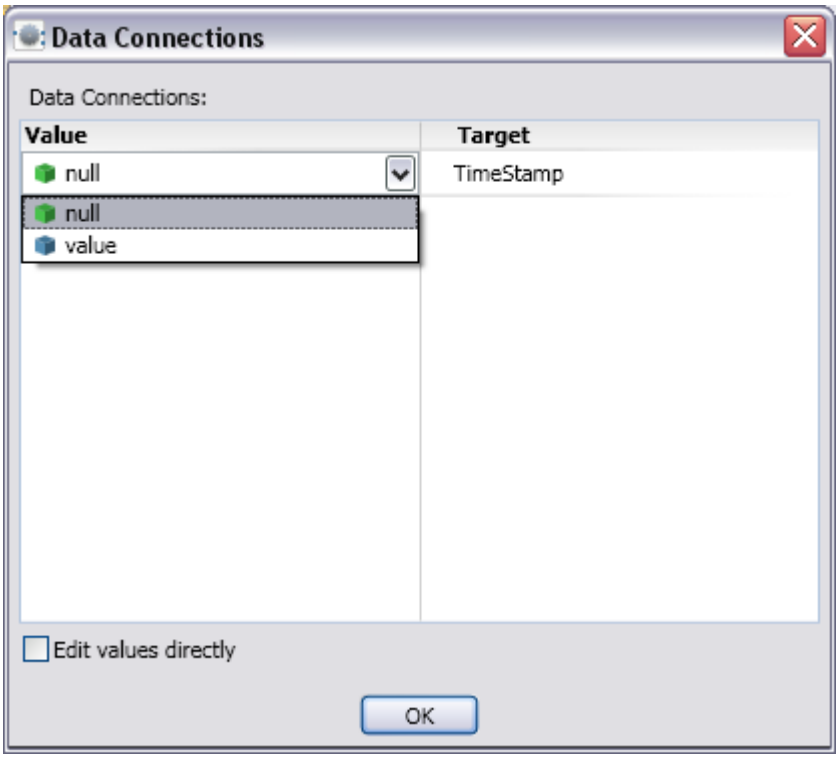

Figura 7. Ajust de dades

Si les dades connectades no coincideixen, es mostrarà el símbol .

En MSRS es pot crear un diagrama:

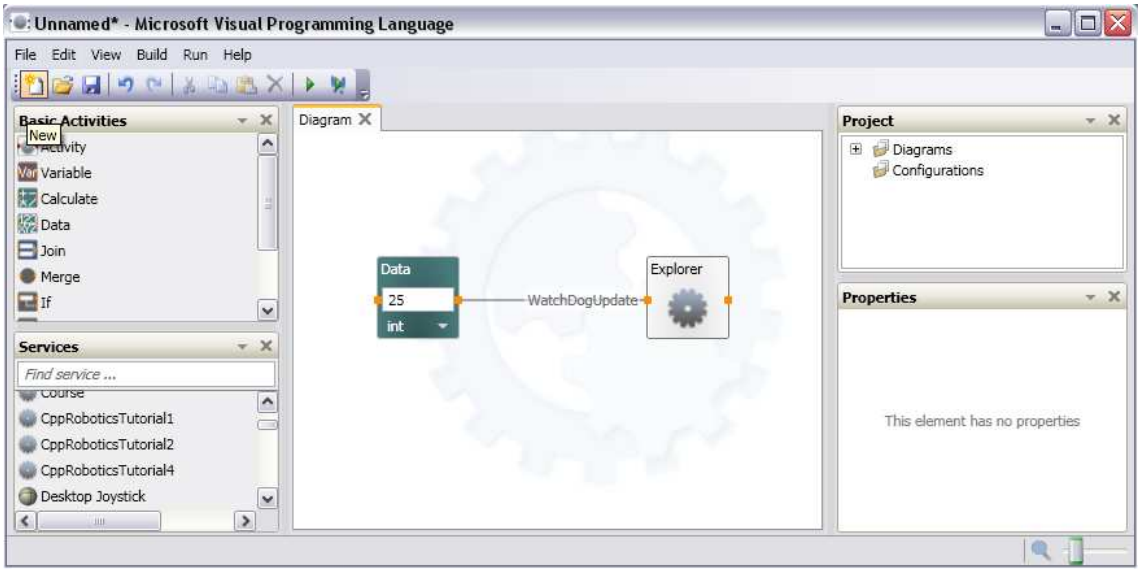

Figura 8. Creació d'un diagrama

També es pot crear una activitat arrossegant i deixant anar l'activitat bàsica "Activitat" dins el diagrama (es mostrarà una altra etiqueta clicant sobre l'activitat).

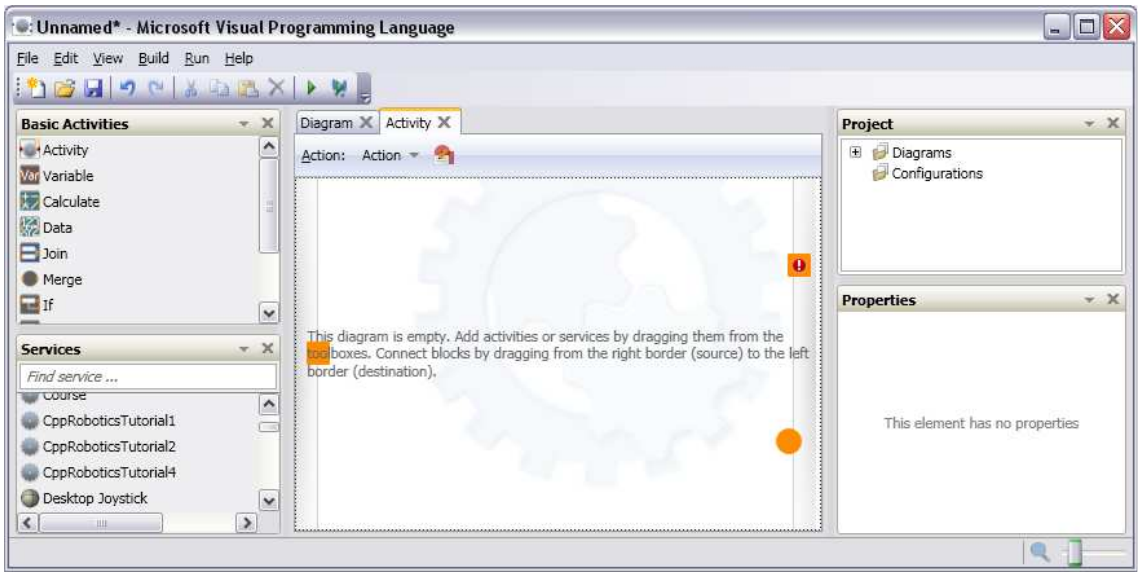

Figura 9. Creació d'una activitat

És important fixar-se en què l'activitat té pins d'entrada i sortida, els quals es poden editar des del menú *Edit*, fent clic sobre *Actions and Notifications*.

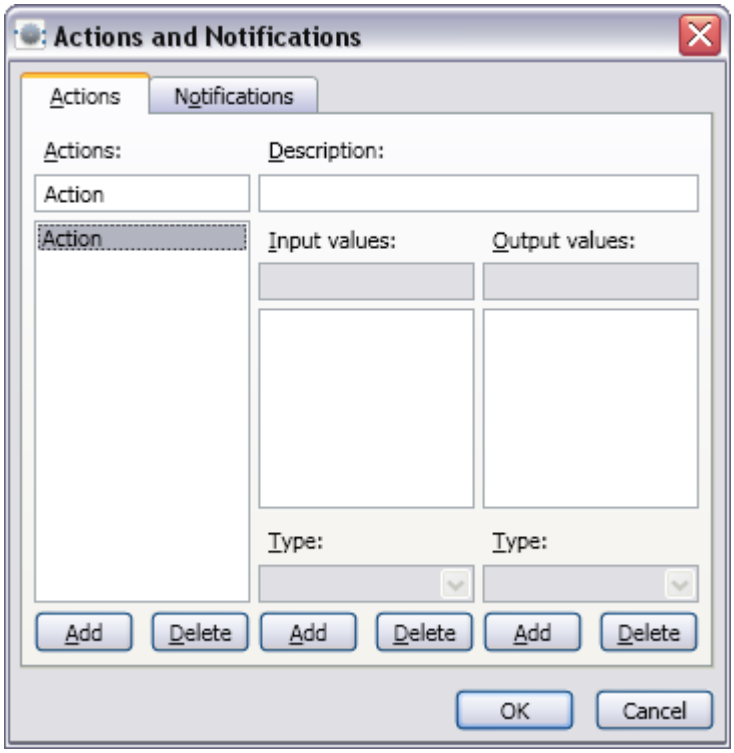

Figura 10. Accions i notificacions

Existeixen diverses toolboxes (caixes d'eines) en una finestra de VPL: Activitats Bàsiques, Serveis, Projectes i Propietats (*Basic Activities, Services, Project and Properties)*, les quals es poden moure i plegar, de manera que poden ser reordenades dins de la finestra principal de VPL. En el menú Vista (*View*) es poden amagar o mostrar de nou.

Per començar un diagrama de fluxe de dades, cal arrosegar i deixar anar blocs d'activitat de flux de dades o de servei des de les toolboxes fins l'àrea del diagrama i connectar-los. També es pot afegir una nova activitat fent clic dues vegades a l'entrada de la seva toolbox. Situar-se amb el punter sobre l'entrada d'una toolbox mostra una ajuda que inclou una descripció de com es pot fer servir l'activitat.

Les activitats es poden tallar, copiar, pegar o eliminar, seleccionant-les al bloc del diagrama amb les commandes del menú Edició (*Edit*) o el menú contextual de les pròpies activitats.

En la part esquerra es pot veure la toolbox d'Activitats Bàsiques (*Basic Activities*):

Figura 11. Basic Activities toolbox

Aquestes són les activitats bàsiques que es poden incloure en el diagrama.

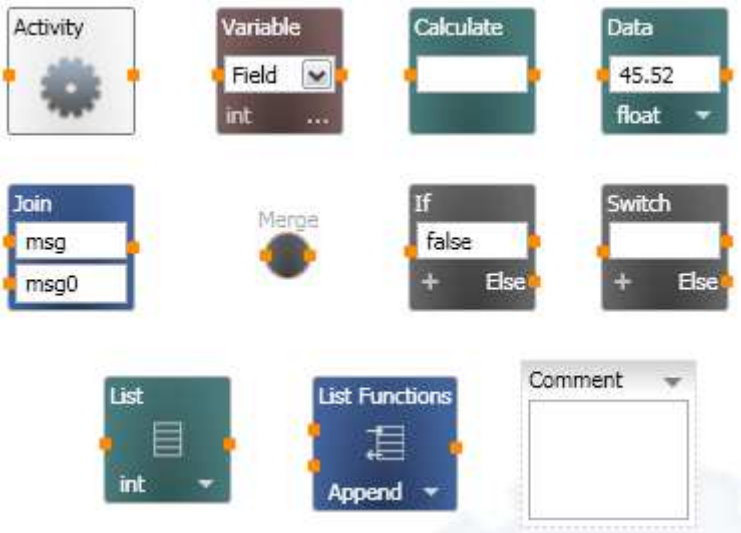

Figura 12. Activitats Bàsiques

Aquestes activitats estan disenyades per ser usades com en un llenguatge de programació de codi:

- **Activity**: permet crear activitats pròpies.
- **Variable:** permet crear una variable i determinar o obtenir el seu valor (o des del menú Edició/*Edit*, seleccionant *Variables*).
- **Calculate**: porta a terme operacions aritmètiques simples o operacions lògiques, sobre una expressió donada (cal incloure valors numérics, el valor del missatge, variables o valors predeterminats).
- **Data**: subministra un valor de dada simple a una altra activitat o servei de forma directa, sense fer servir variables.
- **Join**: combina el flux de dos (o més) fluxes de dades. Tots els missatges han de ser rebuts en totes les connexions entrants abans de que l'activitat passi les dades. Funciona com un AND.
- **Merge**: combina el flux de dos (o més) fluxes de dades sense dependència condicional. Funciona com un OR.
- **If**: proveeix sortides per continuar el missatge entrant basant-se en una condició introduïda. És com un *if* clàssic en llenguatge de programació.
- **Switch**: es pot utilitzar per a direccionar missatges basant-se en si el missatge d'entrada quadra amb l'expressió introduïda a la caixa de text. Funciona com un switch clàssic.
- **List**: crea una llista d'elements de dades.
- **List Functions**: permet modificar les llistes existents.
- **Comment**: permet afegir blocs de text al diagrama. Funciona com els comentaris clàssics.

Es poden utilitzar els elements arrossegant i deixant anar al diagrama.

Al costat dret de la finestra es pot veure la **Properties Window**. Aquesta finestra mostra les propietats de l'element seleccionat. Això permet a l'usuari ajustar la configuració exposada pel servei o activitat.

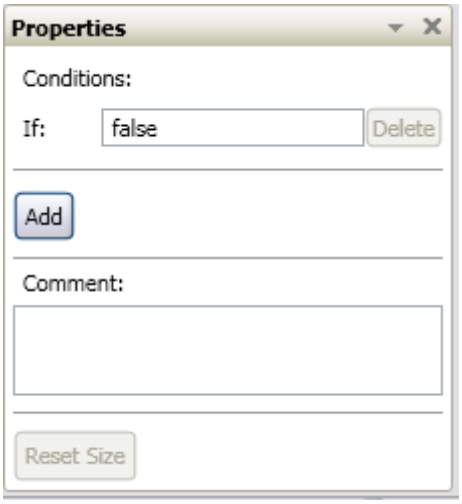

Figure 13. Propietats de l'activitat "If"

A la part esquerra es troba la **Service Toolbox**

Figure 14. Service toolbox

Aquesta caixa mostra els serveis que són disponibles a Visual Programming Language.

Quan s'arrossega un servei que ja existeix al diagrama, es pregunta a l'usuari si vol crear una nova instància del servei o vol crear una referència al servei ja creat. Per exemple es pot utilitzar el mateix servei per una sèrie de sensors o motors, però amb diferents configuracions.

La *Service Toolbox* disposa d'un buscador. Si escrius una paraula, els serveis mostrats seran els que continguin aquesta paraula. Es poden salvar els filtres de recerca polsant el botó +.

L'estat inicial dels serveis s'ha de configurar:

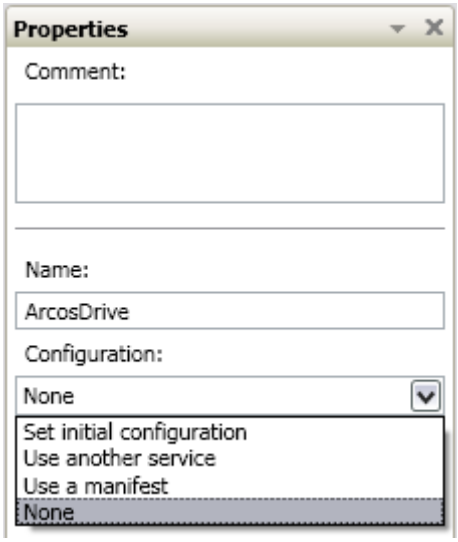

Figure 15. Configuració inicial

Quan es selecciona un servei, la finestra de propietats mostra les característiques inicials que es poden escollir. Depenent del servei, es pot seleccionar des d'una a quatre opcions com a configuració inicial:

- **Set initial configuration**: Aquí l'usuari defineix el valor inicial per al servei (instància) i també pot seleccionar quin partner service (serveis que li proveeixen suport) utilitza
- **Use another service**: Es pot seleccionar un servei específic. Això permet seleccionar els serveis genèrics al diagrama i canviar fàcilment el servei actual sense canviar el diagrama
- **Use a manifest**: Aquesta opció permet seleccionar un fitxer de manifest existent per a inicialitzar el servei. Un manifest és un arxiu especial que descriu els serveis que s'han d'inicialitzar i la seva configuració. La comanda "Import manifest" mostra una llista dels manifests disponibles que es poden escollir. Si es vol utilitzar un robot, s'ha d'escollir el seu manifest (tant si és real com simulat)
- **None**: Aquesta opció indica que la rutina DSS busqui o creï el servei apropiat.

A la part dreta hi ha una finestra anomenada **Project**. Aquesta finestra mostra els diagrames i els fitxers de configuració (per serveis) inclosos al projecte. Es poden afegir o esborrar diagrames del projecte des d'aquí.

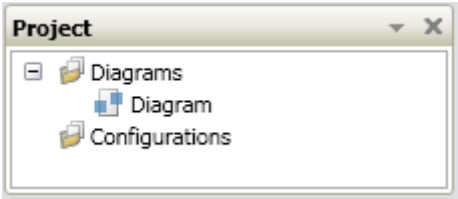

Figure 16. Finestra Project

A més es pot utilitzar un **Zoom** (tant per ampliar com per a reduir la imatge) sobre el diagrama. La barra per a fer-ho estan situats a baix a la dreta.

Visual Programming Language – Executar

Per a executar un projecte VPL s'ha de seleccionar *Start* al menú *Run*. Això iniciarà l'aplicació. Una caixa de diàleg apareixerà amb un enllaç a una traçada de debug així com un botó de Stop que s'utilitza per a parar el projecte i una sèrie de detalls:

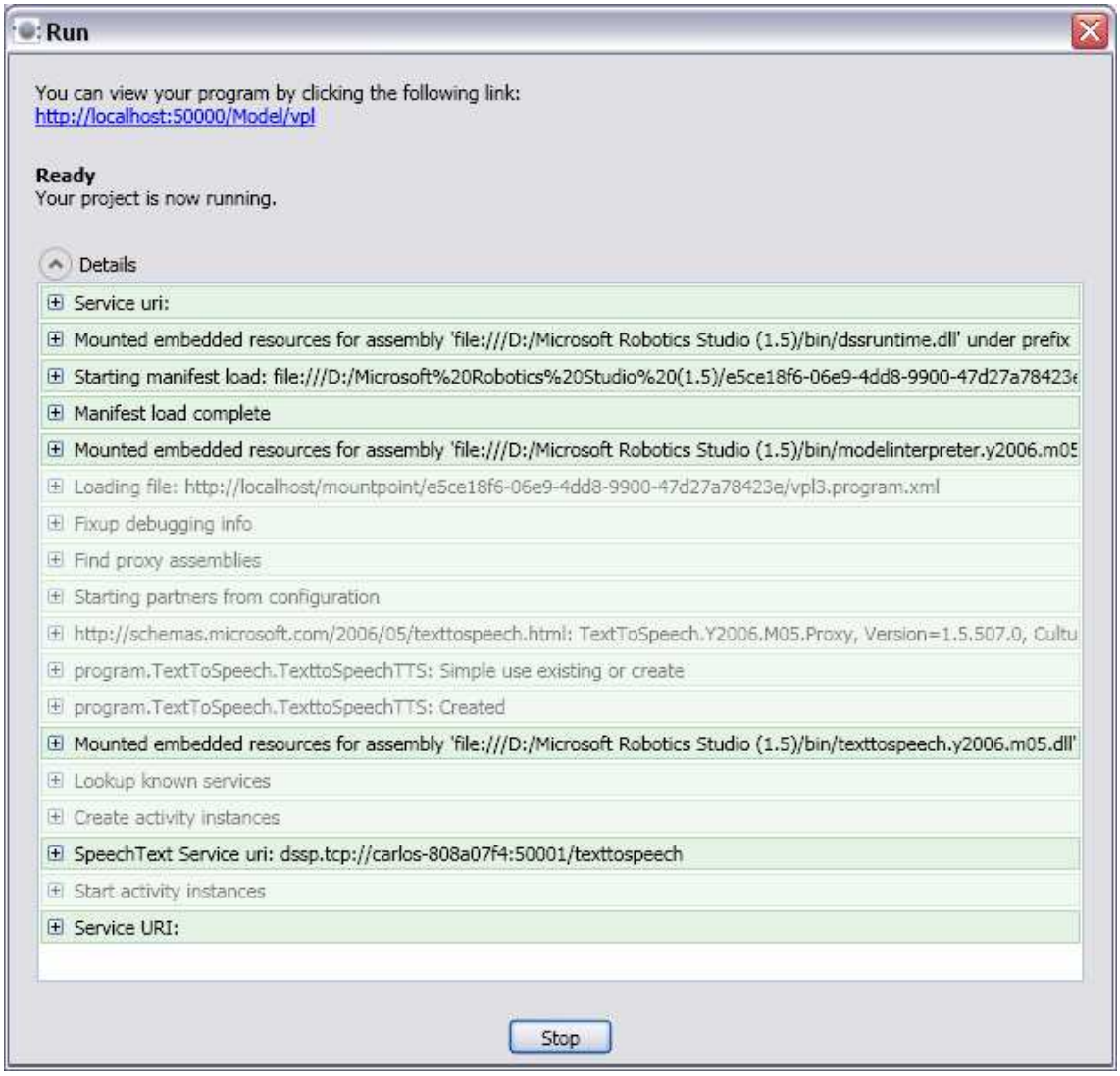

Figure 17. Executant el projecte

Es pot utilitzar la comanda *Debug Start* per a iniciar el projecte i parar-lo al seu primer bloc per a després permetre a l'usuari anar pas a pas. També es poden ocultar els detalls prement la fletxa cap a dalt al costat de la paraula "Details".

També es pot executar l'aplicació VPL independentment de l'entorn de desenvolupament VPL. Per a fer això s'ha de compilar com a servei: comanda *Compile As a Service* del menú *Build*. Un cop fet això es pot executar des de la consola de

comandes (*Command Prompt*) o clicant Run DSS *Node* al grup Microsoft Robotics Studio *(1.5)* dins el menú inici.

Visual Programming Language – Simulador

Si el projecte conté simulació, quan s'executi l'entorn de simulació (Visual Simulation Environment) s'executarà també:

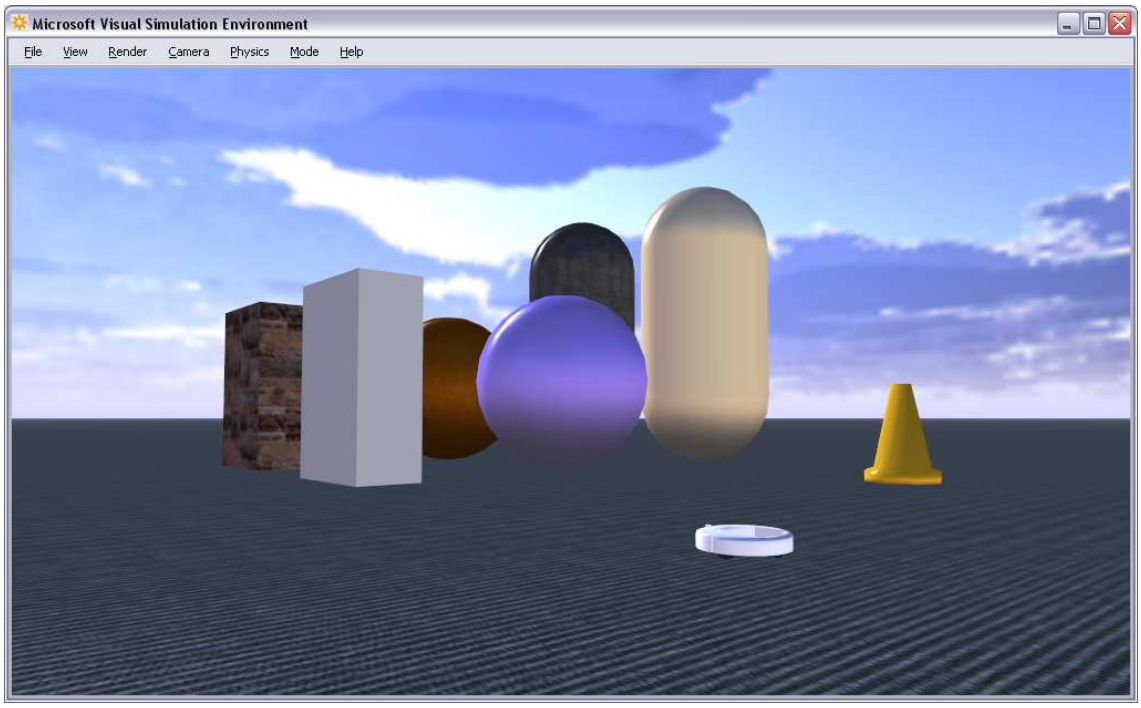

Figure 18. Visual Simulation Environment amb iRobot Create

Es pot moure la càmera amb el teclat i el ratolí prement les tecles w,a,s,d,q,e i clicant i movent el ratolí.

Es poden configurar una ampla varietat de característiques, com velocitat de moviment de la càmera, gravetat, qualitat del renderitzat o fins i tot activar la visió a través de càmeres dels robots.

III. Microsoft Robotics Studio Walk-through

Installation requisites

This guide has been developed with ***Microsoft Robotics Studio (MSRS) 1.5 Refresh***. Everything is license-free and can be downloaded from the Microsoft website.

Microsoft Robotics Studio 1.5 can be downloaded from here:

- http://www.microsoft.com/downloads/details.aspx?FamilyId=73092FF6-E37B-45C6-8E5E-C23D5D632B1E&displaylang=en

The application is not 100% compatible with other versions, so it's recommended to use the same version. As well, you have to install this **update packages**:

- *Runtime and Tools Update for Microsoft Robotics Studio (1.5):* http://www.microsoft.com/downloads/details.aspx?FamilyId=2F747403-7D94-43F0-836B-84247FEE66C6&displaylang=en
- *Samples Update for Microsoft Robotics Studio (1.5):* http://www.microsoft.com/downloads/details.aspx?FamilyId=7EEB9B70-0E86-4E3E-92AF-6148BDA34B7C&displaylang=en

MSRS requires *Ageia Physics Engine* and XNA and NET 3.0 runtimes, which are installed automatically. The XNA version including the package is 2.0, but the simulation environment could need the 1.0 version, so it's recommended to install it after the whole application. You can download it here:

- http://www.microsoft.com/downloads/details.aspx?FamilyId=A7DA4763-6807-4BD5-8D18-18C60C437F93&displaylang=en

If you have any other problem please search the MSDN forums for help:

- http://forums.microsoft.com/qawizard/ask.aspx?siteid=1

System description

Microsoft Robotics Studio (MSRS) consists of a set of applications. When you installed MSRS, it should have created a folder in your Start Menu containing a shortcut for "Robotics Studio Command Prompt". Open one of these windows now.

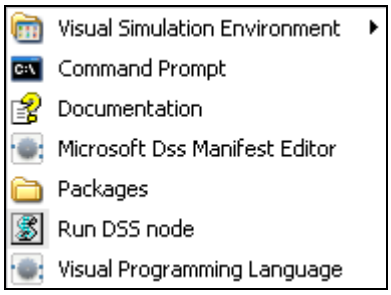

Figure 19. MSRS Program Contents

- **Visual Simulation Environmen**t: the simulation environment, it can be executed for a range of robots.
- **Command Prompt**: it opens a DOS Command Prompt. Many applications can be executed from this console, like starting a service.
- **Documentation:** the MSRS Official Documentation. This file contains a set of information such as user guides and tutorials of all the set of applications. Reading is highly recommended.
- **Microsoft Dss Manifest Editor**: it allows editing the manifests from a graphic user interface.
- **Packages**: all the update packages installed are here.
- **Run Dss Node**: it executes in the computer a DSS node as host.
- **Visual Programming Language**: it's the application we are interested in. It allows the user to program without a writing code.

Microsoft Robotics Studio Architecture

The primary task of robotics applications is to consume sensory input from a range of sources and orchestrate a set of actuators to respond to the sensory input in a manner that achieves the purpose of the application. As an example, the robotics application data-flow diagram shown below contains a simple bumper (sensor) that reports when it is hit, a message box (actuator) that controls the display, and an orchestrator that connects the pieces together.

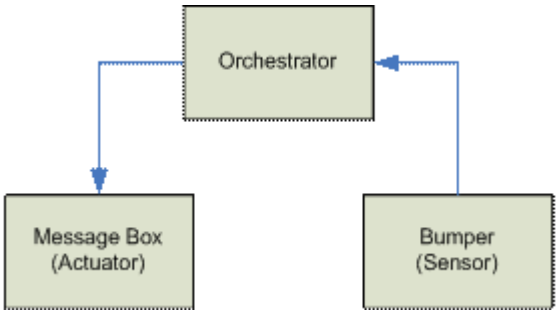

Figure 20. Example

The bases of the MSRS architecture are the runtimes: ***Concurrency and Coordination Runtime (CCR) and Decentralized Software Services (DSS).***

CCR is a managed code library (DLL) accessible from any language targeting the *.NET 2.0. Common Language Runtime*. The CCR addresses the need of service-oriented applications to manage asynchronous operations, deal with concurrency, exploit parallel hardware and deal with partial failure. It enables you to design your application so that its software modules or components can be loosely coupled; that is, so that they can be developed independently, and makes minimal assumptions about their runtime environment and other components.

The CCR is appropriate for an application model that separates components into pieces that can interact only through messages. Components in this model need means to coordinate between messages, deal with complex failure scenarios, and effectively deal with asynchronous programming.

DSS provides a lightweight, service oriented application model that combines key aspects of traditional Web-based architecture (commonly known as *REST*) with pieces of Web Services architecture. The application model defined by DSS builds on the *REST* model by exposing services through their state and a uniform set of operations over that state but extends the application model provided by *HTTP* by adding structured data manipulation, event notification, and service composition.

The primary goal of DSS is to promote simplicity, interoperability, and loose coupling. This makes it particularly suited for creating applications as compositions of services regardless of whether these services are running within the same node or across the network. The result is a highly flexible yet simple platform for writing a broad set of applications. DSS uses HTTP and DSSP as the foundation for interacting with services. DSSP is a lightweight SOAP-based protocol that provides support for manipulation of structured state and for an event model driven by changes to the structured state. DSSP is used for manipulating and subscribing to services and hence compliments HTTP in providing a simple, state-driven application model.

The DSS runtime is built on top of CCR and does not rely on any other components in Microsoft Robotics Studio. It provides a hosting environment for managing services and a set of infrastructure services that can be used for service creation, discovery, logging, debugging, monitoring, and security.

A **service** is the basic building block for writing applications using Microsoft Robotics Studio and is a key component of the DSS application model. Services can be used to represent anything including but not limited to:

- Hardware components such as sensors and actuators
- Software components such as User Interface, storage, directory services, etc.
- Aggregations: Sensor fusion, mash-ups, etc

Services are executed within the context of a DSS Node. A DSS node is a hosting environment that provides support for services to be created and managed until they are deleted or the DSS node is stopped. Services are inherently network enabled and can communicate with each other in a uniform manner regardless of whether they are executed within the same DSS node or across the network.

The DSS service model has been designed to facilitate reuse of services by making them easy to use and compose with each other while enforcing a very loose coupling between them.

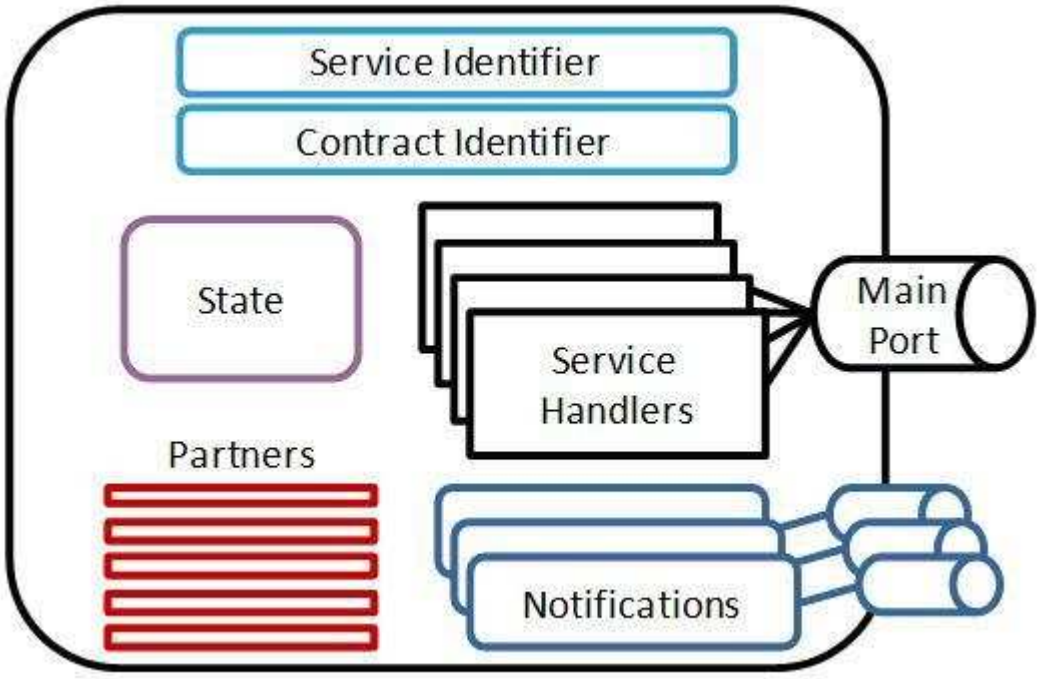

Figure 21. DSS Service Model

All DSS services consist of a common set of components:

- **Service Identifier:** The service identifier provides identity of a service instance and enables other services to communicate with that service as well as accessing DSS Services through a Web Browser.
- **Contract Identifier**: A contract is a condensed description of a service implementation that describes its behavior so that other services can compose and reuse services with a given contract. A contract identifier is a URI that uniquely identifies the contract of a service.

- **Service State**: The service state is a representation of a service at any given point in time. Any information that is to be retrieved, modified, or monitored as part of a DSS service must be expressed as part of the service state.
- **Service Partners**: A critical part of the DSS application model is to enable services to compose with each other to provide higher level functions. Partner services are other services that a service interacts with and possibly depends on in order to function properly. By declaring a set of other services as partners, a service can indicate to the runtime that it wants to be wired up with these services as part of the creation process of the service itself. A partner is declared using the *Partner* attribute.
- **Main Port**: The main port is a CCR port where messages from other services arrive (a.k.a. the operations port). A service can only talk to another service by sending a message to its main port. The main port itself is a private member of the service class and is identified by the *ServicePort* attribute.
- **Service Handlers:** For each of the DSSP operations defined on the main port, service handlers need to be registered to handle incoming messages arriving on that port. Service handlers can be registered declaratively using the *ServiceHandler* attribute.

Visual Programming Language - Basics

VPL is targeted for beginning programmers with a basic understanding of concepts like variables and logic. However, VPL is not limited to novices.

A Microsoft Visual Programming Language data-flow consists of a connected sequence of activities represented as blocks with inputs and outputs which can be connected to other **activity blocks**.

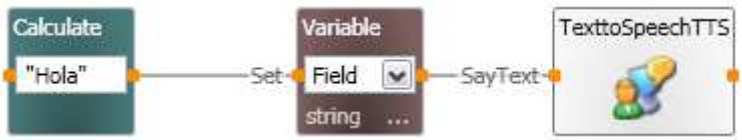

Figure 22. Block diagram with activities

Activities can represent pre-built services, data-flow control, functions, or other code modules. Activities can also include compositions of other of activities. This makes it possible to compose activities and reuse the composition as a building block. In this sense an application built in VPL is itself an activity.

Figure 23. Composed activity

Activity blocks typically include the activity's name and borders that represent its connection points. An activity block may also include graphics to illustrate the purpose

of the activity as well as user interface elements that may enable the user to enter values, assignments, or transformations for data used in an activity.

An activity receives messages containing data through its input connection pins. An activity's input pins are connection points to its predefined internal functions known as actions or handlers (which can be either as functions provided by a service or nested data-flows). A data-flow is executed from **left to right**.

Figure 24. Activity receiving a message through its connection points

An activity may have multiple input connection pins and each with its own set of output connection pins. Output connection pins can be one of two kinds: a *result* output or *notification* output (sometimes also referred to as an event or publication output). Result outputs are displayed as rectangular connection pins while publication outputs have round connection pins.

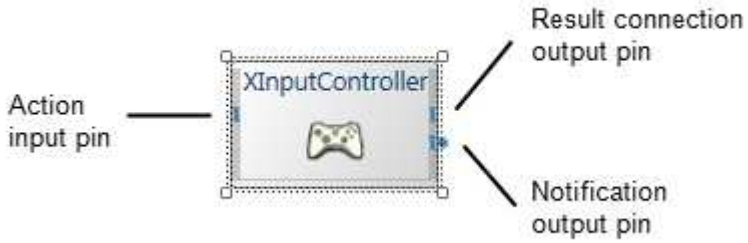

Figure 25. Connection pins

A response output pin is used in situations where an outgoing message (data) is sent as the result of a specific incoming action message. Notification pins can send information resulting from an incoming message, but more typically fire a message as a change their internal state.

Visual Programming Language – How to use

To connect two activity blocks, you drag from the connection pin of one activity to the pin of the other. If there are multiple ways (outputs and inputs) to connect the activities a Connections dialog box will appear so that you can define which of the inputs and outputs you want to connect.

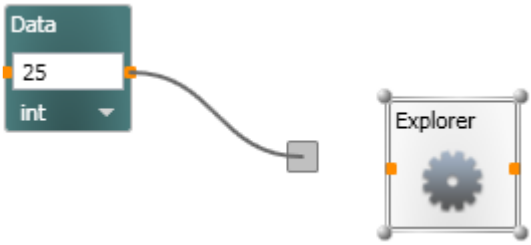

Figure 26. Connecting two blocks

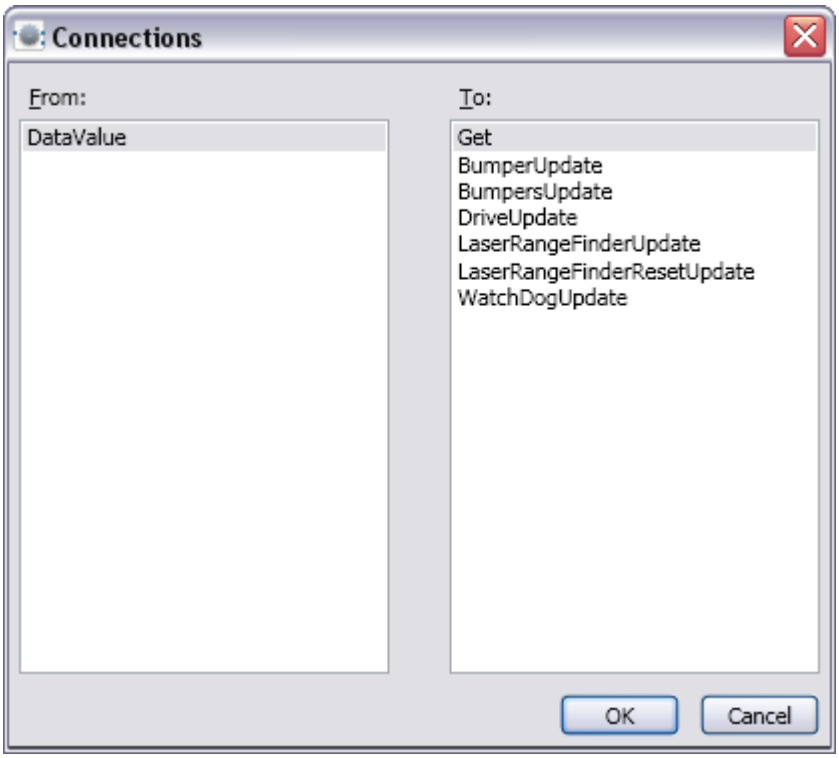

Figure 27. Possible inputs

Sometimes connections between activities also require matching up the data to be passed between the connections. When this occurs a Data Connections dialog appears and displays the message sender's data options on the left side and the receiver position on the right. Incoming data options can include the default value for its data type (0 for numeric types, false for Boolean types, and null for all other types), the value of the data, or sub-data (data member) values.

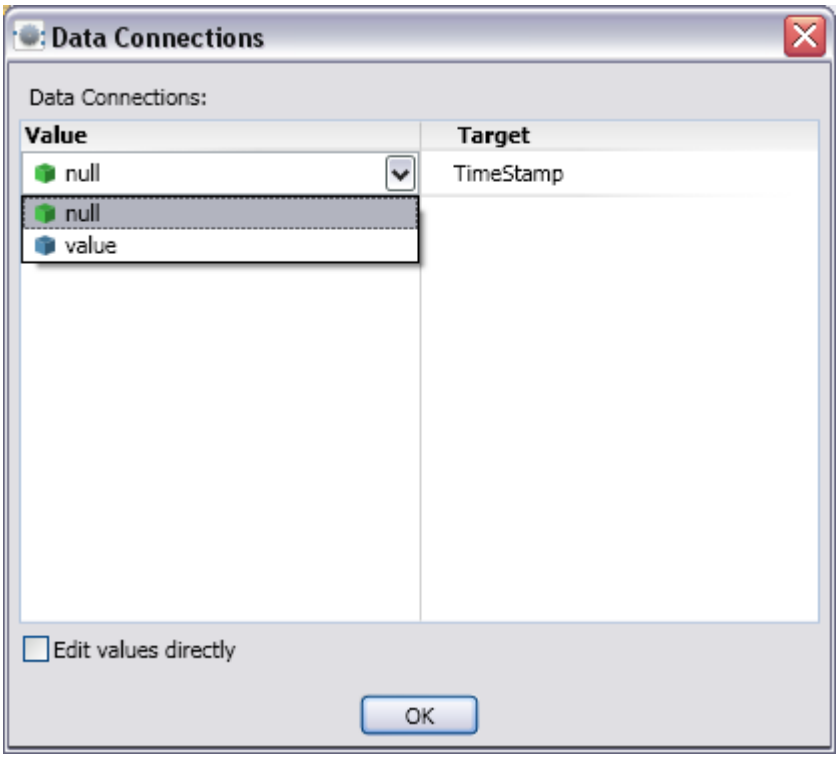

Figure 28. Matching data

If the data connection doesn't match you will be warned with .

In MSRS you can create a diagram:

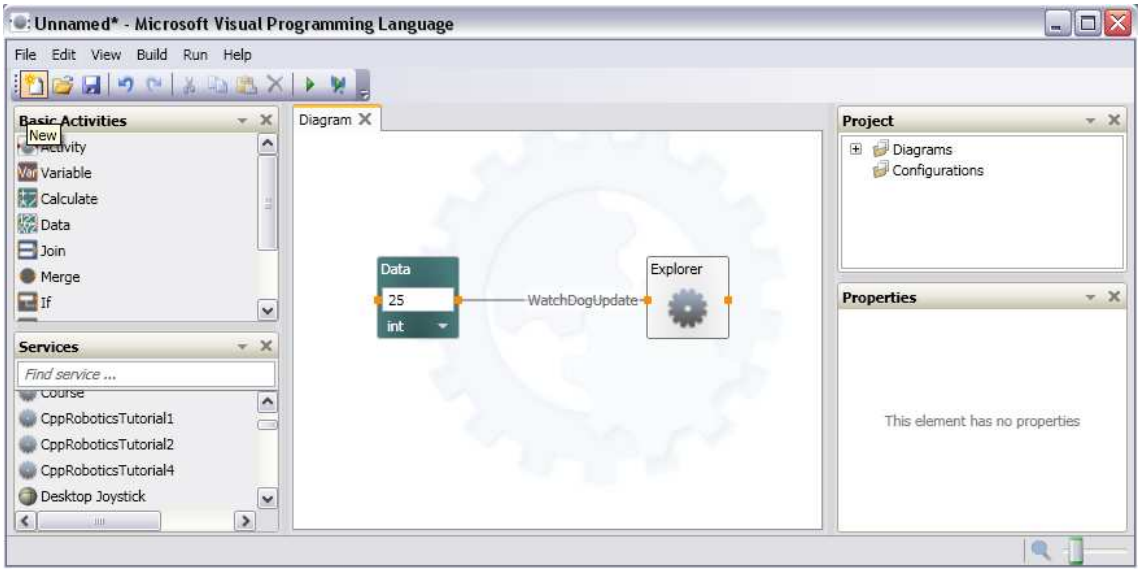

Figure 29. Creating a diagram

Or you can create an activity by dragging and dropping the basic activity "Activity" in the diagram (you will show another tab by clicking on the activity):

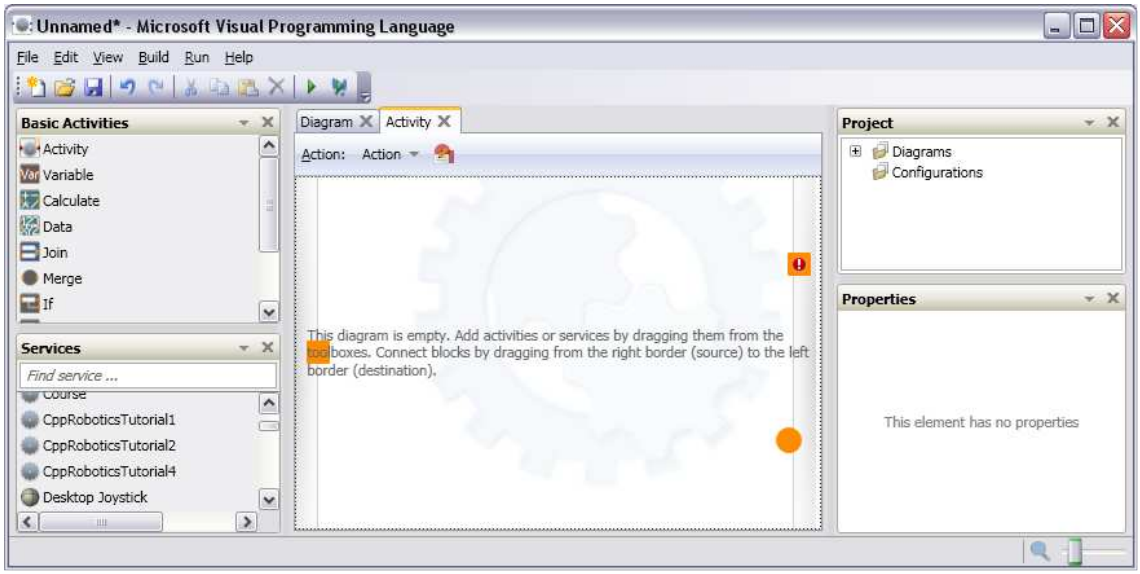

Figure 30. Creating an activity

Note that the activity has input and output pins. You can edit them from *Edit* menu by clicking *Actions and Notifications*.

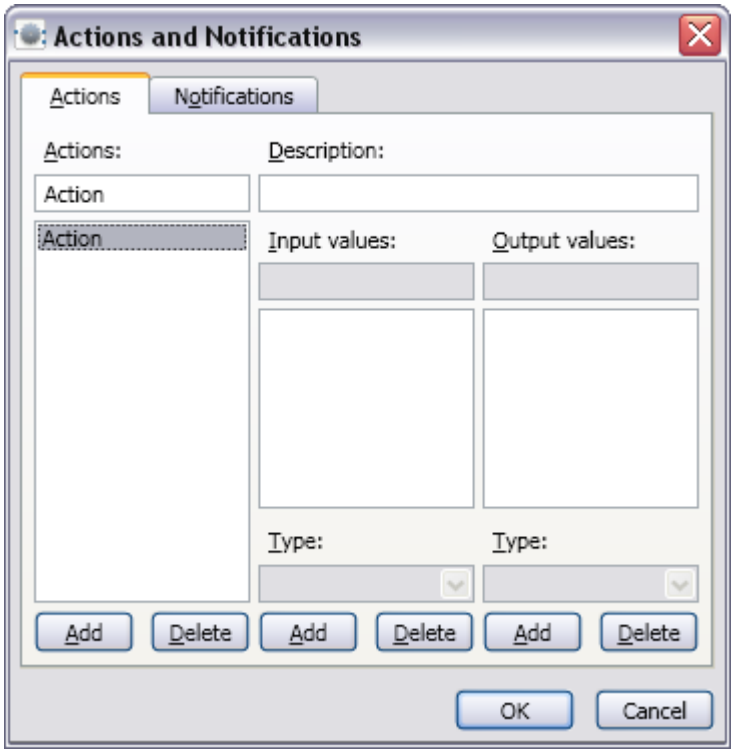

Figure 31. Actions and notifications

There are some toolboxes in a VPL window: *Basic Activities, Services, Project and Properties*. The toolboxes are movable and collapsible, so you can rearrange them within the main VPL window. In the *View* menu, you can hide or redisplay them.

To begin a dataflow diagram, drag and drop dataflow activity or service blocks from the toolboxes to the diagram area (tabbed page) and connect them. You can also add a new activity by double clicking its toolbox entry. Hovering over a toolbox entry with the pointer displays a tooltip that includes a description of how the activity may be used.

You can cut, copy, paste, or delete activities by selecting the block in the diagram using the commands on the Edit menu or the activities pop-up context menu.

At the left side you can see **Basic Activities toolbox**:

Figure 32. Basic Activities toolbox

These are the basic activities you can include in your diagram.

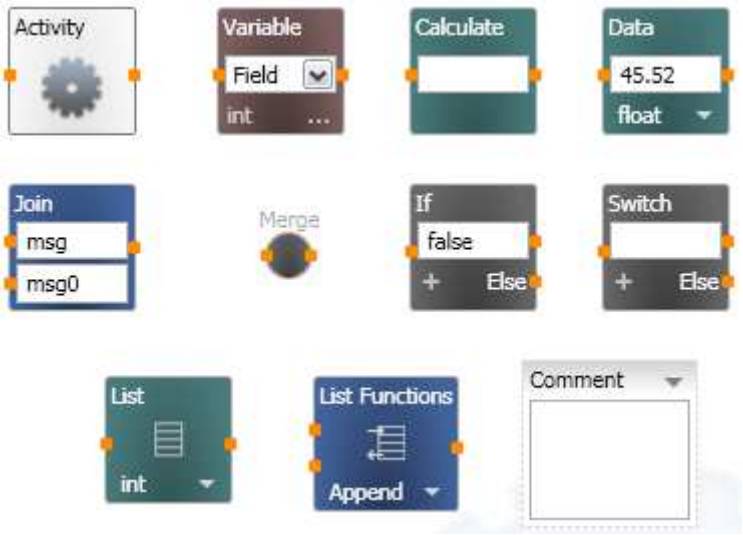

Figure 33. Basic Activities

These activities are designed to be used like in a code programming language:

- **Activity**: this activity block is used to enable you to create your own activities
- **Variable**: enables you to create a variable and set or get its value (or from the *Edit* menu selecting *Variables*)
- **Calculate**: performs simple arithmetic or logical operations on the expression entered (can include numeric values, the value of the message, its data members, or predetermined values)
- **Data**: used to supply a simple data value to another activity or service directly, without using variables
- **Join**: combines the flow of two (or more) data-flow. All messages must be received on all incoming connections before the activity passes on the data. It operates like an AND.
- **Merge**: simply merges the flow of two (or more) data-flows together without conditional dependency. It operates like an OR.
- **If**: provides a choice of outputs to forward the incoming message based on a condition you enter. It's like classical *If* in code programming languages.
- **Switch**: can be used to route messages based on the whether the incoming message matches the expression entered into the text box. Like the classical *Switch*.
- **List**: creates an empty list of data items.
- **List Functions**: enables you to modify an existing list.
- **Comment**: enables you to add a block of text to a diagram. Like comments in code programming language.

You can use activities by dragging and dropping in the diagram.

At the right side of the window you can see the **Properties window**. The Properties window displays the properties for the current selected item. This allows you to adjust the settings exposed by that service or activity.

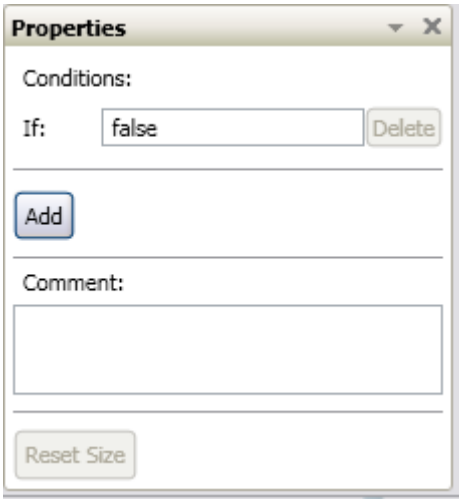

Figure 34. Properties of *If* activity

At the left side, there is the **Service Toolbox**:

Figure 35. Service toolbox

The Services toolbox displays services that are compatible with VPL.

When you drag a service from the toolbox that already exists in your diagram, you are asked whether you want to create a new instance of the service or you want to create a reference to the service already in your diagram. For example you might use the same service for a set of sensors or motors, but with different configuration settings.

There is a search filter. If you write a word, the services displayed in the toolbox are the services that contain this word. You can save the search filter by clicking the + button.

Services initial state settings need to be configured:

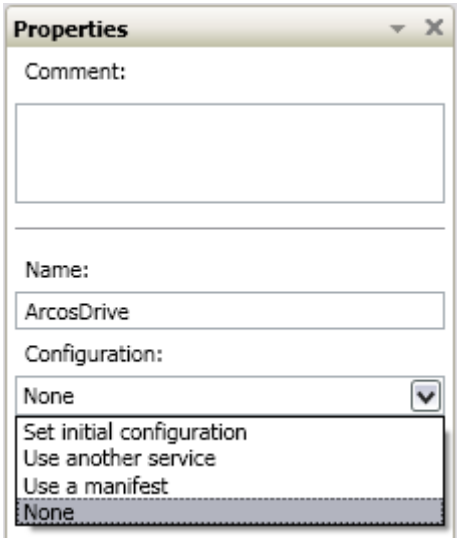

Figure 36. Starting Configuration

When you select the service, the properties window displays the initial settings that you can choose. Depending on the service, you may have one of four options to set the initial configuration of a service.

- **Set initial configuration**: Here you set the initial state values for the service (instance) and can also select what partner services (services that provide support for the service) the service uses.
- **Use another service**: You can select a specific service to use. This enables you to use generic services in your diagram and simply change the actual service used without changing your diagram.
- **Use a manifest**: This option enables you to select an existing manifest file to start the service. A manifest is a special file that describes the services to be started and their configuration. Use the Import Manifest command to display a list of existing manifests which can select. If you want to use a robot configuration, you have to choose his manifest (even simulated or real robot)
- **None**: This option indicates you want the DSS runtime to find or create an appropriate service.

In the right side there is a window called **Project**. This window shows the diagrams and configuration files (for services) included in your project. You can add or delete diagrams for this project from here.

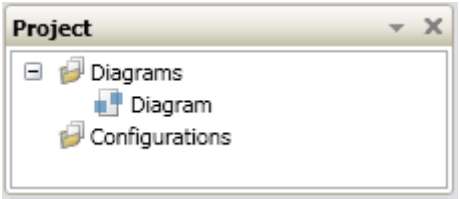

Figure 37. Project window

You can use *Zoom* on your diagram, it is located at the bottom of the window on the right side.

Visual Programming Language – Run

To run your VPL project, select *Start* from the *Run* menu. This will start your application. A dialog box appears with a hyperlink to a debug trace as well as a Stop button to stop your project and a set of details:

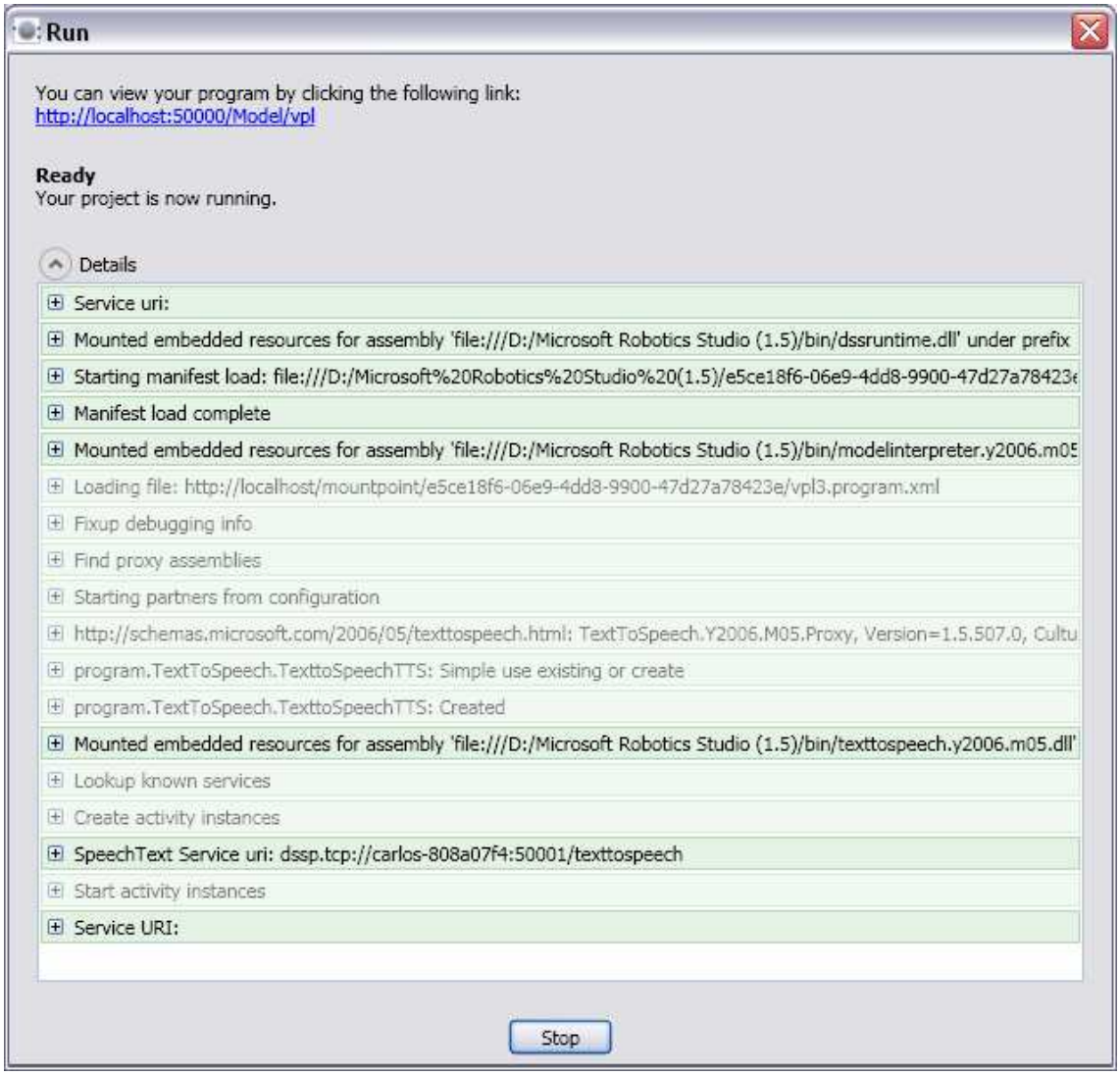

Figure 38. Running the project

You can use the *Debug Start* command to start your project and stop at its first block, and enable you to step through the program. You can hide the Details by clicking the up arrow.

You can also run your VPL application independently from the VPL development environment. To do this use the *Compile As a Service* command from the Build menu. You can run the service created from the ***Command Prompt*** or clicking ***Run*** DSS ***Node*** on the *Start Menu* MSRS group.

Visual Programming Language – Simulator

If the project contains simulation, when you run it the Visual Simulation Environment will be displayed:

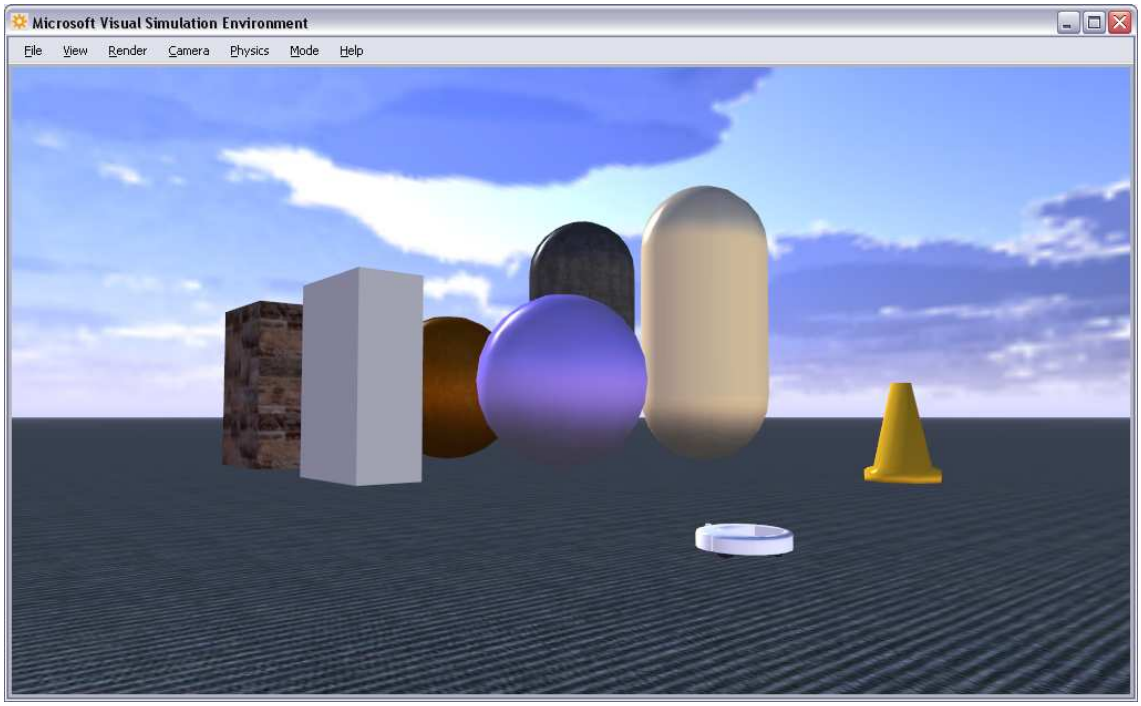

Figure 39. Visual Simulation Environment with iRobot Create

You can move the camera with the keyboard and the mouse by pushing the keys in the keyboard: *w,a,s,d,q,e* and clicking and moving the mouse.

You can configure a wide variety of settings, like the speed of the camera, the render quality or even see from a robot cam!

Walk-through

Introduction

This guide is divided in two blocks:

- Basic programming
- Robot programming

In basic programming you will learn how to program very simple routines with Visual Programming Language, like in C language or other programming languages.

In this tutorial you will learn how to use most of the basic activities to program an application and some services. The application will be a desktop joystick with buttons (0-9) and you will assign to some of them a function.

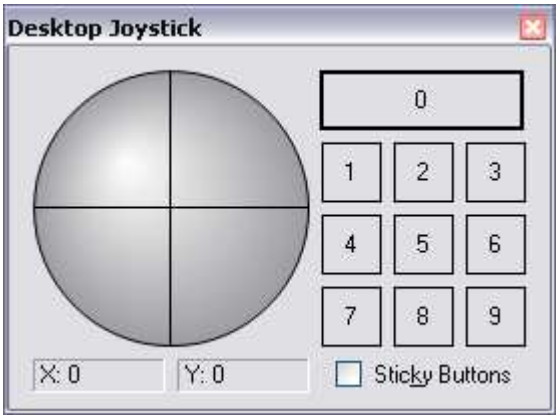

Figure 40. Desktop Joystick

The functions associated with the buttons are:

- A counter:
  - 0: reset the counter
  - 1: increment counter in one
  - 2: show a simple dialog with the counter value
- A simple dialog showing if some buttons are pressed or not pressed
- The computer will speak (text to speech service)

This tutorial introduces the user to the environment with many features but only in one practice.

In robot programming you will learn how to program and control a robot and simulate it. It's divided in three practices:

- Controlling a robot
- Moving a robot
- Sensing and acting

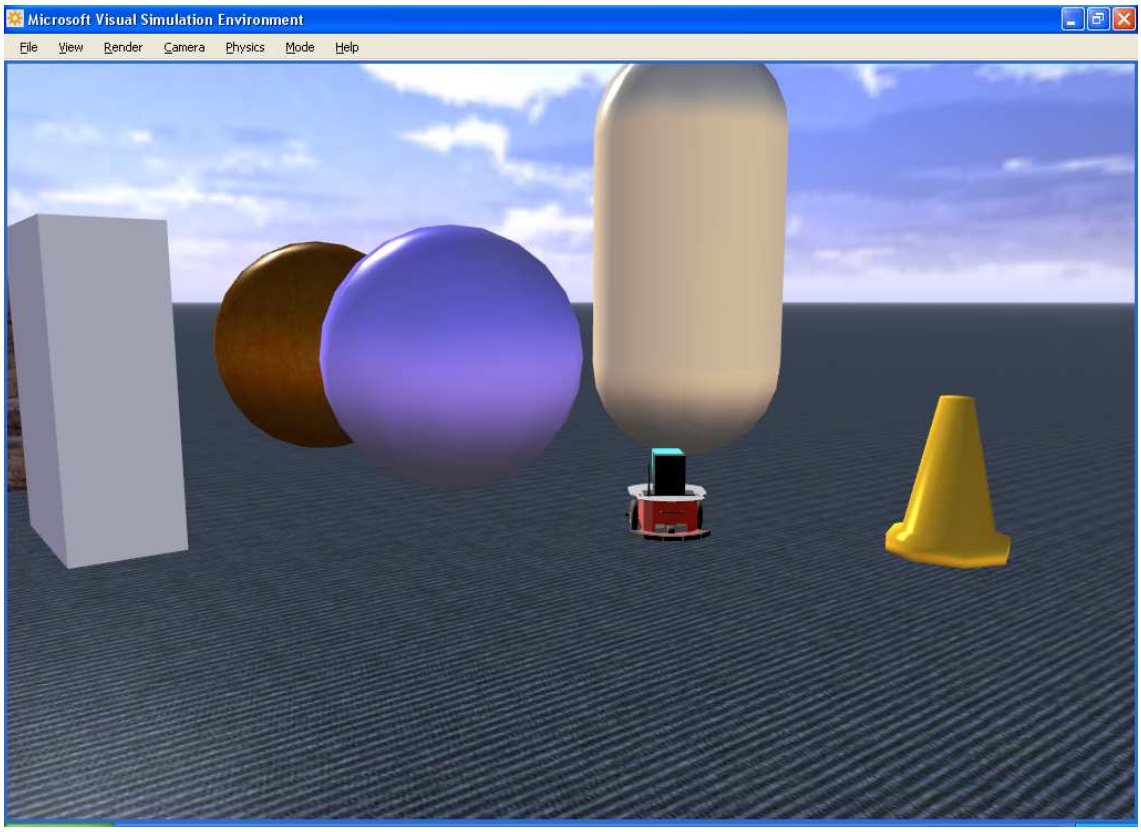

Figure 41. Robot simulation environment

In these tutorials you will learn how to control a robot with a desktop joystick (quite similar to the basic programming application), how to program a set of movements and how to access to its sensors and actuators to get data from them or to send orders.

Basic programming

To begin, create a new project:

1. File – New

An empty page will appear. Now you have to search the **Desktop Joystick** in the **Services list**. You have to install the samples update package to have the Desktop Joystick service.

1. Go to **services** and write in the box: **Desktop Joystick**
2. Drag to the diagram the **Desktop Joystick** service

Figure 42. Desktop Joystick icon

Now drag an **If activity**:

1. Go to **Basic Activities** and drag the **If** in the diagram

You have to connect the notification pin (the round one) from the **Desktop Joystick** to the **If** input pin. **Notification output** will send a message every time an event occurs with the joystick, we will use it when you push a button.

When you connect the pins a window will appear:

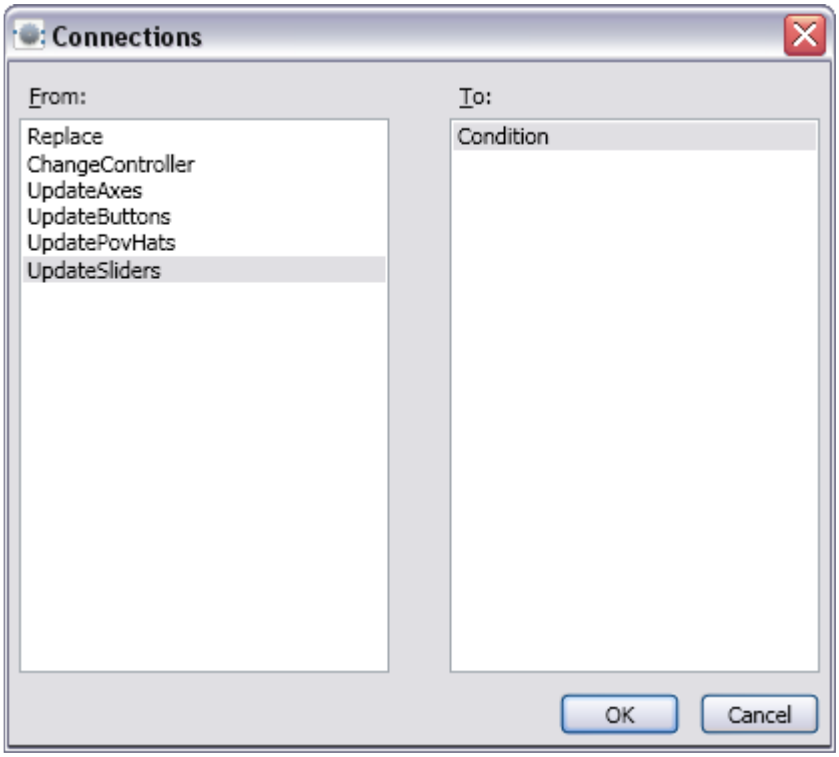

Figure 43. Connections

In the **From** side you have to select what will be sent by the output pin of the **Desktop Joystick**. In this case we are interested in the buttons, so select **Update Buttons**. The **If** input doesn't have options, only **Condition**. You should have a simple diagram like this:

Figure 44. Diagram

If you write in the **If** condition you will see a set of options as a condition. The only **Desktop Joystick** options are **Pressed** and **TimeStamp**.

We are interested in **Pressed**. It is a list (in fact, it works like an array in classical programming language) of Booleans that are associated to all the buttons. For example, to access button "5", you have to write Pressed[5].

We will add conditions to **If**:

1. Select the **If**
2. In the right side, in **Properties** you will see a condition.
3. Write: **Pressed[0] = true**
4. Now add more conditions by pressing **Add** button:
   - Pressed[1] = true
   - Pressed[2] = true
   - Pressed[8] = true
   - Pressed[9] = true

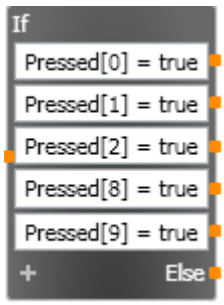

Figure 45. If activity

We can't use a switch because **Pressed** is not a variable, it is a list. Note that you can change the zoom of the view of the diagram in the right bottom of the window.

You will create the counter. The button 0 will reset the counter (pushing the 0 value to the variable), the button 1 will adds 1 (counter+1) and the button 2 will show a dialog with the counter value.

1. Drag a **Data** activity (don't change the value, we want a 0 integer)
2. Drag another **Data** activity, change the data type to **string** and write: Counter reset
3. Drag **Variable** activity and create an **integer** variable named "Counter" (you will access a window from where to define variables by clicking "…")
4. Add a **Simple Dialog** service

You already have all the necessary to make running a button. Connect the boxes:

1. Connect *"If pressed[0]"* to the first **Data** box (0 int)
2. Connect this **Data** box to the another one and the **Variable** box
   - Select **SetVariable**
3. Connect the **Data** box with the text "Counter reset" to the **Simple Dialog** service
   - Select **AlertDialog**
   - Select "**value**" in the next window

If you Run the program now the button 0 will have the desired behavior.

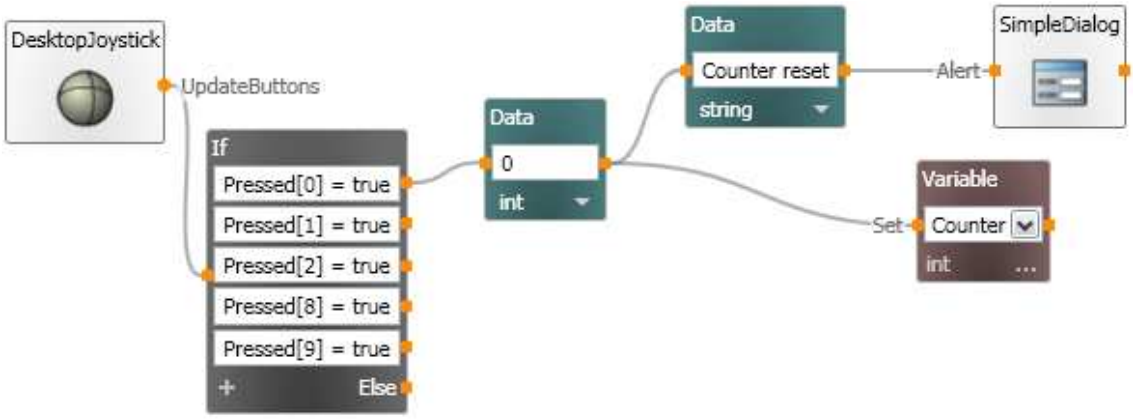

Figure 46. The diagram should look like this

The next button is *"If pressed[1]"*:

1. Copy and paste the **Variable** you used (note that it must contain the variable *Counter*) or drag a new **Variable** box and select *Counter*.
2. Connect the output pin of the **If box** (*If pressed[1]*) to the input pin of the **Variable**
   - Select **GetValue**.
3. Drag a **Calculate Activity** and connect it with the **Variable**.
4. Write in the **Calculate**: Counter + 1
5. Now place a **Merge Activity** on top of the line of the variable from the *If pressed[0]* sequence. It will split the line.
6. Connect the **Calculate** box with the **Merge** you have placed in step 5

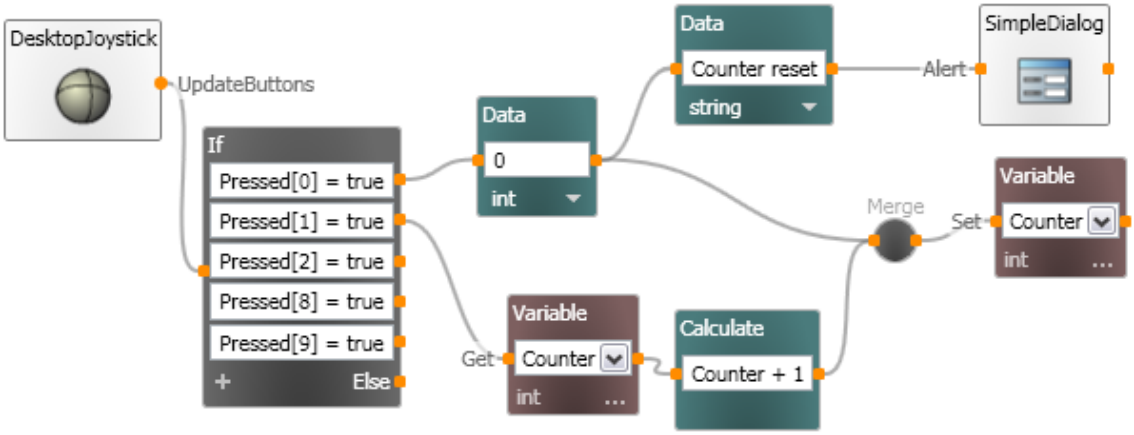

Figure 47. Diagram with 2 buttons

By pushing "2" this dialog will be displayed:

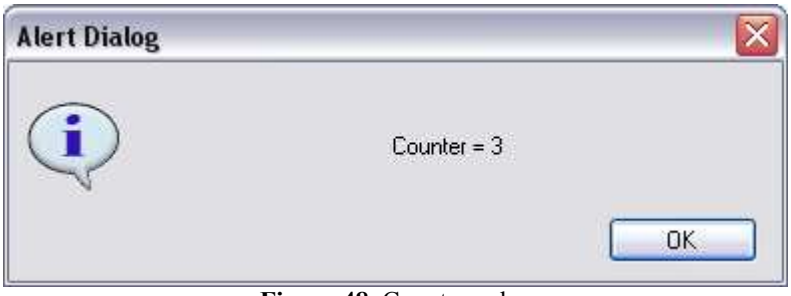

Figure 48. Counter value

Note that you can use zoom or scroll left or right. The next button is *"If pressed[2]":*

1. Copy and paste the **Variable** you used (note that it has to content the variable *Counter*) or drag a new **Variable** box and select *Counter*.
2. Connect the **If** box output for button 2 with the **Variable**
   - Select **GetValue**.
3. Drag a **Calculate Activity** and connect it with the **Variable**.
4. Write in the **Calculate** the whole line(don't copy it, it won't work)**:** "Counter = " + Counter *(it works quite similar with printf in C)*
5. Now place a **Merge Activity** on top of the line that is connected to the **Simple Dialog** activity.
6. Connect the **Calculate** with that **Merge** (which is connected to **Simple Dialog**)
7. **Run**

When you run an application you have to save it. Save whenever you want, with the name you wish. The counter application is now **complete**:

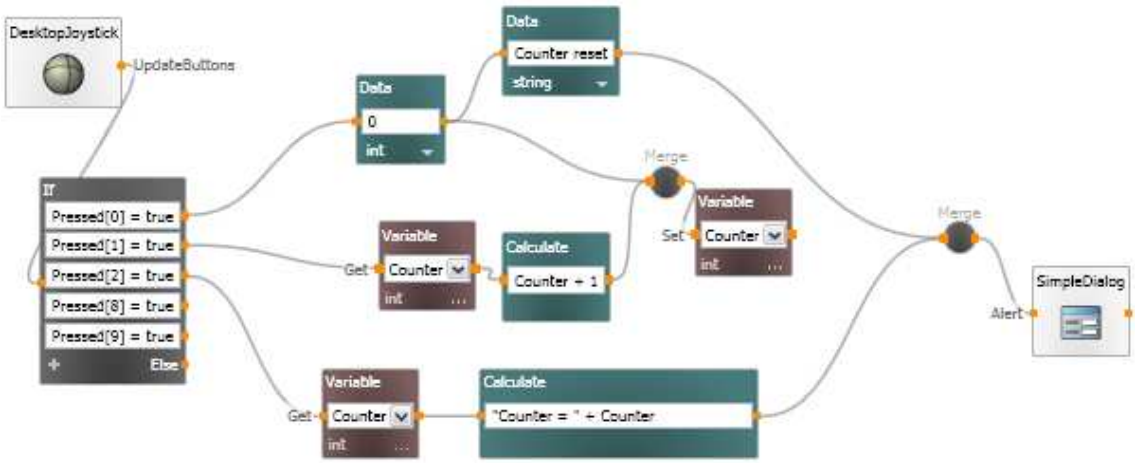

Figure 49. Counter application

It will count the number of times you pushed 1. You can reset this value by clicking 0 and show it by clicking 2.

Now you will add some new features to the application (button 8):

1. Drag two **Calculate** activities to the diagram
2. Connect the *If pressed[8]* to both
3. Write in the boxes the whole line(don't copy it, it won't work):
   - "Button 8 state: " + Pressed[8] + "\n"
   - "Button 9 state: " + Pressed[9] + "\n"
4. Drag a **Join** activity and connect each button with each message.
5. Place a new **Calculate** activity and write: msg + msg0
6. Connect the **Join** output with this **Calculate** input
7. Connect the **Calculate** with the **Merge** connected to the **Simple Dialog** you have created before (in counter application).

The last button that you will use (9):

1. Add a **Data** activity to the diagram and connect button 9 to it.
2. Select **string** and write: You have pushed button 9

3. Drag a **TextToSpeech** service. Search for it in the Service Toolbox "Find service" box. This service is used to talk by the computer speakers.
4. Connect the **Data** activity to the **service**:
    - Select **SayText** in the first formulary
    - Select **Value** in the last one

Now you can **Run** it. If you get a message asking whether to unblock the application or not, select Unblock. The diagram should look like this:

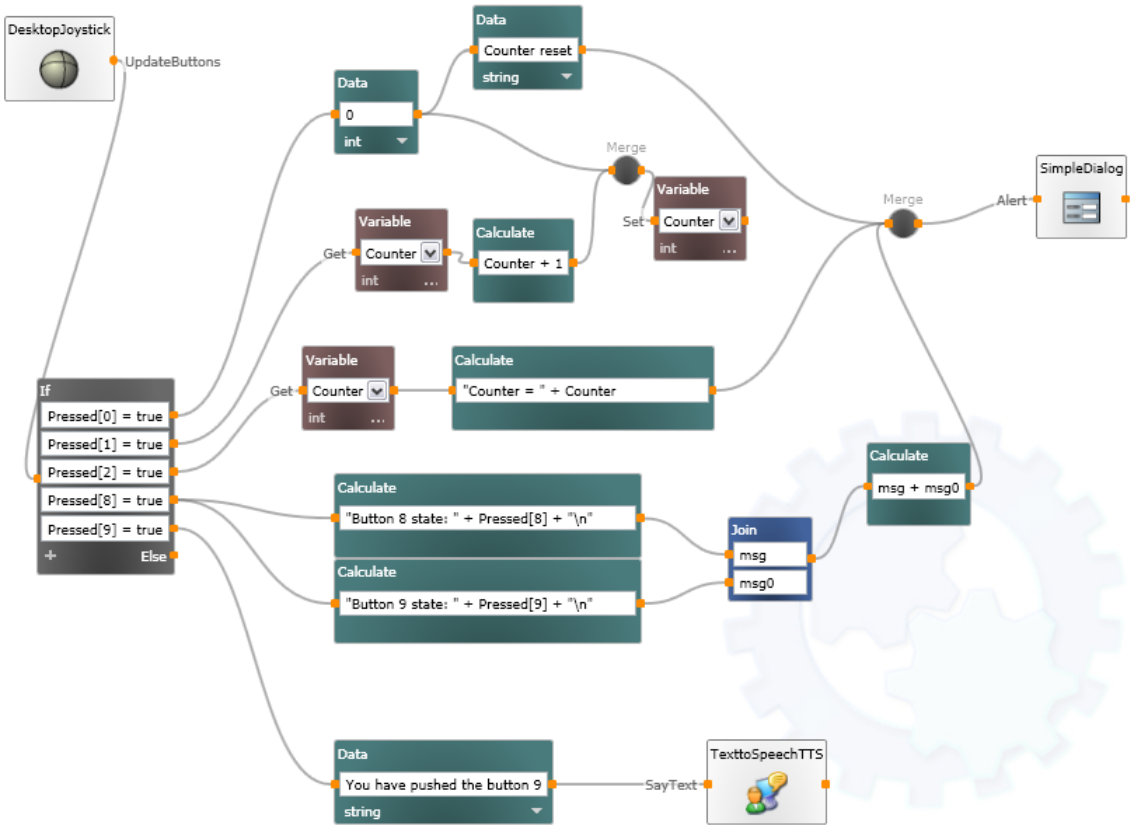

Figure 50. Basic Programming Diagram

You have finished **Basic Programming Tutorial**! The functions associated with the buttons are:

- [0-2] Counter application:
    - 0: resets the counter
    - 1: increments the counter in one
    - 2: shows a simple dialog with the counter value
- [8] A simple dialog that shows if buttons 8 and 9 are pressed or not
- [9] The computer will speak (text to speech service)

Robot programming

This tutorial is designed to be executed in several steps. First, you will create a diagram with a robot and you will control it. Secondly, you will create a robot that moves on its own and finally you will create a robot with sensors.

Controlling a robot

1. File-New
2. Add the service: **DirectionDialog**

Direction Dialog is a virtual controller (similar to Desktop Joystick) with only four directions and a *Stop* button.

Figure 51. Direction Dialog

With that dialog you will be able to control a robot to go forwards, backwards or stop it. Next:

1. Drag a **Calculate** activity
2. Connect the **notification** pin of **DirectionDialog** to the **Calculate** input
    a. Select **ButtonPress**
3. In the **Calculate** box write Name

Name is the variable containing the name of the button pressed. The names are:

1. Forwards
2. Backwards
3. Stop
4. Left
5. Right

We will use a **Switch**:

1. Add **Switch** activity
2. Add the **cases**:
    a. "Forwards"
    b. "Backwards"
    c. "Stop"
3. Connect **Calculate** box with the switch
4. Create 3 **Data** activities
5. Write in them:

   a. 0.5
   b. -0.5
   c. 0
6. Connect the 3 cases with the 3 diagrams
   a. “Forwards”_ 0.5
   b. “Backwards” _ -0.5
   c. “Stop” _ 0

These are the values for the robot movement. Now you will add a Generic Differential Drive service; it is called generic because it supports common operations available on most differential drives, but is not associated by default with a specific drive system. One of the benefits of Robotics Studio is that you can write general programs which will run on a variety of robots.

1. Add a **Merge** activity
2. Connect the 3 **Data** boxes to it
3. Add **Generic Differential Drive** service
4. Connect the merge to it
   a. Select **SetDriveSpeed**
   b. Select **Value** in both cases

The diagram should be like this:

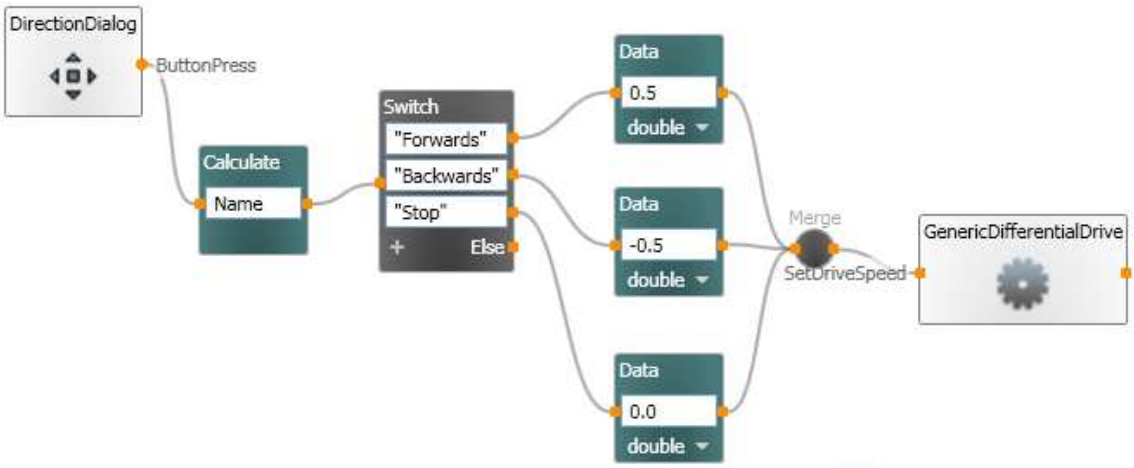

Figure 52. Control a robot diagram

Generic Differential Drive is used to control robots that commonly have wheels. You can control left wheels and right wheels separately. However, in this case, by introducing the same value for both wheels the robot will move in one direction. You must configure the drive:

1. Select **Generic Differential Drive**
2. In the properties toolbox go to **Configuration** and select **Use a manifest**
3. Click on **Import**
4. Select **MobileRobots.P3DX.Simulation.manifest.xml**

To run in **simulation environment** you must select a manifest that could be simulated. This robot has a camera that can be used in the simulation environment, so you will

select it instead of another one. Usually the manifests you can simulate have the word "Simulation" on their names. Let's see it:

1. Click on **Run** or press **F5** (if you have some problem please refer to **Installation requisites)**.

Now you should be able to see that:

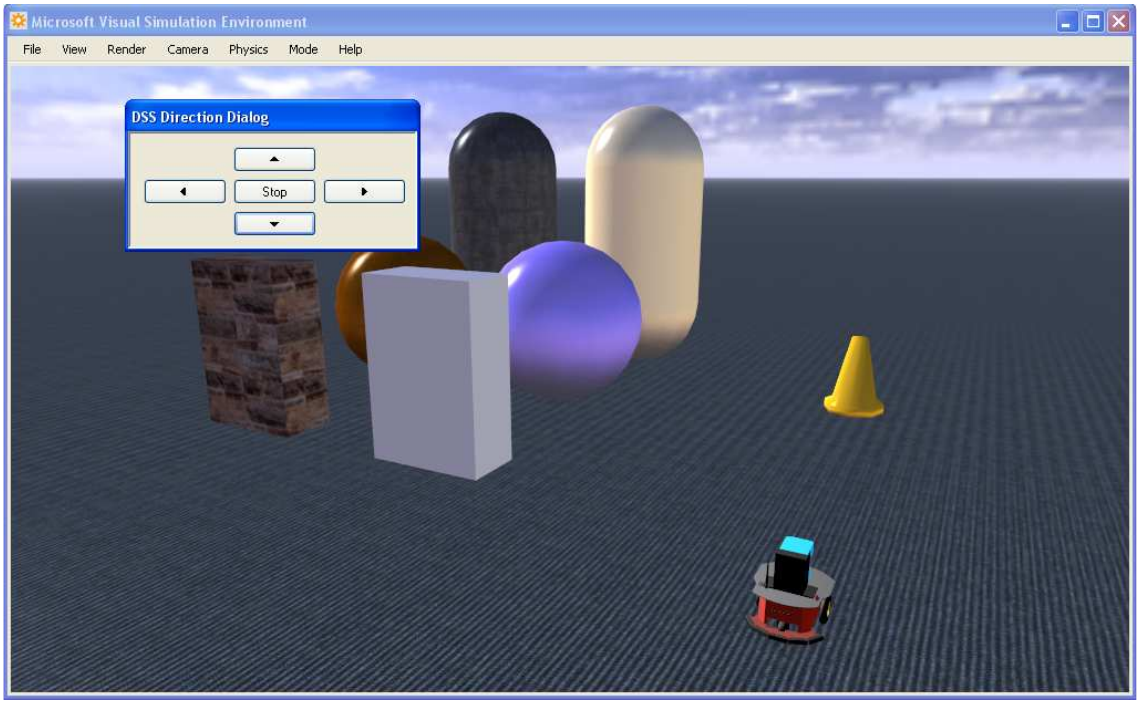

Figure 53. Simulation environment

If you click the buttons *forwards* or *backwards* the robot will move in that direction and if you click the *Stop* button it will stop its movement.

You can move around the simulation world with the mouse and the keyboard. Keyboard: arrows or: a,s,d,q,w,e.

The simulation environment has multiple options. Let's configure some of them:

1. Click on **Render** menu
2. Click on **Graphics Settings**

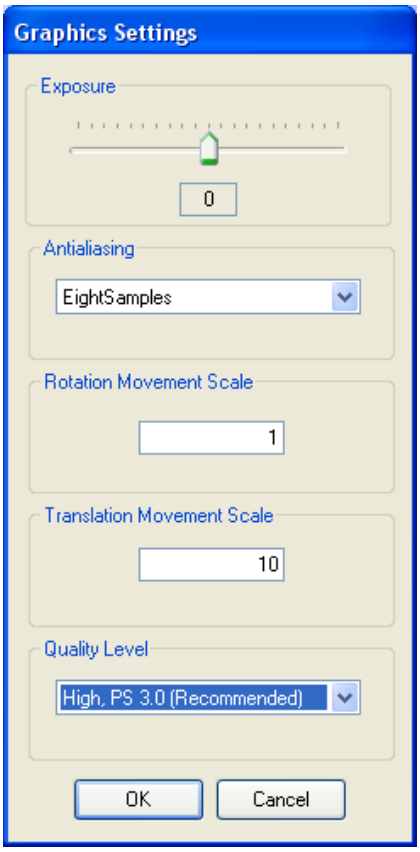

Figure 54. Graphics Settings

In this menu you can configure some graphic properties related with graphics' quality. Configure **Antialising** and **Quality Level** as your computer permits. Rotation and Translation are settings of movement. Try some values and choose the most comfortable for you.

In the **Camera** menu you can select **MainCamera** or **RoboCam**. If you select the latter you will be able to see what the robot is seeing through its integrated cam.

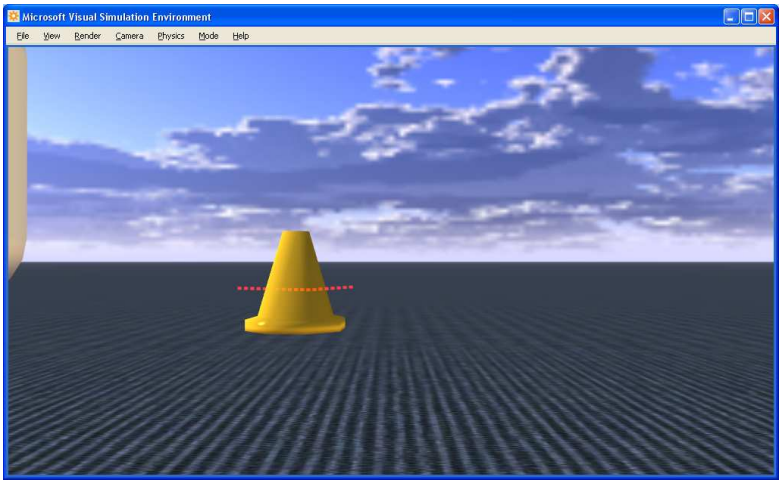

Figure 55. RoboCam

Phase one (controlling a robot) is complete. Now you will make a robot move on its own.

Moving a robot

In this practice you will create a robot that moves on its own. It will have some movements and a timer. The timer will control the time that every movement is executed. When a movement finishes it starts next one.

Start a new project.

1. File -> New
2. Drag two **Data** activities
3. Drag a **Variable** activity
4. Drag a **Timer** service

Firstly, you will set a starting value to Timer and variable. In this tutorial you will use a Timer to control the robot movement. The timer value is set in milliseconds (data integer), and it must be initialized.

1. Set a **Data** activity to 1500 integer and let the another to be a 0 integer
2. Create an **integer** variable and name it Case
3. Connect the **Data** activity (1500) to the **Timer**
    a. Select **SetTimer**
4. Connect the another **Data** activity to the **Variable**

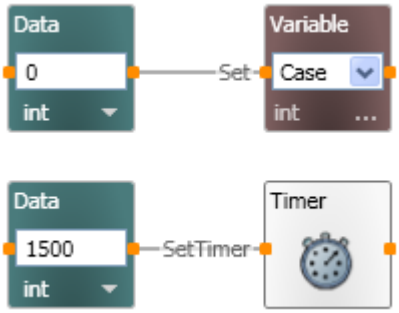

Figure 56. Initializing values

Now you will create the application.

1. Copy and paste the **Timer** (it must be the same timer, another instance, not a new one)
2. Copy and paste the **variable Case**
3. Connect them from the notification pin to the input pin
    a. Set **TimerComplete**
    b. Set **GetValue**
4. Drag a **Calculate** activity and connect the **variable** to it
5. Write in the calculate box: Case (the name of the variable)
6. Drag a **Switch** with the values 0,1 and 2
7. Connect the **Calculate** box to it

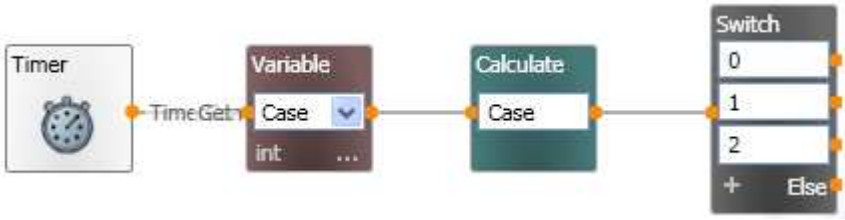

Figure 57. The resulting diagram

When the timer is complete the next iteration starts. The switch has 4 cases:

1. Move forwards
2. Move backwards
3. Move turning left or right
4. Restart the loop

Every new iteration will reset the timer and will add 1 to the case variable. You will program this now, and then all the cases must be connected to this figure:

1. Drag two **Variables** (Case variable, you can copy the instance)
2. Drag a **Calculate** activity
3. Connect the first **Case** to the **Calculate**
4. Write: Case + 1
5. Connect the **Calculate** to the another **Variable**
    a. Select **SetValue**
6. Create a **Data** integer with the value: 2000
7. Create another instance of the same **Timer** you have created before
8. Connect the **Data** activity to it
    a. Select **SetTimer**
9. Drag a **Merge** activity
10. Connect **it** to the two flows
    a. Connect it to the first **Variable**
        i. Select **GetValue**
    b. Connect the merge to the **Data** (2000)

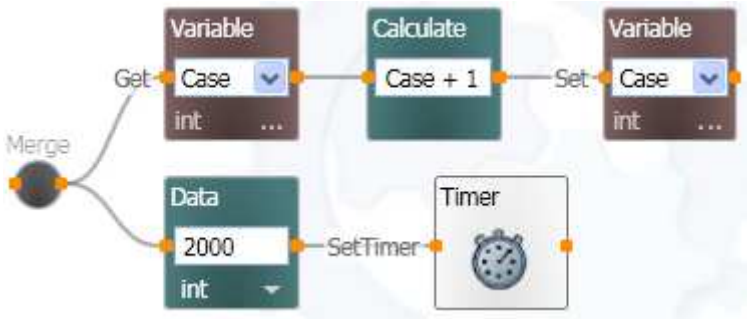

Figure 58. New iteration requisites

The three cases of the switch will be connected to this figure to make the application advance. The "Else" case must reset the application (calculate=0).

1. Copy the entire figure (Figure 58)
2. Connect **Case "else"** to the Merge activity
3. Change the **Calculate** activity with a **Data** activity with the value 0 integer

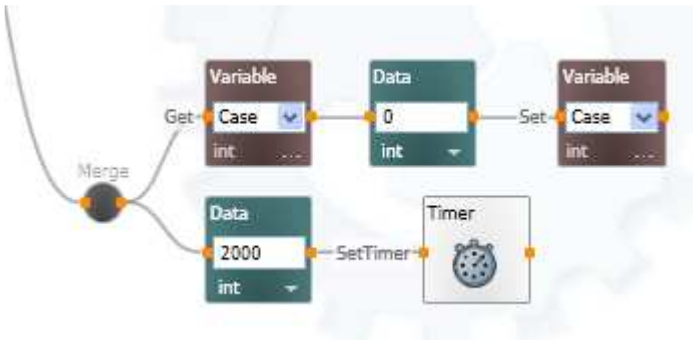

Figure 59. "Else" diagram

Now you will program the robot movement:

1. Drag a **Data** activity: set as **Double** with value 1.0
2. Drag a **Generic Differential Drive**
    a. Configuration: **Use a Manifest**
    b. Select: **MobileRobots.P3DX.Simulation.manifest.xml**
3. Connect the switch **case 0** to the **Data** activity
4. Connect the **Data** activity to the **Generic Differential Drive**
    a. Select **SetDrivePower**
5. Connect the switch **case 0** to the **Merge** you have created before

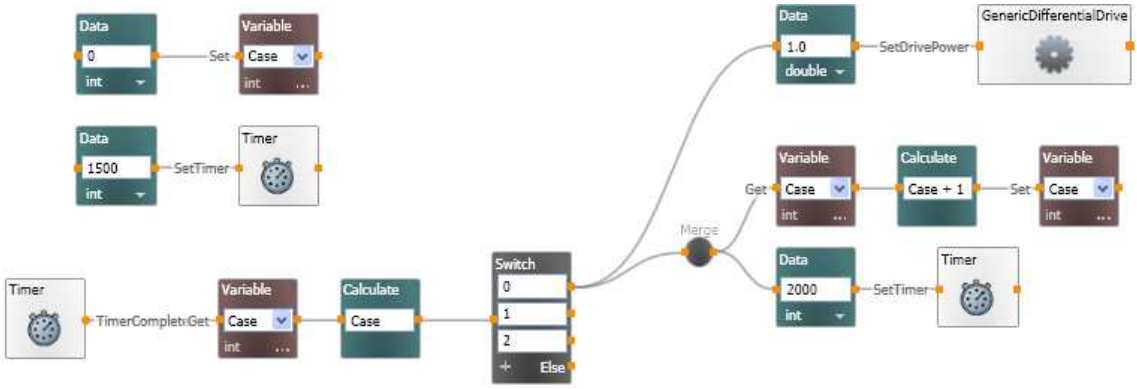

Figure 60. Resulting diagram

This is the diagram with the first case working. The second case is very easy:

1. Connect **Case 1** to the **Merge**
2. Drag a **Data** with value -1.0
3. Copy the **GenericDifferentialDrive** service (a new instance of the same one) and connect the **Data** box to it
    a. Select **SetDrivePower**
4. Connect **Case 1** to this **Data** box

Note: you can copy the **Case 1** diagram and change only the attributes.

You will now add the **Case 2** control program. You will use a new service: **TurningRadiusToWheelPowers**. You will use this service with the Generic Differential Drive. This service is used to move the robot in circles. You can select the side you want to move and the radius.

1. Connect **Case 2** to the **Merge (**controlling loops)
2. Drag two **Data** activities with the values:

   a. 6.0 Double
   b. False Bool
3. Connect **Case 2** to both
4. Drag a **Join** activity and name its messages as:
   a. TurningRadius
   b. TurnRight
5. Connect them:
   a. 6.0 Double - TurningRadius
   b. False Bool - TurnRight
6. Drag a **TurningRadiusToWheelPowers** service
7. Connect the **Join** output to it
   a. Select **CalculateWheelPowers**
   b. Select values with the same name
8. Create another instance of the **GenericDifferentialDrive** (by dragging the service or by copying another one)
9. Connect **TurningRadiusToWheelPowers** output to it
   a. Select **CalculateWheelPowers – Success**
   b. Select **SetDrivePower**
   c. Select **Left** for **LeftWheelPower** and do the same for the Right one
10. **Run**!

You can see the robot moving on its own and doing the same movement all the time. The diagram should be like this:

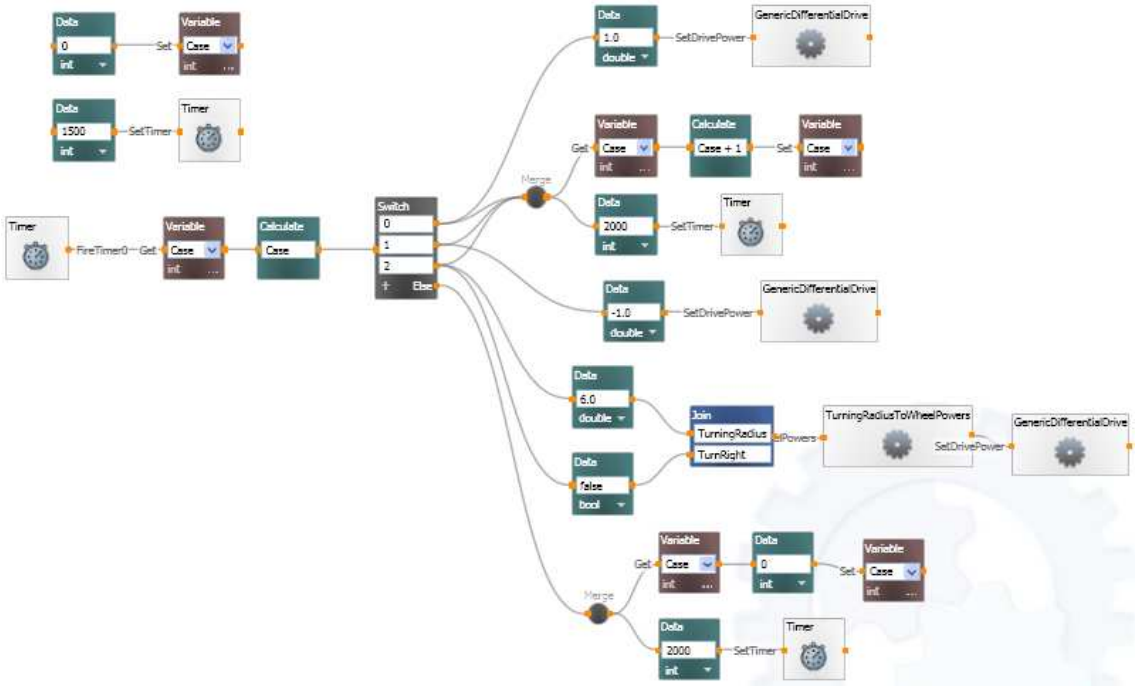

Figure 61. Final diagram of second robot

The robot will crash with the objects and will throw them to the ground:

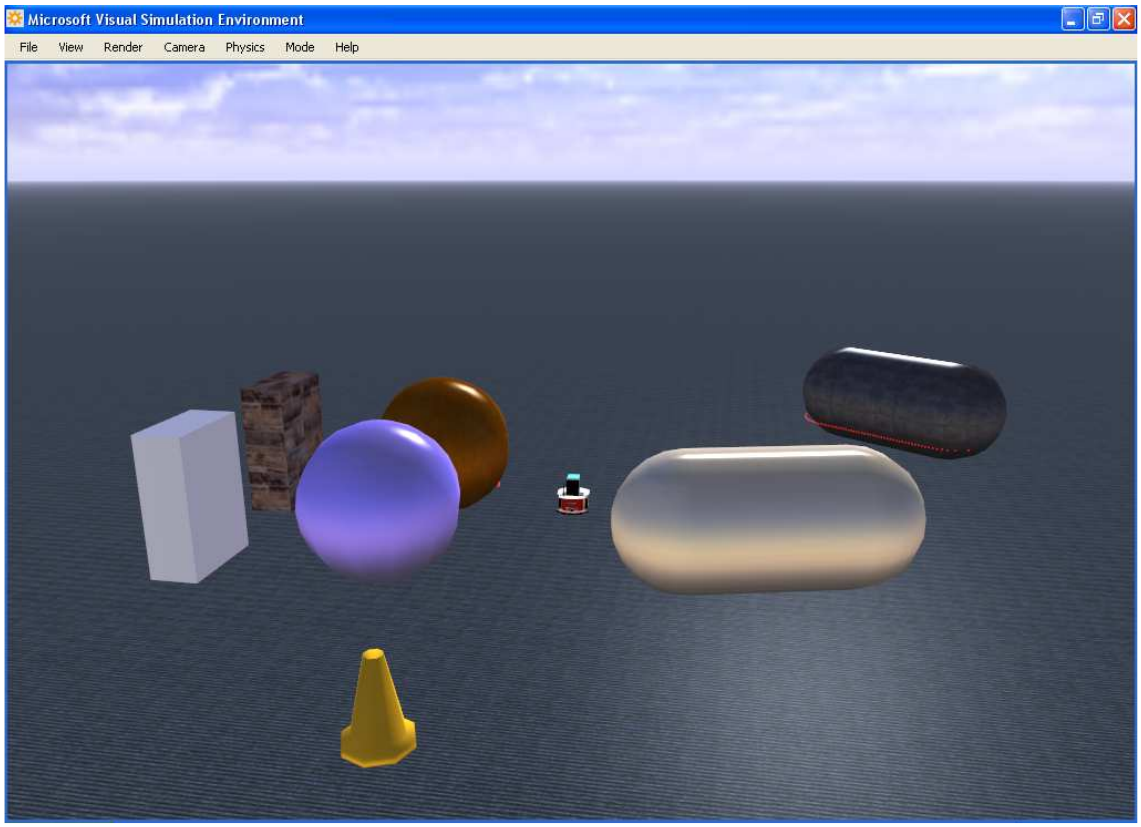

Figure 62. Robot moving itself and throwing objects

Sensing and acting

In this last practice you will interact with an *iRobot Create* and his sensors and actuators.

Figure 63. iRobot Create

It has sensors to detect walls and objects and a "music" player with some "songs". These songs are beep sounds of the computer. This is a very simple tutorial to know how to interact with the robot.

You will create two flows: one to move forwards and one that works like an interruption (active when sensor is updated) that makes the robot turn.

1. File->New
2. Create a **Data** integer with value 2000 and a **Timer**
3. Connect them

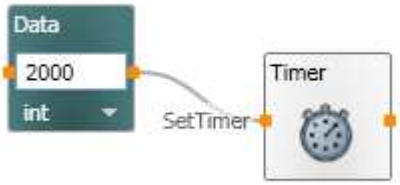

Figure 64. Initialization of the variable

This is the initialization of the timer. You will create the robot behavior now. First you will program the forwards movement:

1. Copy an instance of the **Timer**
2. Drag a **Data** double with the value 1.0 (you can choose higher speeds, but they may cause the robot to overturn)
3. Connect the **Timer** output notification pin with the **Data** input pin
    a. Select **TimerComplete**
4. Drag a **GenericDifferentialDrive** service
5. Connect the **Data** box
    a. Select **SetDrivePower**
    b. Select **Value** for the left and right wheels
6. Drag a **Data** integer with the value 2000
7. Place a **Merge** activity (that you will use later) and connect it to the **Data** box
8. Connect the **GenericDifferentialDrive** to this **Merge** box
    a. Select **SetDrivePower – Success**
9. Copy and paste another instance of the **Timer**
10. Connect the **Data** box to it
    a. Select **SetTimer**
    b. Select **Value**

Your diagram should be like this one:

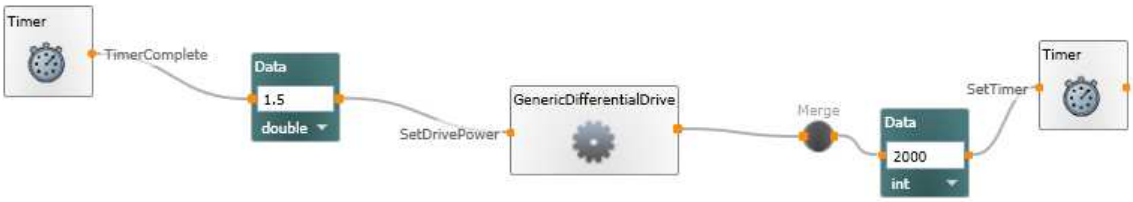

Figure 65. Diagram

Now you will create the interruption diagram:

1. Drag an **iRobotCreateLiteSimulation** service
    a. Set Configuration: **use a manifest**
    b. Select **Import**
    c. Select **testwallsensorsim.Manifest.xml**
2. In the **GenericDifferentialDrive** created before

   a. Set Configuration: **use a manifest**
   b. Select **SimulatedGenericDifferentialDrive in testwallsensorsim.manifest.xml**

You will use a special scenario for the iRobot Create and you are selecting it as a differential drive.

3. Drag a **Merge** activity
4. Connect the notification pin of the **iRobotCreateLiteSimulation** service to the input pin of the **Merge** activity
   a. Select **UpdateBumpsCliffsAndWalls**
5. Create a **Data** integer with the value 1
6. Connect the **Merge** to it
7. Copy and paste an instance of the **iRobotCreateLiteSimulation**
8. Connect the **Data** to it
   a. Select **PlaySong**
   b. Select **Value**

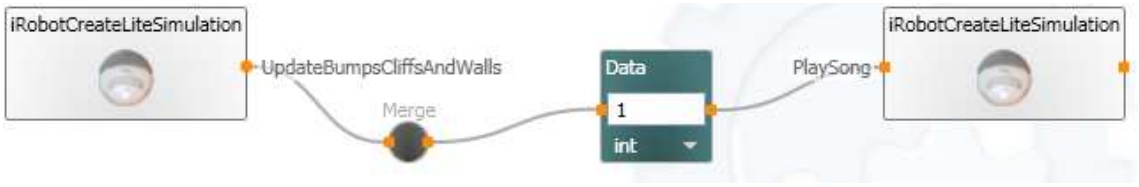

Figure 66. Playing song

This is part of the interruption. When a sensor is updated the robot will turn. When the flow is turning the iRobot will play a song (computer ”beeps”).

Let’s configure the robot to turn:

1. Create two **Data** double activities with the values: -0.2 and 0.2 respectively
2. Connect the last **Merge** (Figure 66) to both of them
3. Create a **Join** activity with the values: **Left** and **Right** respectively
4. Connect the **Data** boxes to them
   a. -0.2 **Left**
   b. 0.2 **Right**
5. Copy and paste an instance of the **GenericDifferentialDrive**
6. Connect the **Join** activity to the input pin of the **GenericDifferentialDrive**
   a. Select **SetDrivePower**
   b. Select **Left** for **LeftWheelPower** and do the same for the right wheel
7. Connect this **GenericDifferentialDrive** to the Merge you have created before in the forward diagram (Figure 65)
   a. Select **SetDrivePower - Success**

It will override the timer to make a turning of the same time every forward flow. The resulting final diagram should be like this one:

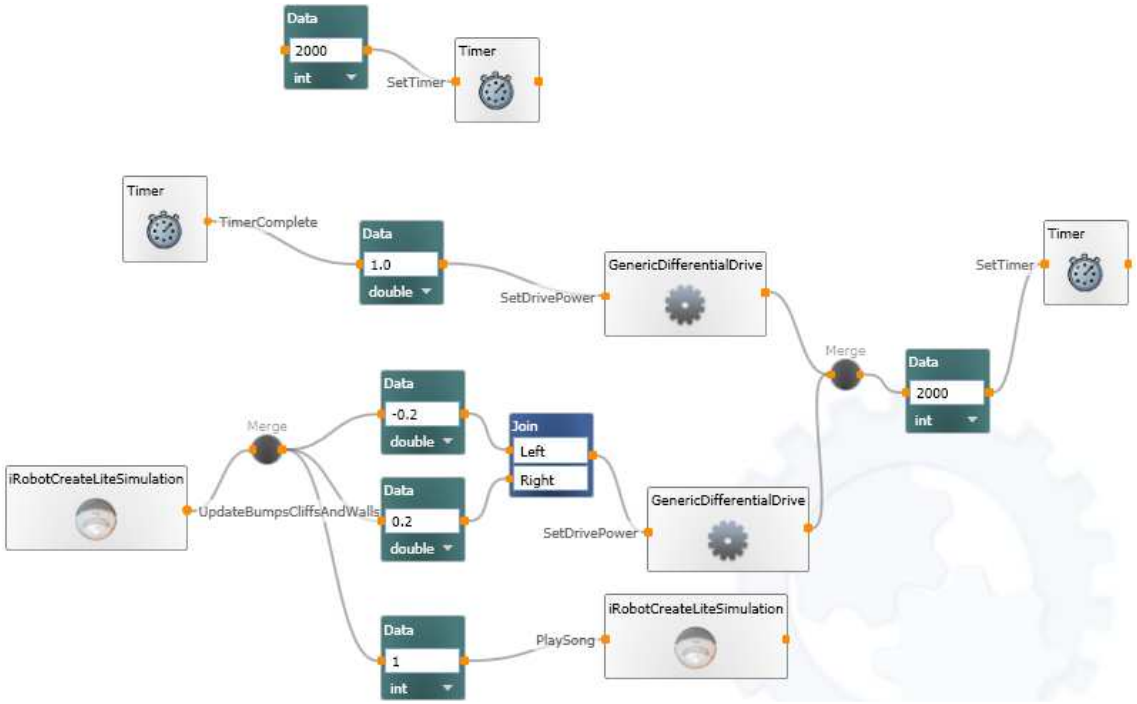

Figure 67. Final diagram

You will see an environment with walls and objects. When the robot crashes with any of them it will turn left and keep moving. Note that crashing with objects doesn't work in 100% of the cases because the robot moves them.

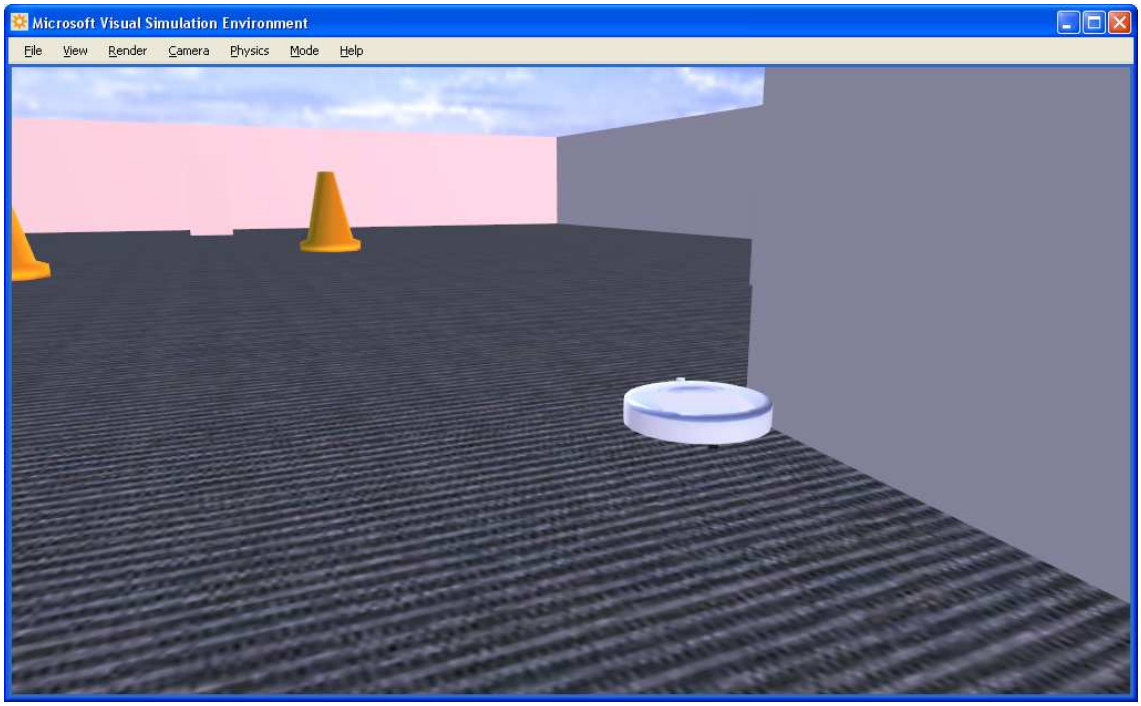

Figure 68. Robot turning after a collision

The robot created will move over the entire scenario:

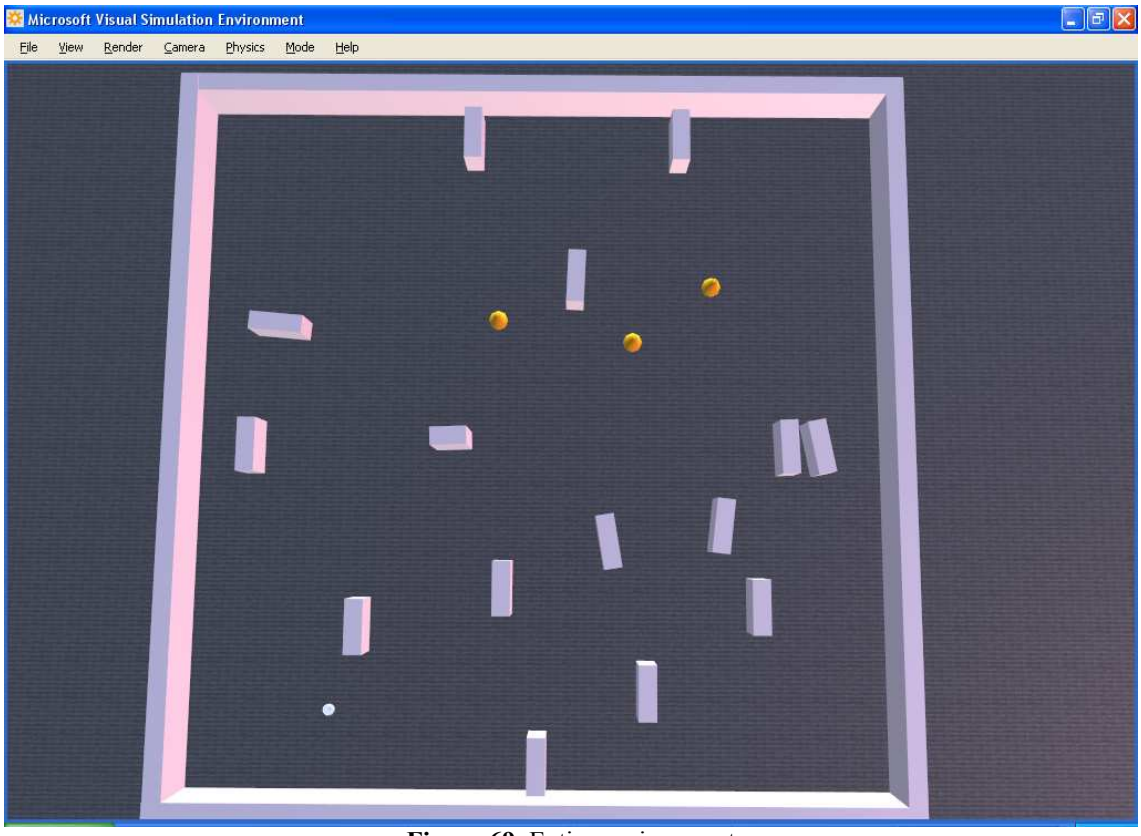

Figure 69. Entire environment

Glossari

En un projecte tècnic com aquest sembla necessari definir alguns dels termes que apareixen. En aquest apartat es definiran breument aquests termes per a facilitar al lector la lectura de la memòria i establir aquest glossari com a referència.

A

Aibo: Robot de Sony similar a un gos
API: Application Programming Interface

B

BDK: Bluetooth Developer Kit
Blob: Objectes bàsics (en imatges)
Bluetooth: Especificació industrial de xarxes inalàmbriques
BSD: Sistema Operatiu derivat de UNIX

C

C++: Llenguatge de programació
CCR: Concurrency and Coordination Runtime
Community Technical Preview: Versió inacabada de demostració d'un programa
Cyborg: Criatura que combina parts orgàniques i mecàniques

D

Dashboard: Tabler de control
DSS: Decentralized Software Services

E

E-Puck: Robot petit, rodó i pla que porta rodes i una sèrie de sensors/actuadors

G

Gazebo: Simulador 3D per múltiples robots
GNU: GNU is not Unix. Sistema operatiu lliure
GPL: Llicència pública general (basada en GNU)
Grau de llibertat: Mínim número de coordenades necessàries e independents per a determinar completament la posició de totes les parts d'un sistema en un instant
GUI: Graphical User Interface

H

HDK: Hardware Developer Kit
Hemisson: Robot petit, rodó i pla que porta rodes i una sèrie de sensors/actuadors
Hosting: Espai per a emmagatzemar informació o oferir un servei
HTML Compilat: Fitxer html autoaccessible
HTML: HyperText Markup Language. Llenguatge de creació de pàgines web
HTTP: HyperText Transfer Protocol. Protocol utilitzat en transaccions web

I

i-Cybie: Robot amb forma de gos
IR: InfraRed
iRobot Create: Robot petit, rodó i pla que porta rodes i una sèrie de sensors/actuadors

J

Jaquemart: Ninots de dues o més posicions que fan cops a campanes accionats per mecanismes de rellotgeria asiàtics
Java: Llenguatge de programació
Javascript: Llenguatge de programació interpretat, usualment utilitzat en pàgines web

K

Kephera: Robot petit, rodó i pla que porta rodes i una sèrie de sensors/actuadors

L

Lego Mindstorms: Kit de robòtica desenvolupat per Lego

M

Matlab: Llenguatge de programació
MRS (o MSRS): Microsoft Robotics Studio
MSDN: Microsoft Developer Network

N

Nao: Robot programable amb una aparença d'esser humà, desenvolupat per l'empresa francesa Aldebaran Robotics
NET (.NET): Plataforma de desenvolupament de Software creada per Microsoft
Nintendo Wiimote: Controlador de Nintendo per a la consola Wii. Incorpora sensors de moviment
NXT (Bloc): Part central de Lego Mindstorms, millorant RCX. Conté la lògica

O

OpenGL: Estàndard que defineix una API per a escriure aplicacions que produeixin gràfics en 2D i 3D

P

Phyton: Llenguatge de programació
Pioneer 3 DX: Robot amb rodes i una càmera
Player/Stage/Gazebo: Software de programació i control de robots
Player: Interfície de gestió de robots
Plugin: Complement per a una aplicació

R

RCX (Bloc): Part central de Lego Mindstorms. Conté la lògica
Render: Generar una imatge des d'un model
REST: Representational State Transfer. Arquitectura basada en web tradicional
RF: Radio freqüència
Robocup: Projecte internacional de competició de robots
Robot: Màquina o aparell electrònic programable que té la capacitat de manipular objectes i realitzar operacions que abans eren possible només mitjançant persones
Robòtica: Ciència i tecnologia dels robots, així com el seu disseny, fabricació i aplicació
Roomba: Robot aspiradora fabricat i distribuït per l'empresa iRobot

S

SDK: Software Developer Kit
Stage: Simulador per múltiples robots
Student Partners: Cada any Microsoft premia i reconeix els estudiants més talentosos del món i els fa "student partner"

T

Third-party: Empreses que desenvolupen software per a diverses plataformes
URBI: Universal Real-time Behavior Interface. Software de programació i control de robots

U

USB: Universal Serial Bus. Port per a connectar perifèrics

V

Visual Basic: Llenguatge de programació
VPL: Visual Programming Language

W

Walk-through: Guia
Webots: Software de programació i control de robots
Wii: Videoconsola de sobretaula produïda per Nintendo
WWW: World Wide Web

X

Xbox360: Videoconsola de sobretaula produïda per Microsoft
XML: Extensible Markup Language. Llenguatge extensible a pàgines web
XNA Framework: API desenvolupada per Microsoft per al desenvolupament de videojocs per PC i Xbox360

Resum

Cada cop més s'utilitzen robots en molts àmbits de la vida, els quals han de ser programats, i el fet que Microsoft s'impliqui aportant una eina de programació resulta interessant.

Aprendre a programar robots i colònies de robots amb aquesta eina, la qual està tenint una bona acollida, és molt important, per això en aquest projecte s'ha fet una anàlisi crítica de MRS i del material docent associat a l'aplicació.

Els resultats i conclusions obtinguts han estat, d'una banda, que és la millor eina de programació de robots, en termes generals, i d'altra, una guia iniciativa a l'entorn MRS, accessible a la direcció: http://shades.uab.cat/MSRS

Resumen

Cada vez más se utilizan robots en muchos ámbitos de la vida, los cuales han de ser programados, y el hecho de que Microsoft se implique aportando una herramienta de programación resulta interesante.

Aprender a programar robots y colonias de robots con esta herramienta, la cual está teniendo una buena acogida, es muy importante, es por esto que en este proyecto se ha hecho un análisis crítico de MRS y del material docente asociado a la aplicación

Los resultados y conclusiones obtenidos han sido, por un lado que es la mejor aplicación de programación de robots, en términos generales, y por otro, una guía iniciativa en el entorno MRS, accesible desde la dirección: http://shades.uab.cat/MSRS

Abstract

Every time more robots are used in many areas of the life, which have to be programmed, and the fact that Microsoft is implied bringing a tool of programming is interesting.

To learn to program robots and colonies of robots with this tool, which is having a good reception, is very important, in this project a critical analysis of MRS and of the educational material associated with the application has been made because of that.

The results and conclusions obtained in the analysis have been, on the one hand, that MRS is the best tool of robots programming, in general terms, and of another, an initiative guide in the environment MRS, accessible to the direction: http://shades.uab.cat/MSRS
