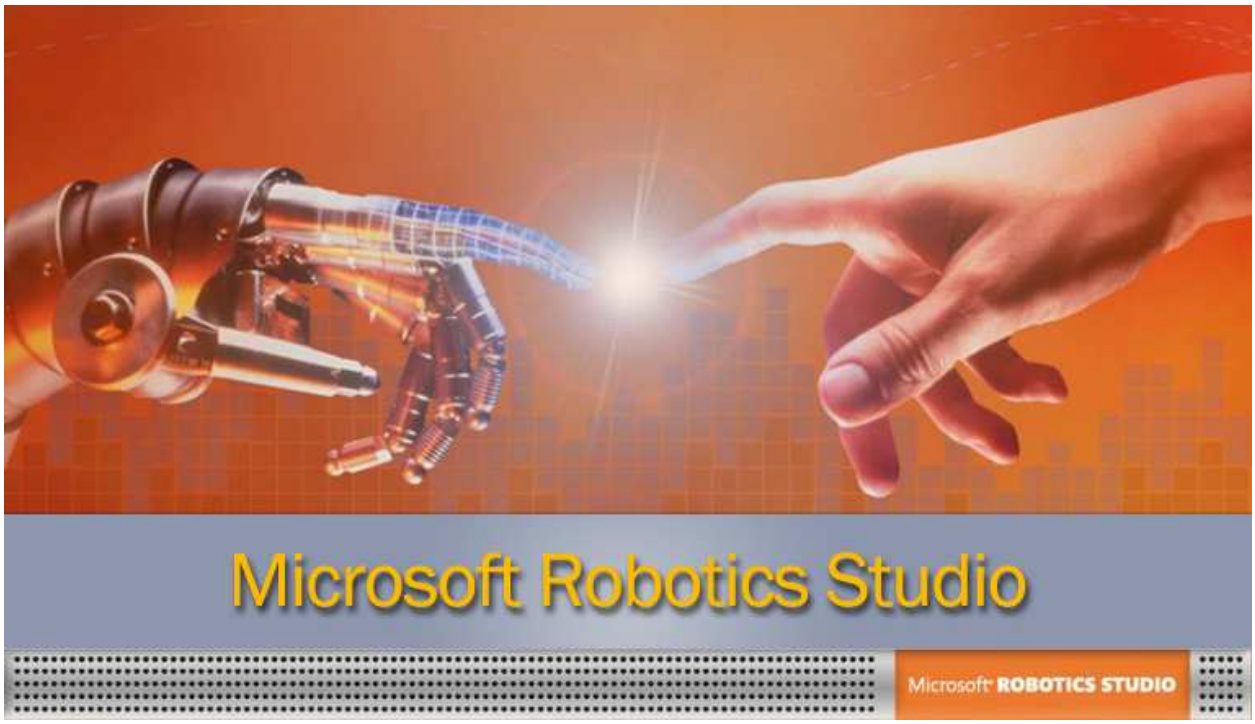
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Θέμα: Ανάπτυξη λογισμικού ευφυούς αυτόνομου ρομπότ.



Κοντογεωργάκος Σπυρίδων
Λούτα Παναγιώτα - Βασιλική

Υπεύθυνος καθηγητής
Παλαμάς Γεώργιος

Πρόλογος

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε με την ανάλυση χρήσης του προγράμματος Microsoft Robotics Developer Studio 2008 Express και την παρουσίαση διαφόρων demos λειτουργίας του προγράμματος.

Ο σκοπός της παρούσας Πτυχιακής εργασίας είναι η ανάπτυξη και προσομοίωση αυτόνομων ρομπότ με το πακέτο Microsoft Robotics Developer Studio 2008 Express.

Θα θέλαμε να ευχαριστήσουμε μέσα από την καρδιά μας τον επιβλέποντα αυτής της εργασίας καθηγητή Παλαμά Γεώργιο για την συνεχή βοήθεια, παρότρυνση καθώς και τις επισημάνσεις του σε κάθε στάδιο περάτωσης αυτής της δουλειάς. Επίσης, θέλουμε να ευχαριστήσουμε ιδιαίτερα τους γονείς μας για όλη τους την προσπάθεια όλα αυτά τα χρόνια που μας στάθηκαν σε οποιοδήποτε πρόβλημα που παρουσιάστηκε.

Ηράκλειο Κρήτης
2009-2010

Κοντογεωργάκος Σπύρος
Λούτα Παναγιώτα-Βασιλική

Περιεχόμενα

Περιεχόμενα 2
Εισαγωγή 1
Προσομοιωτές Ρομπότ 1
Εισαγωγή..... 1
Τι είναι ένας προσομοιωτής (Simulator)..... 1
Πλεονεκτήματα Προσομοίωσης 3
Μειονεκτήματα Προσομοίωσης..... 4
Χαρακτηριστικά..... 5
Προσομοιωτές 6
Παρουσίαση του MICROSOFT ROBOTICS STUDIO (MSRS)..... 7
Απαιτήσεις hardware και software 7
Αντικείμενο της εργασίας 8
Οργάνωση της εργασίας..... 8
Κεφάλαιο 1 9
Εισαγωγή στο 9
Microsoft Robotics Developer Studio 2008 Express..... 9
1.1 Εισαγωγή..... 9
1.2 Χαρακτηριστικά..... 13
1.2.2 DirectX 14
1.2.3 .NET Framework 16
1.2.4 Microsoft XNA 17
Κεφάλαιο 2 20
2.1 Εισαγωγή στη VPL..... 20
2.2 Δημιουργία Διαγράμματος..... 23
2.3 Το μενού της VPL (VPL MENUS)..... 27
File 27
Edit 28
View 28
Run..... 29
Help 29
2.4 Βασικές Δραστηριότητες (Basic Activities) 30
2.5 Τύποι Δεδομένων (Data Types)..... 35
2.6 Δημιουργία και επεξεργασία δραστηριοτήτων 36
2.7 Διαμόρφωση Υπηρεσίας (configuring service) 38
2.8 Σύνδεση Δραστηριοτήτων (Connecting activities) 40
2.9 Εκτέλεση των προγραμμάτων (Running and Debugging) και Παραγωγή κώδικα (Code generation)..... 43
2.10 Τεχνάσματα και συμβουλές για το Microsoft Robotics Developer Studio..... 47
2.10.1 Εύρεση NXT Services στη VPL 47
2.10.2 Θέτοντας τις σωστές τιμές στη VPL..... 48
Κεφάλαιο 3 51
3.1 Εισαγωγή στον DSS Manifest Editor..... 51
3.2 Ξεκινώντας τον DSS Manifest Editor..... 52
3.3 DSS Manifest Editor Menus 54
The File Menu..... 54
The Deploy Menu 56
The Help Menu 57
3.4 Δημιουργία και Επεξεργασία των DSS Manifests..... 57
Κεφάλαιο 4 61
4.1 Microsoft Visual Simulation Environment 61

4.1.1 Γενικά για το Simulation ... 61
4.1.2 Επισκόπηση του χρόνου εκτέλεσης προσομοίωσης ... 64
4.1.3 Προγραμματισμός του Simulation (Simulation programming) ... 64
4.1.4 Εικόνες από Simulations ... 65
4.2 Ξεκινώντας το περιβάλλον Visual Simulation ... 67
4.3 Το μενού του Visual Simulation ... 69
4.4 Η χρήση Keyboard και Mouse στο Visual Simulation ... 72
4.5 Το εργαλείο Obj-to-Bos File Converter ... 74
Κεφάλαιο 5 ... 75
5.1 Εισαγωγή ... 75
5.2 Περίληψη των υπηρεσιών ... 75
5.3 Διαμόρφωση υπηρεσίας Brick (Brick Service Configuration) ... 77
5.4 Οι υπηρεσίες του LEGO MINDSTORMS NXT ... 78
5.5 Υπηρεσία κίνησης (Motor service) ... 80
5.6 Η υπηρεσία Drive ... 81
5.7 Υπηρεσίες Light Sensor ... 83
Κεφάλαιο 6 ... 84
6.1 Προσομοίωση του κόσμου με την χρήση του Microsoft Robotics Studio ... 84
6.3 Οντότητες και τύποι οντοτήτων ... 86
6.4 Δημιουργία αρχείου Mesh ... 93
6.5 Δημιουργία αρχείου Manifest ... 97
6.6 Δημιουργία οντότητας χωρίς διαφορική κίνηση ... 101
Κεφάλαιο 7 ... 102
7.1 Εισαγωγή ... 102
7.1.1Θεωρία της διαφορικής κίνησης ... 102
7.2 Δημιουργία ενός οχήματος διαφορικής κίνησης ... 103
7.3 Σχέδιο κίνησης Ρομπότ ... 107
Συμπληρωματικά ... 110
7.4 ΘΕΩΡΙΑ ΚΙΝΗΣΕΩΝ ΤΩΝ ΡΟΜΠΟΤ ... 112
7.4.1 Κατηγορίες ρομπότ ... 112
7.4.2 Τρόποι κινήσεως των ρομπότ ... 112
7.4.3 Σταθερότητα των ρομπότ ... 114
7.4.4 Βηματισμός των ρομπότ ... 115
7.5 ΠΕΡΙΓΡΑΦΗ ΡΟΜΠΟΤ ΑΝΑΛΟΓΑ ΜΕ ΤΟΝ ΑΡΙΘΜΟ ΤΩΝ ΑΚΡΩΝ ... 116
7.5.1 Δίτροχα ρομπότ ... 116
7.5.2 Κίνηση ρομπότ με ένα πόδι ... 116
7.5.3 Κίνηση ρομπότ με δύο πόδια ... 116
7.5.4 Κίνηση ρομπότ με τέσσερα πόδια ... 117
7.5.5 Κίνηση ρομπότ με έξη πόδια ... 117
7.5.6 Κίνηση ρομπότ με περισσότερα από έξι πόδια ... 118
Κεφάλαιο 8 ... 119
8.1 Σκοπός ... 119
8.2 Αισθητήρες Ρομπότ ... 120
8.3 Διάφοροι τύποι αισθητήρων ... 123
8.4 Αισθητήρες ρομπότ /Αισθητήρες απόστασης ... 124
8.5 CCD εναντίον CMOS ... 131
8.6 Lego Mindstorms ... 132
8.6.1 Robotics Invention System - RCX brick ... 133
8.6.2 Lego Mindstorms NXT ... 135
8.6.3 Λίστα αισθητήρων ... 138
Κεφάλαιο 9 ... 142
9.1 Εφαρμογή Text-to-Speech ... 142
9.2 Ρομπότ σε πορεία τετραγώνου ... 147
9.3 Κίνηση ρομπότ το οποίο ανιχνεύει τοίχο και αλλάζει πορεία ... 154
9.4 Παιχνίδι Sumo ... 163

9.5 Κίνηση ρομπότ με joystick 168
9.5.1 Ο Κώδικας σε γλώσσα VPL..... 168
9.5.2 Ο Κώδικας σε γλώσσα C Sharp (C#)..... 170
9.6 Αναγνώριση περιοχών χρωμάτων 177
9.6.1 Επεξήγηση βασικών εννοιών σχετικών με την κάμερα 177
9.6.2 Η αναγνώριση περιοχών χρωμάτων σε C#.....186
Κεφάλαιο 10……192
10.1 Κίνηση ρομπότ το οποίο ανιχνεύει και ακολουθεί μία κόκκινη μπάλα καθώς κινείται τυχαία……..192
10.1.1 Παρουσίαση προγράμματος στην VPL και σχολιασμός της λειτουργίας του……...192
10.1.2 Εικόνες της προσομοίωσης του προγράμματος……194
10.2 Κίνηση ρομπότ σε λαβύρινθο όπου ανιχνεύει την έξοδο ……...195
10.2.1 Παρουσίαση προγράμματος στην VPL και σχολιασμός της λειτουργίας του……195
10.2.2 Εικόνες της προσομοίωσης του προγράμματος……...198
Παράρτημα……...199
Βιβλιογραφία 2233
Συγγράμματα 2233
Ιστοσελίδες 2233

Εισαγωγή

Προσομοιωτές Ρομπότ

Εισαγωγή

Οι προσομοιωτές των ρομπότ άρχισαν να συνδέονται με τη ρομποτική πριν από αρκετό καιρό. Στις μέρες μας, έχουν γίνει απαραίτητα εργαλεία στη σχεδίαση και τον προγραμματισμό των βιομηχανικών ρομπότ και μελετών. Επίσης έπαιζαν σημαντικό ρόλο στον ερευνητικό τομέα, όπου έχουν γίνει κριτικές για την ανάπτυξη και αναπαράσταση πολλών αλγορίθμων και τεχνικών. Ο κύριος λόγος για αυτήν την επιτυχία είναι το ότι δεν κοστίζουν πολύ και είναι μία γρήγορη εναλλακτική ελέγχου πρωτότυπων και συντονισμού αλγορίθμων.

Ακόμα και σήμερα οι προσομοιωτές ρομπότ χρησιμοποιούνται για αυτό το σκοπό αλλά αναπτύσσονται και αναπαρίστανται και πολλές ερευνητικές εφαρμογές με τη χρήση τους. Αντιπροσωπεύουν ένα πολύτιμο και χρήσιμο εργαλείο για τους ερευνητές των ρομπότ. Αυτός είναι ο λόγος για τον οποίο προτείνονται για αναθεώρηση των διαθέσιμων εργαλείων λογισμικού.

Χρησιμοποιώντας έναν προσομοιωτή (simulator) οι ερευνητές μπορούν να χτίσουν πειραματικά περιβάλλοντα σύμφωνα με την δική τους φαντασία. Η πολυπλοκότητα, η πραγματικότητα, η εξειδίκευση μπορούν σταδιακά να αυξηθούν σε ένα επίπεδο όπου τα ρομπότ εικονικής πραγματικότητας μπορούν να έρθουν αντιμέτωπα με τις πραγματικές προκλήσεις του φυσικού κόσμου.

Τι είναι ένας προσομοιωτής (Simulator)

Η προσομοίωση χρησιμοποιείται ως μία πιο γρήγορη και απλή μέθοδος για έλεγχο ιδεών, θεωριών και λογισμικού με χρήση ρομπότ. Αν ο χρήστης δεν γνωρίζει τι πρόκειται να ελέγξει, τότε δε χρειάζεται ο προσομοιωτής. Στη συνέχεια παρουσιάζονται μερικά παραδείγματα.

Έστω ότι δεν έχει κατασκευαστεί ακόμα το ρομπότ, αλλά ο χρήστης θέλει να ελέγξει μία ιδέα χωρίς να ξοδέψει χρόνο και χρήματα για την κατασκευή του ρομπότ ώστε να την υλοποιήσει. Με ένα προσομοιωτή δημιουργείται ένα στιγμιαίο ρομπότ για να εργαστεί. Αυτό είναι βολικό για αυτούς που προτιμούν να προγραμματίζουν από το να δουλεύουν με φυσικά εξαρτήματα υπολογιστή (hardware).

Έστω ότι ο χρήστης έγραψε έναν αλγόριθμο και θέλει να ελέγξει τον κώδικα όχι μόνο για λάθη, αλλά επίσης για την αποτελεσματικότητα του αλγόριθμου για ένα ρομπότ. Ο έλεγχος μπορεί να γίνει χρησιμοποιώντας ένα πραγματικό ρομπότ. Όμως αυτή είναι μια μεγάλη και κουραστική διαδικασία. Αρχικά πρέπει να τεθεί σε λειτουργία το ρομπότ, να προγραμματιστεί, να τοποθετηθεί στη θέση εκκίνησης, στη συνέχεια να τρέξει, και μετά να γίνει καταγραφή του τι έγινε. Αυτό χρειάζεται τυπικά να γίνει πολλές φορές ώστε να γίνουν οι υπολογισμοί. Κάθε φορά που το ρομπότ αφήνεται να τρέξει χρειάζονται μερικά λεπτά, χωρίς να λαμβάνονται υπόψη καταστάσεις με επαναφόρτιση μπαταριών και αποτυχίες υλικού hardware, που μπορεί να

εμφανιστούν, με αποτέλεσμα να υπάρχει επιπλέον χάσιμο χρόνου από ώρες μέχρι και μέρες. Με ένα προσομοιωτή, μπορεί να εκτελεστούν δωδεκάδες έλεγχοι μέσα σε λίγα λεπτά.

Οι προσομοιώσεις υπάρχουν για λειτουργικούς λόγους και όχι για επίδειξη. Τα γράμματα και οι αριθμοί σε μία γραμμή εντολών είναι πιο αποδοτικοί από μία γραφική απεικόνιση.

Ο προσομοιωτής ρομπότ (robotics simulator) χρησιμοποιείται για την δημιουργία ενσωματωμένων εφαρμογών (embedded applications) για ένα συγκεκριμένο (ή όχι) ρομπότ χωρίς να εξαρτάται από αυτό, εξοικονομώντας έτσι χρόνο και κόστος. Σε μερικές περιπτώσεις, αυτές οι εφαρμογές μπορεί να μεταφερθούν στο πραγματικό ρομπότ χωρίς τροποποιήσεις.

Ο όρος robotics simulator μπορεί να αναφέρεται σε πάρα πολλές διαφορετικές εφαρμογές ρομποτικής προσομοίωσης. Για παράδειγμα, σε mobile robotics εφαρμογές, οι simulators επιτρέπουν στους χρήστες να δημιουργήσουν απλούς κόσμους άκαμπτων αντικειμένων και να προγραμματίσουν ρομπότ ώστε να αλληλεπιδρούν με αυτούς τους κόσμους. Η προσομοίωση που βασίζεται στη συμπεριφορά (behaviour-based simulation) επιτρέπει ενέργειες που είναι πιο βιολογικές στη φύση σε σύγκριση με τους simulators που είναι πιο δυαδικοί, ή υπολογιστικοί. Επίσης, οι behaviour-based simulators μπορούν να «μάθουν» από τα λάθη και είναι ικανοί να αναπαριστούν την ανθρωπόμορφη ποιότητα αντοχής.

Εικόνα1 : Ο simulator Robologix.

Μία από τις πιο διάσημες εφαρμογές των προσομοιωτών ρομπότ είναι η τρισδιάστατη μοντελοποίηση και διαχείριση ενός ρομπότ και του περιβάλλοντός του. Αυτό το τύπο ρομποτικού λογισμικού έχει ένας simulator που είναι ένα ρομπότ εικονικής πραγματικότητας, το οποίο είναι ικανό να μιμείται την κίνηση ενός πραγματικού ρομπότ σε ένα πραγματικό χώρο εργασίας. Μερικοί προσομοιωτές ρομποτικής όπως ο Robologix (**Εικόνα1**), χρησιμοποιούν μία φυσική μηχανή για την ακριβή απόδοση και κίνηση του ρομπότ σε τρισδιάστατο χώρο. Η χρήση ενός προσομοιωτή ρομπότ για την ανάπτυξη ενός προγράμματος ελέγχου του ρομπότ συνίσταται πάρα πολύ είτε ένα πραγματικό ρομπότ είναι διαθέσιμο είτε όχι. Ο simulator επιτρέπει στα προγράμματα ρομποτικής να μπορούν να γραφτούν και να διορθωθούν άνετα σε κατάσταση off-line με την τελική έκδοση του προγράμματος που θα ελεγχθεί σ' ένα πραγματικό ρομπότ.

Πλεονεκτήματα Προσομοίωσης

Εάν υπάρχει τόσο μεγάλο πρόβλημα με την προσομοίωση γιατί αξίζει να την χρησιμοποιούμε? Μία όχι τόσο επιστημονική απάντηση είναι ότι η προσομοίωση είναι πιο φθηνή. Ο ερευνητής δε χρειάζεται να πληρώσει για ένα ρομπότ. Δεν είναι απαραίτητη η αγορά κάποιων αισθητήρων ή άλλων υλικών και είναι επίσης εφικτό να σχεδιαστεί ένα θεωρητικά φυσικό αλλά στην ουσία τεχνητό περιβάλλον. Ένα ρομπότ εικονικής πραγματικότητας δε μπορεί να καταστραφεί και δε μπορεί να καταστρέψει το περιβάλλον του.

Η διαδικασία ενός πειράματος με ένα ρομπότ περιλαμβάνει και άλλα προβλήματα: καλώδια που ευθύνονται για τον έλεγχο και την παροχή ενέργειας, ή εάν δεν υπάρχει καλώδιο, οι συσσωρευτές των ρομπότ θα πρέπει να επαναφορτίζονται κατά καιρούς.

Άλλη μία ενδιαφέρουσα τεχνική πιθανότητα είναι η παράλληλη εκτέλεση της προσομοίωσης. Οι επιστήμονες μπορεί να εκτελέσουν το πείραμα με διαφορετικές παραμέτρους την ίδια στιγμή σε πολλούς υπολογιστές, σε αντίθεση με το να προσπαθούν να λύσουν το πρόβλημα πολλές φορές σε ένα ρομπότ. Τα αποτελέσματα σίγουρα θα φανούν πιο γρήγορα.

Ένας άλλος σημαντικός παράγοντας είναι ότι τα πειράματα γίνονται πιο γρήγορα στο προσομοιωτή παρά στη πραγματικότητα. Πλεονεκτεί επίσης στο ότι μπορεί να συνεργαστεί με πραγματικούς ελέγχους κι έτσι να δημιουργήσει γρήγορα το πρωτότυπο ενός προβλήματος. Πέρα της εγκατάστασης νέου προγράμματος, η εκτέλεση του πειράματος μπορεί να είναι πιο αποτελεσματική. Στη περίπτωση εκμάθησης αλγορίθμων, όπως τα τεχνητά νευρωνικά δίκτυα, το πραγματικό ρομπότ μπορεί να μάθει μόνο από προηγούμενα συγκεντρωμένα δεδομένα, έτσι δε μπορεί να περιπλανηθεί στο περιβάλλον του πολλές φορές λόγω χρονικού περιορισμού και περιορισμών του ρομπότ γενικώς. Μέσω του υπολογιστή είναι πιθανή η τηλε-εκπαίδευση. Το πρόβλημα είναι περισσότερο εμφανές σε εξελικτικούς αλγορίθμους. Σε αυτή τη περίπτωση αμέτρητες προσπάθειες πρέπει να γίνουν σε πολλές οντότητες διαφόρων γενιών ενός μεγάλου πληθυσμού ώστε να καθοριστεί η ζωτικότητα κάθε ατόμου. Στα πραγματικά ρομπότ αυτό είναι πιθανό να συμβεί μόνο για τις μικρές φιγούρες, σε σύγκριση με έναν προσομοιωτή.

Σε σύγκριση με ένα ελεγκτή προγράμματος, ένας προσομοιωτής μπορεί να βοηθήσει στην ανάπτυξη της δομής του ρομπότ. Η πληκτρολόγηση, οι αριθμοί, η θέση, ο τρόπος κίνησης, και η ανάλυση αντίληψης, είναι μερικές από τις ποικίλες ιδιότητές του.

Μια άλλη διαφωνία σχετικά με την υποστήριξη της προσομοίωσης είναι ότι οι επιδράσεις της απόκλισης από τον πραγματικό κόσμο θα πρέπει να μειωθούν. Η εξέλιξη (Coevolution) του κώδικα του ρομπότ και του προσομοιωτή μπορεί να οδηγήσει σε πραγματικά πολύ αποδοτικά ρομπότ.

Προσομοίωση ή πραγματικότητα? Από τα παραπάνω είναι προφανές ότι οι σχεδιαστές και οι ερευνητές πρέπει να είναι ιδιαίτερα προσεχτικοί πριν πάρουν οποιαδήποτε απόφαση. Και οι δύο μέθοδοι έχουν δυνατά σημεία και

αδυναμίες, για αυτό – όπως συνηθίζεται- η λύση μπορεί να είναι μια μέση οδός. Το κύριο μέρος του ελεγκτικού προγράμματος ενός ρομπότ μπορεί να σχεδιαστεί σε ένα αρκετά λεπτομερή προσομοιωτή, ο οποίος αφού μεταφερθεί σε πραγματικές συνθήκες μπορεί να εκτελεστεί. Τα μελλοντικά ρομπότ σχεδιάζονται πιο αποτελεσματικά από την συνεργασία προσομοίωσης και πραγματικότητας.

Μειονεκτήματα Προσομοίωσης

Υπάρχουν πολλά μειονεκτήματα στη προσομοίωση ενός ρομπότ. Αυτό συμβαίνει επειδή οι προσομοιώσεις προσομοιώνουν μόνο ότι τους λες να προσομοιώσουν.

Ακριβώς επειδή το ρομπότ θα λειτουργήσει στην προσομοίωση δεν σημαίνει ότι θα λειτουργήσει στην πραγματική ζωή! Φυσικά μπορείτε να ξοδέψετε πολύ χρόνο καθιστώντας την προσομοίωσή σας τέλεια, αλλά εάν απαιτείτε τις ρεαλιστικές τοποθετήσεις του 100% θα ήταν καλύτερο να κάνετε ελέγχους με ένα πραγματικό ρομπότ. Πριν σχεδιάσετε την προσομοίωσή σας, πρέπει να αποφασίσετε τι μπορείτε να απλοποιήσετε και τι πρέπει να παραμείνει ρεαλιστικό. Η προσομοίωση μερικές φορές χρειάζεται πολύ χρόνο για να υλοποιηθεί.

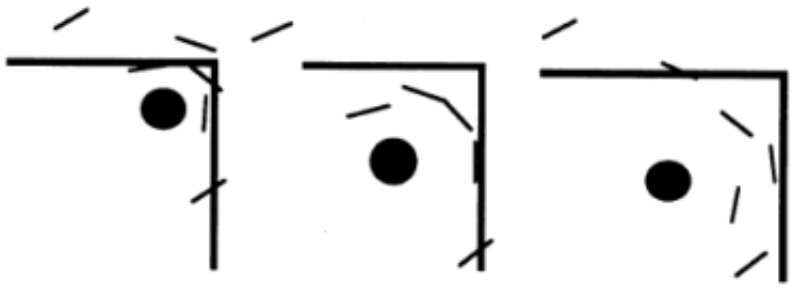
Οι προσομοιώσεις είναι συχνά λανθασμένες. Είναι λανθασμένες επειδή ο πειραματιστής κάνει ένα λάθος, ή δεν είναι βέβαιος για το ποια χαρακτηριστικά γνωρίσματα είναι τα σημαντικότερα και ως εκ τούτου υπεραπλουστεύουν τις νέες πειραματικές θεωρίες. Συχνά πρέπει η προσομοίωση να συγκρίνεται και να συμβαδίζει με τα πραγματικά πειράματα ώστε να εξασφαλιστεί ότι η προσομοίωση είναι ακριβής. Υπό αυτήν τη μορφή, η προσομοίωση μπορεί να χρησιμοποιηθεί ως ένα καλό εργαλείο, αλλά δεν είναι μια λύση για όλα.

Η δημιουργία της προσομοίωσης έχει μερικά επιπρόσθετα υπολογιστικά χρέη τα οποία δεν υπάρχουν στο φυσικό κόσμο: η σύνθεση ενός τρισδιάστατου κόσμου, δημιουργία αντίληψης, κινηματικού και δυναμικού μηχανισμού. Ο προσομοιωτής πρέπει να υπολογίσει τις πληροφορίες που εμφανίζονται στον αισθητήρα που έχει το ρομπότ στην περίπτωση του φάσματος συμπεριλαμβάνοντας τις σκιές και τις επικαλύψεις. Περαιτέρω στόχος είναι η προετοιμασία μιας κατάλληλης τροχιάς λαμβάνοντας υπόψη τους περιορισμούς της αρχιτεκτονικής του ρομπότ και να δοθεί η κατάλληλη προσοχή στα αποτελέσματα απροσδόκητων γεγονότων όπως για παράδειγμα οι συγκρούσεις.

Μία σημαντική απειλή είναι ότι το προσομοιωμένο ρομπότ δουλεύει κατάλληλα, αλλά λόγω κάποιων διαφορών μεταξύ προσομοίωσης και πραγματικότητας, η φυσική εκδοχή είναι άχρηστη. Για να παρακαμφτεί αυτό το πρόβλημα οι ιδιότητες του κόσμου για να διαμορφωθούν πρέπει να καθοριστούν προσεχτικά.

Αξίζει να ληφθεί υπόψη ότι τα δύο στοιχεία δεν είναι ποτέ τα ίδια, δύο τροχοί έχουν διαφορετική επιτάχυνση, δύο κάμερες επιστρέφουν διαφορετικούς χάρτες χρωμάτων. Για να μοντελοποιηθούν τα πραγματικά

χαρακτηριστικά είναι θορυβώδες και αβέβαιο σε αντίληψη και δράση. Οι τροχοί του ρομπότ μπορεί να γλιστρήσουν, αλλά η ίδια δύναμη κίνησης έχει ως αποτέλεσμα ποικίλες προοδευτικές ταχύτητες σε ένα χαλί ή στο παρκέ, κι ακόμη και η κατεύθυνση των ινών του χαλιού πρέπει να είναι καθορισμένη. Ένα σήμα εκπεμπόμενο από ένα sonar απεικονίζεται ευδιάκριτα από τις ανόμοιες επιφάνειες εκθέτοντας τις διάφορες πληροφορίες απόστασης. Λόγω των αντανακλάσεων το σήμα μπορεί ακόμη και να φθάσει στο δέκτη ενός άλλου sonar που επηρεάζει μια προφανώς ανακριβή αξία. Η γωνία της πρόσπτωσης του σήματος μπορεί να είναι επίσης σχετική: στην **Εικόνα2** το ρομπότ (μαύρος κύκλος) που εξοπλίζεται με sonars πλησιάζει μια γωνία. Οι μετρήσεις είναι σωστές εάν η γωνία της πρόσπτωσης είναι κάθετη, αλλά όσο μεγαλύτερη είναι η απόκλιση από αυτήν την τιμή τόσο μεγαλύτερο είναι το σφάλμα όπως φαίνεται κι από τα μαύρα τμήματα γραμμών . Αυτά τα γεγονότα δείχνουν ότι οι αισθητήρες του πραγματικού κόσμου πρέπει να βαθμονομηθούν, και αυτό πρέπει να εμφανιστεί και στον προσομοιωτή.



Εικόνα2 : Sonar error

Η προσομοίωση μπορεί να προκαλέσει προβλήματα που δεν εμφανίζονται στην πραγματικότητα. Ένα απλό παράδειγμα είναι η σύγκρουση δύο ρομπότ σε έναν πλεγματικό κόσμο. Σε αυτήν την περίπτωση ο σχεδιαστής πορειών πρέπει να λύσει την κατάσταση αδιεξόδου. Κανονικά τα ρομπότ δεν φθάνουν συγχρόνως στην ίδια θέση, κι ως εκ τούτου το φαινόμενο δεν χρειάζεται καμία λύση.

Χαρακτηριστικά

Οι σύγχρονοι simulators τείνουν να παρέχουν τα ακόλουθα χαρακτηριστικά:

- **Γρήγορα ρομπότ για δημιουργία πρωτοτύπων (Fast robot prototyping)**
    - Χρησιμοποιώντας τον ίδιο το simulator σαν εργαλείο δημιουργίας (Webots, R-Station, Marilou).
    - Χρησιμοποιώντας εξωτερικά εργαλεία (Gazebo uses Blender).
- **Φυσικές μηχανές (Physics engines)** για πραγματικές κινήσεις. Οι περισσότεροι simulators χρησιμοποιούν ODE (Gazebo, Marilou, Webots) ή PhysX (Microsoft).

- **Ρεαλιστική τρισδιάστατη απόδοση**. Τρισδιάστατα εργαλεία μοντελοποίησης ή άλλα ειδικά εργαλεία μπορούν να χρησιμοποιηθούν για να χτιστούν τα περιβάλλοντα.

- **Δυναμικές ρομποτικές μονάδες με κείμενο εργασίας**. Οι γλώσσες URBI, MATLAB, Python χρησιμοποιούνται από Webots, η Python χρησιμοποιείται από Gazebo.

Προσομοιωτές

Open source simulators

- Gazebo: Πιθανόν ο πιο ολοκληρωμένος open source simulator.

- OpenSim Simulator: για ρομπότ που αποτελούνται από ρόδες ή από αρθρώσεις με μία ευρεία κλίμακα χαρακτηριστικών. Η περεταίρω ανάπτυξη σταμάτησε το 2006.

- Simbad 3d Robot Simulator : simulator βασισμένος στη Java

- breve 3d simulator : χρησιμοποιώντας Python

- Khepera Simulator :Προφανώς πριν τα Webots έγινε εμπορικός ήταν open source και ονομαζόταν Khepera Simulator

- Blender: είναι ένα ώριμο πακέτο υπολογιστών κινούμενων εικόνων που υποστηρίζει συσκευές ρομπότ με τη μορφή οπλισμών.

Προσομοιωτές έρευνας

- UWA Robot Simulators

Εμπορικοί προσομοιωτές

- anyKode Marilou

- Webots Ένας γενικός simulator που μπορεί να προγραμματιστεί σε C/C++, Java, Python και MATLAB

- Microsoft Robotics Studio

- RoboLogix

- Visual Components

- Cogmation robotSim

- SimplySim Content για τρισδιάστατους προσομοιωτές (περιβάλλοντα, μοντέλα, αισθητήρια)

Παρουσίαση του MICROSOFT ROBOTICS STUDIO (MSRS)

Τον Ιούνιο του 2006 ο Trower μαζί με μία ομάδα ερευνητών ανακοίνωσαν την ύπαρξη του MSRS. Το MSRS προέκυψε μέσα από έρευνα του Trower πάνω στην ρομποτική. Επρόκειτο για μία πλατφόρμα η οποία θα μπορούσε να επιλύει προβλήματα που αντιμετώπιζαν οι επιστήμονες της ρομποτικής αλλά και ερασιτέχνες που ασχολούνταν με το πρόγραμμα αυτό.

Το MSRS είναι ένα περιβάλλον που προσφέρει εποπτικά εργαλεία και εγχειρίδια διδασκαλίας. Το πακέτο λογισμικού MSRS παρέχει μία πλατφόρμα και εργαλεία που μπορούν να χρησιμοποιηθούν από επιστήμονες τόσο σε ερευνητικό όσο και σε εμπορικό επίπεδο. Το MSRS προσφέρει επίσης μία γλώσσα προγραμματισμού, Visual Programming Language (VPL), που μπορεί να χρησιμοποιηθεί από μαθητές και ερασιτέχνες. Η έκδοση του MSRS version 1.5 είναι πολύ πιο βελτιωμένη από την έκδοση 1.0 συμπεριλαμβάνοντας αξιοσημείωτες ρυθμίσεις στο περιβάλλον προσομοίωσης και στη γλώσσα προγραμματισμού VPL.

Η νέα αυτή έκδοση του MSRS δημιουργήθηκε πολύ γρήγορα παρόλο που η ομάδα που το σχεδίασε αποτελούνταν μόνο από 11 ερευνητές. Το MSRS βασίζεται στο Concurrency and Coordination Runtime (CCR). Το CCR είναι μία a.NET βιβλιοθήκη σχεδιασμένη έτσι ώστε να γίνεται ο ασύγχρονος προγραμματισμός ευκολότερα από πριν. Αναπτύχθηκε την ίδια χρονική περίοδο που ο Trower προσπαθούσε να κάνει την ρομποτική πιο εύκολη. Η ομάδα που δημιούργησε το MSRS αποτελούνταν και από δύο ερευνητές της ομάδας δημιουργίας του CCR .

Η τελευταία έκδοση του MSRS έχει τόσες βελτιώσεις που κάνει το προϊόν ακόμα πιο εύκολο στη χρήση. Το MSRS είναι μια δημιουργία που βρίσκεται σε εξέλιξη. Γίνεται προσπάθεια για συνεχείς βελτιώσεις και ρυθμίσεις στα εργαλεία και στις υπηρεσίες του MSRS.

Απαιτήσεις hardware και software

• Απαιτήσεις hardware

Συνίσταται να χρησιμοποιείτε ένα laptop ή desktop υπολογιστή.

Οι ελάχιστες απαιτήσεις για τη συσκευή που θα επιλέξετε είναι οι εξής:

Υπολογιστή με 600MHZ PENTIUM III –συμβατό επεξεργαστή 1GHZ ή παραπάνω.

512 MB μνήμη RAM (συνίσταται 1GB ή παραπάνω).

200 MB ελεύθερο χώρο σκληρού δίσκου για την εγκατάσταση του MSRS.

Πάνω από 1,3 GB ελεύθερου χώρου στο σκληρό δίσκο απαιτείται για την εγκατάσταση του Visual Studio Express Edition.

Super VGA (1024*768) ή υψηλότερης ανάλυσης οθόνη και προσαρμοστή οθόνης υπολογιστή.

Εσωτερική κάρτα γραφικών, ικανή να υποστηρίζει τη σκίαση δύο τεμνόμενων πλευρών, VS_2_0 ή μεγαλύτερη, και τη σκίαση των εικονοστοιχείων, PS_2_0 ή μεγαλύτερη.

Πληκτρολόγιο και ποντίκι Microsoft ή συμβατή συσκευή στίξης.

- **Απαιτήσεις Software**

Θα χρειαστεί να εγκατασταθεί ολόκληρη η έκδοση του MSRS, η οποία είναι διαθέσιμη στην ιστοσελίδα της Microsoft. Θα χρειαστεί επίσης το Visual Studio όπου μπορεί να χρησιμοποιηθεί η Express Edition που βρίσκεται στην ιστοσελίδα της Microsoft.

Αντικείμενο της εργασίας

Σκοπός αυτής της Πτυχιακής εργασίας είναι η ανάπτυξη και προσομοίωση αυτόνομων ρομπότ με το πακέτο Microsoft Robotics Developer Studio 2008 Express. Στα κεφάλαια, που ακολουθούν, παρουσιάζονται αναλυτικά τα βήματα για την ολοκλήρωση της εργασίας. Σε κάθε ένα από τα κεφάλαια παρουσιάζεται αναλυτικά το θεωρητικό υπόβαθρο για την κατανόηση των θεμάτων που πραγματεύονται.

Οργάνωση της εργασίας

- Στο 1ο Κεφάλαιο παρουσιάζονται τα εισαγωγικά στοιχεία για το Microsoft Robotics Developer Studio.

- Στο 2ο Κεφάλαιο, γίνεται παρουσίαση της γλώσσας προγραμματισμού visual programming language (VPL).

- Στο 3ο Κεφάλαιο, θα ασχοληθούμε με τον Microsoft DSS manifest editor.

- Στο 4ο Κεφάλαιο γίνεται ανάλυση του περιβάλλοντος εξομοίωσης «Microsoft visual simulation environment».

- Στο 5ο κεφάλαιο θα ασχοληθούμε με τις λειτουργίες του Lego Mindstorms NXT.

- Στο 6^{ο} κεφάλαιο παρουσιάζεται η κατασκευή ενός απλού κόσμου με το Microsoft Robotics Developer Studio.

- Στο 7^{ο} κεφάλαιο παρουσιάζεται η κατασκευή ενός κινούμενου ρομπότ με διαφορική κίνηση από απλά υλικά .

.

- Στο 8^{ο} κεφάλαιο γίνεται παρουσίαση των αισθητήρων.

- Στο 9^{ο} κεφάλαιο γίνεται παρουσίαση προγραμμάτων σε γλώσσα VPL του Microsoft Robotics Developer Studio.

- Στο 10^{ο} κεφάλαιο γίνεται παρουσίαση Advanced προγραμμάτων σε γλώσσα VPL του Microsoft Robotics Developer Studio.

Κεφάλαιο 1

Εισαγωγή στο

Microsoft Robotics Developer Studio 2008 Express

Φιλοσοφία του MSRS(Microsoft Robotics Studio)

Το Microsoft Robotics Studio στοχεύει να δημιουργήσει τις κοινές τεχνολογικές υποστηρίξεις για τη σχετικά αποκομμένη βιομηχανία της ρομποτικής. Προορίζεται για τη χρήση σε ένα ευρύ φάσμα ρομπότ, στα εργοστάσια από εκείνους, στον αυξανόμενο αριθμό προσανατολισμένων προς τον καταναλωτή ρομπότ.

Το Microsoft Robotics Studio , όπως συνήθως, θα τρέχει μόνο στα Windows.

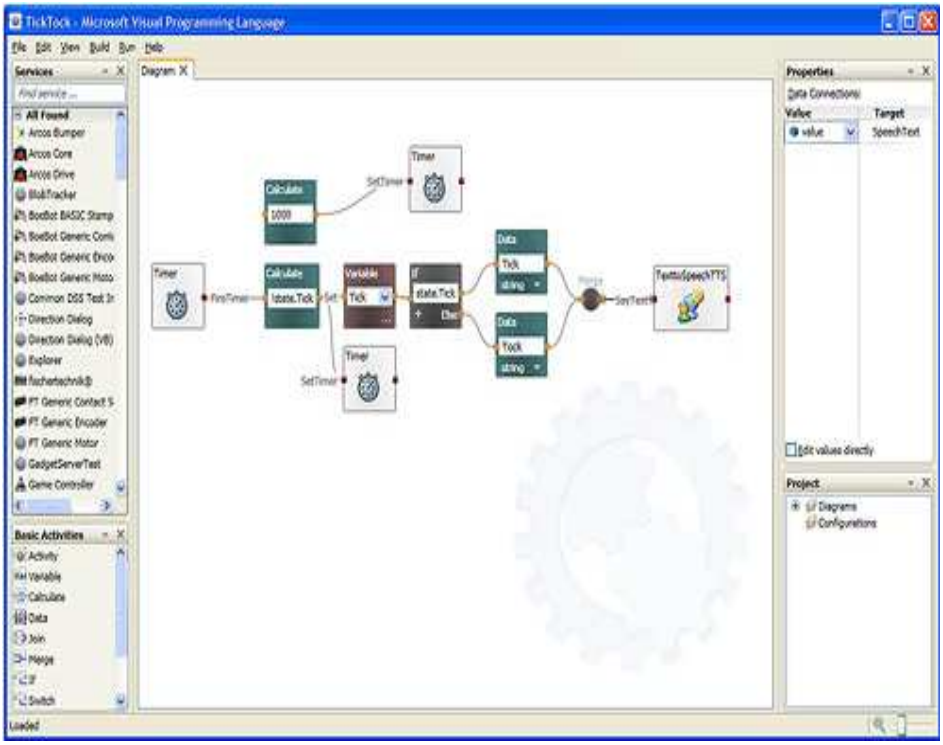
Θα έχει τη δυνατότητα να δημιουργήσει τρισδιάστατα υπολογιστικά μοντέλα των ρομπότ, χρησιμοποιώντας φυσική πραγματικού κόσμου, αφήνοντας τους υπεύθυνους για την ανάπτυξη να δουν πώς τα προγράμματα θα δουλέψουν, πριν δημιουργήσουν το ρομπότ .

1.1 Εισαγωγή

Το Microsoft Robotics Developer Studio 2008 Express είναι ένα περιβάλλον βασισμένο στα Windows κατάλληλο για ερασιτέχνες, σπουδαστές ή ερευνητές, το οποίο παρέχει τη δυνατότητα δημιουργίας ρομποτικών εφαρμογών για μία ποικιλία από ρομπότ. Περιλαμβάνει εργαλεία παραγωγής κώδικα, εργαλεία εξομοίωσης, παραδείγματα και εγχειρίδια .

- Το Microsoft Robotics Developer Studio 2008 Express δίνει την δυνατότητα σε χρήστες να δημιουργήσουν υπηρεσίες για μια ευρύτατη ποικιλία ρομπότ. Όλοι μπορούν να δημιουργήσουν εφαρμογές των ρομπότ χρησιμοποιώντας ένα οπτικό περιβάλλον προγραμματισμού.

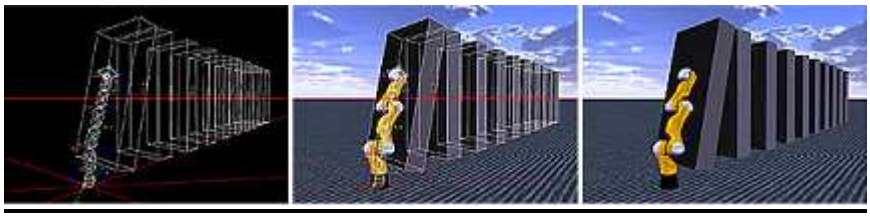
- Η Visual Programming Language (γλώσσα προγραμματισμού) δίνει την δυνατότητα στον οποιοδήποτε να δημιουργήσει και να τρέξει προγράμματα για ρομπότ πολύ εύκολα. Με απλό σύρσιμο τοποθετούνται τα blocks που αντιπροσωπεύουν υπηρεσίες και μετά συνδέονται (**Εικόνα 1.1.1**). Επίσης υπάρχει δυνατότητα να επιλεγεί μια συλλογή από συνδεδεμένα blocks και να χρησιμοποιηθεί σαν ένα ξεχωριστό block οπουδήποτε στο πρόγραμμα .



Εικόνα 1.1.1 : Παράδειγμα διαγράμματος στην VPL με την χρήση block υπηρεσιών.

- Υπάρχει δυνατότητα προσομοίωσης των εφαρμογών των ρομπότ χρησιμοποιώντας ρεαλιστικά τρισδιάστατα προσομοιωμένα μοντέλα. Το εργαλείο της προσομοίωσης στο Microsoft Robotics Studio περιλαμβάνει το AGEIA PsysX Technology από το AGEIA Technologies Inc. Μία πρωτοποριακή τεχνολογία που καθιστά ικανή τη προσομοίωση ρομποτικών μοντέλων σε τεχνολογία πραγματικού κόσμου (**Εικόνες 1.1.2** και **1.1.3**).

Οι προσομοιώσεις PsysX μπορούν επίσης να επιταχυνθούν χρησιμοποιώντας ειδικό AGEIA Hardware το οποίο είναι σαν ένας συνεπεξεργαστής που θα μπορεί να εισέρχεται σαν πρόσθετο υλικό ή σαν ολοκληρωμένη πατέντα επεξεργαστή ή ως GPU συνεπεξεργαστή ή ως υπομονάδα AGEIA στο σύστημα ή να είναι ενσωματωμένο στην κάρτα γραφικών. Ο ρόλος του είναι να αναλαμβάνει πλήρως τους φυσικούς υπολογισμούς οι οποίοι θα είναι απαραίτητοι για ένα απαιτητικό σχέδιο ενός παιχνιδιού ή μιας προσομοίωσης κλπ.



Εικόνα 1.1.2 : Παράδειγμα εικόνας τεχνολογίας πραγματικού κόσμου.

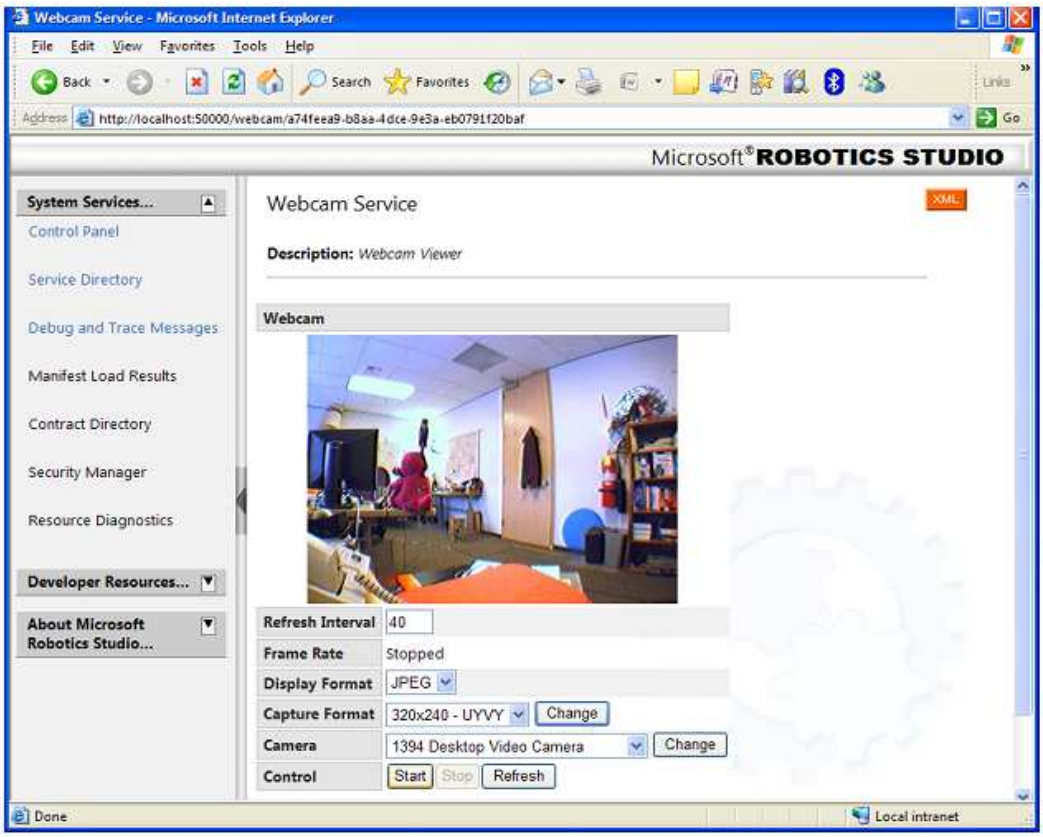
Εικόνα1.1.3 : Δεύτερο παράδειγμα εικόνας τεχνολογίας πραγματικού κόσμου.

- Υπάρχει η δυνατότητα αλληλεπίδρασης με τα ρομπότ χρησιμοποιώντας Windows ή περιβάλλοντα βασισμένα στο διαδίκτυο. Ο χρήστης μπορεί να δημιουργήσει εφαρμογές που θα του επιτρέπουν να ελέγξει και να ρυθμίσει ένα ρομπότ εξ αποστάσεως χρησιμοποιώντας έναν Web – browser και να του στέλνει εντολές χρησιμοποιώντας υπάρχουσες Web τεχνολογίες όπως HTML φόρμες και JavaScript. Με την προσθήκη κάμερας στα ρομπότ παρέχεται επιπλέον η δυνατότητα ελέγχου τους από απόσταση καθώς θα εξετάζουν μία τοποθεσία.

- Πώς ο Ασύγχρονος Προγραμματισμός Απλοποιείται!

Με το CCR (Concurrency Coordination Runtime), υπάρχει η δυνατότητα γραφής προγραμμάτων με απλό τρόπο ώστε να γίνεται ο χειρισμός ασύγχρονων εισόδων από πολλαπλούς αισθητήρες των ρομπότ.

- Ο πραγματικός χρόνος ελέγχου των αισθητήρων των ρομπότ και ανταπόκρισης των κινητήρων και των μηχανισμών κίνησης γίνεται με το μοντέλο DSS. Η εφαρμογή του μοντέλου DSS (decentralized software services) απλοποιεί την πρόσβαση και την ανταπόκριση σε μία κατάσταση του ρομπότ, χρησιμοποιώντας μία εφαρμογή Web-browser ή Windows – based εφαρμογή **(Εικόνα 1.1.4)**.



Εικόνα 1.1.4 : Χρησιμοποίηση του Microsoft Internet Explorer μέσω του μοντέλου DSS.

- Παρέχεται η δυνατότητα χρήσης υπηρεσιών που αποτελούνται από υπομονάδες μέσω ενός μοντέλου που μπορεί να αναπαραχθεί. Η δημιουργία υψηλού επιπέδου λειτουργίας γίνεται με τη χρήση απλών στοιχείων παράγοντας έτσι κώδικα για επαναχρησιμοποίηση. Για παράδειγμα μία χαμηλότερου επιπέδου υπηρεσία αισθητήρα μπορεί να ολοκληρωθεί σε μία υπηρεσία εύρεσης πορείας .

- Το μοντέλο προγραμματισμού του Microsoft Robotics Developer Studio 2008 Express μπορεί να εφαρμοσθεί σε μία ποικιλία από πλατφόρμες hardware των ρομπότ, καθιστώντας ικανούς τους χρήστες να εφαρμόσουν τις ικανότητές τους διαμέσου των πλατφορμών. Μέσω προγραμματισμού μπορούν να αναπτυχθούν εφαρμογές για ρομπότ χρησιμοποιώντας επεξεργαστές 8bit,16bit ή 32bit, είτε μεμονωμένα είτε πολλαπλά συνδεδεμένους .

- Εύκολα επεκτείνεται η λειτουργικότητα του Microsoft Robotics Developer Studio 2008 Express. Τρίτα μέρη μπορούν να επεκτείνουν τη λειτουργικότητα του Microsoft Robotics Developer Studio 2008 Express παρέχοντας επιπρόσθετες βιβλιοθήκες και υπηρεσίες. Πωλητές του hardware και του software μπορούν να κάνουν τα προϊόντα τους εύκολα συμβατά με το Microsoft Robotics Developer Studio2008 Express .

- Ανάπτυξη χρησιμοποιώντας μία ευρεία κλίμακα από γλώσσες προγραμματισμού. Με το Microsoft Robotics Developer Studio 2008 Express,

οι εφαρμογές των ρομπότ μπορούν να αναπτυχθούν χρησιμοποιώντας μία συλλογή από γλώσσες προγραμματισμού, συμπεριλαμβάνοντας εκείνες στο Microsoft Visual Studio και στο Microsoft Visual Studio Express (C, VB.NET) όπως επίσης γλώσσες σαν την Microsoft Iron Python .

1.2 Χαρακτηριστικά

1.2.1 Γενικά

- Το Visual Programming Language είναι ένας απλός και εύκολος τρόπος προγραμματισμού. Blocks κώδικα που αναπαριστούν services (υπηρεσίες).Επαναχρησιμοποίηση κώδικα. Εφαρμογή σε πολλά διαφορετικά πεδία. Τα πάντα είναι visual .
- Ρεαλιστική 3D(τρισδιάστατη) εξομοίωση με τις τεχνολογίες AGEIA PsyX και Microsoft XNA Framework.

Προκλήσεις: ακριβός εξοπλισμός, δύσκολη και χρονοβόρα αντιμετώπιση προβλημάτων, δυσκολία στην ταυτόχρονη χρήση από πολλούς.

Λύσεις: μηδενικό αρχικό κόστος, αφαιρετική ικανότητα, δημιουργία πρωτοτύπου, εφαρμογή στην εκπαίδευση.

Προβλήματα: έλλειψη προβλημάτων πραγματικού περιβάλλοντος, αδυναμία πλήρους μοντελοποίησης, χρονοβόρες μικρορυθμίσεις.

- Επικοινωνία με τα μέρη του ρομπότ (κινητήρες , αισθητήρες κ.α.)
    1. Υποστήριξη διαφορετικών τρόπων επικοινωνίας μεταξύ ρομπότ και υπολογιστή (serial port ,Bluetooth ,usb κα)
    2. Υποστήριξη εφαρμογών διαφορετικών ειδών και απαιτήσεων .

    <u>Δυνατότητες :</u>
    1. Παρακολούθηση και επικοινωνία με τα ξεχωριστά μέρη του ρομπότ καθώς η εφαρμογή τρέχει .
    2. Σωστός συντονισμός εισόδου από πολλές πηγές .
    3. Τοπικός ή απομακρυσμένος έλεγχος του ρομπότ .
    4. Ελαφρύ runtime(χρόνος εκτέλεσης) κατάλληλο για διάφορα περιβάλλοντα .

    Το runtime αποτελείται από τα : Concurrency and Coordination Runtime (CCR), Decentralized Software Services (DSS).

- Επαναχρησιμοποίηση κώδικα και τμηματική σχεδίαση. Δημιουργία συναρτήσεων υψηλού επιπέδου χρησιμοποιώντας απλά στοιχεία . Αυξημένη αξιοπιστία . Εύκολη επαναχρησιμοποίηση τμημάτων κώδικα .

- Ευέλικτη πλατφόρμα. Υποστηρίζει πολλά ρομποτικά περιβάλλοντα. Παράγει κώδικα για 8, 16, 32 bit συστήματα, για ένα ή πολλούς πυρήνες.

- Υποστήριξη εξίσου προγραμμάτων απομακρυσμένων ή αυτόνομων ρομπότ.

Υποστήριξη επικοινωνίας με serial port, Bluetooth, 802.11, RF. Τα προγράμματα μπορούν να εκτελεστούν σε Microsoft Windows περιβάλλοντα .

- Υποστηρίζει πλήθος γλωσσών προγραμματισμού. Δυνατότητα δημιουργίας προγραμμάτων μέσω Microsoft Visual Studio, Microsoft Visual Studio Express (C, VB.NET), Microsoft Iron Python.

1.2.2 DirectX

Το **Microsoft DirectX** είναι μια συλλογή εφαρμογών προγραμματιζόμενων διεπαφών (APIs-application programming interfaces) που χειρίζεται θέματα σχετικά με πολυμέσα, και ειδικά με προγραμματιζόμενα παιχνίδια και βίντεο, στις πλατφόρμες της Microsoft.

Το Direct3D (η τρισδιάστατη γραφική παράσταση API μέσα σε DirectX) χρησιμοποιείται ευρέως στην ανάπτυξη των τηλεοπτικών παιχνιδιών για τα Microsoft Windows, το Microsoft Xbox, και το Microsoft Xbox 360. Το Direct3D χρησιμοποιείται επίσης από άλλες εφαρμογές λογισμικού για θέματα γραφικής απεικόνισης όπως η εφαρμοσμένη μηχανική CAD/CAM. Δεδομένου ότι το Direct3D είναι το ευρύτατα κοινοποιημένο συστατικό του DirectX, είναι συνηθισμένο να βλέπεις τα ονόματα "DirectX" και "Direct3D" να χρησιμοποιούνται εναλλακτικά.

Το λογισμικό ανάπτυξης του DirectX, το SDK δηλαδή, αποτελείται από τις βιβλιοθήκες χρόνου εκτέλεσης με δυαδική μορφή, μαζί με τη συνοδευτικές αποδείξεις και επιγραφές για τη χρήση στην κωδικοποίηση. Αρχικά, οι χρόνοι εκτέλεσης ρυθμίζονταν μόνο από τα παιχνίδια ή αποκλειστικά από το χρήστη.

Στοιχεία

Τα στοιχεία του DirectX είναι

- **DirectX Graphics**, αποτελείται από πάρα πολλές εφαρμογές APIs:
    - DirectDraw: για τον σχεδιασμό γραφικών 2D Graphics. Δεν προτιμάται λόγω του Direct2D, αν και χρησιμοποιείται ακόμα από πολλά παιχνίδια και σαν διακομιστής βίντεο (video renderer) σε εφαρμογές πολυμέσων.
    - Direct3D (D3D): για τον σχεδιασμό τρισδιάστατων γραφικών
    - DXGI: για απαρίθμηση των προσαρμοστών και των οθόνων ώστε να επιτευχθεί ανταλλαγή αλυσίδων για Direct3D 10 και πάνω.
    - Direct2D για γραφικά 2D Graphics
    - DirectWrite ή Fonts
- **DirectCompute** για υπολογισμούς GPU
- **DirectInput:** για τη διασύνδεση με τις συσκευές εισαγωγής συμπεριλαμβανομένου πληκτρολόγιων, ποντικιών, joysticks, ή άλλων μέσων ελέγχου παιχνιδιών. Δεν χρησιμοποιήθηκε ιδιαίτερα μετά την έκδοση 8 λόγω του XInput για ελεγκτές Xbox 360 ή για συγκεκριμένα παράθυρα μηνυμάτων WM INPUT για επεξεργασία μέσω πληκτρολογίου ή ποντικιού.

- **DirectPlay:** για την επικοινωνία πέρα από μία περιοχή δικτύου τοπική ή ευρεία.
- **DirectSound**: για την αναπαραγωγή ήχου και την καταγραφή των κυματοειδών ήχων.
- **DirectSound3D (DS3D):** για την αναπαραγωγή 3D ήχων.
- **DirectMusic:** για συσκευές αναπαραγωγής ήχου τμημάτων ταινίας εξουσιοδοτημένα από την DirectMusic Producer.
- **DirectX Media:** περιλαμβάνει DirectAnimation για 2D/3D δικτυακή δυναμική κίνηση ομοιωμάτων, DirectShow για αναπαραγωγή ήχου πολυμέσων και μέσων ροής (streaming media), DirectX Transform για δικτυακή αλληλεπίδραση, και Direct3D Retained Mode για γραφικά υψηλού τρισδιάστατου επιπέδου. DirectShow περιέχει DirectX plugins για διαδικασίες σημάτων ακουστικών συχνοτήτων 15-20000Hz (audio signal processing) και DirectX Video Acceleration για επιτάχυνση αναπαραγωγής βίντεο ήχου.
- **DirectX Diagnostics (DxDiag):** ένα εργαλείο για την διάγνωση και αναπαραγωγή αναφορών σε στοιχεία που σχετίζονται με την DirectX, όπως ήχο, βίντεο και drivers' εισαγωγής.
- **DirectX Media Objects:** υποστηρίζεται από αντικείμενα ροής όπως είναι οι κωδικοποιητές, οι αποκωδικοποιητές, και τα εφέ.
- **DirectSetup:** για την εγκατάσταση στοιχείων της DirectX, και την ανίχνευση της τρέχουσας έκδοσης DirectX.

Η λειτουργικότητα του DirectX παρέχεται μέσω αντικειμένων και τύπου COM και αλληλεπιδράσεων. Επίσης, αν και οι συνιστώσες του DirectX από μόνες τους διαχειρίζονται αντικείμενα, βασίζονται σε κάποια κομμάτια του DirectX, όπως είναι οι διαχειριστές Direct3D 9 και η βιβλιοθήκη γραφικών XNA .

Συμβατότητα (Compatibility)

Οι διάφορες εκδόσεις των Windows έχουν συμπεριλάβει και έχουν υποστηρίξει τις διάφορες εκδόσεις του DirectX, επιτρέποντας στις νεότερες εκδόσεις του λειτουργικού συστήματος να συνεχίσουν τις εφαρμογές που σχεδιάζονταν από τις προηγούμενες εκδόσεις DirectX έως ότου να μπορούν εκείνες οι εκδόσεις να καταργηθούν σταδιακά υπέρ νεότερων APIs, drivers, και του υλικού hardware.

Εφαρμογές APIs όπως είναι η Direct3D και η DirectSound πρέπει να αλληλεπιδράσουν με το υλικό hardware, και το κάνουν αυτό μέσω ενός οδηγού συσκευών. Οι κατασκευαστές υλικού hardware πρέπει να γράψουν αυτούς τους οδηγούς για μία συγκεκριμένη έκδοση DirectX (ή DDI), και να εξετάσουν κάθε μεμονωμένο κομμάτι του υλικού hardware για να το κάνουν συμβατό με το σύστημα DirectX. Μερικές μόνο συσκευές υλικού hardware έχουν συμβατούς οδηγούς DirectX (με άλλα λόγια, κάποιος πρέπει να εγκαταστήσει το DirectX προκειμένου να χρησιμοποιήσει το συγκεκριμένο το υλικό hardware). Οι πρώρες εκδόσεις DirectX περιλάμβαναν μια ενημερωμένη βιβλιοθήκη όλων των συμβατών οδηγών DirectX που είναι διαθέσιμη και σήμερα.

Ωστόσο, η πρακτική αυτή σταμάτησε, λόγω του συστήματος οδηγού ενημέρωσης που βασίζεται στο δικτυακό Windows Update, το οποίο έδινε τη δυνατότητα στους χρήστες να κατεβάσουν μόνο τους κατάλληλους οδηγούς για το δικό τους υλικό hardware παρά ολόκληρη την βιβλιοθήκη.

Πριν από το DirectX 10, ο χρόνος εκτέλεσης DirectX είχε σχεδιαστεί ώστε να είναι συμβατός με τους παλαιότερους οδηγούς, εννοώντας ότι οι νεότερες εκδόσεις του APIs είχαν ως σκοπό να επικοινωνήσουν με τους παλαιότερους οδηγούς που γράφτηκαν έναντι του DDI μιας προηγούμενης έκδοσης. Ο προγραμματιστής της εφαρμογής έπρεπε να ρωτήσει τις διαθέσιμες ικανότητες του υλικού hardware χρησιμοποιώντας ένα σύνθετο σύστημα όπου κάθε κομμάτι αφορούσε ένα ιδιαίτερο χαρακτηριστικό υλικού hardware. Παραδείγματος χάριν, ένα παιχνίδι που σχεδιάστηκε για να τρέχει σε Direct3D 9 με έναν οδηγό προσαρμογής γραφικών που σχεδιάστηκε για Direct3D 6 θα λειτουργούσε αλλά πιθανότατα με υποβαθμισμένη λειτουργία.

Όμως ο χρόνος εκτέλεσης του Direct3D 10 στα Windows vista δεν μπορεί να τρέξει στους παλαιότερους οδηγούς υλικού hardware λόγω του σημαντικά ενημερωμένου DDI, το οποίο απαιτεί ένα ενοποιημένο σετ χαρακτηριστικών.

Ο χρόνος εκτέλεσης του Direct3D 11 θα εισαγάγει τα Direct3D 9, 10, και 10.1 "επίπεδα χαρακτηριστικών", με τρόπους συμβατότητας οι οποίοι θα επιτρέπουν μόνο τη χρήση των χαρακτηριστικών υλικού hardware που καθορίζονται στη διευκρινισμένη έκδοση Direct3D. Για το υλικό hardware του Direct3D 9, θα υπάρξουν τρία διαφορετικά επίπεδα χαρακτηριστικών γνωρισμάτων, που ομαδοποιούνται από τις κοινές ικανότητες "low", "med" και "high-end" των καρτών βίντεο. Ο χρόνος εκτέλεσης θα χρησιμοποιεί άμεσα το Direct3D 9 DDI που παρέχεται σε όλους τους οδηγούς WDDM.

1.2.3 .NET Framework

Κατά τη διάρκεια του GDC 2006 η Microsoft παρουσίασε το XNA Framework, μια νέα διοικούμενη έκδοση του DirectX (παρόμοια αλλά όχι ίδια με αυτή του DirectX) που προορίζεται να βοηθήσει την ανάπτυξη των παιχνιδιών κάνοντας ευκολότερη την ενσωμάτωση του DirectX, της γλώσσας Shader υψηλού επιπέδου (HLSL) και άλλων εργαλείων σε ένα πακέτο. Υποστηρίζει επίσης την εκτέλεση του διοικούμενου κώδικα στο Xbox 360. Το XNA Game Studio Express RTM δημιουργήθηκε στις 11 Δεκεμβρίου 2006. Αντίθετα, ο χρόνος εκτέλεσης του DirectX, το Managed DirectX, το XNA Framework ή το Xbox 360 APIs (XInput, XACT κ.λπ.) δεν φορτώνονται ως μέρος των Windows. Οι υπεύθυνοι ανάπτυξης περιμένουν να ανακατανείμουν τα τμήματα του χρόνου εκτέλεσης μαζί με τα παιχνίδια ή τις εφαρμογές τους. Κανένα προϊόν της Microsoft συμπεριλαμβανομένων των πιο πρόσφατων εκδόσεων XNA, δεν παρέχει το DirectX 10 για την υποστήριξη του .NET Framework.

Η άλλη προσέγγιση για το DirectX στις διοικούμενες γλώσσες είναι να χρησιμοποιηθούν οι βιβλιοθήκες όπως η SlimDX για το Direct3D, το DirectInput (συμπεριλαμβανομένου του Direct3D 10), το Direct Show .NET για το υποσύνολο DirectShow ή τα Windows API CodePack για το .Net Framework που είναι μια ανοικτή βιβλιοθήκη πηγής από τη Microsoft.

1.2.4 Microsoft XNA

Το Microsoft XNA (νέα αρχιτεκτονική Xbox) είναι ένα σύνολο εργαλείων με ένα διοικούμενο περιβάλλον χρόνου εκτέλεσης που παρέχεται από τη Microsoft το οποίο διευκολύνει την ανάπτυξη και τη διαχείριση παιχνιδιών στους υπολογιστές. Το XNA προσπαθεί να απελευθερώσει τους υπεύθυνους για την ανάπτυξη παιχνιδιών από το γράψιμο του επαναλαμβανόμενου κώδικα και παρουσιάζει διαφορετικές πτυχές της παραγωγής παιχνιδιών σε ένα ενιαίο σύστημα.

Εισαγωγή

XNA Framework

Το XNA Framework είναι βασισμένο στην εφαρμογή του .NET Compact Framework 2.0 για την Xbox 360 ανάπτυξη και του .NET Framework 2.0 στα Windows. Περιλαμβάνει ένα εκτενές σύνολο βιβλιοθηκών, ειδικό για την ανάπτυξη παιχνιδιών, για να προωθήσει τη μέγιστη επαναχρησιμοποίηση κώδικα στις πλατφόρμες. Το framework τρέχει σε μια έκδοση του Common Language Runtime_ το οποίο βελτιστοποιείται για παιχνίδια ώστε να παρέχει ένα διοικούμενο περιβάλλον εκτέλεσης. Ο χρόνος εκτέλεσης είναι διαθέσιμος για Windows XP, vista, και Xbox 360 .

Δεδομένου ότι τα παιχνίδια XNA γράφονται για το χρόνο εκτέλεσης, μπορούν να τρέξουν σε οποιαδήποτε πλατφόρμα που υποστηρίζει το XNA Framework με ελάχιστη ή καμία τροποποίηση. Τα παιχνίδια που εκτελούνται στο framework δεν μπορούν τεχνικά να γραφτούν σε οποιαδήποτε .NETγλώσσα, αλλά μόνο σε C# και XNA Game Studio Express του IDE και όλες τις εκδόσεις του Visual Studio 2005 που υποστηρίζονται επίσημα.

Το XNA Framework τοποθετεί τις χαμηλού επιπέδου τεχνολογικές λεπτομέρειες που περιλαμβάνονται στην κωδικοποίηση ενός παιχνιδιού, έτσι ώστε το ίδιο το framework να φροντίζει τη διαφορά μεταξύ των πλατφορμών όταν τα παιχνίδια περνούν από μια συμβατή πλατφόρμα σε άλλη, και με αυτόν τον τρόπο επιτρέπει στους υπεύθυνους για την ανάπτυξη παιχνιδιών να εστιάσουν περισσότερο στο περιεχόμενο και στην εμπειρία των παιχνιδιών.

Το XNAFramework έχει ενσωματωμένο ένα αριθμό εργαλείων, όπως το Cross-platform Audio Creation Tool (XACT), για να βοηθήσει στην δημιουργία περιεχομένου. Το XNA Framework παρέχει την υποστήριξη και για δυσδιάστατη και για την τρισδιάστατη δημιουργία παιχνιδιών και επιτρέπει τη χρήση ελεγκτών και ταλαντώσεων (παλμών) στο Xbox 360.

XNA Build

Το XNA Build είναι ένα σύνολο διοικητικών εργαλείων παιχνιδιών, το οποίο βοηθά με τον καθορισμό, τη διατήρηση, τη διόρθωση, και τη βελτιστοποίηση των μεμονωμένων προσπαθειών ανάπτυξης παιχνιδιών. Ένα μέσο πληροφόρησης για τα στοιχεία ενός παιχνιδιού (game asset pipeline) περιγράφει τη διαδικασία από την οποία το περιεχόμενο παιχνιδιών, όπως τα

κείμενα και τα τρισδιάστατα μοντέλα, τροποποιούνται σε μια μορφή κατάλληλη προς χρήση από τη μηχανή του παιχνιδιού. Το XNA Build βοηθά να προσδιοριστούν οι απαιτήσεις του μέσου πληροφόρησης, και παρέχει επίσης την πρόσβαση API για να επιτρέψει την περαιτέρω επεξεργασία των στοιχείων εξάρτησης. Τα στοιχεία εξάρτησης μπορούν να αναλυθούν για να βοηθήσουν στη μείωση του μεγέθους ενός παιχνιδιού με την εύρεση περιεχομένου που δεν χρησιμοποιείται πραγματικά.

XNA Game Studio

Το XNA Game Studio είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη των παιχνιδιών. Πέντε εκδόσεις έχουν δημιουργηθεί μέχρι τώρα.

XNA Game Studio Professional

Το **XNA Game Studio Professional** ήταν μια προγραμματισμένη έκδοση του XNA IDE που στόχευε στους επαγγελματίες δημιουργούς παιχνιδιών. Βασισμένο στο Visual Studio 2005 Team System, το XNA Studio παρέχει μια δομή για τη συνεργασία μεταξύ των δημιουργών περιεχομένου, των προγραμματιστών, της διαχείρισης και των ελεγκτών. Οι στόχοι διαχείρισης του προγράμματος, όπως η αξιολόγηση διαχείρισης (asset management), η ανίχνευση ατέλειας, η αυτοματοποίηση προγράμματος, και οι κατάλογοι θεμάτων εργασίας, είναι κάπως αυτοματοποιημένοι από το XNA Studio.

Το XNA Game Studio Professional δεν είναι πλέον υπό ενεργή ανάπτυξη.

XNA Game Studio Express

Το XNA Game Studio Express είναι σαφές ότι προορίζεται για τους σπουδαστές, τους χομπίστες, και τους ανεξάρτητους (και οικιακούς (homebrew) χρήστες παιχνιδιών. Το Express παρέχει τις βασικές λειτουργίες για τη γρήγορη ανάπτυξη των συγκεκριμένων παιχνιδιών, όπως την πλατφόρμα, τη στρατηγική σε πραγματικό χρόνο, και first-person shooters .

XNA Game Studio 2.0

Το XNA Game Studio 2.0 εκδόθηκε στις 13 Δεκεμβρίου, το 2007. **Το XNA Game Studio 2.0** χαρακτηρίζει τη δυνατότητα να χρησιμοποιείται με όλες τις εκδόσεις Visual Studio 2005 (συμπεριλαμβανομένης της έκδοσης Visual C# 2005 Express Edition), μια δικτύωση API χρησιμοποιώντας το Xbox Live και στα Windows και στο Xbox 360 και καλύτερος χειρισμός συσκευών.

XNA Game Studio 3.0

Το XNA Game Studio 3.0 (για το Visual Studio 2008 ή για το Visual C# 2008 Express Edition) επιτρέπει την παραγωγή των παιχνιδιών που

απευθύνονται στην πλατφόρμα Zune και προσθέτει την υποστήριξη επικοινωνίας του Xbox Live. Η beta του συνόλου εργαλείων εκδόθηκε τον Σεπτέμβριο του 2008. Η τελική έκδοση έγινε στις 30 Οκτωβρίου του 2008. Το XNA Game Studio 3.0 τώρα υποστηρίζει την γλώσσα C# 3.0, LINQ και τις περισσότερες εκδόσεις του Visual Studio 2008. Υπάρχουν επίσης αρκετά ακόμα νέα χαρακτηριστικά γνωρίσματα του XNA Game Studio 3.0 , όπως ο δοκιμαστικός τρόπος που προστίθεται στο XNA Game Studio 3.0 ο οποίος θα επιτρέπει στους δημιουργούς να προσθέσουν εύκολα το απαραίτητο δοκιμαστικό χαρακτηριστικό γνώρισμα στα παιχνίδια τους, χαρακτηριστικά γνωρίσματα του Xbox LIVE όπως προκλήσεις μέσω παιχνιδιών, τα οποία δημιουργούν παιχνίδια διαγώνιων - πλατφορμών που λειτουργούν στα Windows, Xbox 360 και Zune.

XNA Game Studio 3.1

Το XNA Game Studio 3.1 αναγγέλθηκε στη διάσκεψη υπεύθυνων για την ανάπτυξη παιχνιδιών στο Σαν Φρανσίσκο στις 24 Μαρτίου 2009. Το API περιλαμβάνει την υποστήριξη για την τηλεοπτική αναπαραγωγή ήχου, ένα αναθεωρημένο ακουστικό API, το Xbox LIVE σύστημα και την υποστήριξη για τα παιχνίδια ώστε να χρησιμοποιηθεί το Xbox 360 .

XNA Framework Content Pipeline

Το **XNA Framework Content Pipeline** είναι ένα σύνολο εργαλείων που επιτρέπει στο Visual Studio και XNA Studio ″ως βασικό σημείο σχεδιασμού γύρω από την οργάνωση και την κατανάλωση του τρισδιάστατου περιεχομένου″.

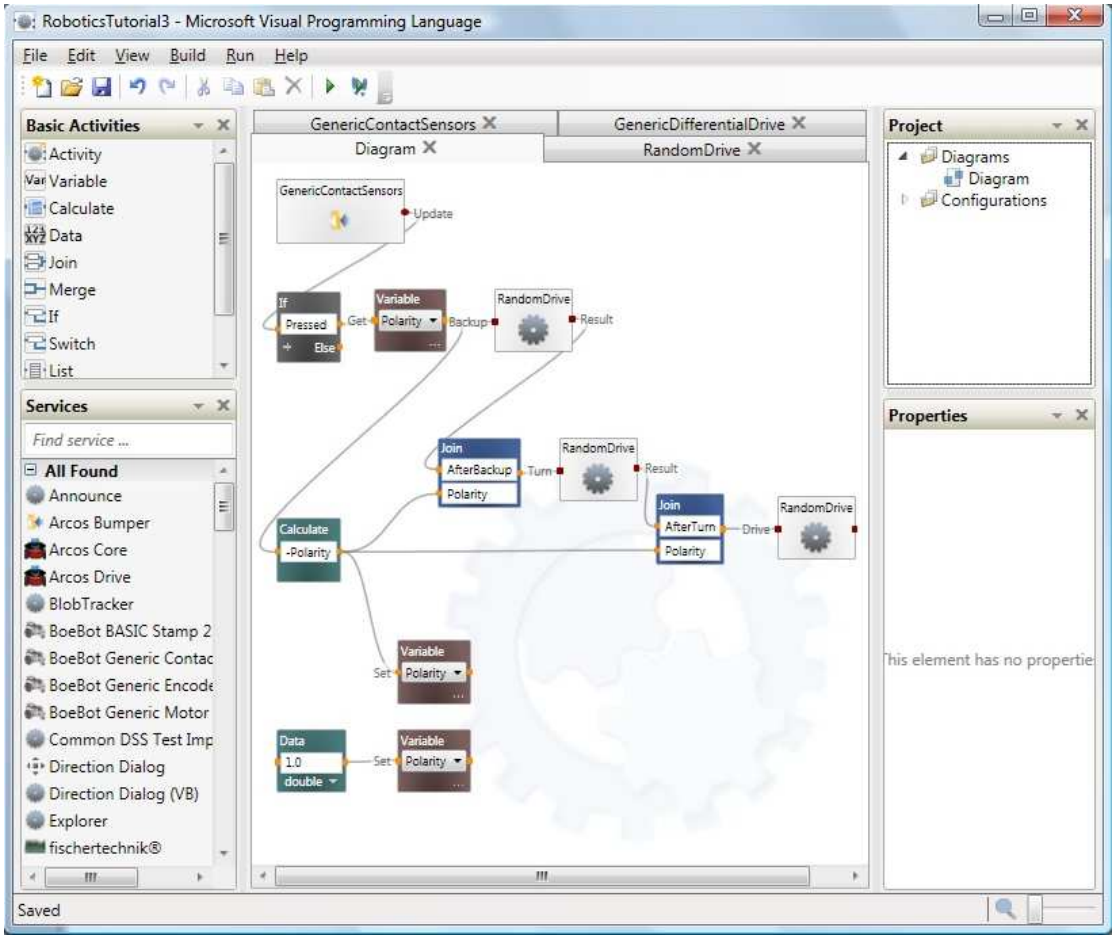
Κεφάλαιο 2

Microsoft Visual Programming Language (VPL)

2.1 Εισαγωγή στη VPL

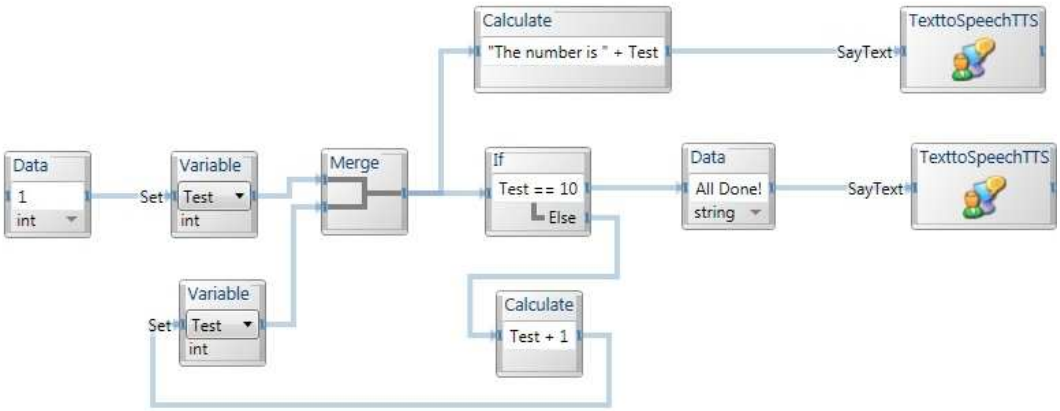
Η Microsoft Visual Programming Language (VPL) είναι ένα περιβάλλον ανάπτυξης μίας εφαρμογής, σχεδιασμένο σε ένα γραφικό μοντέλο προγραμματισμού ροής δεδομένων. Παρά τη σειρά εντολών που εκτελούνται διαδοχικά, ένα πρόγραμμα ροής δεδομένων μοιάζει με μια σειρά εργαζομένων, οι οποίοι έχουν συγκεκριμένο στόχο καθώς τα υλικά φθάνουν (**Εικόνα 2.1.1**). Κατά συνέπεια η VPL ταιριάζει καλά με τον προγραμματισμό ποικίλων ταυτόχρονων ή διανεμημένων σεναρίων επεξεργασίας.

Η VPL στοχεύει για τους αρχάριους προγραμματιστές να κατανοήσουν βασικές έννοιες όπως τις μεταβλητές και τη λογική. Εντούτοις, η VPL δεν περιορίζεται στους αρχαρίους. Η σύνθετη φύση της γλώσσας προγραμματισμού μπορεί να απευθυνθεί στους πιο προηγμένους προγραμματιστές για τη γρήγορη διαμόρφωση πρωτοτύπου ή την ανάπτυξη κώδικα. Επιπλέον, ενώ η εργαλειοθήκη της προσαρμόζεται στις αναπτυσσόμενες εφαρμογές του ρομπότ, η αρχιτεκτονική δεν περιορίζεται στα προγραμματιζόμενα ρομπότ και θα μπορούσε να εφαρμοστεί σε άλλες εφαρμογές επίσης. Κατά συνέπεια, η VPL μπορεί να απευθυνθεί σε ένα ευρύ κοινό χρηστών συμπεριλαμβανομένων των σπουδαστών, ερασιτεχνών, καθώς επίσης και ενδεχομένως των web developers και των επαγγελματιών προγραμματιστών.



Εικόνα 2.1.1 :Παράδειγμα διαγράμματος στην γλώσσα VPL.

Μια ροή πληροφοριών της VPL αποτελείται από μια συνδεδεμένη ακολουθία δραστηριοτήτων που αντιπροσωπεύει τα blocks με εισόδους και εξόδους που μπορούν να συνδεθούν με άλλα blocks δραστηριοτήτων όπως βλέπετε στην **Εικόνα 2.1.2**.



Εικόνα 2.1.2 : Μία ροή πληροφοριών της VPL.

Οι δραστηριότητες μπορούν να αντιπροσωπεύσουν τις υπάρχουσες υπηρεσίες, τον έλεγχο ροής πληροφοριών, τις λειτουργίες, ή άλλες ενότητες κώδικα. Η προκύπτουσα εφαρμογή επομένως αναφέρεται συχνά ως

orchestration, ή αλληλουχία των χωριστών διαδικασιών.

Οι δραστηριότητες μπορούν επίσης να συμπεριλάβουν συνθέσεις άλλων δραστηριοτήτων. Αυτό καθιστά πιθανό να συνθέτονται δραστηριότητες και να επαναχρησιμοποιείται η σύνθεση ως block. Από αυτή την άποψη μια εφαρμογή που δημιουργείται σε VPL είναι από μόνη της μία δραστηριότητα.

Τα blocks δραστηριοτήτων περιλαμβάνουν το όνομα και τα σύνορα της δραστηριότητας που αντιπροσωπεύουν τα σημεία σύνδεσής τους. Ένα block δραστηριότητας μπορεί επίσης να περιλαμβάνει γραφικά για να αναπαραστήσει το σκοπό της δραστηριότητας καθώς επίσης στοιχεία που μπορούν να επιτρέψουν στο χρήστη να εισάγει τιμές, αναθέσεις, ή μετασχηματισμούς για τα στοιχεία που χρησιμοποιούνται σε μια δραστηριότητα (**Εικόνα 2.1.3**).

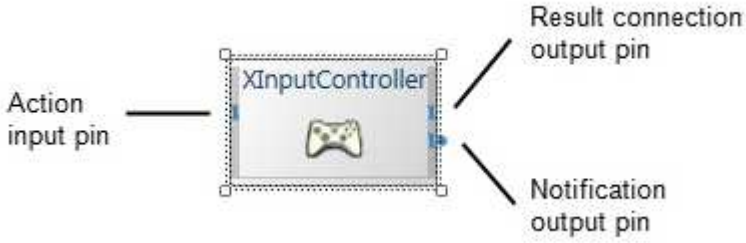
Εικόνα 2.1.3 : Ένα block τύπου Data και άλλο ένα τύπου Text to SpeechTTS.

Οι δραστηριότητες συνδέονται μέσω των ακίδων σύνδεσής τους. Μια ακίδα σύνδεσης στην αριστερή πλευρά μιας δραστηριότητας αντιπροσωπεύει το σημείο σύνδεσης για τα εισερχόμενα/εισαγμένα μηνύματα και μια ακίδα στα δεξιά αντιπροσωπεύει το σημείο σύνδεσής για τα εξερχόμενα/μηνύματα .

Μια δραστηριότητα λαμβάνει μηνύματα που περιέχουν στοιχεία μέσω των ακίδων σύνδεσης εισόδου. Οι ακίδες εισόδου μιας δραστηριότητας είναι σημεία σύνδεσης στις προκαθορισμένες εσωτερικές λειτουργίες γνωστές ως ενέργειες ή χειριστές (που μπορούν είτε ως λειτουργίες να παρέχονται από μια υπηρεσία).

Ένα block δραστηριότητας ενεργοποιεί και επεξεργάζεται τα εισερχόμενα μηνυμάτων μόλις θα λάβει ένα έγκυρο εισερχόμενο μήνυμα. Όλα τα δεδομένα που στέλνονται στη δραστηριότητα καταναλώνονται από τη δραστηριότητα. Για τα στοιχεία που διαβιβάζονται μέσω της εξόδου της δραστηριότητας, η λαμβάνουσα δραστηριότητα πρέπει να «ξαναδιπλώσει» τα στοιχεία και να τα βάλει στην έξοδο της σύνδεσης.

Μια δραστηριότητα μπορεί να έχει πολλαπλές ακίδες σύνδεσης εισόδου και κάθε μια το δικό της σετ ακίδων σύνδεσης εξόδου . Οι ακίδες σύνδεσης εξόδου μπορούν να είναι ένα από δύο είδη: έξοδος αποτελέσματος (result output) ή έξοδος ανακοίνωσης (notification output). Οι έξοδοι αποτελεσμάτων (result outputs) εμφανίζονται ως ορθογώνιες ακίδες σύνδεσης ενώ οι έξοδοι ανακοίνωσης (publication outputs) έχουν κυκλικές ακίδες σύνδεσης (**Εικόνα 2.1.4**).



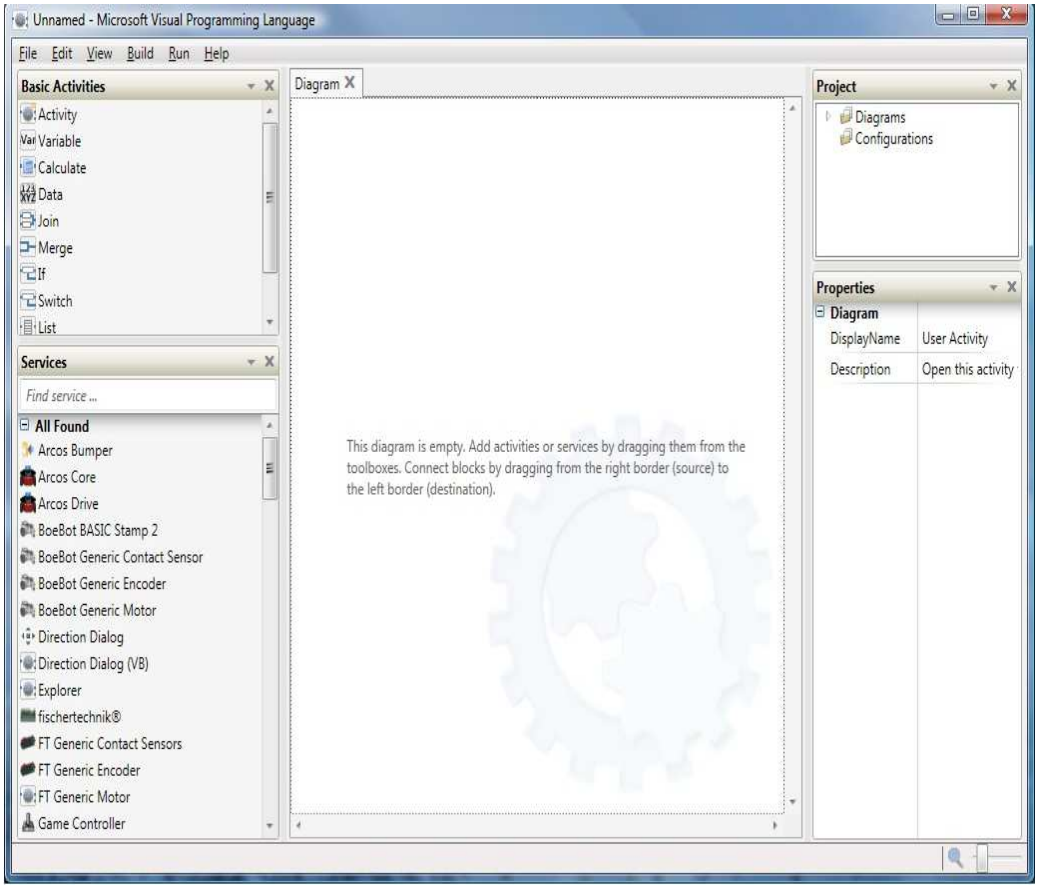
Εικόνα 2.1.4 : Ορθογώνια ακίδα και κυκλική ακίδα εξόδου .

Η ακίδα ανταπόκρισης εξόδου (result output) χρησιμοποιείται στις καταστάσεις όπου ένα εξερχόμενο μήνυμα (στοιχεία) στέλνεται ως αποτέλεσμα ενός συγκεκριμένου εισερχόμενου μηνύματος δράσης. Οι ακίδες ανακοίνωσης (notification pins) μπορούν να στείλουν τις πληροφορίες ως αποτέλεσμα ενός εισερχόμενου μηνύματος.

Μπορούν επίσης να παραγάγουν μηνύματα πολλές φορές, ενώ μια ακίδα αποτελέσματος στέλνει ένα ενιαίο μήνυμα στην παραλαβή ενός εισερχόμενου μηνύματος. Έτσι οι ακίδες ανακοίνωσης εξόδου χρησιμοποιούνται για την αποστολή μηνυμάτων χωρίς να πρέπει επανειλημμένα να ζητήσουν ή να καταγράψουν για την κατάσταση μιας δραστηριότητας.

2.2 Δημιουργία Διαγράμματος

Για την έναρξη της VPL, επιλέγεται η είσοδό της από το μενού έναρξης. Όταν η VPL ξεκινήσει εμφανίζεται ένα παράθυρο με ένα σύνολο από μενού, tool boxes, και μιας σελίδας διαγραμμάτων όπως φαίνεται παρακάτω στην **Εικόνα 2.2.1**.



Εικόνα 2.2.1 : Παράθυρο δημιουργίας διαγράμματος στην VPL.

Τα tool boxes εμφανίζουν τα περιεχόμενα του τρέχοντος αρχείου του project και των δραστηριοτήτων που μπορούν να χρησιμοποιηθούν για τη δημιουργία ενός διαγράμματος. Τα tool boxes είναι κινητά και πτυσσόμενα, κι έτσι μπορούν να ρυθμιστούν εκ νέου μέσα στο κύριο παράθυρο VPL.

Υπάρχει επίσης η δυνατότητα να εμφανιστεί ένα κρυμμένο παράθυρο tool box ή να κρυφτεί επιλεκτικά από το χρήστη, χρησιμοποιώντας την εντολή tool boxes στο μενού View.

Για να αρχίσει ένα διάγραμμα ροής πληροφοριών, 'σύρατε και αφήστε' δεδομένα ή blocks από τα toolboxes στην περιοχή διαγραμμάτων και συνδέστε τα. Μπορείτε επίσης να προσθέσετε μια νέα δραστηριότητα με το να κάνετε διπλό κλικ στην είσοδο του toolbox.

Μπορείτε να κόψετε, να αντιγράψετε, να κολλήσετε, ή να διαγράψετε τις δραστηριότητες με την επιλογή του block στο διάγραμμα χρησιμοποιώντας τις εντολές στο μενού Edit .

- **Οι βασικές δραστηριότητες του toolbox window**

Οι βασικές δραστηριότητες του toolbox window περιλαμβάνουν blocks για τον έλεγχο της ροής πληροφοριών και τη δημιουργία των στοιχείων και των μεταβλητών (δείτε **Εικόνα2.2.2**). Επίσης περιλαμβάνεται μια δραστηριότητα σχόλιου (comment activity) που επιτρέπει σε σας να τοποθετήσετε blocks σχόλιου κειμένων στο διάγραμμά σας.

Εικόνα 2.2.2 : Τα blocks που αντιπροσωπεύουν τις βασικές δραστηριότητες (basic activities).

- **Οι υπηρεσίες του toolbox window**

Οι υπηρεσίες του toolbox window εμφανίζουν υπηρεσίες που είναι συμβατές με την VPL (**Εικόνα 2.2.3**). Όταν μια υπηρεσία μεταφερθεί με σύρσιμο από το toolbox που υπάρχει ήδη στο διάγραμμα, γίνεται ερώτηση στο χρήστη για το εάν θέλει να δημιουργήσει μια νέα περίπτωση της υπηρεσίας ή θέλει να δημιουργήσει μια πανομοιότυπη της υπηρεσίας που υπάρχει ήδη στο διάγραμμα. Παραδείγματος χάριν μπορεί να χρησιμοποιηθεί η ίδια υπηρεσία για ένα σύνολο αισθητήρων ή μηχανών, αλλά με διαφορετικές ρυθμίσεις διαμόρφωσης. Υπάρχει η δυνατότητα αλλαγής των ονομάτων των υπηρεσιών που εισάγονται στο διάγραμμα με τη πληκτρολόγηση του νέου ονόματος στο text box στην κορυφή του toolbox. Οι υπηρεσίες που εμφανίζονται σε αυτό το toolbox window εξαρτώνται από το περιεχόμενο του box (πλαισίου) φίλτρων αναζήτησης. Η είσοδος μιας λέξης αναγκάζει μόνο εκείνες τις υπηρεσίες με εκείνη την λέξη να εμφανιστούν στο παράθυρο. Η αναζήτηση ψάχνει τις αντιστοιχίες και στο όνομα της υπηρεσίας καθώς επίσης και στην περιγραφή της. Ένα κενό box αναζήτησης βρίσκει όλες τις υπηρεσίες. Μπορούν να εισαχθούν πολλαπλάσιες λέξεις (που χωρίζονται με τα διαστήματα) και ο κατάλογος φιλτράρεται από όλες εκείνες τις λέξεις. Η τοποθέτηση του α **+** μεταξύ των λέξεων αναγκάζει το φίλτρο να υποθέσει ότι και οι δύο λέξεις πρέπει να εμφανιστούν. Χρησιμοποιώντας το α **-** αποκλείει τη λέξη. Παρέχεται ακόμη η δυνατότητα αποθήκευσης ενός φίλτρου αναζήτησης κάνοντας κλικ στο + κουμπί που βρίσκεται στο All Found. Αυτό δημιουργεί ένα νέο τμήμα στο service toolbox window με τα αποτελέσματα του φίλτρου αναζήτησης χρησιμοποιώντας τις λέξεις που χρησιμοποιήθηκαν ως φίλτρο αναζήτησης. Για τη δημιουργία ενός τίτλου για το φίλτρο αναζήτησης χωρίς τις λέξεις του φίλτρου, πρέπει να προστεθεί "**= τίτλος** " στο κείμενο αναζήτησης. Τώρα όταν πατηθεί κλικ στο + κουμπί στο All Found, το νέο τμήμα θα χρησιμοποιήσει ακριβώς εκείνο το κείμενο. Παραδείγματος χάριν εάν εισαχθούν τα εξής:

NXT RCX = LEGO services

και έπειτα πατηθεί κλικ στο + κουμπί, θα εμφανιστεί ένα νέο τμήμα με ακριβώς εκείνες τις υπηρεσίες των οποίων το όνομα ή η περιγραφή περιλαμβάνει εκείνες τις λέξεις, αλλά με τον τίτλο ως υπηρεσίες LEGO.

Εικόνα 2.2.3 : Οι υπηρεσίες του toolbox window.

- **Το project window**

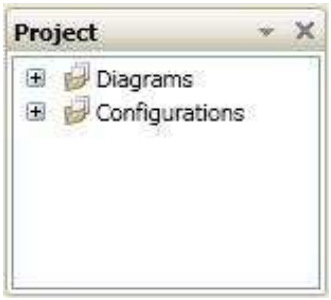
Το project window παρουσιάζει τα διαγράμματα και τα αρχεία διαμόρφωσης (για τις υπηρεσίες) που περιλαμβάνονται στο πρόγραμμα (**Εικόνα 2.2.4**).

Για να κλείσει η τρέχουσα κατάσταση του διαγράμματος, πρέπει να πατηθεί κλικ στο κουμπί X.

Για να εμφανιστεί ξανά το διάγραμμα, πρέπει να πατηθεί δύο φορές το όνομα του διαγράμματος στο project window. Εάν το όνομα διαγράμματος δεν εμφανίζεται, πρέπει να πατηθεί κλικ στο + κουμπί για να επεκτείνει την είσοδο διαγράμματος.

Για αλλαγή ονόματος του διαγράμματος, πρέπει να πατηθεί κλικ κατά την εισαγωγή του στο project window. Για να προστεθούν τα πρόσθετα διαγράμματα στο πρόγραμμα, κάνετε κλικ στο πρόσθεση διαγράμματος.

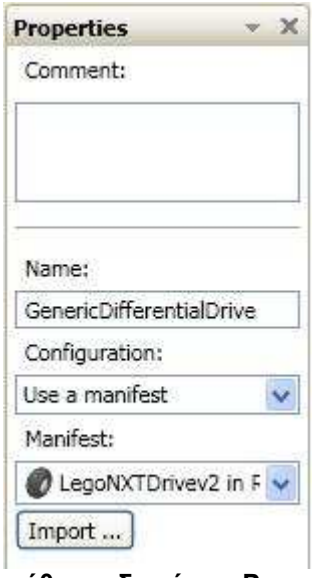
Για διαγραφή ενός διαγράμματος κάνετε κλικ στην διαγραφή.



Εικόνα2.2.4 : Το project window.

- **Το παράθυρο ιδιοτήτων**

Το παράθυρο ιδιοτήτων (Properties window) εμφανίζει τις ιδιότητες για το τρέχον επιλεγμένο στοιχείο (**Εικόνα 2.2.5**). Αυτό επιτρέπει στο χρήστη να διαμορφώσει τις ρυθμίσεις που εκτίθενται από εκείνη την υπηρεσία ή δραστηριότητα.

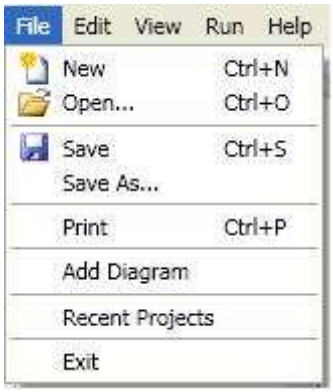


Εικόνα 2.2.5 : Το παράθυρο ιδιοτήτων Properties Window.

2.3 Το μενού της VPL (VPL MENUS)

- **File**

Οι λειτουργίες της επιλογής File αναλύονται παρακάτω και φαίνονται και στην **Εικόνα 2.3.1** που ακολουθεί.



Εικόνα 2.3.1 : Οι λειτουργίες της επιλογής File.

New – Επιτρέπει τη δημιουργία ενός καινούριου project.
Open – Ανοίγει ένα υπάρχον project αρχείο
Save – Αποθηκεύει το τρέχων project.
Save As – Αποθηκεύει το project με ένα συγκεκριμένο όνομα.
Recent Projects – Εκθέτει και ανοίγει πρόσφατα project αρχεία

Exit – Έξοδος από την VPL

- **Edit**

Οι λειτουργίες της επιλογής Edit αναλύονται παρακάτω και φαίνονται και στην **Εικόνα 2.3.2** που ακολουθεί.

Εικόνα 2.3.2 : Οι λειτουργίες της επιλογήςEdit.

Undo – Αντιστρέφει την τελευταία δράση
Redo – Αποκαθιστά την τελευταία ανεστραμμένη δράση.
Cut – Αφαιρεί το επιλεγμένο αντικείμενο
Copy – Αντιγράφει το επιλεγμένο αντικείμενο.
Paste – Εισάγει την τελευταία περικοπή ή το αντιγραμμένο αντικείμενο.
Del – Αφαιρεί το επιλεγμένο αντικείμενο.
Actions and Notifications – Εμφανίζει ένα πλαίσιο διαλόγου- dialog box που επιτρέπει την προσθήκη ή τη λειτουργία ενεργειών δραστηριότητας χρηστών, τα αποτελέσματα, και τις ανακοινώσεις.

Define Variables – Εμφανίζει ένα πλαίσιο διαλόγου- dialog box που επιτρέπει τον καθορισμό μεταβλητών και του τύπου τους.

Connections – Εμφανίζει ένα πλαίσιο διαλόγου- dialog box για να τεθούν οι συνδέσεις μεταξύ δύο δραστηριοτήτων.

Data Connections – Εμφανίζει ένα πλαίσιο διαλόγου- dialog box για να τεθούν οι συνδέσεις στοιχείων για μια σύνδεση.

Set Configuration – Εμφανίζει το πλαίσιο διαμόρφωσης που επιτρέπει τον καθορισμό της διαμόρφωσης μιας υπηρεσίας και τις τοποθετήσεις των συνεταιρικών υπηρεσιών.

- **View**

Οι λειτουργίες της επιλογής View αναλύονται παρακάτω και φαίνονται και στην **Εικόνα 2.3.3** που ακολουθεί.

Εικόνα 2.3.3 : Οι λειτουργίες της επιλογήςView.

Toolboxes – Εκθέτει ή κρύβει τις βασικές δραστηριότητες, τις υπηρεσίες, το πρόγραμμα, και τα παράθυρα ιδιοτήτων.
Grid – Εκθέτει ή κρύβει ένα οπτικό πλέγμα για το διάγραμμα.
Toolbar – Εκθέτει ή κρύβει την VPL μπάρα εργαλείων.

- **Run**

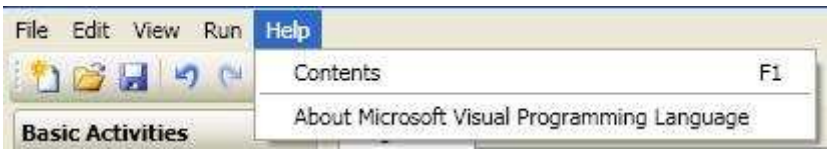
Οι λειτουργίες της επιλογής Run αναλύονται παρακάτω και φαίνονται και στην **Εικόνα 2.3.4** που ακολουθεί.

Εικόνα 2.3.4 : Οι λειτουργίες της επιλογής Run.

Start – Ξεκινά να εκτελείται το τρέχον project.
Debug Start – Ξεκινά να εκτελείται το τρέχον project, αλλά σταματάει στη 1η δραστηριότητα, και εκθέτει το αποτέλεσμα.
Port Settings – Εκθέτει ένα πλαίσιο διαλόγου - dialog box που μας δίνει τη δυνατότητα να θέσουμε θύρες τις οποίες το project χρησιμοποιεί όταν το εκτελείτε.

- **Help**

Οι λειτουργίες της επιλογής Help αναλύονται παρακάτω και φαίνονται και στην **Εικόνα 2.3.5** που ακολουθεί.



Εικόνα 2.3.5 : Οι λειτουργίες της επιλογής Help.

Contents – Εκθέτει το VPL αρχείο βοήθειας.

2.4 Βασικές Δραστηριότητες (Basic Activities)

Η γλώσσα προγραμματισμού Microsoft Visual Programming Language περιλαμβάνει ένα σύνολο βασικών δραστηριοτήτων που χρησιμοποιούνται για να βοηθήσουν να δημιουργηθεί ένα πρόγραμμα ροής στοιχείων. Αυτά τα block χρησιμοποιούνται για τη σύνδεση των block υπηρεσιών, αλλά μπορούν επίσης να συνδεθούν μεταξύ τους.

Activity

Αυτό το block (**Εικόνα 2.4.1**) χρησιμοποιείται για να μπορεί ο χρήστης να δημιουργήσει τις δικές του δραστηριότητες. Κάθε μία δραστηριότητα μπορεί να έχει διαγράμματα ροής στοιχείων. Τα διαγράμματα αυτά μπορούν να χρησιμοποιηθούν ώστε να δημιουργηθούν διαγράμματα που μπορούν να αντιπροσωπευθούν ως ενιαία blocks σε άλλα διαγράμματα. Τα block αυτά μπορούν επίσης να συνταχθούν στις υπηρεσίες που θα μπορούσαν να χρησιμοποιηθούν με άλλες υπηρεσίες.

Εικόνα 2.4.1 : Το block Activity.

Calculate

Η Calculate δραστηριότητα (**Εικόνα 2.4.2**) εκτελεί τις απλές αριθμητικές ή λογικές διαδικασίες στην έκφραση που εισάγεται που μπορεί να συμπεριλάβει τις αριθμητικές τιμές, την αξία του μηνύματος, τα μέλη στοιχείων της, ή τις προκαθορισμένες τιμές που παρέχονται από άλλες υπηρεσίες στο διάγραμμα.

Εικόνα 2.4.2 : Το block Calculate.

Για αριθμητικά δεδομένα χρησιμοποιούνται τα εξής:

+	add

-	subtract/minus
*	multiply
/	divide
%	mod

Το συν (+) μπορεί επίσης να χρησιμοποιηθεί για να συνδέσει (να συνδυάσει) τις συμβολοσειρές. Αυτό μπορεί επίσης να χρησιμοποιηθεί για να συνδυάσει το κείμενο και τα αριθμητικά στοιχεία με τη χρησιμοποίηση των διπλών εισαγωγικών (π.χ. "η απάντηση είναι" + x/4).

Για λογικούς χειριστές χρησιμοποιούνται τα ακόλουθα:

&&	AND
||	OR
!	NOT

Η παρένθεση χρησιμοποιείται επίσης για την υποστήριξη της προτεραιότητας (διαταγή της αξιολόγησης) της έκφρασης που εισήχθη.

Με κλικ στο textbox του Calculate box εμφανίζεται ένας κατάλογος συμπεριλαμβανομένης της αξίας του εισερχόμενου μηνύματος, οποιονδήποτε μελών στοιχείων, καθώς επίσης και των προκαθορισμένων τιμών που μπορούν να παρασχεθούν από άλλες υπηρεσίες.

Comment

Η δραστηριότητα Comment (**Εικόνα 2.4.3**) επιτρέπει την προσθήκη ενός block κειμένου σε ένα διάγραμμα.

Εικόνα 2.4.3 : Το block Comment.

Στο παραπάνω παράθυρο που θα εμφανιστεί μπορεί να εισαχθεί το κείμενο που θα εμφανίζετε στο textbox.

Το block Comment δεν υποστηρίζει οποιεσδήποτε συνδέσεις. Μπορεί όμως να τοποθετηθεί οπουδήποτε στο διάγραμμα.

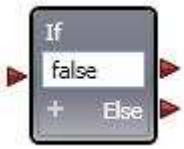
Data

Η δραστηριότητα Data (**Εικόνα 2.4.4)** χρησιμοποιείται για να παρέχει μια απλή αξία στοιχείων σε μια άλλη δραστηριότητα ή υπηρεσία. Για τον καθορισμό ενός συγκεκριμένου είδους ημερομηνίας, μπορείτε να επιλέξετε τον τύπο του από το dropdown (κυλιόμενος κατάλογος) κάτω από το textbox, και έπειτα εισάγετε μια αξία στο textbox.

Εικόνα 2.4.4 : Το block Data.

IF

Η δραστηριότητα If (**Εικόνα 2.4.5)** παρέχει μια επιλογή των εξόδων για να διαβιβαστεί το εισερχόμενο μήνυμα βασισμένο σε έναν όρο που εισάγεται. Εάν ο όρος είναι αληθινός, η πρώτη εξερχόμενη σύνδεση υποδέχεται το εισερχόμενο μήνυμα (και τα στοιχεία του). Εάν δεν είναι αληθινό, τότε η έξοδος Else χρησιμοποιείται.



Εικόνα 2.4.5 : Το block If.

Η υποθετική έκφραση μπορεί να χρησιμοποιήσει τους ακόλουθους χειριστές για αποτίμηση :

= or ==	equals
!= or <>	not equals
<	less than
>	greater than
<=	less than or equals
>=	great than or equals

Υπάρχει δυνατότητα να χρησιμοποιηθούν οι ίδιοι χειριστές στη δραστηριότητα Calculate, υπό τον όρο ότι ολόκληρη η δήλωση θα αξιολογηθεί σε αληθινός ή ψεύτικος.

Επίσης μπορεί να προστεθούν όροι στη δραστηριότητα με κλικ στο Add (+) κουμπί του block activity.

Join

Η Join δραστηριότητα (**Εικόνα 2.4.6)** συνδυάζει τη ροή δύο (ή περισσότερων) ροών πληροφοριών .

Εικόνα 2.4.6 : Το block Join.

Είναι σημαντικά διαφορετικό από το Merge στο οποίο τα δεδομένα των μηνυμάτων από τις εισερχόμενες συνδέσεις συνδυάζονται και όλα τα μηνύματα πρέπει να παραληφθούν σε όλες τις εισερχόμενες συνδέσεις πριν η δραστηριότητα περαστεί στα δεδομένα. Το κείμενο που εισάγεται στα textboxes είναι ονόματα των τοπικών μεταβλητών που αντιπροσωπεύουν τα μηνύματα. Μπορεί οι μεταβλητές να χρησιμοποιηθούν άμεσα ή με διάφορες αναφορές όπως (π.χ. x.a) για παραπομπή σε ένα μέλος στοιχείων (υπό-στοιχείων).

List

Η δραστηριότητα List (**Εικόνα 2.4.7)** δημιουργεί έναν κενό κατάλογο στοιχείων.

Εικόνα 2.4.7 : Το block List.

Για τη δημιουργία ενός καταλόγου, επιλέγετε αρχικά το τύπο για τα αντικείμενα (τύπος στοιχείων) από το dropdown του καταλόγου επιλογών στο block δραστηριότητας (ή στο properties view).

Για προσθήκη εισόδων στον κατάλογο, χρησιμοποιείτε το List Functions activity.

Για αποθήκευση ενός καταλόγου ώστε να χρησιμοποιηθεί σε άλλο μέρος στο διάγραμμα, δημιουργείτε μια μεταβλητή καταλόγων μέσω της Variable δραστηριότητας.

List Functions

Η δραστηριότητα List Function (**Εικόνα 2.4.8)** επιτρέπει την τροποποίηση ενός υπάρχοντα καταλόγου.

Εικόνα 2.4.8 : Το block List Functions.

Με χρήση της λίστας dropdown του block επιλέγετε τη λειτουργία που θα εφαρμοστεί σε έναν κατάλογο .

Merge

Η δραστηριότητα Merge (**Εικόνα 2.4.9)** συγχωνεύει απλά τη ροή δύο (ή περισσότερων) πληροφοριών από κοινού. Δεν υπάρχει καμία εξάρτηση στον άλλο κλάδο όταν δίνονται τα μηνύματα. Ο ρόλος της δραστηριότητας είναι απλά να περαστεί το μήνυμα εμπρός στην επόμενη δραστηριότητα.

Εικόνα 2.4.9 : Το block Merge.

Switch

Η δραστηριότητα Switch (**Εικόνα 2.4.9a)** μπορεί να χρησιμοποιηθεί για την δρομολόγηση των μηνυμάτων και βασίζεται στο ότι το εισερχόμενο μήνυμα ταιριάζει με την έκφραση που εισάγεται στο textbox.

Εικόνα 2.4.9a : Το block Switch.

Παρέχεται η δυνατότητα προσθήκης κλάδων περίπτωσης - Case branches (όροι αντιστοιχιών) στο block δραστηριότητας με κλικ στο Add (+) κουμπί της block δραστηριότητας.

Variable

Η δραστηριότητα Variable (**Εικόνα 2.4.9b)** επιτρέπει τη δημιουργία μιας μεταβλητής για την οποία τίθεται ή λαμβάνεται η αξία της.

Εικόνα 2.4.9b : Το block Variable.

Μια μεταβλητή επιλέγεται από το dropdown κατάλογο που εμφανίζεται με κλικ στο drop κουμπί που βρίσκεται δίπλα στο textbox. Εάν δεν έχει καθοριστεί η μεταβλητή ή επιθυμείτε τη δημιουργία μιας νέας μεταβλητής, επιλέγετε το Define Variables από τον κατάλογο (ή από το Edit μενού). Αυτό θα εμφανίσει το Define Variables dialog box όπου παρέχεται η δυνατότητα προσθήκης μιας μεταβλητής και ο καθορισμός του τύπου της. Οι τύποι μεταβλητών περιλαμβάνουν τους τύπους καταλόγων.

Τα ονόματα μεταβλητών διακρίνονται σε κεφαλαία και μικρά. Έτσι κατά την παραπομπή μιας μεταβλητής πρέπει να δίνετε ιδιαίτερη προσοχή ώστε να χρησιμοποιείτε η σωστή κάθε φορά. Τα ονόματα πρέπει επίσης να αρχίσουν με ένα γράμμα και να περιλάβουν μόνο τους αλφαριθμητικούς χαρακτήρες.

Οι δραστηριότητες variable είναι απλά κατασκευάσματα που υποστηρίζουν μια σύνδεση GetValue (για να πάρει την αξία του) καθώς επίσης και μια σύνδεση SetValue (για να θέσει την αξία του). Κατά τη χρησιμοποίηση της σύνδεσης SetValue, η σύνδεση εξόδου της δραστηριότητας περνά επίσης τη μεταβλητή σε εξερχόμενη σύνδεση.

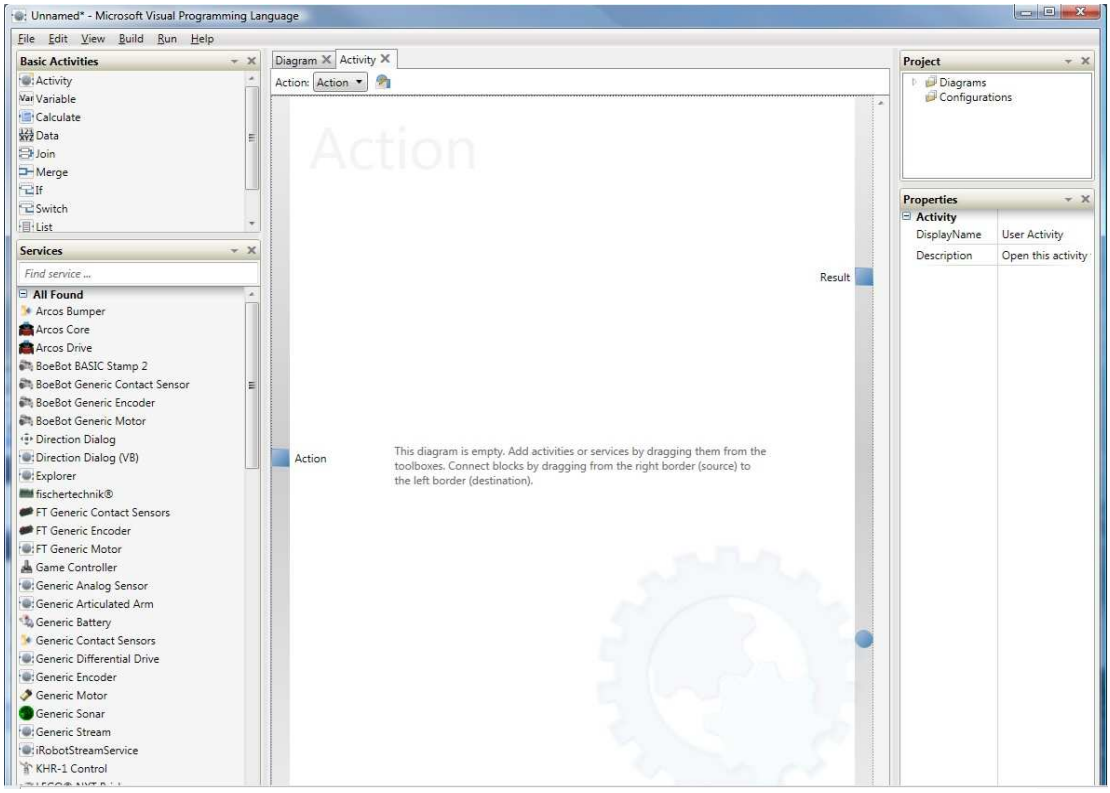
2.5 Τύποι Δεδομένων (Data Types)

Η Microsoft Visual Programming Language υποστηρίζει .NET Visual C# τύπο δεδομένων .

VPL Type	Description
Bool	Boolean values: true, false
Byte	8 bit unsigned integer
Sbyte	8 bit signed integer
Char	character
Decimal	fixed point decimal number
Double	double precision floating point number
Float	single precision floating point number
Int	32 bit signed integer
Uint	32 bit unsigned integer
Long	64 bit signed integer
Ulong	64 bit unsigned integer
Short	16 bit signed integer
Ushort	16 bit unsigned integer
String	character string (text)

2.6 Δημιουργία και επεξεργασία δραστηριοτήτων

Με την VPL μπορεί να δημιουργηθούν δραστηριότητες. Αυτό επιτυγχάνεται με σύρσιμο και άνοιγμα ενός εικονιδίου Activity από το Basic Activities toolbox. Αυτό θα ανοίξει ένα νέο διάγραμμα για τη δραστηριότητα όπως φαίνεται και στην **Εικόνα 2.6.1** που ακολουθεί.

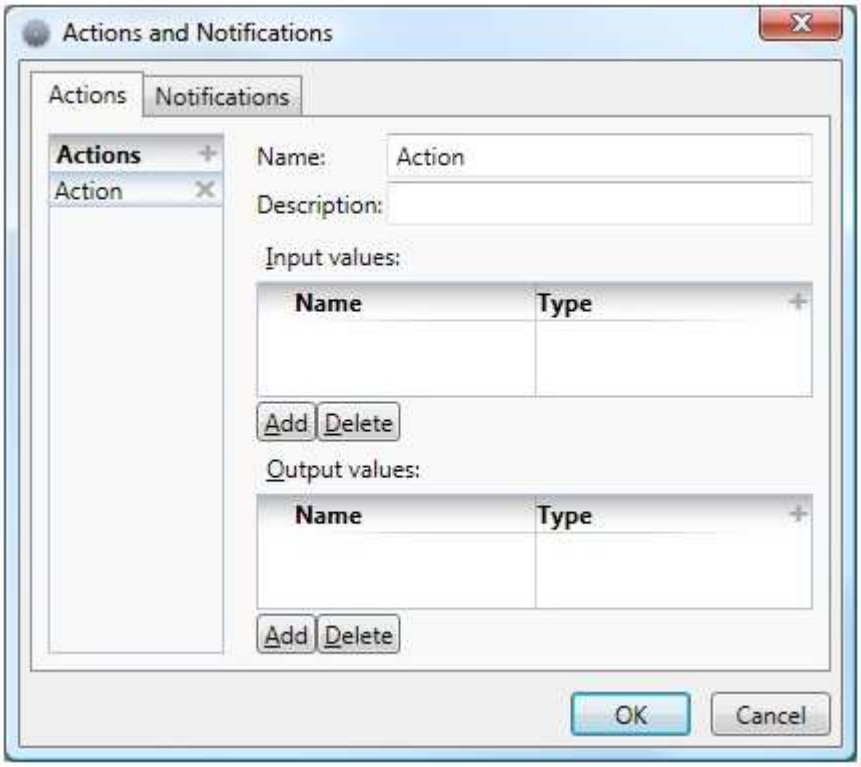


Εικόνα 2.6.1 : Το παράθυρο της VPL που μας δείχνει ένα νέο διάγραμμα της δραστηριότητας που επιλέξαμε.

Η VPL δημιουργεί μία φόρμα Action. Εάν η σελίδα Action δεν εμφανίζεται όταν ανοίγει η δραστηριότητα, χρησιμοποιείτε το Action dropdown ακριβώς κάτω από την tab περιοχή. Παρέχεται στη συνέχεια η δυνατότητα παρεμβολής άλλων δραστηριοτήτων καθώς και η σύνδεσή τους όπως και στο κύριο διάγραμμα ροής πληροφοριών.

Για να μπορεί η εσωτερική ροή των πληροφοριών να είναι χρησιμοποιήσιμη μέσα από μια άλλη ροή πληροφοριών, πρέπει η εσωτερική ροή πληροφοριών να συνδεθεί με τις εξωτερικές ακίδες της δραστηριότητας. Αυτό γίνεται με σύρσιμο μιας σύνδεσης από την ακίδα action στην είσοδο, στην ακίδα σύνδεσης action στα αριστερά και άλλη μία σύνδεση από την ακίδα στην έξοδο στην ακίδα σύνδεσης Result στα δεξιά.

Για τον καθορισμό του ονόματος και του τύπου δεδομένων (data type) για τις εξωτερικές ακίδες σύνδεσης της δραστηριότητας, επιλέγεται η εντολή Actions and Notifications από το μενού View (**Εικόνα 2.6.2**). Για αλλαγή του ονόματος ή του τύπου επιλέγεται η είσοδος από τον κατάλογο, γίνονται οι απαραίτητες αλλαγές στις καταχωρήσεις τους και στη συνέχεια με κλικ στο ok (εντάξει) ολοκληρώνεται η διαδικασία.



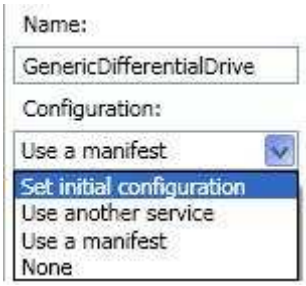
Εικόνα 2.6.2 : Το παράθυρο της επιλογής Actions and Notifications.

Εάν η ροή πληροφοριών που δημιουργείται μέσα στη δραστηριότητα παράγει έξοδο πολλαπλές φορές, πρέπει να συνδεθεί η έξοδός του με μια σύνδεση Notification. Μπορεί να χρησιμοποιηθεί το dialog Actions and Notifications, με κλικ στο Notifications tab, κατόπιν με κλικ στο κουμπί Add και στη συνέχεια με εισαγωγή του ονόματος και του τύπου. Η διαδικασία ολοκληρώνεται με κλικ στο ok (ΕΝΤΑΞΕΙ). Τώρα με σύρσιμο μπορεί να γίνει η σύνδεση από την έξοδο της ροής πληροφοριών στην υπόλοιπη notification σύνδεση που δημιουργήθηκε. Μπορεί επίσης να χρησιμοποιηθεί το dialog Actions and Notifications για τη δημιουργία πρόσθετων χειριστών δράσης (action handlers) για τη δραστηριότητά σας. Η προσθήκη ενός άλλου χειριστή δράσης, γίνεται με κλικ στο Actions tab σε αυτό το dialog και στη συνέχεια με κλικ στο Add κουμπί. Για μεταπήδηση στην πρόσφατα δημιουργημένη δράση, επιλέγετε με κλικ από το Action dropdown στην κορυφή της σελίδας της δραστηριότητας. Μερικές φορές μπορεί να δημιουργηθεί μια δραστηριότητα που εκτελεί κάποια λειτουργία, αλλά δεν χρειάζεται συνδέσεις εισόδου ή εξόδου. Σε αυτές τις περιπτώσεις χρησιμοποιείται το Start action. Η VPL περιλαμβάνει αυτόματα μια σελίδα Start action για τη δραστηριότητα. Αυτό μπορεί να χρησιμοποιηθεί για την υποβολή μιας ροής πληροφοριών που πρόκειται να εκτελεστεί όταν αρχίζει η δραστηριότητα, ανεξάρτητα από το τι αποτελούν οι συνδέσεις της.

2.7 Διαμόρφωση Υπηρεσίας (configuring service)

Οι υπηρεσίες απαιτούν διαμόρφωση πληροφοριών για το πώς πρέπει να αρχίσουν (ρυθμίσεις για την αρχική κατάσταση) και τι μπορεί να χρειάζονται οι "συνεταιρικές υπηρεσίες" (partner services) για να αρχίσουν με την υπηρεσία. Αυτά εμφανίζονται στο Properties window όταν επιλέγεται η υπηρεσία ή εμφανίζεται στην σελίδα Set Configuration page με κλικ στην Set Configuration εντολή από το Edit μενού ή με διπλό κλικ στο block υπηρεσιών. Ανάλογα με την υπηρεσία, παρέχονται μια από τις τέσσερις επιλογές (οι οποίες φαίνονται παρακάτω στην **Εικόνα 2.7.1**) για να τεθεί η αρχική διαμόρφωση μιας υπηρεσίας.

- **Αρχική διαμόρφωση**



Εικόνα 2.7.1 : Οι τέσσερις επιλογές κατάστασης που μπορεί να έχει μία υπηρεσία και η επιλογή της αρχικής διαμόρφωσης.

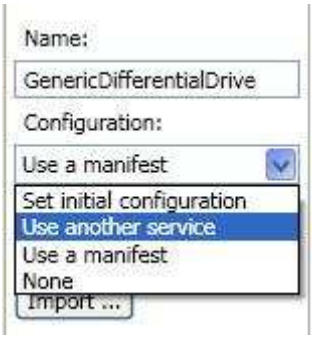
Εδώ ορίζονται οι τιμές αρχικής κατάστασης για την υπηρεσία και παρέχεται η δυνατότητα επιλογής των συνεταιρικών υπηρεσιών (υπηρεσίες που παρέχουν την υποστήριξη για την υπηρεσία) που χρησιμοποιούν την υπηρεσία. Για τις συνεταιρικές υπηρεσίες που έχουν επιλεγεί μπορεί να επιλεγεί η χρήση του καθορισμού της υπηρεσίας για το πώς να θέσει τις συνεταιρικές ή να επιλέξετε μια άλλη υπηρεσία να χρησιμοποιηθεί.

Μια υπηρεσία μπορεί να έχει μία κατάσταση που περιλαμβάνει τη διαμόρφωσή της. Με τον καθορισμό της αρχικής κατάστασης διαμορφώνεται η υπηρεσία. Η διαμόρφωση μπορεί να περιλάβει ιδιότητες της κατάστασης όπως σειριακές πόρτες επικοινωνίας και κλίμακα baud, καθώς επίσης φυσικές ιδιότητες όπως τη διάμετρο και την απόσταση μεταξύ των τροχών. Μπορείτε να θέσετε αυτές τις αρχικές ιδιότητες διαμόρφωσης είτε στο Properties window είτε στο Initial State tab της σελίδας Set Configuration page .

Μερικές υπηρεσίες καθορίζουν τις συνεταιρικές που χρησιμοποιούνται για τη λειτουργία τους. Παραδείγματος χάριν μια υπηρεσία κίνησης differential drive χρειάζεται δύο υπηρεσίες μηχανών. Προαιρετικά μπορεί επίσης να χρησιμοποιήσει δύο κωδικοποιητές τροχών. Παρέχετε η δυνατότητα καθορισμού των υπηρεσιών που θα χρησιμοποιηθούν για εκείνες τις συνεταιρικές με το σύρσιμο τους από το toolbox στις εισόδους τους στο Partner tab της σελίδας Set Configuration page ή με την επιλογή τους από την λίστα στο Properties window. Οι υπηρεσίες που έχουν επιλεγεί ως συνεταιρικές μπορούν επίσης να διαμορφωθούν.

Όταν τεθεί η αρχική διαμόρφωση μιας υπηρεσίας, η VPL προσθέτει αυτό ως αρχείο στο πρόγραμμα. Εάν διαγραφεί μια υπηρεσία, κι έχει τεθεί η αρχική διαμόρφωσή της, η VPL θα εμφανίσει μια προειδοποίηση ρωτώντας το χρήστη για να επιβεβαιώσει τη διαγραφή. Αν επιβεβαιωθεί τότε θα διαγραφεί και το αρχείο διαμόρφωσης. Μπορεί επίσης να διαγραφεί ένα αρχείο διαμόρφωσης με την επιλογή του και την επίδειξη του μενού περιεχομένου και με κλικ στη διαγραφή.

- **Χρησιμοποιήστε μια άλλη υπηρεσία**



Εικόνα 2.7.2 : Οι τέσσερις επιλογές κατάστασης που μπορεί να έχει μία υπηρεσία και η επιλογή της χρησιμοποίησης μίας άλλης υπηρεσίας.

Εδώ χρησιμοποιείται η εφαρμογή μιας άλλης υπηρεσίας. Μπορεί να επιλεγεί μια συγκεκριμένη υπηρεσία για χρήση ή ο dss χρόνος εκτέλεσης που βρίσκει ή δημιουργεί μια αρμόδια υπηρεσία. Αυτό επιτρέπει στο χρήστη να χρησιμοποιήσει τις γενικές (generic) υπηρεσίες στο διάγραμμα και να αλλάξει την υπηρεσία που χρησιμοποιείται χωρίς αλλαγή του διαγράμματος. Η υπηρεσία που έχει επιλεγεί μπορεί επίσης να διαμορφωθεί.

- **Χρησιμοποιήστε ένα manifest**

Εικόνα 2.7.3 : Οι τέσσερις επιλογές κατάστασης που μπορεί να έχει μία υπηρεσία και η επιλογή της χρησιμοποίησης ενός manifest αρχείου .

Αυτή η επιλογή (δείτε **Εικόνα 2.7.3**) επιτρέπει να επιλεχτεί ένα υπάρχον manifest αρχείο για να αρχίσει την υπηρεσία. Ένα *manifest* είναι ένα ειδικό αρχείο που περιγράφει τις υπηρεσίες που αρχίζουν και τη διαμόρφωσή τους. Για να εμφανιστεί ο κατάλογος των υπαρχουσών manifests που μπορούν να επιλεχθούν χρησιμοποιείται η Import Manifest command. Στη συνέχεια υπάρχει η δυνατότητα επιλογής μιας υπηρεσίας που αρχίζει από αυτό το Manifest. Ακόμη μπορεί να επιλεγεί ο dss χρόνος εκτέλεσης για να βρεθεί ή να δημιουργηθεί μια αρμόδια υπηρεσία.

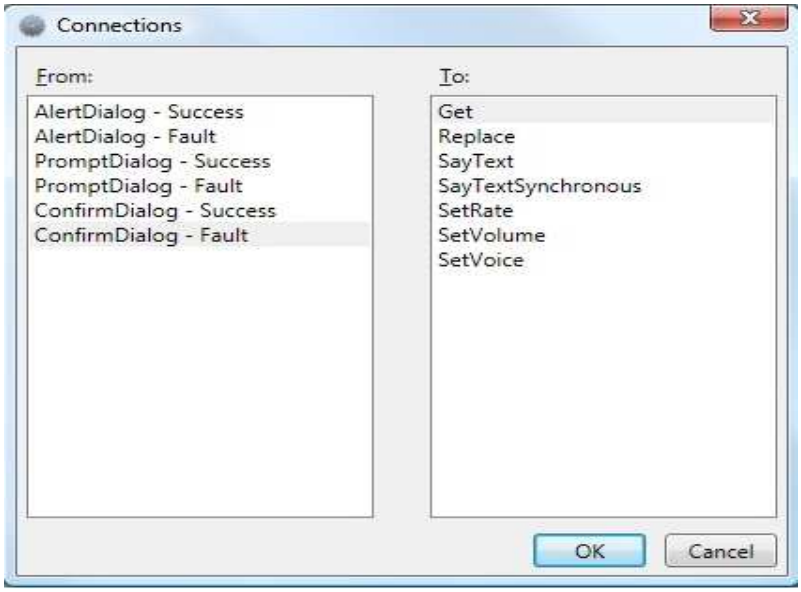
- **Επιλογή «None»**

Εικόνα 2.7.4 : Οι τέσσερις επιλογές κατάστασης που μπορεί να έχει μία υπηρεσία και η επιλογή της κατάστασης None .

Αυτή η επιλογή (δείτε **Εικόνα 2.7.4**) δείχνει ότι χρειάζεται ο dss χρόνος εκτέλεσης για να βρεθεί ή να δημιουργηθεί μια αρμόδια υπηρεσία. Αυτή η επιλογή είναι κατάλληλη για τις περισσότερες απλές υπηρεσίες (παραδείγματος χάριν, οι λειτουργίες Math, ο απλός διάλογος, ή οι text-To-Speech υπηρεσίες).

2.8 Σύνδεση Δραστηριοτήτων (Connecting activities)

Για να συνδεθούν δύο block δραστηριότητας, σύρετε μία σύνδεση από την ακίδα σύνδεσης της μιας δραστηριότητας στην ακίδα της άλλης. Εάν υπάρχουν πολλαπλές έξοδοι και είσοδοι για να συνδεθούν οι δραστηριότητες, ένα Connection Dialog box θα εμφανιστεί έτσι ώστε να καθοριστούν ποιες από τις εισόδους και εξόδους πρόκειται να συνδεθούν (δείτε **Εικόνα 2.8.1**).

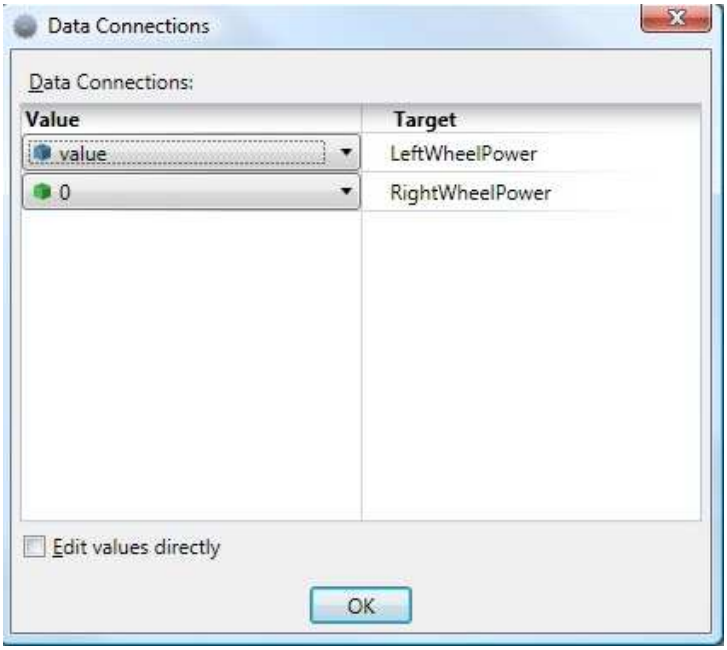


Εικόνα 2.8.1 : Το Connection Dialog Box όπου ρυθμίζουμε τις εισόδους και εξόδους που θέλουμε να συνδεθούν μεταξύ των δραστηριοτήτων.

Για να ολοκληρωθεί η σύνδεση, πρέπει να επιλεγεί μια σύνδεση εξόδου αποτελέσματος από τον αριστερό κατάλογο και μια δράση από το δεξί κατάλογο και έπειτα κλικ στο OK. Μια σύνδεση θα εμφανιστεί μεταξύ των δραστηριοτήτων. Εάν δεν υπάρχει καμία επιλογή σύνδεσης, το Connection Dialog δεν εμφανίζεται. Συχνά οι επιλογές που είναι διαθέσιμες για την έξοδο μιας δραστηριότητας μπορούν να εξαρτηθούν από αυτό που είναι η σύνδεση εισόδου της.

Μερικές φορές οι συνδέσεις μεταξύ των δραστηριοτήτων απαιτούν επίσης να ταιριάξουν τα στοιχεία που θα περαστούν μέσω των συνδέσεων. Όταν αυτό συμβεί εμφανίζεται το Data Connections dialog (δείτε **Εικόνα 2.8.2**) και εμφανίζει τις επιλογές στοιχείων του αποστολέα μηνυμάτων στην αριστερή πλευρά και τη θέση δεκτών στα δεξιά.

Οι εισερχόμενες επιλογές στοιχείων μπορούν να περιλάβουν την default τιμή για τον τύπο των στοιχείων (0 για τους αριθμητικούς τύπους,false για τους Boolean τύπους, και null για όλους τους άλλους τύπους), την τιμή των στοιχείων, ή τις τιμές υπό-στοιχείων .



Εικόνα 2.8.2 : Το Data Connections dialog εμφανίζει τις επιλογές στοιχείων του αποστολέα μηνυμάτων στην αριστερή πλευρά και τη θέση δεκτών στα δεξιά.

Υπάρχει η δυνατότητα σύνδεσης από μια δραστηριότητα σε άλλη, επιλέγοντας τη σύνδεση link και σύρσιμο του τέλους της δραστηριότητας που πρόκειται να επανασυνδεθεί. Ακόμη μπορεί να γίνει αλλαγή συνδέσεων ή στοιχείων matchups με την επιλογή των υπερεμφανιζόμενων επιλογών πλαισίου στη σύνδεση (link). Οι συνδέσεις links μπορούν να διαγραφούν με την επιλογή της link και επιλογή διαγραφής από το Edit μενού ή τις υπερεμφανιζόμενες επιλογές πλαισίου για τη σύνδεση (ή να πιεστεί το πλήκτρο **Del**).

Εάν μια δραστηριότητα υποστηρίζει διάφορες συνδέσεις εισόδου (χειριστές δράσης), παρέχεται η δυνατότητα σύνδεσης μόνο με έναν χειριστή κάθε φορά. Για να συνδεθεί μια σύνδεση με μια από τις ακίδες εισόδου μιας δραστηριότητας ενώ είναι ήδη συνδεδεμένη με μια άλλη ακίδα, πρέπει να δημιουργηθεί ένα αντίγραφο (αναφορά) του block δραστηριότητας και να γίνει σύνδεση με αυτό.

Δεν υπάρχει δυνατότητα σύνδεσης πολλαπλών άμεσων συνδέσεων στην ίδια ακίδα σύνδεσης εισόδου. Παρόλα αυτά μπορεί να χρησιμοποιηθούν δραστηριότητες Merge ή Join activities για να δρομολογηθούν ή να παγιωθούν τα μηνύματα από τις πολλαπλές δραστηριότητες στην ίδια ακίδα σύνδεσης εισόδου.

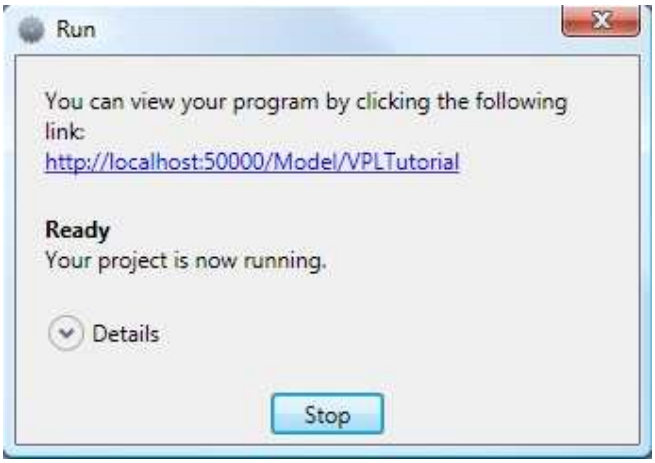
Μια δραστηριότητα μπορεί επίσης να πολλαπλασιάσει συνδέσεις εξόδου αλλά θα πρέπει να συνδέεται μόνο με μία από αυτές κάθε φορά. Έτσι εάν μια δραστηριότητα έχει και τις ακίδες σύνδεσης εξόδου αποτελέσματος και ανακοίνωσης, πρέπει να χρησιμοποιηθούν οι χωριστές αναφορές της δραστηριότητας για να συνδεθεί με τις άλλες συνδέσεις της. Εντούτοις, μπορούν να δημιουργηθούν πολλαπλές συνδέσεις στην ίδια ακίδα εξόδου.

Δεν είναι δυνατό να συνδεθούν δύο (ή περισσότερες) συνδέσεις ανακοίνωσης με μια Join δραστηριότητα. Αυτό ισχύει και για τη σύνδεση δύο ή περισσότερων ροών στοιχείων που περιλαμβάνουν τις συνδέσεις

ανακοίνωσης ανεξάρτητα από το πού βρίσκονται στη ροή στοιχείων. Παρόλα αυτά μπορεί να χρησιμοποιηθεί μια merge δραστηριότητα.

2.9 Εκτέλεση των προγραμμάτων (Running and Debugging) και Παραγωγή κώδικα (Code generation)

Για να εκτελεστεί το VPL πρόγραμμα, αρχικά επιλέγεται έναρξη από το run μενού. Αυτό θα αρχίσει την εφαρμογή. Ένα dialog box εμφανίζεται (δείτε **Εικόνα 2.9.1)** με έναν σύνδεσμο υπέρ-κειμένου καθώς επίσης και ένα κουμπί stop για να σταματήσει το πρόγραμμα.



Εικόνα 2.9.1 :Το dialog box που εμφανίζεται κατά την εκτέλεση ενός προγράμματός μας στην VPL.

Με κλικ στο απομακρυσμένο link εμφανίζεται μια συνοπτική ιστοσελίδα. Παρέχεται η δυνατότητα χρήσης των κουμπιών στη σελίδα για να σταματήσει η διαδικασία και να επιτευχθεί μετάβαση μέσω της λειτουργίας του διαγράμματος.

Επειδή το πρόγραμμα μπορεί να ολοκληρωθεί πριν επιτευχθεί η πρόσβαση στην απομακρυσμένη σελίδα, υπάρχει πιθανότητα να μην είναι εφικτός ο έλεγχος της λειτουργίας της.

Παρόλα αυτά μπορεί να χρησιμοποιηθεί η εντολή έναρξης Debug για να αρχίσει το project και να σταματήσει στο πρώτο block του, και να επιτρέψει την συνέχιση της διαδικασίας μέσω του προγράμματος. Με την επιλογή debug view υπάρχει δυνατότητα να εμφανιστούν οι τιμές των μηνυμάτων κατά τη διαδικασία μέσω του διαγράμματος. Εάν το διάγραμμα δεν είναι ορατό στη σελίδα debug, χρησιμοποιείται η εντολή Refresh για να ξαναφορτώσει η σελίδα. Το πρόγραμμα σταματά με κλικ στο κουμπί stop στο run dialog.

Δεδομένου ότι το πρόγραμμα είναι το ίδιο με μια υπηρεσία, όταν τρέχει το πρόγραμμα η VPL προωθεί το dss χρόνο εκτέλεσης και φορτώνει την υπηρεσία. Για να γίνει αυτό χρειάζεται μια θύρα HTTP και TCP. Η VPL θέτει αυτόματα τις προκαθορισμένες τιμές. Παρόλα αυτά, οι τιμές αλλάζουν με κλικ στο settings command του run menu. Για πρόσθετες debug πληροφορίες που αφορούν την εφαρμογή, χρησιμοποιείται ένας web browser για τον οποίο

πρέπει να καθοριστεί η διεύθυνσή του στη θέση της θύρας. Μια σύνδεση με αυτό εμφανίζεται στη συνέχεια στη Debug σελίδα.

Η VPL εφαρμογή μπορεί να τρέξει ανεξάρτητα από το περιβάλλον ανάπτυξης VPL. Για να γίνει αυτό χρησιμοποιείται το Compile ως εντολή υπηρεσιών από το build μενού. Αυτό θα δημιουργήσει μια υπηρεσία που μπορεί να τρέξει από το dss χρόνο εκτέλεσης χρησιμοποιώντας μια από τις ακόλουθες μεθόδους:

1. Με κλικ στην είσοδο του run dss node από τις επιλογές έναρξης (κάτω από την είσοδο του Microsoft robotics studio) ξεκινά ένας dss κόμβος. Στη συνέχεια πρέπει να πατηθεί κλικ στην είσοδο του control panel, και κατόπιν να πληκτρολογηθεί το όνομα της εργασίας στο search text box. Αυτό πρέπει να εμφανίσει το όνομα του προγράμματος με ένα Create κουμπί. Κάνοντας κλικ στο Create κουμπί θα αρχίσει το πρόγραμμα.

2. Έναρξη του Microsoft Robotics Studio Command Prompt από το μενού έναρξης (κάτω από την είσοδο του Microsoft Robotics Developer Studio). Κατόπιν πληκτρολογείτε μια εντολή που αρχίζει το dss με / p επιλογή (θύρα που θέτει) και / c (σύμβαση που θέτει) με το όνομα της σύμβασης (το οποίο μπορεί να βρεθεί στην VPL κάνοντας κλικ στην είσοδο του διαγράμματος στο Project window), του τρέχοντος έτους, του τρέχοντος μήνα, του ονόματος αρχείου του προγράμματος, και των λέξεων "diagram.html".

dsshost.exe/p:50000/c:http://schemas.tempuri.org/2006/12/helloworld/diagram.html

Εισαγωγή

Οι προηγούμενες εκδόσεις της VPL περιέχουν μια απλή ικανότητα παραγωγής κώδικα. Εντούτοις, το νέο χαρακτηριστικό γνώρισμα παραγωγής κώδικα επιτρέπει σε ένα διάγραμμα σχεδιασμένο σε VPL να μετατραπεί σε κώδικα C που θα τρέχει σε μια ή περισσότερες dss υπηρεσίες. Αυτό επιτρέπει στην ευκολία της ανάπτυξης για να συνδυαστεί με τη δύναμη και την ταχύτητα της εκτέλεσης του κώδικα C .

Επιπλέον, ο κώδικας που παράγεται μπορεί να χρησιμοποιηθεί ως αφετηρία για έναν developer για να επεκτείνει την ανάπτυξη ενός διαγράμματος στις περιοχές που η VPL δεν αντιστοιχεί, παραδείγματος χάριν, την εκτέλεση των σύνθετων υπολογισμών.

- **Παραγωγή του κώδικα**

Υπάρχουν δύο βήματα στην παραγωγή του κώδικα: διαμόρφωση και παραγωγή του κώδικα.

1 Διαμόρφωση Παραγωγής κώδικα

Το διάγραμμα πρέπει να διαμορφωθεί σωστά για την παραγωγή κώδικα.

1. Στο Project toolbox κάνετε κλικ στο φάκελο Diagrams.
2. Στο Properties toolbox πρέπει να εξασφαλιστεί ότι το Key Location προσδιορίζει σωστά τη θέση ενός ισχυρού ονόματος κλειδιού (.Snk αρχείο) Αυτό μπορεί συνήθως να βρεθεί στο" samples\mrisamples.snk»
3. Στο Properties toolbox πρέπει να διευκρινιστεί το Source Location.

Αυτό διαμορφώνει μια θέση όπου τα αρχεία κώδικα θα παραχθούν.
Σημείωση: Αυτά είναι τα ίδια βήματα για την παραγωγή του κώδικα C .

- **Ξεκίνημα παραγωγής κώδικα**

Από το Build μενού επιλέγεται το ΄Compile as a Service΄, ώστε να εμφανιστεί το dialog box "Create a Service ". Όταν ολοκληρωθεί επιτυχώς το Compile τότε θα εμφανιστεί το μήνυμα "Compilation complete”. Εάν βρεθεί λάθος κατά τη διάρκεια του Compile της υπηρεσίας θα εμφανιστεί ένα κατάλληλο μήνυμα λάθους.

Εφόσον εκτελεστεί επιτυχώς το "Compile as a Service" μία νέα υπηρεσία θα έχει δημιουργηθεί και θα έχει αντιγραφεί στον κατάλογο "bin ". Στη θέση που διευκρινίζεται στο Source Location θα υπάρξουν διάφορα αρχεία **.**cs και ένα αρχείο **.**csproj **.** Αυτά μπορούν να χρησιμοποιηθούν για τροποποίηση του παραγόμενου κώδικα και να γίνει επαναμεταγλώττιση (recompile) εάν είναι απαραίτητο. Αν παραχθεί ο κώδικας πάλι, θα επικαλύψει οποιεσδήποτε αλλαγές που γίνονται στον παραγόμενο κώδικα χωρίς έκδοση προειδοποίησης.

- **Εκτέλεση υπηρεσίας (Running a Generated Service)**

Όπως σε όλες τις υπηρεσίες του Microsoft Robotics Studio υπάρχουν πολλοί τρόποι για να εκτελεστεί μια παραγόμενη υπηρεσία.

1. Αρχικά γίνεται εκκίνηση ενός dss κόμβου και μέσω του control panel αρχίζει η παραγόμενη υπηρεσία. Τα βήματα που ακολουθούνται παρουσιάζονται στη συνέχεια:

    a. Αρχικά τρέχει το **dsshost/p: 50000/t: 50001**
    b. Ακολουθεί πλοήγηση στο **http://localhost:50000/controlpanel**
    c. γίνεται εντοπισμός της πρόσφατα παραγόμενης υπηρεσίας στον κατάλογο υπηρεσιών
    d. Τέλος επιλέγεται με κλικ το κατάλληλο "Create” κουμπί στο control panel της σελίδας.

2. Στη συνέχεια δημιουργείτε ένα αρχείο manifest το οποίο να αναφέρει την πιο πρόσφατα δημιουργημένη υπηρεσία με σύμβαση (η σύμβαση για την υπηρεσία σας θα βρίσκεται στο παραγόμενο αρχείο Diagram Types.cs, και μπορεί να τροποποιηθεί με την έκδοση του Contract Prefix θέτοντας στις ιδιότητες διαγραμμάτων πριν παραχθεί ο κώδικας) και μέσω του προφανούς αρχείου στην **/** m παράμετρο ξεκινήματος ξεκινά ο dss κόμβος.
3. Για να ολοκληρωθεί η διαδικασία πρέπει να κλείσει και να ανοίξει πάλι η VPL, να δημιουργηθεί ένα νέο διάγραμμα που χρησιμοποιεί την πιο πρόσφατα παραγόμενη υπηρεσία (η οποία θα εμφανιστεί τώρα στον κατάλογο υπηρεσιών). Όταν εκτελείται εκείνο το διάγραμμα από VPL με τον συνηθισμένο τρόπο θα αναγκάσει την παραγόμενη υπηρεσία να εκτελεστεί.

Γνωστά ζητήματα

- οι λειτουργίες καταλόγων δεν υποστηρίζονται ακόμα.

- σε ορισμένες περιστάσεις το block If που ακολουθείται από ένα merge, μπορεί να προκαλέσει την παραγωγή ανακριβή κώδικα. Πρέπει να είναι δυνατό να αντιμετωπιστεί εάν αυτό συμβαίνει με την εξέταση του κώδικα.

- μια πρόσφατα παραγόμενη υπηρεσία δεν εμφανίζεται αυτόματα στον κατάλογο υπηρεσιών σε VPL έως ότου έχει κλείσει η VPL και έχει ανοιχτεί πάλι.

- οι υπηρεσίες δεν παράγονται για. .NET Compact Framework(CF) - είναι δυνατό να πάρει την υπηρεσία που παράγεται και να την μετατρέψει σε συνεργασία με το .NET CF ακολουθώντας την ίδια διαδικασία που χρησιμοποιείται κατά τη μετατροπή οποιασδήποτε άλλης υπηρεσίας σε συνεργασία με το . NET CF . Ο κώδικας που παράγεται δεν χρησιμοποιεί τις μεθόδους που δεν είναι υποστηριγμένο το .NET CF.

- οι πληροφορίες διαμόρφωσης δεν συμπεριλαμβάνονται στις παραγόμενες υπηρεσίες. Εάν έχεις ένα διάγραμμα που ενορχηστρώνει τις υπηρεσίες και τις διαμορφώνει, είτε θέτοντας την αρχική κατάσταση και τις συνεταιρικές άμεσα είτε με την εισαγωγή ενός manifest, η παραγόμενη υπηρεσία δεν θα αρχίσει σωστά.

Υπάρχουν δύο τρόποι εργασίας γύρω από αυτό:

1. Αφού έχει παραχθεί ο κώδικας για το διάγραμμα πρέπει να κλείσει η VPL, και να ανοίξει πάλι. Στη συνέχεια πρέπει να δημιουργηθεί ένα νέο διάγραμμα και να συρθεί η υπηρεσία που συντάχθηκε ακριβώς από τον κατάλογο υπηρεσιών. Η υπηρεσία αυτή πρέπει να διαμορφωθεί και να εκτελεστεί το νέο διάγραμμα. Αυτό το νέο διάγραμμα δεν περιέχει κανέναν κώδικα, υπάρχει μόνο για να διαμορφώσει τη υπηρεσία που συντάχθηκε.

2. Αρχικά πρέπει να δημιουργηθεί ένα manifest για την πρόσφατα παραγόμενη υπηρεσία που διαμορφώνει σωστά τις συνεταιρικές υπηρεσίες. Εάν εφαρμοστεί αυτή η μέθοδος θα πρέπει να τροποποιηθούν οι ιδιότητες συνεργατών (partners) στον παραγόμενο κώδικα υπηρεσιών για να αλλάξει το PartnerCreationPolicy τους από UseExistingOrCreate σε UsePartnerListEntry για οποιουσδήποτε συνεργάτες που πρέπει να διαμορφωθούν.

2.10 Τεχνάσματα και συμβουλές για το Microsoft Robotics Developer Studio

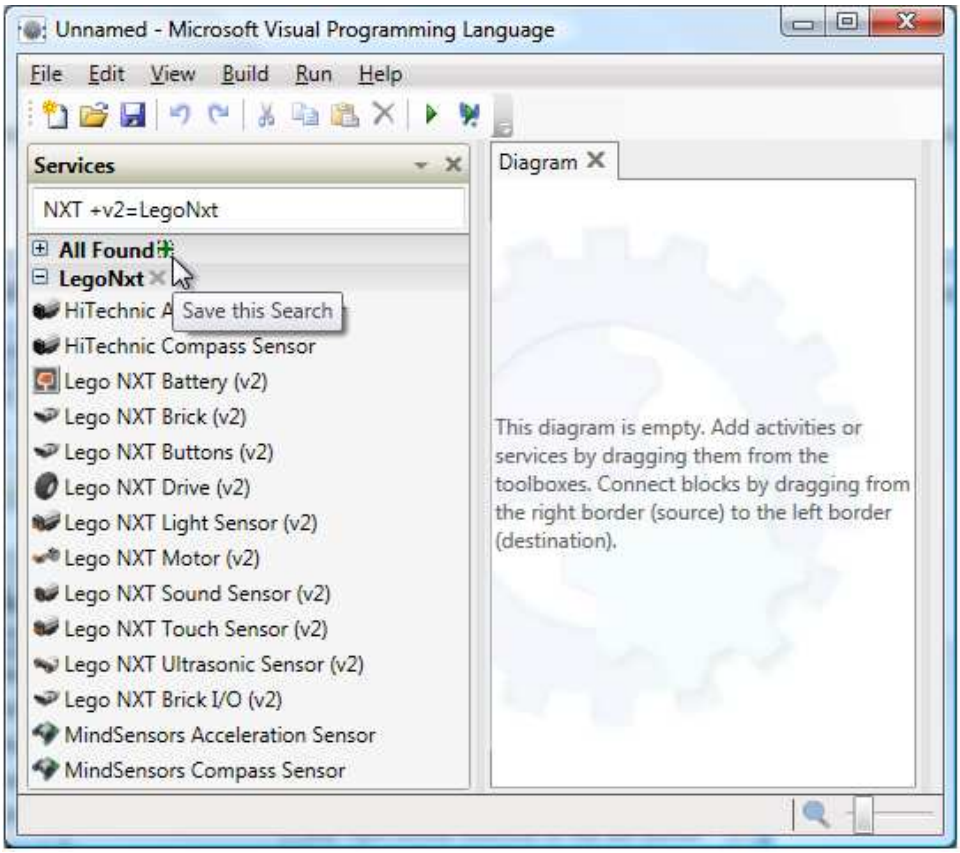
2.10.1 Εύρεση NXT Services στη VPL

Δημιουργία ενός γκρουπ όλων των υπηρεσιών LegoNxt

Στην VPL παρέχεται η δυνατότητα δημιουργίας γκρουπ (ομάδες) υπηρεσιών που επιτρέπουν την ευκολότερη εύρεση υπηρεσιών που χρησιμοποιούνται συχνότερα κατά τη δημιουργία διαγραμμάτων. Κάνοντας άνοιγμα στο visual programming language της Microsoft και με την επιλογή **new** από το File Menu δημιουργείται ένα γκρουπ υπηρεσιών με δικό του όνομα.

Όπως φαίνεται στην παρακάτω **Εικόνα2.10.1.1**, αριστερά που εμφανίζονται οι υπηρεσίες, υπάρχει ένα textbox.

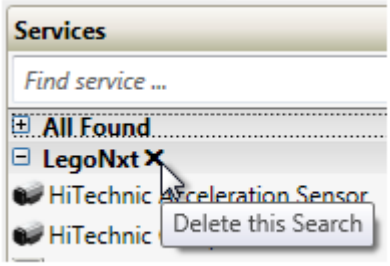
Πληκτρολογώντας: NXT + v2=LegoNxt και επιλέγοντας στη συνέχεια το **+** που βρίσκεται δεξιά από το All Found κάτω ακριβώς από το textbox, θα δημιουργήσει ένα μόνιμο γκρουπ με όνομα *LegoNxt*



Εικόνα 2.10.1.1: Παράθυρο τηςVisual Programming Language.

Αφαίρεση ενός γκρουπ από τη λίστα υπηρεσιών .

Μπορεί να αναιρεθεί το γκρουπ υπηρεσιών στην VPL με επιλογή του **X που βρίσκεται** δίπλα στο όνομα του γκρουπ όπως δείχνει η **Εικόνα 2.10.1.2** παρακάτω.

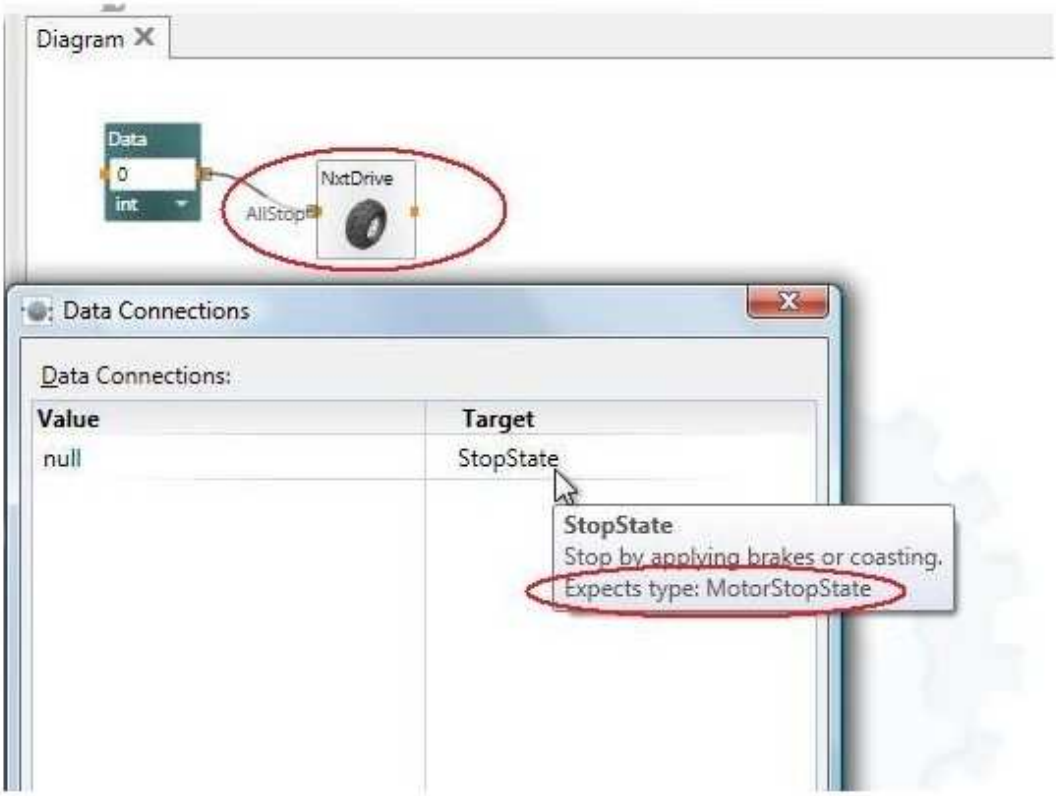


Εικόνα 2.10.1.2: Παράθυρο τηςVisual Programming Language,επίδειξη διαγραφής ενός γκρουπ υπηρεσιών.

2.10.2 Θέτοντας τις σωστές τιμές στη VPL

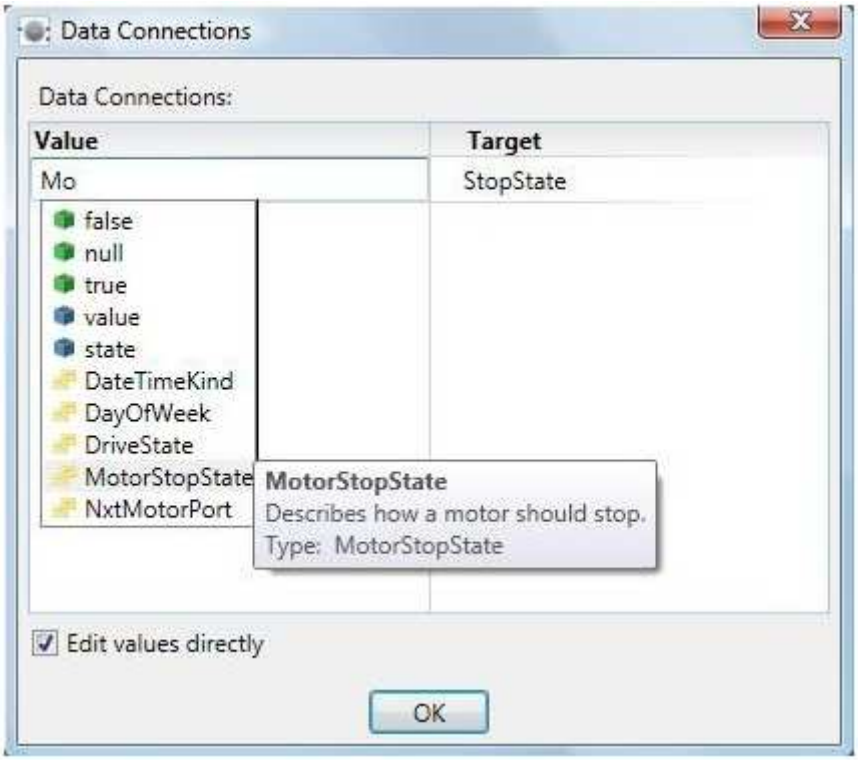
Το να γνωρίζει κανείς τι να πληκτρολογήσει στην τιμή μιας λειτουργίας μπορεί μερικές φορές να είναι <παγίδα>. Για παράδειγμα, η λειτουργία **StopState** της εντολής **AllStop**, που είναι διαθέσιμη στο **NxtDrive**.

Εάν ο δείκτης του ποντικιού τοποθετηθεί πάνω από το **StopState**, εμφανίζεται ένα παραθυράκι που λέει ποιος τύπος αναμένεται. Σε αυτήν την περίπτωση, είναι το **MotorStopState (δείτε Εικόνα 2.10.2.1).**



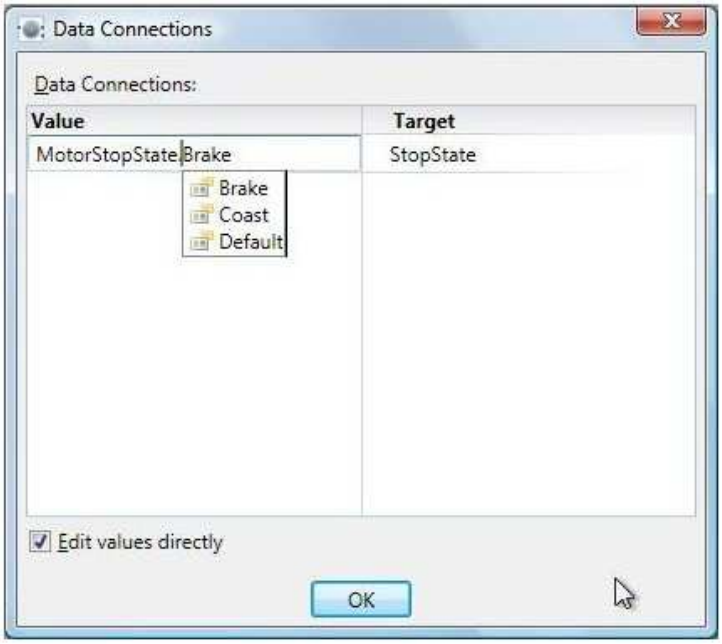
Εικόνα 2.10.2.1: Παράθυρο του Data Connections.

Από το πεδίο value όπου έχει ένα textbox (όπως δείχνει η **Εικόνα2.10.2.2**), μπορεί να πληκτρολογηθεί "MotorStopState" ώστε να εμφανιστεί ένας κυλιόμενος κατάλογος όπου στη λίστα του θα υπάρχει **το MotorStopState.** Αφού επιλεγεί από τη λίστα με OK ολοκληρώνεται η διαδικασία.



Εικόνα 2.10.2.2: Παράθυρο του Data Connections.

Με πληκτρολόγηση μιας τελείας (".") μετά από το **MotorStopState** εμφανίζεται ένα άλλο κυλιόμενο μενού με τις τιμές **του MotorStopState,** απ' όπου μπορεί να επιλεγεί μια από τις τιμές (**Εικόνα 2.10.2.3**).



Εικόνα 2.10.2.3: Παράθυρο του Data Connections.

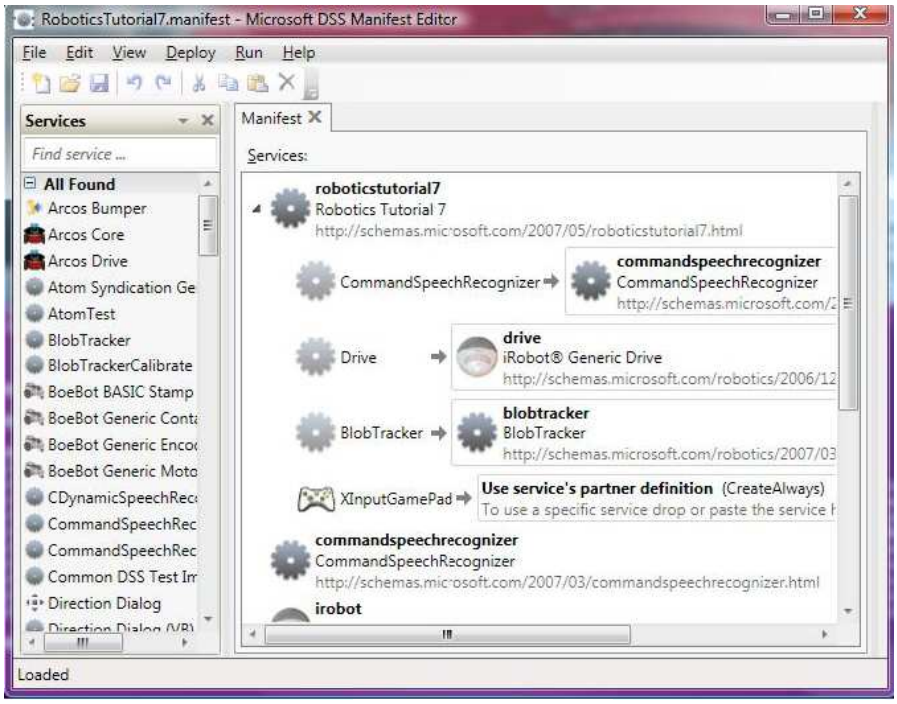
Κεφάλαιο 3

Microsoft DSS Manifest Editor

3.1 Εισαγωγή στον DSS Manifest Editor

Ο Manifest Editor (DSSME) είναι σχεδιασμένος με τέτοιο τρόπο έτσι ώστε να επιτρέπει τη δημιουργία και την έκδοση αρχείων manifests για τις dss υπηρεσίες. Τα αρχεία manifest υπηρεσιών είναι αρχεία XML που περιέχουν μία λίστα υπηρεσιών και τη διαμόρφωσή τους. Αυτά χρησιμοποιούνται για να αρχίσουν οι υπηρεσίες χρησιμοποιώντας το dss χρόνο εκτέλεσης (Manifest Loader Service).

Το παράθυρο (window) του dss manifest editor περιλαμβάνει μενού για δημιουργία, φόρτωση, και αποθήκευση των manifests καθώς επίσης και για την έκδοση των καταχωρήσεων που εισάγονται στο manifest αρχείο (δείτε **Εικόνα 3.1.1**). Συνολικά ο editor δημιουργεί ένα project που αποτελείται από το manifest αρχείο καθώς επίσης και από οποιαδήποτε σχετικά αρχεία διαμόρφωσης που χρησιμοποιούνται. Το DSSME περιλαμβάνει επίσης μία μπάρα εργαλείων για τη γρήγορη πρόσβαση στις κοινές εντολές, και τρία παράθυρα εργαλειοθηκών: το παράθυρο service toolbox window, που επιδεικνύει ποιες υπηρεσίες είναι διαθέσιμες για να συμπεριληφθούν στο manifest αρχείο, το project toolbox window που επιδεικνύει το manifest αρχείο και οποιαδήποτε σχετικά αρχεία διαμόρφωσης, και το properties toolbox window που επιδεικνύει τις παραμέτρους διαμόρφωσης για τις επιλεγμένες υπηρεσίες.



Εικόνα 3.1.1 :Το παράθυρο του dss manifest editor ,φαίνονται πολλές από τις επιλογές του.

3.2 Ξεκινώντας τον DSS Manifest Editor

Για να αρχίσεις το Microsoft DSS Manifest Editor, επιλέγεται η είσοδος σε αυτό από το μενού έναρξης. Όταν το DSSME φορτώνει, εμφανίζεται ένα παράθυρο με ένα σύνολο από μενού, tool boxes, και μιας τοποθετημένης σελίδας διαγραμμάτων.

Τα toolbox windows είναι κινητά και διευθετήσιμα, έτσι μπορεί να ρυθμιστούν εκ νέου μέσα στο κύριο παράθυρο DSSME με το σύρσιμο τους ή με τα σύνορά τους και επίσης μπορεί παρουσιαστούν ή όχι χρησιμοποιώντας το μενού View Toolboxes .

- **The services toolbox window**

Εικόνα 3.2.1 : Το παράθυρο services toolbox window με τις υπηρεσίες που είναι διαθέσιμες .

Το παράθυρο services toolbox window (**δείτε Εικόνα 3.2.1**) εμφανίζει τις υπηρεσίες που είναι διαθέσιμες για να συμπεριληφθούν σε ένα manifest αρχείο.

Μπορεί να αλλάξει το όνομα μιας υπηρεσίας που μπαίνει στο manifest αρχείο με τη πληκτρολόγηση του ονόματος που επιδεικνύεται στο properties toolbox όταν επιλέγεται η υπηρεσία.

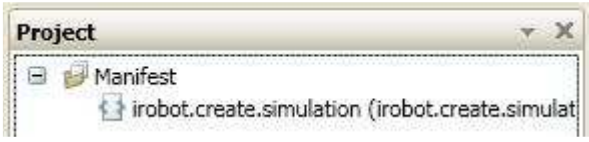
Οι υπηρεσίες που επιδεικνύονται στο services toolbox window εξαρτώνται από το περιεχόμενο κειμένου του textbox (το οποίο περιέχει φίλτρο αναζήτησης) που βρίσκεται πάνω πάνω. Πληκτρολογώντας μία λέξη εμφανίζονται στο παράθυρο μόνο εκείνες οι υπηρεσίες που περιέχουν αυτήν την λέξη. Όταν το textbox είναι κενό εμφανίζει όλες τις υπηρεσίες.

Μπορεί να πληκτρολογηθούν πολλές λέξεις (που χωρίζονται με τα κενά) και η εργαλειοθήκη θα εμφανίσει τις υπηρεσίες που περιλαμβάνουν όλες αυτές τις λέξεις. Η τοποθέτηση α + μεταξύ των λέξεων αναγκάζει τη λίστα

να εμφανίσει μόνο εκείνες τις υπηρεσίες όπου όλες οι λέξεις εμφανίζονται. Χρησιμοποιώντας το α - αποκλείει τη λέξη /-εις.

Μπορεί να αποθηκευτεί ένα φίλτρο αναζήτησης κάνοντας κλικ στο + Add (προσθήκη) κουμπί. Αυτό δημιουργεί έναν τίτλο για το φίλτρο αναζήτησης που χρησιμοποιεί τις λέξεις που χρησιμοποιήθηκαν ως φίλτρο αναζήτησης. Για τη δημιουργία ενός τίτλου για το φίλτρο αναζήτησης χωρίς τις λέξεις φίλτρου, αρκεί να προστεθεί "= title" στο κείμενο αναζήτησης. Τώρα όταν πατηθεί κλικ στο + Add (προσθήκη) κουμπί στο All Found, το νέο τμήμα θα χρησιμοποιήσει ακριβώς εκείνο το κείμενο.

- **The project toolbox window**



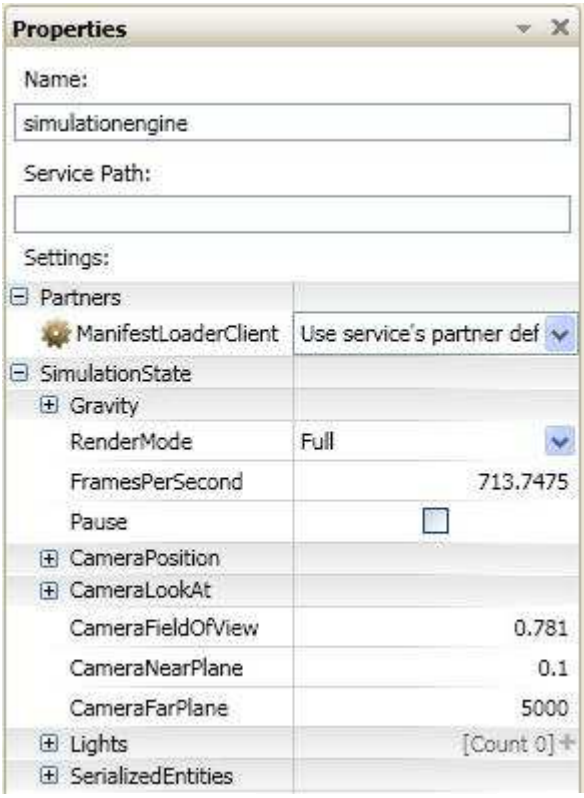
Εικόνα 3.2.2 : Το παράθυρο project toolbox window.

Τα παράθυρα των project (**Εικόνα3.2.2**) εμφανίζουν το manifest αρχείο και οποιαδήποτε αρχεία διαμόρφωσης για τις υπηρεσίες που περιλαμβάνονται. Τα αρχεία διαμόρφωσης βρίσκονται ιεραρχικά σε λίστα κάτω από το manifest αρχείο. Μπορεί να παρουσιαστεί ή να κρυφτεί η εμφάνισή τους στο παράθυρο του project χρησιμοποιώντας τα + (expand) ή - (collapse) κουμπιά.

Όταν χρησιμοποιείται η εντολή Save ή Save as αποθηκεύονται αυτόματα οποιεσδήποτε αλλαγές στα αρχεία manifest και διαμόρφωσης.

Για να διαγραφεί ένα αρχείο διαμόρφωσης αρκεί να επιλεγεί και να πατηθεί κλικ στη διαγραφή.

- **The properties toolbox window**



Εικόνα 3.2.3 : Το παράθυρο properties toolbox window.

Το παράθυρο properties toolbox (**Εικόνα 3.2.3**)εμφανίζει τις ιδιότητες για την τρέχουσα επιλεγμένη υπηρεσία.

3.3 DSS Manifest Editor Menus

- **The File Menu**

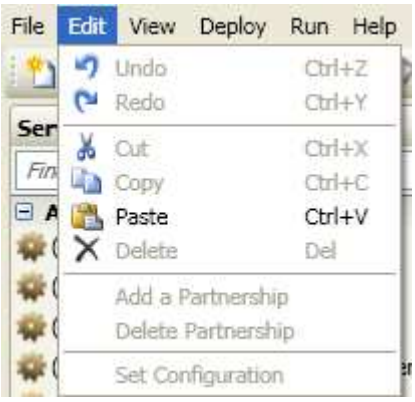
Εδώ παρουσιάζονται οι επιλογές του μενού File (**Εικόνα 3.3.1**)

Εικόνα 3.3.1 : Οι επιλογές του μενού File .

New – Επιτρέπει τη δημιουργία ενός νέου manifest.
Open – Ανοίγει ένα προϋπάρχον manifest αρχείο (και οποιαδήποτε αρχεία διαμόρφωσης που σχετίζονται). **Save** - Αποθηκεύει το manifest (και οποιαδήποτε αρχεία διαμόρφωσης που σχετίζονται).
Save As - Αποθηκεύει το manifest με ένα διευκρινισμένο όνομα. **Import Manifest** - Φορτώνει ένα προϋπάρχον manifest.
Print - Τυπώνει το τρέχον ανοικτό manifest αρχείο.
Recent Projects – Επιδεικνύει και ανοίγει τα πρόσφατα projects (manifests και οποιαδήποτε σχετικά αρχεία διαμόρφωσης).
Exit – Εγκαταλείπει το DSSME .

- **The Edit Menu**

Εδώ παρουσιάζονται οι επιλογές του μενού **Edit** (**Εικόνα 3.3.2**)

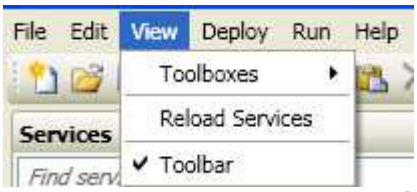


Εικόνα 3.3.2 : Οι επιλογές του μενού Edit.

Redo – Αποκαθιστά την τελευταία τροποποίηση.
Undo – Αντιστρέφει την τελευταία δράση.
Cut – Αφαιρεί το επιλεγμένο αντικείμενο.
Copy – Αντιγράφει το επιλεγμένο αντικείμενο.
Paste – Εισάγει την τελευταία περικοπή ή το αντιγραμμένο αντικείμενο.
Del – Αφαιρεί το επιλεγμένο αντικείμενο.
Set Configuration – Εμφανίζει το πλαίσιο διαμόρφωσης επιτρέποντας τον ορισμό της διαμόρφωσης μιας υπηρεσίας και τις ιδιότητες μίας συνεταιρικής.

- **The View Menu**

Εδώ παρουσιάζονται οι επιλογές του μενού **View** (**Εικόνα 3.3.3**)



Εικόνα 3.3.3 : Οι επιλογές του μενού View.

Toolboxes – Επιδεικνύει (ή κρύβει) τις υπηρεσίες, το Project, και τα παράθυρα ιδιοτήτων (properties window).
Reload Services - Αναζωογονεί την τρέχουσα εικόνα των διαθέσιμων υπηρεσιών που επιδεικνύονται στην εργαλειοθήκη υπηρεσιών.
Toolbar – Επιδεικνύει (ή κρύβει) τη μπάρα εργαλείων DSSME.

- **The Deploy Menu**

Εδώ παρουσιάζονται οι επιλογές του μενού **Deploy** (**Εικόνα 3.3.4**)

Εικόνα 3.3.4 : Οι επιλογές του μενού Deploy.

Create Deployment Package - Αυτό παράγει ένα dss deployment αρχείο.

- **The Run Menu**

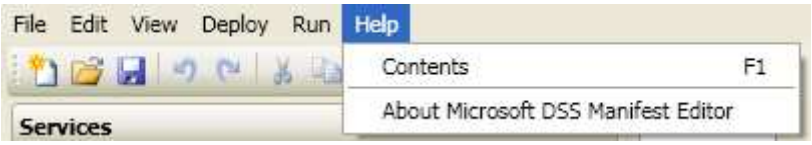
Εδώ παρουσιάζονται οι επιλογές του μενού **Run** (**Εικόνα 3.3.5**)

Εικόνα 3.3.5 : Οι επιλογές του μενού Run.

Run Manifest - Αυτό αρχίζει το τρέχον φορτωμένο manifest σε έναν νέο dss κόμβο.

- **The Help Menu**

Εδώ παρουσιάζονται οι επιλογές του μενού **Help** (**Εικόνα 3.3.6**)

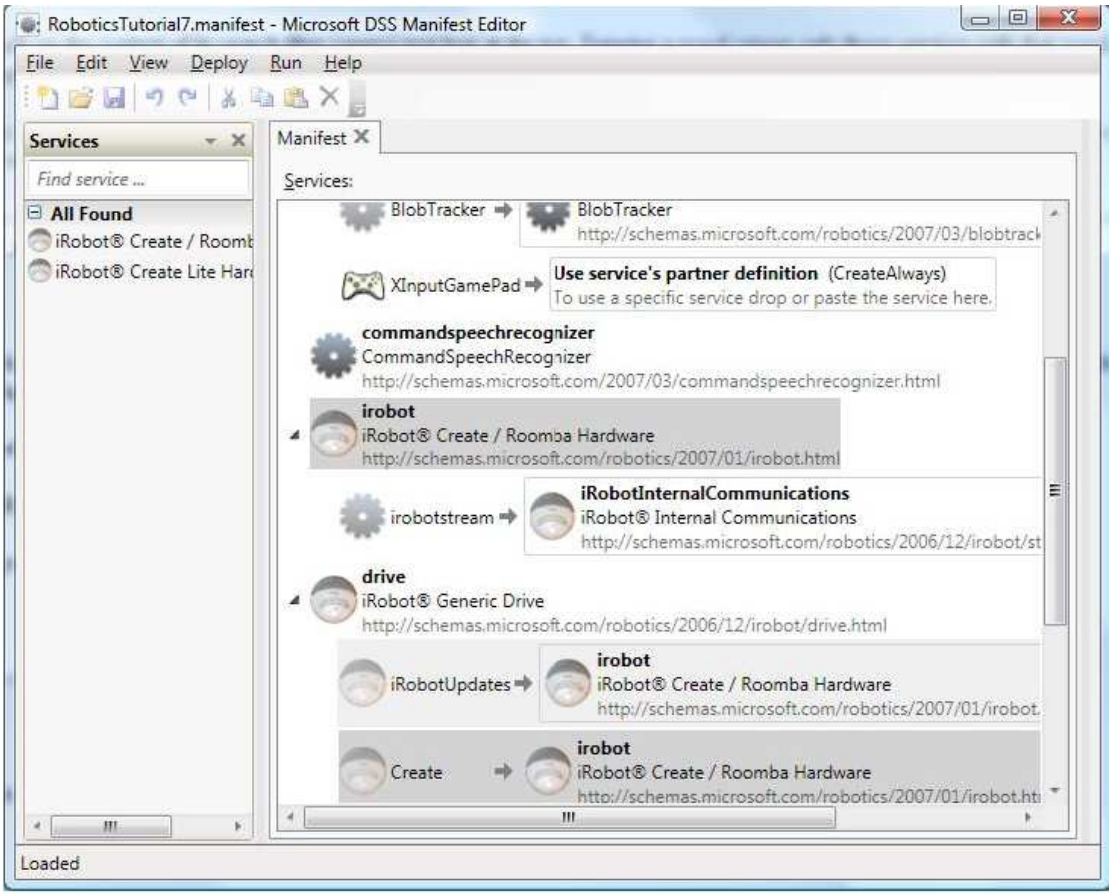


Εικόνα 3.3.6 : Οι επιλογές του μενού Help.

Contents – Επιδεικνύει το αρχείο βοήθειας DSSME.
About – Εμφανίζει τα Πνευματικά δικαιώματα και πληροφορίες για το DSSME

3.4 Δημιουργία και Επεξεργασία των DSS Manifests

Για τη δημιουργία ενός νέου manifest αρχείου, επιλέγετε New από το File Menu (ή κάνετε κλικ στο εικονίδιο New στη μπάρα εργαλείων του DSSME). Για να φορτωθεί και να εκδοθεί ένα υπάρχον manifest αρχείο, επιλέγετε Open από το File Menu (**Εικόνα3.4.1**).



Εικόνα 3.4.1 : Το παράθυρο του DSS Manifest Editor , όπου έχει φορτωθεί ένα υπάρχον manifest αρχείο.

Προσθήκη μιας υπηρεσίας σε ένα manifest

Με σύρσιμο ή διπλό κλικ στην είσοδο της υπηρεσίας από την εργαλειοθήκη υπηρεσιών προστίθεται μια υπηρεσία στο manifest, και προστίθεται στην τοποθετημένη manifest σελίδα.

Τυπικά μια υπηρεσία θα έχει μια ή περισσότερες υπηρεσίες συνεταιρικές (partner services). Αυτές είναι υπηρεσίες που βοηθούν στην υποστήριξη της λειτουργίας ή της λειτουργικότητας της υπηρεσίας. Εάν οι συνεταιρικές είναι ήδη καθορισμένες για μια υπηρεσία, θα εμφανίζεται ένα κουμπί επέκτασης αριστερά του εικονιδίου της υπηρεσίας. Για να εμφανιστεί ο κατάλογος καθορισμένων συνεργασιών, χρησιμοποιείται το κουμπί επέκτασης της λίστας.

Κάθε συνεργασία υπηρεσιών εμφανίζεται χωριστά συμπεριλαμβάνοντας το εικονίδιο που αντιπροσωπεύει την απαιτούμενη συνεταιρική υπηρεσία, το όνομα της συνεργασίας, κάποια χαρακτηριστικά δημιουργίας της συνεταιρικής, επίσης προσδιορίζει πώς η συνεταιρική υπηρεσία ξεκινά ή συνδέεται εάν ο συνεργάτης είναι απροσδιόριστος.

Οι επιλογές παρουσιάζονται στη συνέχεια:

Ιδιότητες Partner	**Περιγραφή**
UsePartnerListEntry	Απαιτεί την συνεταιρική υπηρεσία να διευκρινιστεί.
UseExisting	Απαιτεί η συμβατή υπηρεσία να εκτελείται.
UseExistingOrCreate	Χρησιμοποιεί μία συμβατή υπηρεσία εάν είναι διαθέσιμη ή προσπαθεί να αρχίσει μια συμβατή υπηρεσία.
CreateAlways	Προσπαθεί πάντα να αρχίσει μια συμβατή υπηρεσία .

Αυτές οι επιλογές υπάρχουν όταν δεν διευκρινίζεται μια συνεταιρική υπηρεσία. Για να διευκρινιστεί μία συγκεκριμένη συνεταιρική υπηρεσία μια υπηρεσία πρέπει να συρθεί από την εργαλειοθήκη υπηρεσιών πάνω στην είσοδο συνεργασίας. Για να προσδιοριστούν γρήγορα οι συμβατές υπηρεσίες, επιλέγεται η είσοδος συνεργασίας (partnership entry). Η λίστα υπηρεσιών στην εργαλειοθήκη υπηρεσιών φιλτράρεται για να παρουσιάσει μόνο τις συμβατές υπηρεσίες.

Εάν συρθεί μια υπηρεσία που υπάρχει ήδη στο manifest αρχείο θα εμφανιστεί προτροπή είτε στη χρήση μιας από τις υπάρχουσες υπηρεσίες είτε στη δημιουργία μιας νέας. Όταν προστεθεί η συνεταιρική υπηρεσία στη συνεργασία, θα προστεθεί επίσης μια νέα είσοδος στη λίστα υπηρεσιών του manifest για εκείνη την υπηρεσία. Παρέχεται επίσης η δυνατότητα να συρθεί μια υπάρχουσα υπηρεσία στο manifest ή να αντιγραφεί και να επικολληθεί στην υπηρεσία. Μπορεί να αποκοπεί, να αντιγραφεί, να επικολληθεί ή να

διαγραφεί μια υπηρεσία με την επιλογή της και τη χρησιμοποίηση των αντίστοιχων εντολών από το Edit μενού .

Εναλλακτικά, αντί της διευκρίνισης μιας συνεταιρικής υπηρεσίας μπορεί να δοθεί το URL της συνεταιρικής υπηρεσίας ή το URL ενός dss καταλόγου υπηρεσίας όπου θα τοποθετηθεί μια τρέχουσα υπηρεσία ώστε να χρησιμοποιηθεί ως συνεταιρική. Για να γίνει αυτό, επιλέγεται η είσοδος συνεργασίας και τίθεται μια από τις επιλογές URL.

Διαμόρφωση μιας υπηρεσίας σε ένα manifest

Οι περισσότερες υπηρεσίες είναι διαμορφώσιμες (δηλαδή μπορεί να διαμορφωθούν στην αρχική κατάστασή τους). Ένα αρχείο διαμόρφωσης με προεπιλεγμένες τιμές (default) θα δημιουργηθεί (εάν η υπηρεσία είναι διαμορφώσιμη). Μπορεί να οριστεί μια διαμόρφωση με την επιλογή της υπηρεσίας και να χρησιμοποιηθεί η εργαλειοθήκη ιδιοτήτων (Properties toolbox). Ακόμα, μπορεί να εμφανιστούν οι επιλογές διαμόρφωσης ως μία χωριστή σελίδα, με την επιλογή της υπηρεσίας και έπειτα να τεθεί η διαμόρφωση από το Edit μενού ή με διπλό κλικ να γίνει είσοδος στην υπηρεσία.

Για να μην τεθεί μια διαμόρφωση, αρκεί να επεκταθεί η είσοδος του manifest στο παράθυρο του Project toolbox και να διαγραφεί η είσοδος του αρχείου διαμόρφωσης.

Εάν διαγραφεί μια διαμόρφωση μιας υπηρεσίας, μπορεί να δημιουργηθεί μια κενή διαμόρφωση με την επιλογή της υπηρεσίας και με κλικ στο κουμπί Δημιουργίας αρχικής κατάστασης (Create Initial State button).

Μπορεί επίσης να εισαχθεί ένα υπάρχον αρχείο διαμόρφωσης, με κλικ στο κουμπί αρχικής κατάστασης εισαγωγής (Import Initial State button), στο οποίο μπορεί να επιλεγεί το αρχείο που πρόκειται να εισαχθεί.

Εισαγωγή ενός manifest

Για να ενωθούν δύο manifest μπορεί να χρησιμοποιηθεί η εντολή import manifest command από το File μενού. Αυτό εμφανίζει ένα μενού που επιτρέπει την επιλογή ενός άλλου manifest για να γίνει merge.

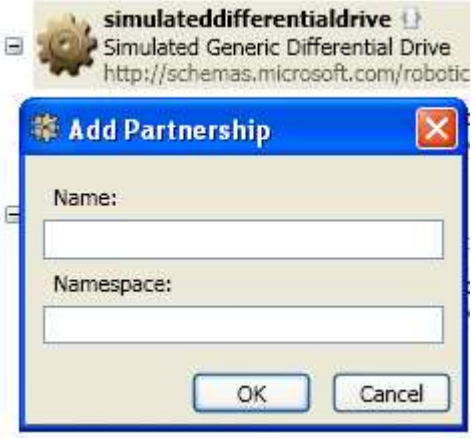
Αποθήκευση ενός manifest

Για να αποθηκευτεί το νέο manifest, επιλέγετε το save από το File μενού (ή κάνετε κλικ στην εικόνα save στη μπάρα εργαλείων DSSME).

Για να προστεθεί μια υπηρεσία στο manifest σύρετε την είσοδο της υπηρεσίας (ή κάνετε διπλό κλικ) στη μπάρα εργαλείων υπηρεσιών (Services tool bar).

Για αποθήκευση του νέου manifest, επιλέγετε save από το File μενού (ή κάνετε κλικ στην εικόνα save στη μπάρα εργαλείων DSSME).

Προσθήκη μιας νέας συνεργασίας(Partnership)



Εικόνα 3.4.2 : Το παράθυροAdd Partnership για την συνεργασία υπηρεσιών.

Μπορεί να προστεθεί μια νέα συνεργασία σε μια υπάρχουσα υπηρεσία. Για να γίνει αυτό, επιλέγεται η υπηρεσία και το add a Partnership από το Edit μενού (**Εικόνα 3.4.2**).

Εμφανίζεται στη συνέχεια ένα πλαίσιο διαλόγου (dialog box) όπου εισάγετε το όνομα της συνεργασίας.Αν μία φορά έχει προστεθεί μια νέα συνεργασία, μπορεί να διευκρινιστεί μία συνεταιρική υπηρεσία για αυτήν την είσοδο.

Καθορισμός διαμόρφωσης ενός manifest

Η διαμόρφωση ενός manifest περιέχει τέσσερις προαιρετικές ρυθμίσεις που χρησιμοποιούνται μόνο όταν χρησιμοποιείται το manifest από το συντάκτη (editor). Οι πρώτες δύο ρυθμίσεις επιτρέπουν να τεθούν οι παράμετροι για τη δημιουργία ενός dss deployment package. Αυτές περιλαμβάνουν την πορεία όπου το deployment package είναι αποθηκευμένο. Οι τελευταίες δύο ρυθμίσεις επιτρέπουν τη διευκρίνιση των θυρών του HTTP και TCP για να εκτελεστεί το manifest στο dss κόμβο. Αυτές οι ρυθμίσεις αποθηκεύονται σε ένα πρόσθετο αρχείο που είναι προαιρετικό και το αρχείο ονομάζεται, χρησιμοποιώντας το όνομα του manifest αρχείου με προέκταση το .editor-settings αντί του .xml. Εάν το αρχείο δεν είναι διαθέσιμο, τότε το DSSME παράγει αυτόματα default ρυθμίσεις.

Κεφάλαιο 4

4.1 Microsoft Visual Simulation Environment

4.1.1 Γενικά για το Simulation

Το Microsoft Robotics Developer Studio2008 Express στοχεύει στο να προσεγγίσει το ευρύ κοινό σε μία προσπάθεια να επιταχυνθεί η ανάπτυξη και η υιοθέτηση της ρομποτικής. Ένα σημαντικό μέρος αυτής της προσπάθειας είναι ο χρόνος εκτέλεσης της προσομοίωσης. Ήταν αμέσως προφανές ότι τα παιχνίδια με τη χρήση των PC είχαν προετοιμάσει το έδαφος για την προσομοίωση της ρομποτικής. Τα παιχνίδια στηρίζονται στις ρεαλιστικές (photo-realistic) απεικονίσεις με την προηγμένη εκτέλεση προσομοίωσης μέσα σε περιορισμούς πραγματικού χρόνου.

Ο χρόνος εκτέλεσης της προσομοίωσης σχεδιάστηκε να χρησιμοποιηθεί σε μία ποικιλία προηγμένων σεναρίων τα οποία έχουν υψηλές απαιτήσεις για ακρίβεια, απεικόνιση, και κλιμάκωση. Συγχρόνως, ένας αρχάριος χρήστης με λίγη εμπειρία στον προγραμματισμό μπορεί να χρησιμοποιήσει την προσομοίωση ,αναπτύσσοντας ενδιαφέρουσες εφαρμογές σε ένα περιβάλλον που μοιάζει με παιχνίδι δηλαδή σε ένα game-like περιβάλλον. Η ολοκλήρωσή των τεχνολογιών AGEIA PhysX μας επιτρέπει να χρησιμοποιούμε ένα πολύ ισχυρό προϊόν προσομοίωσης που είναι «ώριμο» και εξελιγμένο συνεχώς προς τα χαρακτηριστικά γνωρίσματα που θα είναι ανεκτίμητα στη ρομποτική.

Προκλήσεις που τίθενται από την ανάπτυξη ρομποτικής

- **Το Hardware υλικό ρομποτικής μπορεί να είναι πολύ ακριβό και δύσκολο να βρεθεί**

Οι αποτελούμενες από υπομονάδες πλατφόρμες ρομποτικής, όπως το LEGO® MINDSTORMS™ και fischertechnik®, έχουν καταστήσει τη ρομποτική προσιτή σε ένα ευρύ καταναλωτικό κοινό. Αυτές οι πλατφόρμες είναι μια άριστη αφετηρία για την εκπαιδευτική αγορά και για τους ερασιτέχνες. Για τη δημιουργία κλιμακωτών επιπέδων δυσκολίας του ρομπότ ή του αριθμού μεμονωμένων ρομπότ, το κόστος εμποδίζει τους περισσότερους ανθρώπους στο να προχωρήσουν περαιτέρω.

- **Δυσκολία της ανίχνευσης λαθών υλικού hardware**

Η ανίχνευση λαθών υλικού hardware είναι πολύ δύσκολη όπως και για ευρέως διαδεδομένα καταναλωτικά υλικά hardware όπως έναν DVD player ή μια τηλεόραση. Τα ηλεκτρονικά είδη ευρείας κατανάλωσης συμβαίνει να είναι εξαιρετικά αξιόπιστα έτσι οι περισσότεροι άνθρωποι δεν είναι απαραίτητο να ανησυχούν για το αν θα πάνε τα πράγματα στραβά.

- **Δυσκολία της ταυτόχρονης χρήσης**

Η ανάπτυξη ενός προηγμένου ρομπότ (όπως τα οχήματα που συμμετείχαν στους διαγωνισμούς DARPA) μέσω μίας ομάδας ανθρώπων γίνεται ένα κοινό ζήτημα. Μια από τις προκλήσεις είναι ότι συχνά, το ρομπότ που αναπτύσσεται είναι ακριβό και υπάρχει μόνο ένα. Αυτές οι δύο ιδιότητες καθιστούν δύσκολο το να δοκιμαστούν ταυτόχρονα καταστάσεις και χωρίς τον κίνδυνο καταστροφής του ρομπότ. Αυτό αναγκάζει τους ανθρώπους να προσπαθήσουν να αναπτύξουν τα συστατικά μεμονωμένα, κάνοντας την ολοκλήρωση σκληρότερη .

Πλεονεκτήματα προσομοίωσης

1 Κανένα όριο (Low barrier to entry)

Η προσομοίωση επιτρέπει στους ανθρώπους με έναν προσωπικό υπολογιστή να αναπτύξουν πολύ ενδιαφέροντα ρομπότ ή πλήθους ρομπότ με τους κύριους περιοριστικούς παράγοντες που είναι ο χρόνος και η φαντασία. Συγχρόνως, τους περιορίζει έτσι ώστε να στρέψουν την προσπάθειά τους σε κάτι που μπορεί να πραγματοποιηθεί.

2 Σταδιακή προσέγγιση (Staged approach)

Το Microsoft Robotics Developer Studio 2008 Express προσεγγίζει την προσομοίωση σταδιακά, επιτρέποντας έτσι στους ανθρώπους να αντιμετωπίσουν την πολυπλοκότητα στο σωστό χρόνο. Αυτό σημαίνει ότι μπορεί να γίνουν "διορθώσεις" στο προσομοιωμένο ρομπότ αρχίζοντας από βασικά στοιχεία έχοντας τις βασικές γνώσεις. Είναι εξαιρετικά σαφές για να προστεθεί ένα ρομπότ εικονικής πραγματικότητας σε ένα περιβάλλον, ότι πρέπει να υπάρχουν επίσης μερικές απλές μορφές (σχήματα ,οντότητες) για να αλληλεπιδράσει.

3 Δημιουργία προτύπων (Prototyping)

Τα φυσικά πρότυπα (μοντέλα) για ένα ρομπότ και τις υπηρεσίες προσομοίωσης που τα χρησιμοποιούν (μπορούν να αναπτυχθούν ταυτόχρονα από πολλούς ανθρώπους, όπως και από κοινότητες ανάπτυξης λογισμικού), δημιουργούν μια "πλατφόρμα" που πολλοί μπορούν να χρησιμοποιήσουν και να τροποποιήσουν χωρίς ανησυχία για τη καταστροφή των ακριβών και μοναδικών ρομπότ.

4 Εκπαίδευση

Η προσομοίωση μπορεί να είναι μια εξαιρετικά χρήσιμη εκπαιδευτική ενίσχυση. Μπορεί να επιλεγεί, να αναπτυχθεί σταδιακά η πολυπλοκότητα και να ελεγχθεί το περιβάλλον. Μπορεί επίσης να εισαχθούν συστατικά που είναι καθαρά εικονικά, έννοιες που δεν μπορούν να γίνουν εύκολα αντιληπτές, οι οποίες όμως είναι χρήσιμες για μάθηση.

5 Σύστημα εκμάθησης

Μια άλλη πραγματικά ενδιαφέρουσα πτυχή της προσομοίωσης είναι ότι μπορεί να χρησιμοποιηθεί ενώ το ρομπότ «τρέχει»(εκτελεί κάποια λειτουργία), ως προφητικό εργαλείο ή ως εποπτευόμενο εργαλείο μάθησης. Εδώ και αρκετό καιρό, οι άνθρωποι έχουν χρησιμοποιήσει την εκτέλεση

προσομοίωσης ταυτόχρονα με ένα ενεργό ρομπότ για να δοκιμάσουν τα πράγματα στον κόσμο της προσομοίωσης που είναι ενημερωμένος πραγματικός χρόνος με δεδομένα αισθητήρων. Κατόπιν η προσομοίωση μπορεί να μας δείξει εάν κάτι είναι μια καλή ιδέα, "κοιτάζοντας μπροστά" στα διάφορα ενδεχόμενα.

Μειονεκτήματα και περιορισμοί προσομοίωσης

Αυτό που ουσιαστικά γίνεται είναι μια προσπάθεια μετατροπής ενός προβλήματος υλικού hardware σε software. Η ανάπτυξη του λογισμικού software και ενός φυσικού προτύπου έχει τις προκλήσεις του έτσι ώστε η κατάληξη να έχει ένα σύνολο διαφόρων προκλήσεων και περιορισμών. Συνήθως αυτό σημαίνει ότι υπάρχει ένα «γλυκό σημείο», ένα ποσοστό εφαρμογών όπου η προσομοίωση είναι απαραίτητη, και έπειτα ένα ποσοστό εφαρμογών ή σταδίων στην ανάπτυξη, όπου χρησιμοποιώντας ένα πραγματικό ρομπότ η προσομοίωση να είναι ευκολότερη και πιο ουσιαστική . Δεδομένου ότι βελτιώνεται ο χρόνος εκτέλεσης της προσομοίωσης, το ποσοστό εφαρμογών που είναι κατάλληλο επεκτείνεται. Η αύξηση της επεξεργασίας καθώς και η ποιότητα του Microsoft robotics studio βοηθούν στο να επιλυθούν διάφορα ζητήματα

1 Έλλειψη πραγματικών/απροσδιόριστων δεδομένων (Lack of noisy data)

Οι άνθρωποι που συμμετέχουν στις μεγάλες προκλήσεις ρομποτικής λένε ότι πρέπει να ξοδεύεται αρκετός διαθέσιμος χρόνος με το πραγματικό ρομπότ ανεξάρτητα από το πόσο καλή ήταν η προσομοίωση. Από την άλλη όμως χρειάζεται σοβαρή δουλειά για να είναι η προσομοίωση χρήσιμη και ρεαλιστική και αυτό γιατί ο πραγματικός κόσμος είναι απρόβλεπτος και σύνθετος με πολύ θόρυβο που περισυλλέγεται από τους αισθητήρες.

2 Ελλιπή και ανακριβή πρότυπα (μοντέλα)

Ένας μεγάλος αριθμός επιδράσεων στον πραγματικό κόσμο είναι ακόμα ανεξήγητος ή πολύ δύσκολο να μοντελοποιηθεί. Αυτό σημαίνει ότι μπορεί να μην είναι εφικτή η μοντελοποίηση όλων με ακρίβεια, ειδικά στον πραγματικό χρόνο. Για παράδειγμα, σε ορισμένες οντότητες, όπως στα τροχοφόρα οχήματα η κίνηση σε χαμηλές ταχύτητες είναι ακόμα μια μεγάλη πρόκληση για τις μηχανές προσομοίωσης

3 Χάσιμο χρόνου στη συμβατότητα (Lots of time for tuning)

Στο χρόνο εκτέλεσης της προσομοίωσης, είναι πραγματικά πολύ εύκολο να τεθεί ένα ρομπότ στον εικονικό κόσμο έτσι ώστε να κινείται αλληλεπιδρώντας με άλλα αντικείμενα. Απαιτείται ακόμη σημαντική προσπάθεια για να συντονιστεί με το προσομοιωμένο υλικό hardware, ώστε να συμπεριφέρεται με τα αντίστοιχα μέρη του πραγματικού κόσμου. Η χρησιμοποίηση της τεχνολογίας AGEIA PhysX, αποτελεί μια πολύ καλή αφετηρία αλλά απαιτείται περισσότερη προσπάθεια συντονισμού των παραμέτρων προσομοίωσης μέσω των αυτοματοποιημένων εργαλείων.

4.1.2 Επισκόπηση του χρόνου εκτέλεσης προσομοίωσης

Ο χρόνος εκτέλεσης προσομοίωσης αποτελείται από τα ακόλουθα συστατικά:

- **Simulation engine service** - είναι αρμόδια για την απόδοση των οντοτήτων και την πρόοδο του χρόνου προσομοίωσης για τη φυσική μηχανή. Ακολουθεί ολόκληρη την κατάσταση προσομοίωσης και παρέχει την υπηρεσία στο τέλος της προσομοίωσης.

- **Managed physics engine wrapper** - αποσύρει το χρήστη από τη φυσική μηχανή χαμηλού επιπέδου API, παρέχει μια πιο συνοπτική, διοικούμενη διεπαφή στην προσομοίωση.

- **Native physics engine library** - επιτρέπει την επιτάχυνση υλικού hardware μέσω της τεχνολογίας AGEIA PhysX, η οποία υποστηρίζει την επιτάχυνση υλικού hardware μέσω του επεξεργαστή τεχνολογίας AGEIA PhysX.

- **Οντότητες (entities)** - αντιπροσωπεύουν το υλικό hardware και φυσικά αντικείμενα στον κόσμο προσομοίωσης. Διάφορες οντότητες είναι προκαθορισμένες με το Microsoft robotics studio και επιτρέπουν στους χρήστες να τις συγκεντρώσουν γρήγορα και να δημιουργήσουν πλούσιες προσομοιωμένες πλατφόρμες ρομπότ στα διάφορα εικονικά περιβάλλοντα.

Μπορείτε να επιλέξετε να αλληλεπιδράσετε μόνο με τη διοικούμενη φυσική μηχανή API εάν δεν θέλετε οποιαδήποτε απεικόνιση. Εντούτοις, έντονα συνίσταται να χρησιμοποιείται πάντα η υπηρεσία simulation engine και να καθορίζετε τις οντότητες που θέτουν εκτός λειτουργίας την απόδοση. Αυτό απλοποιεί πολύ την εμμονή της κατάστασης, την επιθεώρηση και τη διόρθωση του κώδικα προσομοίωσης.

4.1.3 Προγραμματισμός του Simulation (Simulation programming)

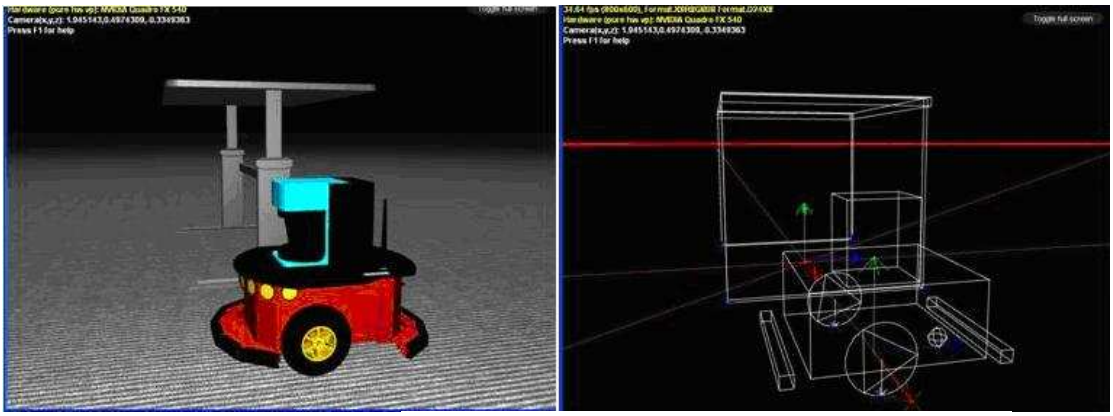
Δύο τμήματα λογισμικού περιλαμβάνονται συνήθως για τη προσομοίωση ενός φυσικού συστατικού και της υπηρεσίας του:

- **Οντότητα (entity)** - είναι το τμήμα λογισμικού που διασυνδέεται με τη φυσική μηχανή και τη μηχανή απόδοσης. Εκθέτει τις κατάλληλες διεπαφές υψηλού επιπέδου για να προσομοιωθεί το υλικό και να κρύψει τη συγκεκριμένη χρήση της φυσικής API.

- **Υπηρεσία (service)** - αυτή χρησιμοποιεί τους ίδιους τύπους και λειτουργίες καθώς η υπηρεσία προσομοιώνεται και παρέχει ολοκληρωμένο τέλος στην οντότητα, ακριβώς όπως στη ρομποτική οι υπηρεσίες παρέχουν ολοκληρωμένο τέλος στο υλικό hardware του ρομπότ.

Τα δείγματα των υπηρεσιών προσομοίωσης που προσομοιώνουν τους αισθητήρες, τους ενεργοποιητές και τις υπηρεσίες υψηλού επιπέδου μπορούν να βρεθούν στο φάκελο δειγμάτων (samples) του καταλόγου του Microsoft robotics studio.

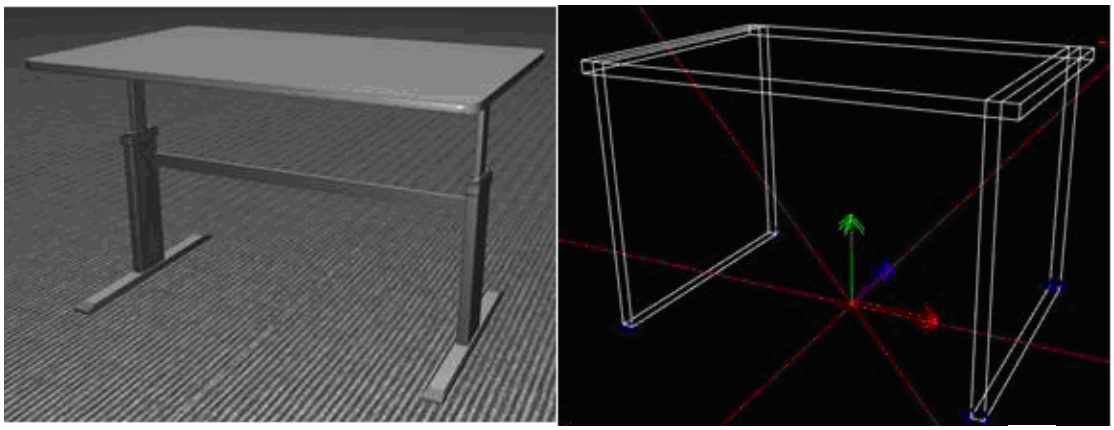
4.1.4 Εικόνες από Simulations



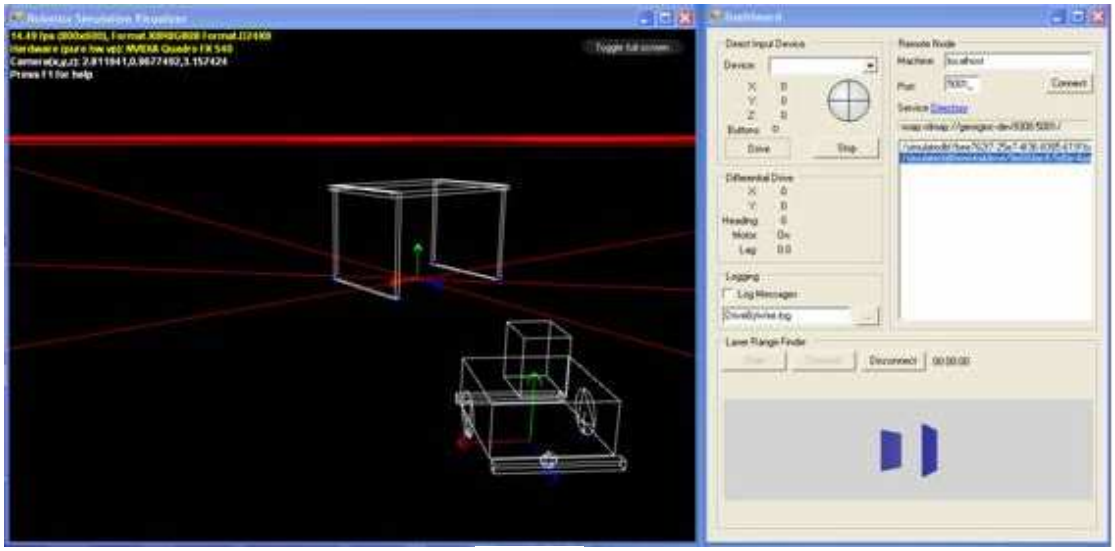
Εικόνα 4.1.4.α :Ρομπότ με διαφορική κίνηση, ανιχνευτή λέιζερ και σειρά προφυλακτήρων(bumper array).

Στην πρώτη εικόνα παρουσιάζεται μια βάση ρομπότ με διαφορική κίνηση (differential drive), ανιχνευτή λέιζερ και σειρά προφυλακτήρων (bumper array, **Εικόνα4.1.4.α**).

Η δεύτερη εικόνα (και στις δύο περιπτώσεις του ρομπότ και του τραπεζιού στην **Εικόνα** παρακάτω **4.1.4.β**) είναι της φυσικής άποψης, η οποία δείχνει πώς το τραπέζι και το ρομπότ προσεγγίζονται από στερεά σχήματα (solid shapes).

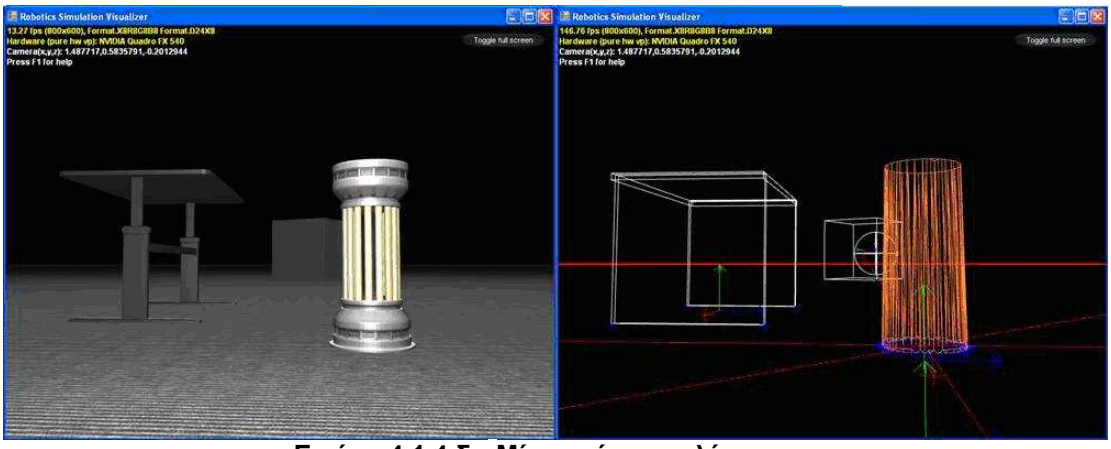


Εικόνα 4.1.4.β : Αναπαράσταση ενός τραπεζιού από στερεό σχήμα αλλιώς μια πολύμορφη οντότητα περιβάλλοντος και το φυσικό πρότυπο της.



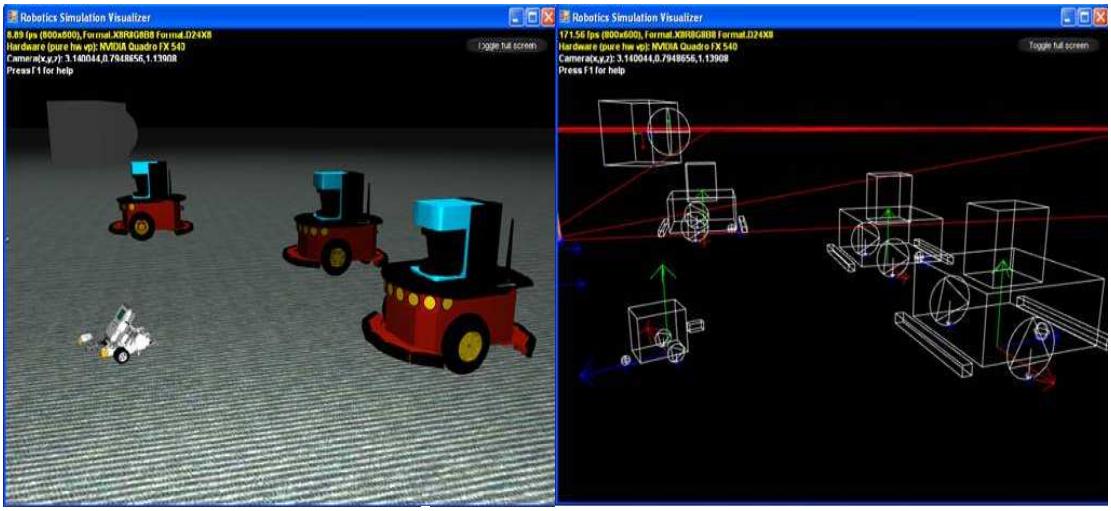
Εικόνα 4.1.4.γ : Ταμπλό ελέγχου του λέιζερ.

Απλό ταμπλό (dashboard) που ελέγχει το προσομοιωμένο λέιζερ κατά την φυσική άποψη(**Εικόνα4.1.4.γ**).



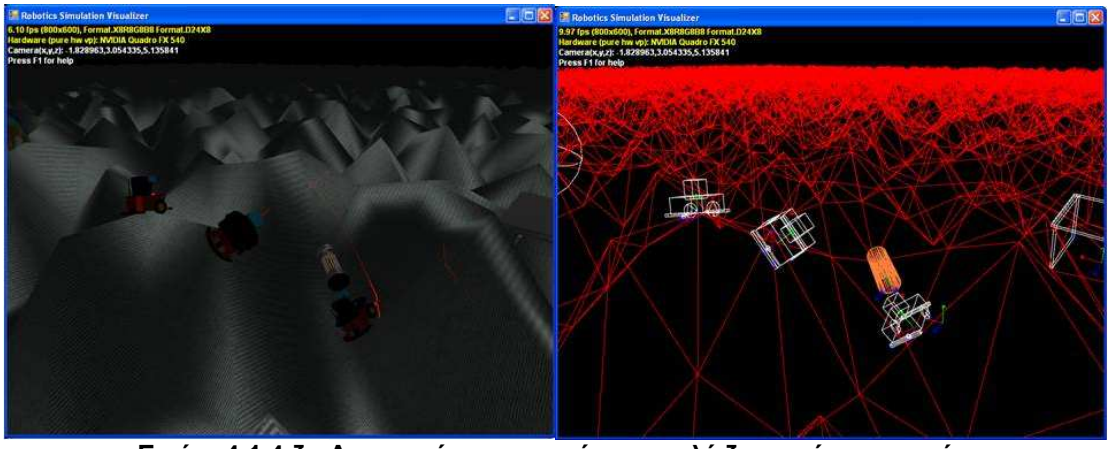
Εικόνα 4.1.4.δ : Μία οντότητα πλέγματος.

Σύνθετο αρχείο βασισμένο σε οντότητα πλέγματος, με χρόνο εκτέλεσης παραγόμενο και απλοποιημένο σε κυρτό πλέγμα (**Εικόνα4.1.4.δ**).



Εικόνα 4.1.4.ε : Αναπαράσταση τεσσάρων ρομπότ.

Friends - τρία ρομπότ Mobile Robots Pioneer3DX με λέιζερ συν το νέο LEGO® MIND STORMS™ NXT (**Εικόνα 4.1.4.ε**).



Εικόνα4.1.4.ζ : Αναπαράσταση αντίκτυπου λέιζερ από το ρομπότ.

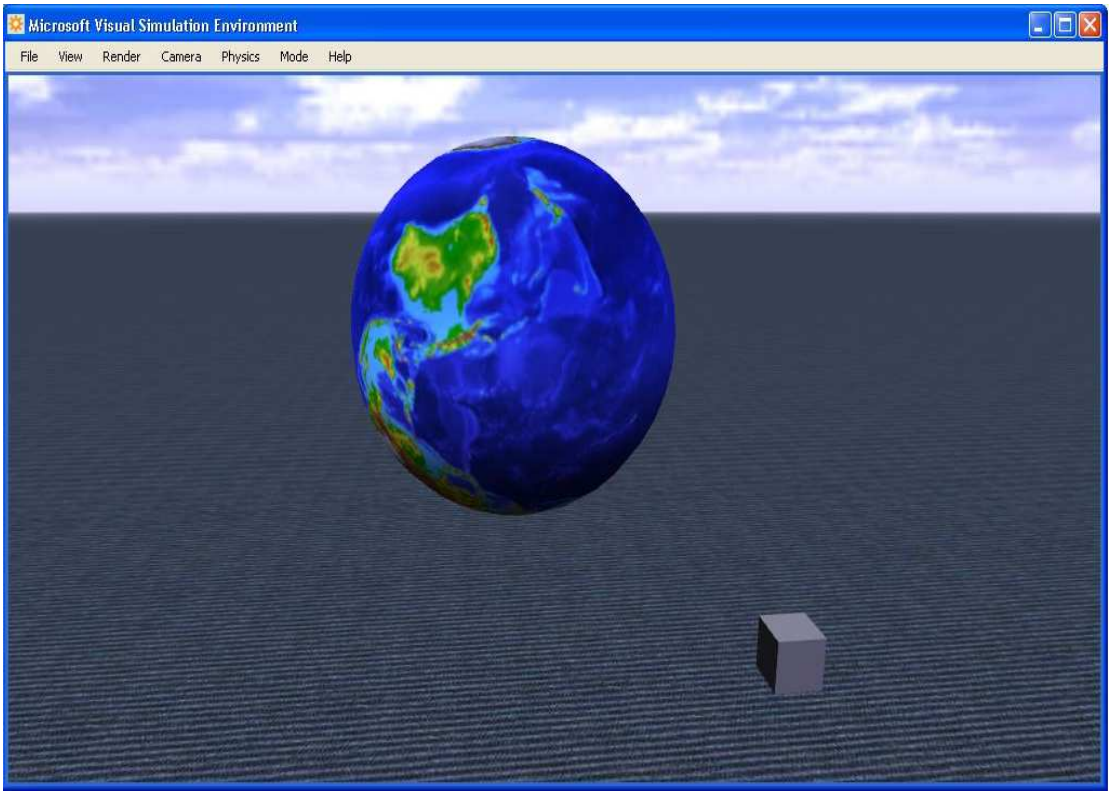
Ανώμαλη επίγεια επιφάνεια.

Ως παράδειγμα, χρησιμοποιείται μια οντότητα σε ανώμαλο έδαφος, με τυχαία θέση με διαφορά απόστασης ενός μέτρου κάθε φορά. Τα κόκκινα σημεία στο αριστερό μέρος της **Εικόνας4.1.4.ζ** είναι η απεικόνιση των σημείων αντίκτυπου του λέιζερ από τις οντότητες ανιχνευτών με λέιζερ.(τα σημεία αυτά ίσα που διακρίνονται στην εικόνα, λόγω του πολύ πολύ μικρού μεγέθους του λέιζερ).

4.2 Ξεκινώντας το περιβάλλον Visual Simulation

Για να αρχίσει το Microsoft Visual Simulation Environment (vse) κάνετε κλικ σε μία από τις εισόδους στο Visual Simulation Environment κάτω από το Microsoft Robotics Developer Studio2008 Express στο μενού έναρξης. Αυτό

θα εμφανίσει το vse παράθυρο (**Εικόνα4.2.1**) που φορτώνεται με τα διάφορα manifests για τα εγχειρίδια προσομοίωσης. Για την καλύτερη απόδοση καλό θα ήταν να κρυφτούν ή να κλείσουν όλα τα παράθυρα που μπορούν να εμφανιστούν πάνω από το παράθυρο προσομοίωσης.



Εικόνα4.2.1 :Παράθυρο προσομοίωσης με αναπαράσταση μίας σφαίρας.

Όταν ο προσομοιωτής κάνει εκτέλεση, μπορεί να κινηθεί η κάμερα με το σύρσιμο του δείκτη του ποντικιού πάνω στην οθόνη. Δεν αλλάζει τη θέση της κάμερας αλλά αλλάζει το σημείο που η κάμερα εξετάζει.

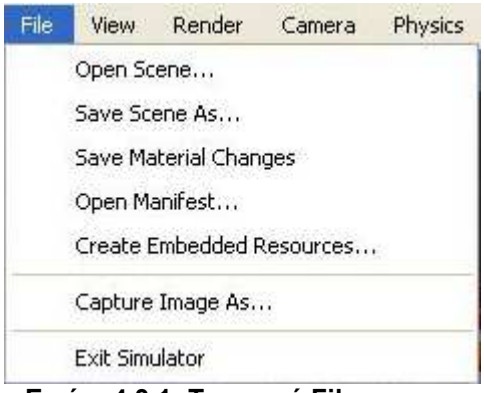
Για την κίνηση της κάμερας, μπορεί να χρησιμοποιηθεί το πληκτρολόγιο όπως φαίνεται στον ακόλουθο πίνακα:

Πλήκτρο	**Δράση**
W	κινείται μπροστά
S	κινείται προς τα πίσω
A	κινήσεις στα αριστερά
D	κινήσεις στα δεξιά
Q	κινείται επάνω
E	κινείται κάτω

4.3 Το μενού του Visual Simulation

Το μενού File έχει διάφορες επιλογές που παρουσιάζονται στη συνέχεια (**Εικόνα 4.3.1**).

- **The File Menu**



Εικόνα4.3.1 :Το μενού File.

Open scene – εμφανίζει μια σκηνή.
Save scene as – αποθηκεύει μια σκηνή. Όταν αποθηκεύετε μια σκηνή, ο προσομοιωτής αποθηκεύει την κατάσταση του προσομοιωτή μαζί με τη κατάσταση για κάθε οντότητα στη σκηνή. Αποθηκεύει επίσης ένα manifest που μπορεί να χρησιμοποιηθεί για να επαναρχικοποιηθεί η σκηνή και οποιεσδήποτε υπηρεσίες που συνδέονται με τις οντότητες.
Open manifest – εμφανίζει ένα manifest.
Capture image as –αποθηκεύει την τρέχουσα κατάσταση της προσομοίωσης σε ένα αρχείο.
Exit – φεύγουμε από τη προσομοίωση και κλείνει ο dss κόμβος.

- **The View Menu**

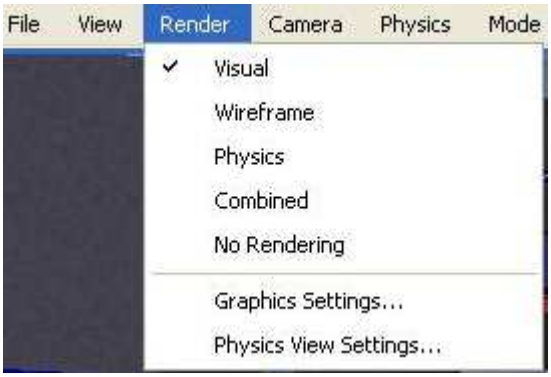
Το μενού View έχει διάφορες επιλογές που παρουσιάζονται στη συνέχεια (**Εικόνα 4.3.2**).

Εικόνα4.3.2 :Το μενού View.

Status bar – εμφανίζει ή κρύβει την μπάρα κατάστασης. Η μπάρα κατάστασης παρουσιάζει το τρέχον ποσοστό πλαισίων σε πλαίσιο-ανά-δεύτερο καθώς επίσης και τη τρέχουσα θέση κάμερας και εξετάζει το σημείο.
Look along - θέτει τη κάμερα σε έναν συγκεκριμένο άξονα εξέτασης.

- **The Render Menu**

Το μενού Render έχει διάφορες επιλογές που παρουσιάζονται παρακάτω(**Εικόνα 4.3.3**).



Εικόνα4.3.3 :Το μενού Render.

Οι πρώτες τέσσερις καταχωρήσεις σε αυτό το μενού επιτρέπουν την αλλαγή του τρόπου με τον οποίο εμφανίζονται οι οντότητες στην προσομοίωση. Μπορεί να γίνει επιλογή μεταξύ αυτών των καταχωρήσεων χρησιμοποιώντας το πλήκτρο F2 του πληκτρολογίου.

Visual - δίνει μια πλήρη τρισδιάστατη εικόνα. Τα πλέγματα που συνδέονται με κάθε οντότητα στη σκηνή δίνονται με ρεαλιστικό φωτισμό και σκίαση.
Wireframe - δίνει τη σκηνή ως κατάσταση wireframe. Έτσι φαίνονται πόσα πολύγωνα δημιουργήθηκαν για κάθε πλέγμα και που είναι οι άκρες των πολυγώνων.
Physics - δίνει τη σκηνή που παρουσιάζει τις φυσικές γραμμές. Έτσι φαίνεται πώς κάθε οντότητα μοντελοποιείται στη φυσική μηχανή. Η σκηνή δεν δίνεται εντελώς εάν η φυσική μηχανή είναι εκτός λειτουργίας.
Combined - δίνει την πλήρη τρισδιάστατη εικόνα με φυσική άποψη. Αυτός ο τρόπος διευκολύνει τον καθορισμό του πόσο καλά ταιριάζουν τα φυσικά σχήματα με το οπτικό πλέγμα για κάθε οντότητα. Δε δίνεται όλο το μέρος αυτής της φυσικής σκηνής αν η φυσική μηχανή είναι εκτός λειτουργίας.
Graphics settings - επιτρέπει την αλλαγή στις ρυθμίσεις που ελέγχουν για το πώς δίνεται η σκηνή.

- **The Camera Menu**

Το μενού Camera έχει διάφορες επιλογές που ακολουθούν παρακάτω (**Εικόνα 4.3.4**).

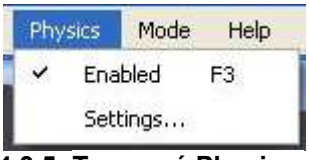
Εικόνα4.3.4 :Το μενού Camera.

Οι επιλογές κάμερας επιτρέπουν την πρόσβαση μεταξύ των καμερών εάν υπάρχουν περισσότερες από μία σκηνές. Πατώντας το F8 η πρόσβαση μεταξύ των καμερών γίνεται γρηγορότερα.

Κύρια κάμερα - θέτει την κατάσταση της προσομοιωμένης κάμερας που παρέχεται σε κατάσταση default(προεπιλογή).
Άλλες κάμερες - θέτει την κατάσταση από άλλες κάμερες που καθορίζονται στο τρέχον manifest.

- **The Physics Menu**

Το μενού Physics έχει διάφορες επιλογές που ακολουθούν παρακάτω(**Εικόνα 4.3.5**).

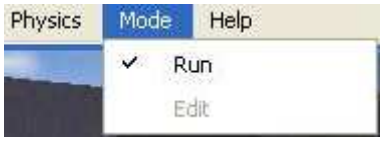


Εικόνα4.3.5 :Το μενού Physics.

Enabled- Επιτρέπει (ή θέτει εκτός λειτουργίας) τις δυνάμεις φυσικής στην προσομοίωση.
Settings - επιτρέπει τον έλεγχο για το αν η κάμερα προεπιλογής αντιμετωπίζεται ως άκαμπτο σώμα και την ρύθμιση βαρύτητας. Εάν η επιλογή κάμερας έχει καθοριστεί, μπορεί να χρησιμοποιηθεί η κάμερα για εστίαση σε αντικείμενα στη σκηνή.

- **The Mode Menu**

Το μενού Mode έχει διάφορες επιλογές που ακολουθούν παρακάτω(**Εικόνα 4.3.6**).



Εικόνα4.3.6 :Το μενού Mode.

Οι ρυθμίσεις σε αυτό το μενού επιτρέπουν σε σας να αλλάξετε τον τρόπο με τον οποίο μπορείτε να αλληλεπιδράσετε με τις οντότητες στη σκηνή.

Run- ο κανονικός τρόπος λειτουργίας για την προσομοίωση.
Edit - ο τρόπος που επιτρέπει τον καθορισμό της κατάστασης των οντοτήτων στην προσομοίωση. Αυτός ο τρόπος θέτει εκτός λειτουργίας αυτόματα τη φυσική. (Physics)

- **The Help Menu**

Το μενού Help έχει διάφορες επιλογές που ακολουθούν παρακάτω(**Εικόνα 4.3.7**).

Εικόνα4.3.7 :Το μενού Help.

Contents- παρουσιάζει ιστοσελίδες που παρέχουν περισσότερες πληροφορίες για το πώς να χρησιμοποιηθούν τα διάφορα χαρακτηριστικά γνωρίσματα του οπτικού περιβάλλοντος προσομοίωσης.
About - παρουσιάζει ένα dialog που επιδεικνύει πληροφορίες για την έκδοση του οπτικού περιβάλλοντος προσομοίωσης και επίσης πληροφορίες για το τρέχον υλικό hardware γραφικών.

4.4 Η χρήση Keyboard και Mouse στο Visual Simulation

Το VSE (visual simulation environment) παράθυρο προσομοίωσης παρέχει λειτουργίες πληκτρολογίου και ποντικιού που μπορούν να χρησιμοποιηθούν για τον έλεγχο της κατάστασης και των στοιχείων γ στο περιβάλλον προσομοίωσης.

- **Χρησιμοποίηση του πληκτρολογίου**

Τα ακόλουθα κλειδιά ελέγχουν τη κίνηση και τον προσανατολισμό της κάμερας στο περιβάλλον προσομοίωσης .

Πλήκτρο	**Δράση**
W	κινείται μπροστά
S	κινείται προς τα πίσω
A	κινήσεις στα αριστερά
D	κινήσεις στα δεξιά
Q	κινείται επάνω
E	κινείται κάτω

Εάν πατηθεί το πλήκτρο Shift με αυτά τα κλειδιά, τα κλειδιά κίνησης κινούν 20 χρόνους γρηγορότερα.

Πλήκτρο	Δράση
F2	Αλλάζει την δεδομένη κατάσταση
F3	αλλαγή της φυσικής μηχανής που είναι ενεργοποιημένη
F5	αλλαγή μεταξύ του edit mode και run mode
F8	Αλλάζει την ενεργό κάμερα

Όταν ο προσομοιωτής είναι στη κατάσταση Edit τα κλειδιά λειτουργούν το ίδιο αλλά μερικές πρόσθετες επιλογές είναι διαθέσιμες όταν είναι πατημένο το αριστερό Ctrl κλειδί. Όταν το αριστερό CTRL πλήκτρο πιέζεται, η τρέχουσα επιλεγμένη οντότητα θα «φωτιστεί» εάν συνδέεται με μια οριοθετημένη σφαίρα. Καθώς το πλήκτρο CTRL πιέζεται, τα ακόλουθα πρόσθετα κλειδιά είναι διαθέσιμα:

Πλήκτρο	Δράση
up arrow	Εμφανίζει το επιλεγμένο αντικείμενο από πάνω (μια θετική απόσταση Y από το αντικείμενο)
Shift+Up arrow	Εμφανίζει το επιλεγμένο αντικείμενο από κάτω (μια αρνητική απόσταση Y από το αντικείμενο)
left arrow	Εμφανίζει το επιλεγμένο αντικείμενο από μια θετική απόσταση X από το αντικείμενο
Shift+Left arrow	Εμφανίζει το επιλεγμένο αντικείμενο από μια αρνητική απόσταση X από το αντικείμενο
right arrow	Εμφανίζει το επιλεγμένο αντικείμενο από μια θετική απόσταση z από το αντικείμενο
Shift+Right arrow	Εμφανίζει το επιλεγμένο αντικείμενο από μια αρνητική απόσταση z από το αντικείμενο

- **Χρησιμοποίηση του ποντικιού**

Στις περισσότερες περιπτώσεις, το σύρσιμο του δρομέα ποντικιού διαμέσου του παραθύρου των γραφικών κρατώντας πατημένο το αριστερό κουμπί και δίνοντας κίνηση στο ποντίκι αναγκάζει την κατάσταση της κάμερας να αλλάξει. Γενικά, οι εντολές πληκτρολογίου έχουν επιπτώσεις στη θέση της κάμερας ενώ η μετακίνηση ποντικιού έχει επιπτώσεις στον προσανατολισμό της κάμερας.

Όταν ο προσομοιωτής είναι στην κατάσταση Edit Mode και το αριστερό Ctrl κλειδί είναι πατημένο, η συμπεριφορά του ποντικιού μπορεί να είναι διαφορετική. Εάν μια οντότητα επιλέγεται και το position property επιλέγεται στο property window, η κίνηση του ποντικιού επιδρά στη θέση της οντότητας.

Εάν οποιοδήποτε συστατικό του άξονα θέσης επιλέγεται, η οντότητα περιορίζεται να κινηθεί μόνο κατά μήκος εκείνου του άξονα.

Ομοίως, εάν η ιδιότητα περιστροφής επιλέγεται στο property window, η κίνηση του ποντικιού επιδρά στον προσανατολισμό της οντότητας. Εάν οποιοδήποτε συστατικό του άξονα περιστροφής επιλέγεται, η οντότητα περιορίζεται για να περιστραφεί μόνο κατά μήκος εκείνου του άξονα.

Το σύστημα συντονισμού του Simulation

Ο προσομοιωτής χρησιμοποιεί ένα σύστημα συντονισμού δεξιοστρεφές. Ο +Y άξονας αντιπροσωπεύει την ανύψωση πάνω από το επίπεδο. Οι άξονες X και Z είναι παράλληλοι στο επίπεδο. Όταν ορίζεται η +X κατεύθυνση, τότε ο + Z άξονας είναι στα δεξιά.

Μερικά εργαλεία διαμόρφωσης χρησιμοποιούν ένα διαφορετικό σύστημα συντονισμού και είναι σημαντικό να εξάγονται τα μοντέλα σας που περιστρέφονται κατά τέτοιο τρόπο ώστε να εμφανίζονται σωστά στο περιβάλλον προσομοίωσης.

4.5 Το εργαλείο Obj-to-Bos File Converter

Το ".obj " to ".bos" File Converter(obj2bos.exe), είναι ένα εργαλείο που μετατρέπει ένα αρχείο Object simulation mesh file σε ένα βελτιστοποιημένο δυαδικό αρχείο.

Παράμετροι:

Επιλογή	**Σύντομο όνομα**	**Περιγραφή**
infile:<string >	/ ι	.obj αρχείο να μετατρέπεται σε .bos . Οι χαρακτήρες * και ? επιτρέπονται.
Level: None|Default| Maximum	/ l	Επίπεδο βελτιστοποίησης. Η υψηλότερη βελτιστοποίηση αυξάνει το χρόνο επεξεργασίας αλλά μειώνει το μέγεθος του πλέγματος.
outfile:<string >	/ o	Προαιρετικό όνομα αρχείου παραγωγής. Η επέκταση .bos δεν απαιτείται. Εάν αυτή η παράμετρος δεν διευκρινίζεται, το αρχείο παραγωγής θα γίνει default στο ίδιο όνομα ως αρχείο εισαγωγής αλλά με μια επέκταση.bos.
Verbosity: Off|Error|Warning|Info|Verbose	/ v	Επίπεδο διεξοδικότητας.

<u>Παράδειγμα</u>

Χρησιμοποίηση μιας εντολής studio robotics της Microsoft για τη μετατροπή του Earth.obj σε earth.bos στον ίδιο φάκελο:

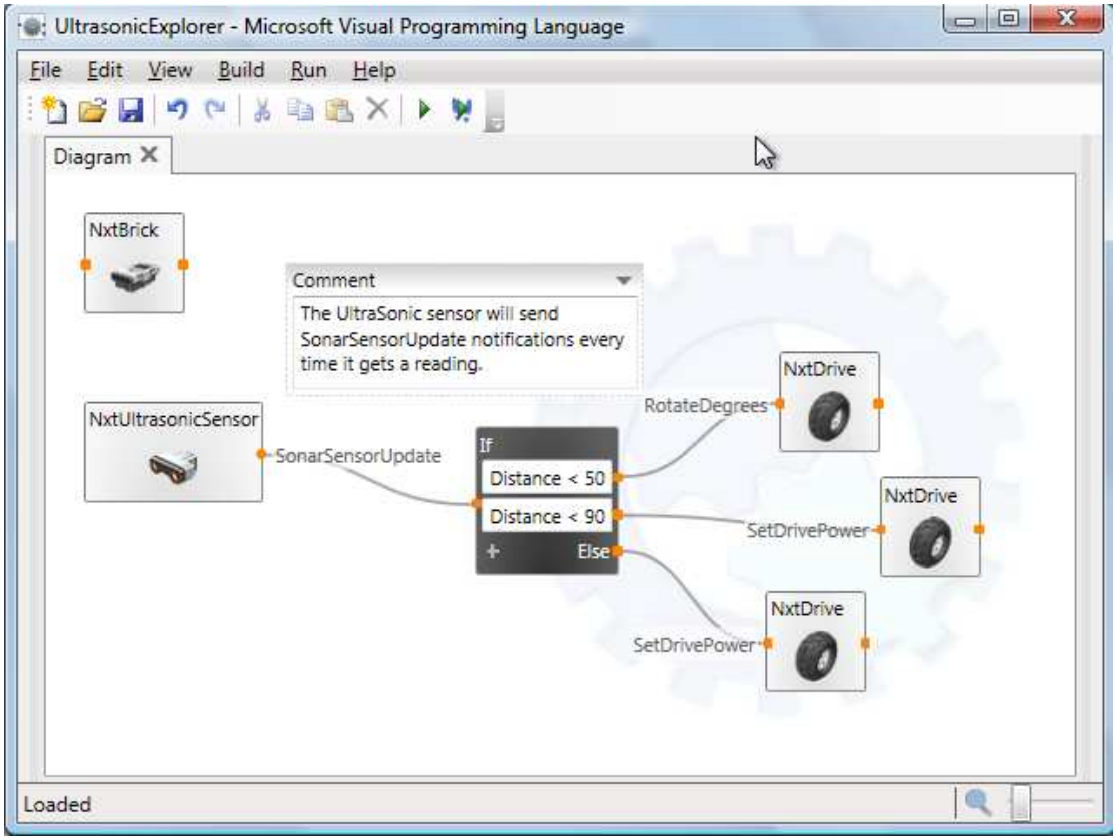
Obj2Bos.exe/i:"store\. Media \ earth .obj"

Κεφάλαιο 5

LEGO MIND STORMS NXT Services

5.1 Εισαγωγή

Οι νέες υπηρεσίες LEGO MINDSTORMS NXT έχουν σχεδιαστεί για να χρησιμοποιηθούν στην γλώσσα προγραμματισμού visual programming language (**Εικόνα 5.1.1**).

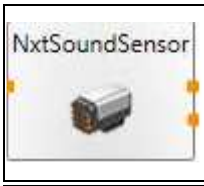
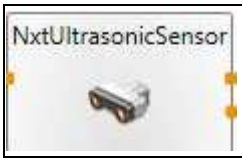


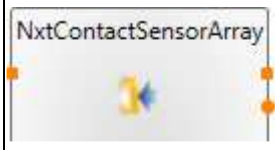
Εικόνα5.1.1 :Το παράθυρο της γλώσσας VPL παρουσιάζοντας τη χρήση υπηρεσιών LEGO MINDSTORMS NXT .

5.2 Περίληψη των υπηρεσιών

Αυτές οι υπηρεσίες είναι εύκολο να διαμορφωθούν και να παρέχουν μια επεκτάσιμη αρχιτεκτονική έτσι ώστε να μπορούν να προστεθούν οποιαδήποτε στιγμή αισθητήρες.

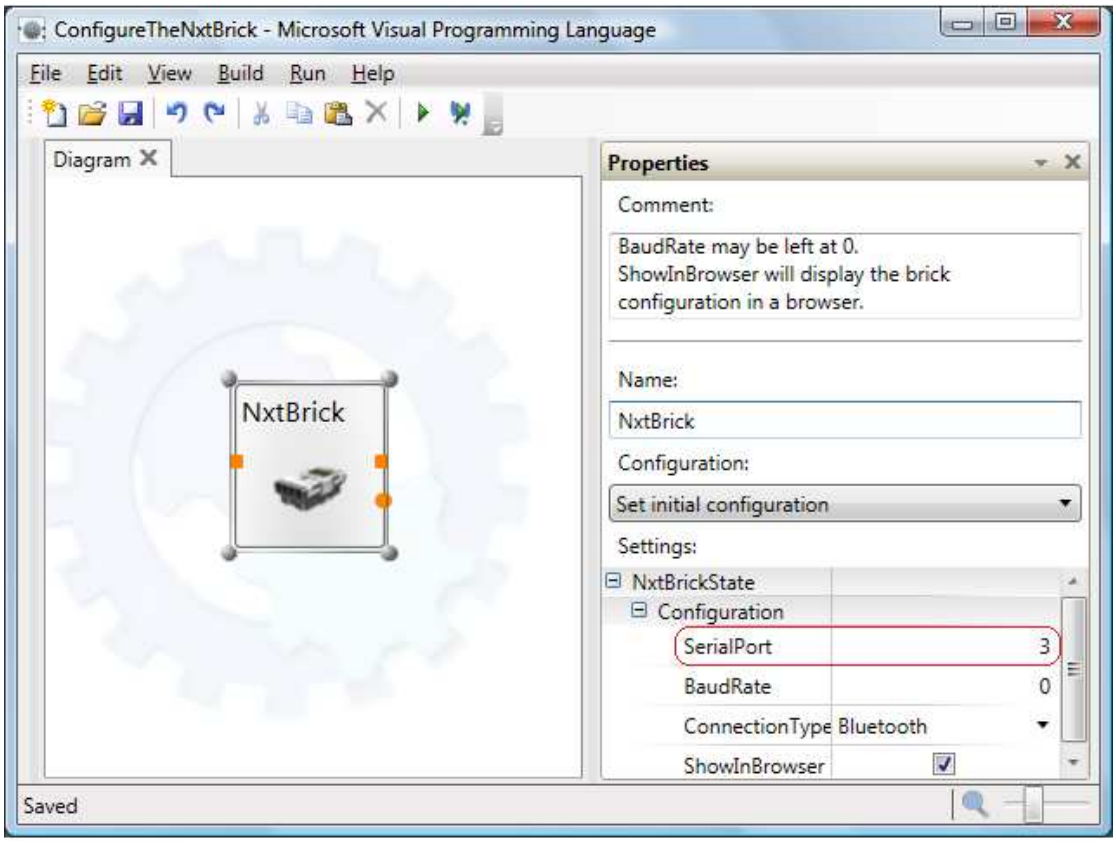
Οι υπηρεσίες που παρέχονται περιλαμβάνουν μηχανές (motors) και αισθητήρες για τις τυποποιημένες συσκευές NXT καθώς επίσης και μερικά παραδείγματα αισθητήρων διαθέσιμων από HiTechnic και MindSensors.

Εικονίδιο στη VPL	Υπηρεσία	Περιγραφή
	Lego NXT Brick (v2)	Ελέγχει τη σύνδεση σε ένα brick LEGO MINDSTORMS NXT. Στο παρόν έγγραφο, θα αναφερθούμε σε αυτό ως NXT brick, ή απλά the brick .
	Lego NXT Motor (v2)	Ελέγχει τη δύναμη σε μια μηχανή και επιστρέφει προαιρετικά τις αναγνώσεις κωδικοποιητών.
	Lego NXT Drive (v2)	Ελέγχει τη δύναμη σε δύο μηχανές που διαμορφώνονται ως Drive. Οι προηγμένες λειτουργίες όπως DriveDistance και RotateDegrees εφαρμόζονται πλήρως.
	Lego NXT Light Sensor (v2)	Επιτρέπει στον προγραμματιστή να ενεργοποιήσει το σημειακό προβολέα και να στείλει τις ανακοινώσεις καθώς η ένταση μέτρησης του φωτός αλλάζει.
	Lego NXT Sound Sensor (v2)	Στέλνει τις ανακοινώσεις που δείχνουν την ένταση του ήχου που παραλαμβάνεται από τον αισθητήρα ήχου.
	Lego NXT Touch Sensor (v2)	Στέλνει μια ανακοίνωση όταν πιέζεται ο αισθητήρας αφής ή απελευθερώνεται.
	Lego NXT Ultrasonic Sensor (v2)	Στέλνει συνεχείς ανακοινώσεις που δείχνουν την απόσταση στο κοντινότερο αντικείμενο μεταξύ του Ultrasonic αισθητήρα.
	Lego NXT Battery (v2)	Παρέχει συχνές ανακοινώσεις της τρέχουσας τάσης μπαταρίας, και διαμορφώσιμες CriticalBattery ανακοινώσεις.
	Lego NXT Buttons (v2)	Στέλνει ανακοινώσεις όταν πιέζονται και απελευθερώνονται τα κουμπιά στο brick NXT.

	MindSensors Acceleration Sensor	Στέλνει συνεχείς ανακοινώσεις που δείχνουν την X,Y, & την περιστροφή z του αισθητήρα επιτάχυνσης.
	MindSensors Compass Sensor	Στέλνει συνεχείς ανακοινώσεις που δείχνουν την τρέχουσα επισήμανση πυξίδων.
	Lego NXT Contact Sensor Array (v2)	Αθροίζει πάνω από τέσσερις αισθητήρες LEGO NXT για να παραγάγει μια σειρά αισθητήρων επαφών.(Contact Sensor Array)
	Lego NXT Brick I/O (v2)	Παρέχει την πρόσβαση στα αρχεία και τα προγράμματα άμεσα στο brick NXT. Αυτή η υπηρεσία επιτρέπει επίσης σε σας να πάρετε και να θέσετε το όνομα του NXT brick σας.

5.3 Διαμόρφωση υπηρεσίας Brick (Brick Service Configuration)

Αρχικά σύρεται η " *Brick* υπηρεσία Lego NXT (v2)" προς την επιφάνεια VPL (**Εικόνα 5.3.1**), κατόπιν επιλέγετε "set initial configuration" και αλλάζει το Serial Port (σειριακή θύρα) ανάλογα με την περίπτωση για τη σύνδεση με το Bluetooth.



Εικόνα5.3.1 :Το παράθυρο της γλώσσας VPL παρουσιάζοντας τη χρήση υπηρεσίας NXT *Brick* .

Αυτό το «πακέτο» υποστηρίζει την τρέχουσα σύνδεση Bluetooth στο NXT brick, και μια προεπιλογή BaudRate 0 (μηδέν) θα συνδεθεί στο κατάλληλο Rate.

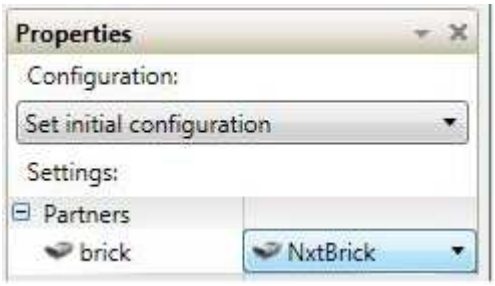
Με την επιλογή "ShowInBrowser" η αλληλεπίδραση του χρήστη για την υπηρεσία LEGO NXT θα ανοίγει σε έναν web browser κάθε φορά που θα αρχίζει η υπηρεσία. Αυτό είναι χρήσιμο για τη διαμόρφωση και την κατάσταση των συσκευών που συνδέονται με το NXT Brick.

5.4 Οι υπηρεσίες του LEGO MINDSTORMS NXT

Το πακέτο LEGO MINDSTORMS NXT έχει τρεις μηχανές και τέσσερις αισθητήρες. Όλες αυτές οι συσκευές συνδέονται με NXT Brick και μοιράζονται μερικές κοινές ρυθμίσεις διαμόρφωσης.

• Η ρύθμιση partners

Όταν χρησιμοποιούνται περισσότερα από ένα LEGO NXT Brick, αυτή η ρύθμιση αναγνωρίζει το brick με το οποίο αυτή η συσκευή είναι συνδεμένη (**Εικόνα 5.4.1**).



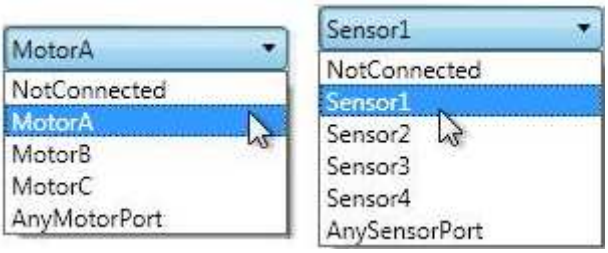
Εικόνα 5.4.1 :Το παράθυρο Properties παρουσιάζοντας την επιλογή NXT *Brick* στα δεξιά και brick στα αριστερά αντίστοιχα .

• Η ονομασία των συσκευών

Οι ίδιες συσκευές μπορούν να ονομαστούν. Εάν μπορούν να συνδεθούν δύο συσκευές του ίδιου είδους με ένα NXT brick, η συσκευή θα έχει μια ιδιότητα ονόματος (Name property) έτσι ώστε να διακρίνονται μεταξύ τους.

• Θύρες μηχανών και αισθητήρων

Το NXT Brick έχει τρεις θύρες μηχανών (MotorA, MotorB, και MotorC,**Εικόνα 5.4.2**), και τέσσερις θύρες αισθητήρων (sensor1-4). Όλες οι εξωτερικές συσκευές για το NXT συνδέονται με μία από αυτές τις θύρες. Οι περισσότερες συσκευές απαιτούν ο χρήστης να γνωρίζει ποια θύρα έχει συνδεθεί με τι.

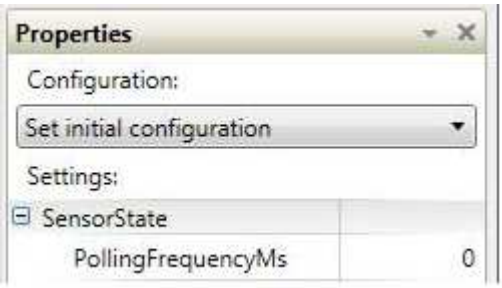


Εικόνα 5.4.2 :Η επιλογή των τριών θυρών μηχανών και των τεσσάρων αισθητήρων του NXT *Brick* .

• Συχνότητα καταγραφής

Οι περισσότεροι αισθητήρες έχουν την ικανότητα της αποστολής ανακοινώσεων ή των αναπροσαρμογών καθώς η τιμή τους αλλάζει. Αυτό επιτυγχάνεται με την συχνή επαφή με το NXT brick και του αισθητήρα. Η συχνότητα καταγραφής επιτρέπει τη διαμόρφωση του πόσο συχνά θα γίνεται αυτό σε κάθε αισθητήρα.

Στις περισσότερες περιπτώσεις, αφήνεται η προεπιλεγμένη συχνότητα καταγραφής που είναι καθορισμένη στο 0 (μηδέν,**Εικόνα 5.4.3**). Μπορεί να υπάρξουν στιγμές όπου θα επιθυμείται η λήψη μίας «ανάγνωσης» του αισθητήρα συχνότερα. Σε αυτήν την περίπτωση, μπορεί να ρυθμιστεί η συχνότητα καταγραφής, η οποία μετριέται σε χιλιοστά του δευτερολέπτου (1/1000 δεύτερα).



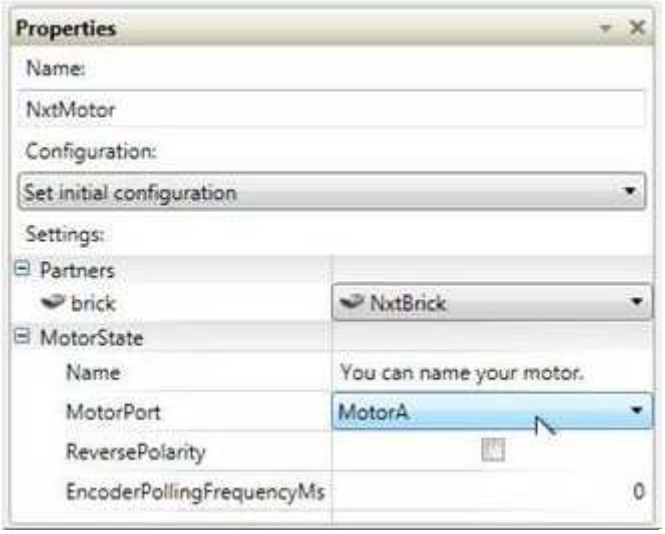
Εικόνα 5.4.3 :Η συχνότητα καταγραφής Polling FrequencyMs ρυθμισμένη στην τιμή μηδέν.

Για τους κωδικοποιητές μηχανών, την αφή, τον ήχο, και τους αισθητήρες φωτός, χρειάζεται περίπου 65ms για να διαβαστούν από τον αισθητήρα. Για τον Ultrasonic αισθητήρα και άλλες συσκευές όπως ο αισθητήρας HiTechnic Acceleration ή ο αισθητήρας MindSensors Compass θα χρειαστούν γύρω στα 200ms. Ρυθμίζοντας την τιμή πολύ χαμηλή δεν θα βλάψει το ρομπότ, αλλά εάν η συχνότητα καταγραφής διαμορφωθεί πολύ χαμηλή για διάφορους αισθητήρες θα γεμίσει το NXT brick με τα αιτήματα, και θα είναι πιο αργός να ανταποκριθεί σε όλα τα αιτήματα αισθητήρων.

Όταν ρυθμίζεται η συχνότητα καταγραφής για έναν αισθητήρα πρέπει να τεθεί η συχνότητα καταγραφής στα 50ms ή λιγότερο. Η χρησιμοποίηση μιας τιμής των 50 ή λιγότερο δείχνει στο σύστημα ότι θα δοθεί προτεραιότητα σε αυτήν την συσκευή, ακόμα κι αν σημαίνει ότι θα λειτουργήσουν πιο αργά άλλοι αισθητήρες. Είναι καλύτερο να τεθεί εκτός λειτουργίας η καταγραφή για οποιουσδήποτε αισθητήρες που δεν χρησιμοποιούνται (με τη ρύθμιση τους σε -1), ή να ρυθμιστούν σε 500 ή υψηλότερα έτσι ώστε να λαμβάνονται περισσότερες «αναγνώσεις» αλλά συνολικά το NXT μπορεί να εστιάσει στα αιτήματα από τον αρχικό αισθητήρα.

5.5 Υπηρεσία κίνησης (Motor service)

Οι μηχανές μπορούν να ελεγχθούν χωριστά ή ως ζευγάρι κίνησης. Οι κωδικοποιητές ενσωματώνονται και στις συσκευές μηχανών και κίνησης, και προσδιορίζουν πόσους βαθμούς η μηχανή έχει περιστραφεί (360 βαθμοί = 1 περιστροφή). Παρακάτω παρουσιάζονται οι επεξηγήσεις μερικών επιλογών του παράθυρου Properties που εμφανίζονται στην **Εικόνα 5.5.1**.



Εικόνα 5.5.1 :Διάφορες επιλογές του παραθύρου Properties ,όπως το Reverse Polarity, MotorPort, Polling Frequency Ms.

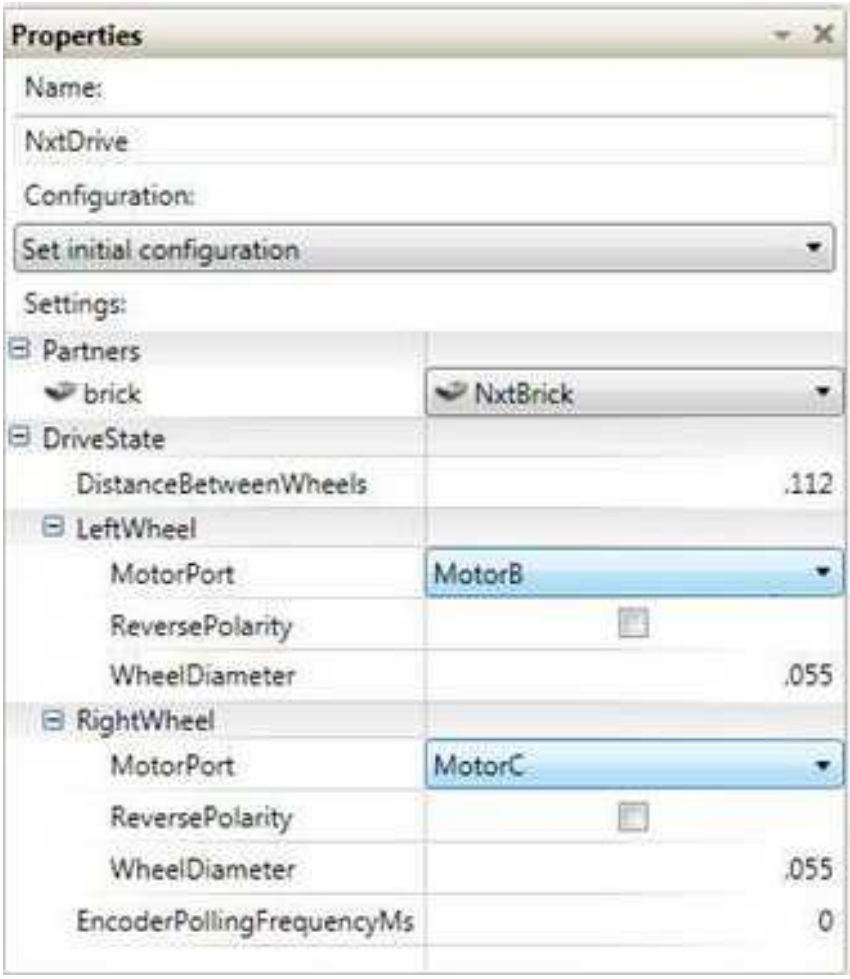
- **Reverse Polarity:** Το ReversePolarity δείχνει την κατεύθυνση (πολικότητα) της μηχανής. Η ενεργοποίηση αυτής της επιλογής (ή ρύθμισή του **σε αληθινό= true**) αντιστρέφει την κατεύθυνση της μηχανής.
- **MotorPort:** Προσδιορίστε τη θύρα (A, b, c) στην οποία η μηχανή έχει συνδεθεί. Εάν διαμορφώσετε τη μηχανή με το **AnyMotorPort**, η υπηρεσία motor θα συνδεθεί με τον πρώτη διαθέσιμη θύρα μηχανών.

Polling Frequency Ms: Αυτή η ιδιότητα διευκρινίζει τη συχνότητα από την οποία οι κωδικοποιητές καταγράφονται:

Τιμή	**Περιγραφή**
0	Προεπιλεγμένη ρύθμιση
-1	Η καταγραφή κωδικοποιητών είναι εκτός λειτουργίας
> 0	ζητούμενη καταγραφή σε χιλιοστά του δευτερολέπτου

5.6 Η υπηρεσία Drive

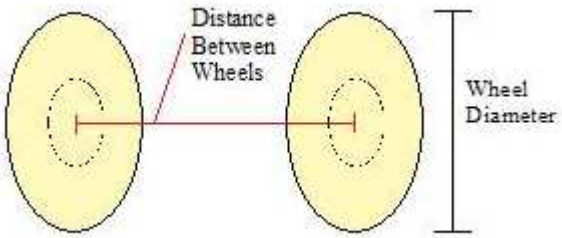
Για να δουλεύει κατάλληλα η υπηρεσία Drive, θα χρειαστεί να διαμορφωθεί η συσκευή αυτή (**Εικόνα 5.6.1**) για να ταιριάξει με τις προδιαγραφές του ρομπότ που έχουν δημιουργηθεί.



Εικόνα 5.6.1 :Διάφορες επιλογές του παραθύρου Properties όπου μπορούμε να τροποποιήσουμε ανάλογα με το τι θέλουμε.

- **Απόσταση μεταξύ τροχών και διάμετρος τροχών**

Θα πρέπει να μετρηθεί η διάμετρος κάθε τροχού και η απόσταση μεταξύ των τροχών (**Εικόνα 5.6.2**). Αυτό θα επιτρέψει στην υπηρεσία να ξέρει πώς να περιστραφεί ή να διανύσει μια συγκεκριμένη απόσταση.



Εικόνα 5.6.2 :Απεικόνιση απόστασης μεταξύ δύο τροχών .

Σημείωση: Στο Microsoft robotics studio οι μετρήσεις είναι πάντα καταγεγραμμένες σε μέτρα. Για το LEGO MINDSTORMS NXT αυτό σημαίνει ότι οι μετρήσεις θα γίνουν σε χιλιοστόμετρα και οι κινήσεις θα γίνουν με το δεκαδικό μέρος τρεις θέσεις προς τα αριστερά. Παραδείγματος χάριν, έστω ότι

οι τροχοί μετρήθηκαν περίπου 55 χιλιοστόμετρα από την κορυφή έως την μέση. Όταν μετατρέπετε αυτό σε μέτρα, έχουμε την τιμή 0,055.

- **Αριστερός και δεξής τροχός μηχανής**

Διαμόρφωση της κίνησης προσδιορίζοντας ποιος τροχός έχει συνδεθεί και με ποια θύρα μηχανής.

- **Αντιστροφή δύναμης (ReversePolarity)**

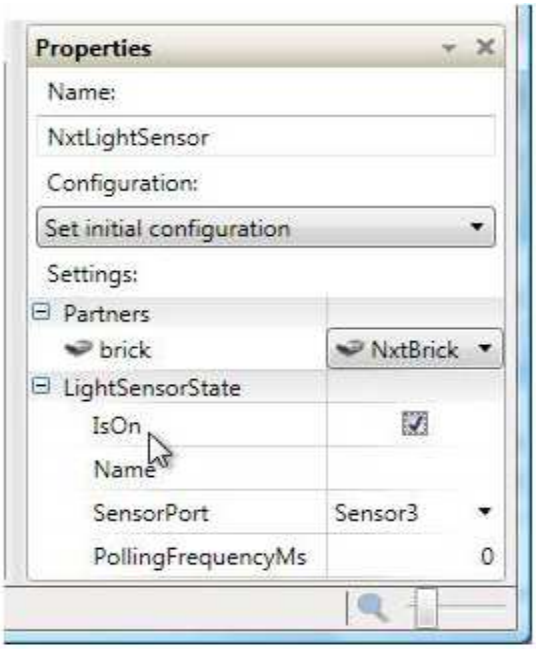
Ανάλογα με το πώς έχει τοποθετηθεί η μηχανή, μπορεί να κινηθεί προς τα πίσω όταν εφαρμόζετε μια θετική δύναμη σε αυτό. Εάν αυτό συμβαίνει, πρέπει να επιλεγεί το box *ReversePolarity*.

- **PollingFrequencyMs**

Οι εντολές drivedistance και rotate degrees θέτουν σε λειτουργία τις μηχανές έως ότου αυτές έχουν διανύσει μια ορισμένη απόσταση. Αυτές οι εντολές χρησιμοποιούν τους κωδικοποιητές που ενσωματώνονται στις μηχανές για να ξέρουν πότε να σταματήσουν. Εάν δεν επιθυμείτε τη χρήση κωδικοποιητών ή αυτών των εντολών τότε πρέπει να ρυθμιστεί η PollingFrequencyMs σε -1.

5.7 Υπηρεσίες Light Sensor

Σε αντίθεση με τις ρυθμίσεις που περιγράφονται στο DeviceServices, η υπηρεσία Light Sensor LEGO MINDSTORMS NXT έχει επιπλέον τη ρύθμιση **IsON (Εικόνα 5.7.1)**. Ο Light Sensor έχει ένα σημειακό προβολέα που μπορεί να ελεγχθεί από την υπηρεσία. Με τη ρύθμιση "IsOn" στη διαμόρφωση, ο σημειακός προβολέας θα ενεργοποιηθεί όταν ξεκινήσει η Light Sensor υπηρεσία.



Εικόνα 5.7.1 :Απεικόνιση ρύθμισης IsON στο παράθυρο Properties.

Κεφάλαιο 6

Κατασκευή ενός απλού κόσμου με το Microsoft Robotics Developer Studio

6.1 Προσομοίωση του κόσμου με την χρήση του Microsoft Robotics Studio

Το Microsoft® Robotics Studio (MSRS) είναι κάτι παραπάνω από το να παίζεις με τα ρομπότ. Το MSRS επιτρέπει τη δημιουργία εφαρμογών που βασίζονται στην εξυπηρέτηση για ένα ευρύ φάσμα συσκευών υλικού hardware. Η σειρά εργαλείων περιλαμβάνει ένα λογισμικό (runtime) το οποίο πρέπει τουλάχιστον να είναι σχετικό με τις εκδόσεις των Windows® Communication Framework (WCF). Επιπλέον, χαρακτηρίζεται από ένα εργαλείο που λέγεται VPL (Visual Programming Language) και ένα περιβάλλον προσομοίωσης VSE (Visual Simulation Environment).

Το MSRS προσφέρει ένα χρονικό διάστημα προσανατολισμένης εξυπηρέτησης, μαζί με τα εργαλεία που απαιτούνται για το σχεδιασμό και την ανάπτυξη εφαρμογών που βασίζονται σε ρομπότ. Περιλαμβάνει οπτικά εργαλεία εκπαιδευτικά έγγραφα, και υλικό σχεδιασμένο για αρχάριους στον νέο κόσμο της ρομποτικής.

Ο χρόνος εκτέλεσης του MSRS αποτελείται από δύο χαμηλότερους χρόνους εκτέλεσης που δεν υπερβαίνουν τον CLR 2,0. Αυτοί οι δύο χρόνοι εκτέλεσης είναι το DSS (Decentralized Software Services) και ο χρόνος εκτέλεσης CCR (Coordination and Concurrency Runtime). Το DSS είναι ένα ουσιαστικά ασήμαντο προσανατολισμένο χρονικό διάστημα εξυπηρέτησης βασισμένο στις αρχές της αντιπροσωπευτικής κατάστασης μεταφοράς REST (Representation State Transfer), οι οποίες χρησιμοποιούνται για να τροφοδοτήσουν το Web. Το CCR είναι μια βιβλιοθήκη .NET Framework της Microsoft. που υποστηρίζει τις ασύγχρονες επεξεργασίες. Αυτό είναι εξαιρετικά σημαντικό για τις εφαρμογές της ρομποτικής, δεδομένου ότι τα δεδομένα στέλνονται συνεχώς και παραλαμβάνονται από πολυάριθμους αισθητήρες και ενεργοποιητές.

Μαζί με το χρόνο εκτέλεσης, το κουτί εργαλείων του MSRS περιλαμβάνει ένα εργαλείο VPL που επιτρέπει τη δημιουργία εφαρμογών της ρομποτικής με απλό σύρσιμο και μεταφορά στοιχείων πάνω σε μια επιφάνεια σχεδίασης. Το MSRS περιλαμβάνει επίσης το VSE που επιτρέπει τον πειραματισμό με σύνθετες προσομοιώσεις που περιλαμβάνουν πολλαπλά ρομπότ και εμπόδια.

6.1.1 Τι είναι η προσομοίωση

Μια προσομοίωση περιέχει μια ή περισσότερες οντότητες και κάθε μια από αυτές τις οντότητες παρέχεται μέσα σε ένα πλαίσιο. Το ποσοστό στο οποίο τα πλαίσια παρέχονται εξαρτάται από τη κάρτα γραφικών της μηχανής.

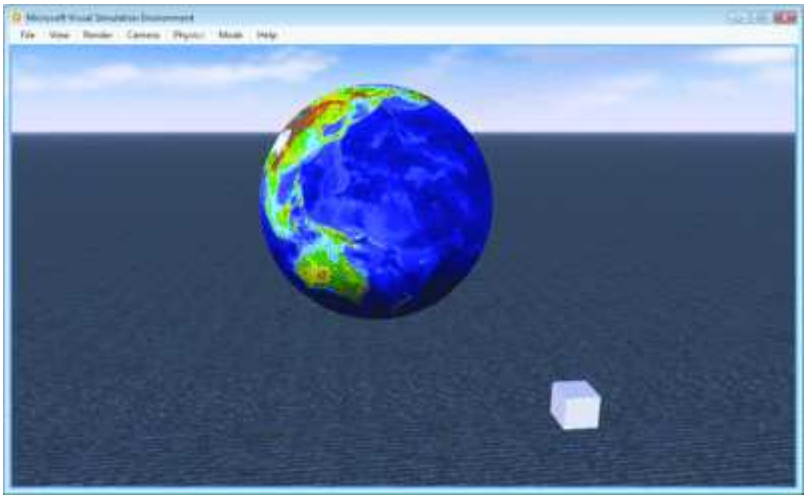
Κάθε πλαίσιο μιας προσομοίωσης δίνεται δυναμικά και οι δυνάμεις που μια οντότητα αντιμετωπίζει μέσα σε κάθε πλαίσιο δεν είναι πάντα γνωστές εκ των προτέρων. Αυτό είναι που καθιστά τις προσομοιώσεις τόσο χρήσιμες. Μπορεί να δημιουργηθεί μία εφαρμογή με ένα ρομπότ ώστε να εμφανιστεί το πώς το ρομπότ θα αλληλεπιδρούσε με τον κόσμο.

6.2 Ξεκινώντας με το πρόγραμμα προσομοίωσης

Το MSRS χρησιμοποιεί τη μηχανή AGEIA PhysX για να ορίσει τη φυσική διάσταση του περιβάλλοντος προσομοίωσης. Χωρίς τη φυσική διάσταση, η προσομοίωση θα ήταν άχρηστη, δεδομένου ότι σε καμία περίπτωση δε θα αντιπροσώπευε τον κόσμο που προσπαθεί να προσομοιώσει. Παραδείγματος χάριν, οι ουσίες θα μπορούσαν να ταλαντεύονται χωρίς βαρύτητα.

Εκτέλεση μιας προσομοίωσης

Πριν τη δημιουργία της προσομοίωσης πρέπει να εκτελεστεί μια από τις προσομοιώσεις που παρέχονται στο MSRS. Μόλις εγκατασταθεί, το MSRS δημιουργεί ένα φάκελο επιλογών που περιέχει εφτά προσομοιώσεις. Η πρώτη προσομοίωση σε αυτόν τον φάκελο, θα παρουσιάσει μια σκηνή παρόμοια με **την Εικόνα6.2.1**.

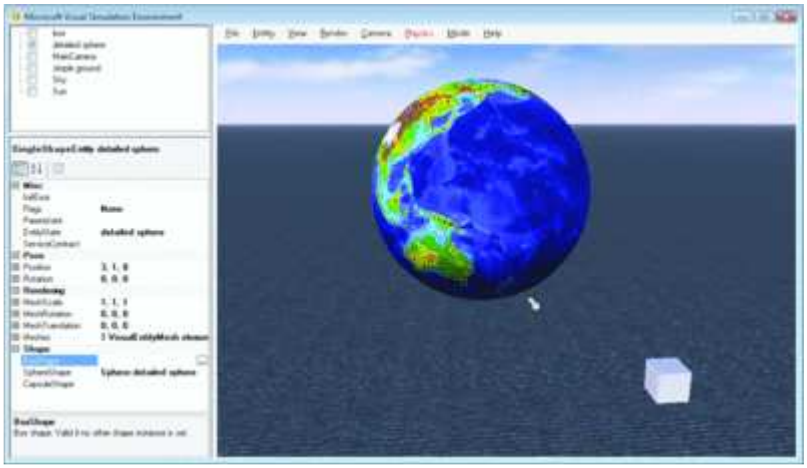


Εικόνα6.2.1: Βασικό περιβάλλον προσομοίωσης.

Θεωρητικά η βασική προσομοίωση περιλαμβάνει μόνο δύο οντότητες: μια παγκόσμια σφαίρα και ένα κουτί. Στην πραγματικότητα, η εικόνα προσομοίωσης περιλαμβάνει επίσης οντότητες που αντιπροσωπεύουν την κύρια κάμερα, το έδαφος, τον ουρανό, και τον ήλιο. Κάνοντας κλικ στο Mode και Edit από το μενού VSE υπάρχει δυνατότητα να εμφανιστούν οι οντότητες που περιλαμβάνονται.

Το Edit, που παρουσιάζεται **στην Εικόνα6.2.2**, παρέχει τη δυνατότητα τροποποίησης ιδιοτήτων που συνδέονται με κάθε οντότητα μέσω του παραθύρου που εμφανίζεται στο αριστερό μέρος της οθόνης. Οι ιδιότητες

αυτές ελέγχουν τα πάντα, από το όνομα της οντότητας μέχρι και τη θέση της στο περιβάλλον προσομοίωσης. Επιπλέον, επιτρέπουν τον ακριβή έλεγχο του πώς η οντότητα μεταδίδεται, επηρεάζοντας τον τρόπο εμφάνισης της προσομοίωσης.



Εικόνα 6.2.2: Καθορισμός τρόπου τροποποίησης των ιδιοτήτων μιας οντότητας

Μέσω της επιλογής Run, παρέχεται η δυνατότητα κίνησης γύρω από την προσομοίωση είτε με τη χρήση του ποντικιού είτε με τα βέλη. Αυτό αλλάζει την οπτική γωνία για την κάμερα, η οποία είναι η οπτική γωνία του χειριστή της κάμερας στην προσομοίωση. Ένα άλλο σημαντικό σημείο που πρέπει να αναφερθεί είναι ότι το VSE επιτρέπει την μετάδοση της εικόνας με διαφορετικούς τρόπους. Η επιλογή Visual είναι η προεπιλογή και είναι μια ρεαλιστική άποψη της εικόνας της προσομοίωσης. Όμως, το Render μενού επιτρέπει την επιλογή μέσω των Wireframe, Physics, Combined, ή καμία επιλογή Render.

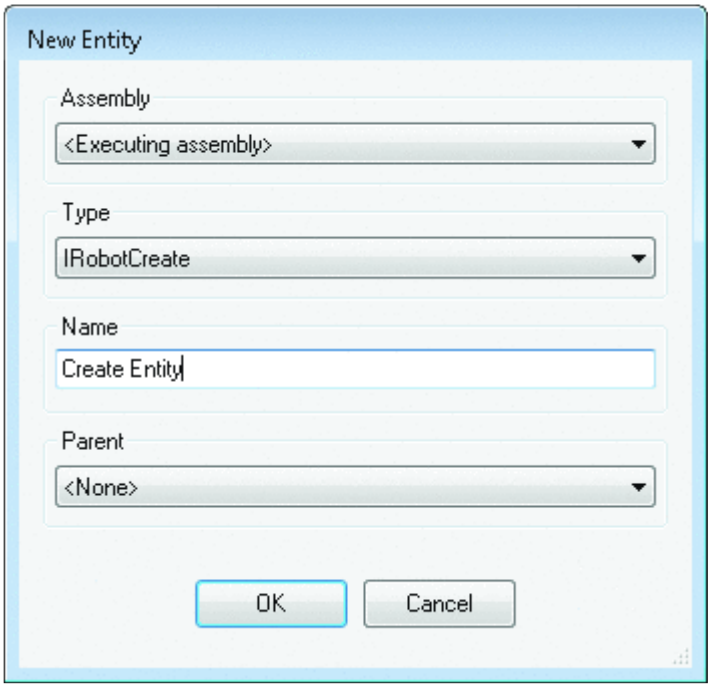
Η επιλογή της μη χρησιμοποίησης Render αντικειμένου συμπεριλαμβάνεται επειδή η εικόνα είναι μόνο μια εκδοχή μιας προσομοίωσης. Αυτό που είναι το πολυτιμότερο για την εκτέλεση μιας προσομοίωσης είναι η αλληλεπίδραση μεταξύ των διάφορων οντοτήτων. Αν οι μετατοπίσεις των οντοτήτων σε μια εικόνα προσομοίωσης είναι ακριβή από άποψη πόρων, τότε η επιλογή της μη χρησιμοποίησης της Render είναι χρήσιμη όταν υπάρχει ένας μεγάλος αριθμός οντοτήτων.

6.3 Οντότητες και τύποι οντοτήτων

Ένας τύπος οντοτήτων επιτρέπει τον καθορισμό μιας νέας εικόνας για ένα ειδικό τύπο οντότητας. Παραδείγματος χάριν, η παγκόσμια σφαίρα που περιλαμβάνεται στο βασικό περιβάλλον προσομοίωσης είναι ένας ενιαίος τύπος οντοτήτων. Ο τύπος οντοτήτων ενεργεί ως πρότυπο για τη νέα οντότητα δεδομένου ότι διευκρινίζει τις ιδιότητες που συνδέονται με έναν ειδικό τύπο οντότητας. Οι τιμές για αυτές τις ιδιότητες μπορούν να αλλάξουν

μόλις δημιουργηθεί η οντότητα, αλλά ο τύπος οντοτήτων καθορίζει ποιες ιδιότητες συμπεριλαμβάνονται.

Το VSE απαιτεί έναν τύπο οντοτήτων προκειμένου να προστεθεί μια οντότητα σε μια προσομοίωση. Παραδείγματος χάριν, για να προστεθεί ένα ρομπότ όπως το Create iRobot, θα πρέπει να προστεθεί μια νέα οντότητα μέσω του VSE κάνοντας κλικ στο Entity (οντότητα) και το New.Αυτό θα εμφανίσει το New Entity dialog box (**Εικόνα6.3.1**).

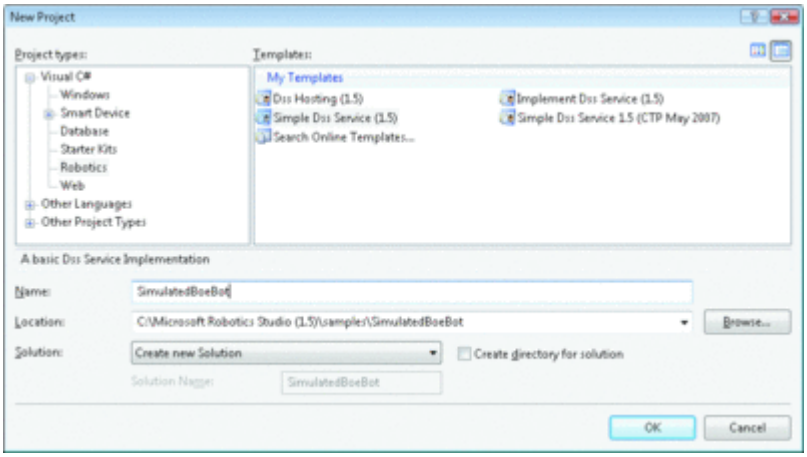


Εικόνα6.3.1 : Εισαγωγή νέας οντότητας στο New Entity Dialog Box.

Αυτό σημαίνει ότι είναι δυνατή η προσθήκη ενός νέου Create ρομπότ στην προσομοίωση χρησιμοποιώντας απλά το New Entity dialog box. Από το New Entity dialog box, επιλέγεται το iRobotCreate ως τύπος και εισάγεται ένα μοναδικό όνομα, όπως "Create Entity" Μόλις επιλεχθεί το OK, το Create ρομπότ εμφανίζεται στην εικόνα της προσομοίωσης.

Η προσθήκη νέα οντότητας μέσω προγραμματισμού.

Το MSRS προσφέρει περισσότερους από έναν τρόπους δημιουργίας και εργασίας με προσομοιώσεις. Εκτός από το VSE, μπορεί να προστεθούν οντότητες προγραμματίζοντας σε μια προσομοίωση δημιουργώντας ένα DSS σχέδιο εξυπηρέτησης. Το MSRS παρέχει ένα Visual Studio πρότυπο που μπορεί να χρησιμοποιηθεί για να δημιουργήσει μια νέα DSS υπηρεσία.. Μόλις εγκατασταθεί το MSRS, πρέπει απλώς να επιλεγεί το Simple DSS Service(1.5) ως πρότυπο κατά τη δημιουργία ενός νέου Visual Studio σχεδίου (**Εικόνα6.3.2**).



Εικόνα6.3.2: Δημιουργία καινούριας DSS Υπηρεσίας στο Visual Studio 2005.

Η δημιουργία μιας νέας DSS υπηρεσίας που χρησιμοποιεί το πρότυπο θα οδηγήσει στη δημιουργία δύο αρχείων κατηγορίας-κλάσης. Η υλοποίηση της κατηγορίας-κλάσης, που έχει εξ ορισμού το ίδιο όνομα με το σχέδιο, είναι όπου θα προστεθεί ο κώδικας που θα δημιουργήσει τη νέα οντότητα. Οι προσομοιώσεις που δημιουργούνται μέσω προγραμματισμού θα απαιτήσουν την πρόσβαση σε συστήματα που δεν περιλαμβάνονται στο απλό DSS πρότυπο υπηρεσιών. Επομένως, θα πρέπει να προστεθούν αναφορές στα συστήματα που απαριθμούνται **στην Εικόνα6.3.3** .

Εικόνα6.3.3: Συναρμολόγηση που σχετίζεται με το σχέδιο της προσομοίωσης

Assembly Name	**Description**
PhysicsEngine	Παρέχει την πρόσβαση στην μηχανή φυσικής λογισμικού AGEIA.
RoboticsCommon	Παρέχει την πρόσβαση στο PhysicalModel namespace, το οποίο χρησιμοποιείται για να καθορίσει τα φυσικά χαρακτηριστικά των ρομπότ.
SimulationCommon	Παρέχει την πρόσβαση στους ορισμούς τύπων οι οποίοι χρησιμοποιούνται κατά την εργασία με τις μηχανές προσομοίωσης και φυσικής.
SimulationEngine	Παρέχει την πρόσβαση στη μηχανή προσομοίωσης.
SimulationEngine.proxy	Αντιπροσωπεύει ένα πληρεξούσιο για τη μηχανή προσομοίωσης, η οποία χρησιμοποιείται κατά τη φόρτωση της μηχανής προσομοίωσης ως συνεργάτη.

Μετά την προσθήκη των αναφορών, θα πρέπει να προστεθούν οι ακόλουθες δηλώσεις namespace στο αρχείο κατηγορίας εφαρμογής:

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

Για να γίνει αντιληπτό ποιος κώδικας απαιτείται για την προσομοίωση, πρέπει πρώτα να εξεταστούν τα εγχειρίδια προσομοίωσης που παρέχονται από το MSRS. Τα δείγματα που παρέχονται στον κατάλογο εγχειριδίων από την εγκατάσταση του MSRS αντιστοιχούν στις διαθέσιμες προσομοιώσεις από τις εγκατεστημένες MSRS επιλογές. Παραδείγματος χάριν, το βασικό περιβάλλον προσομοίωσης είναι το ίδιο με το πρόγραμμα SimulationTutorial1.

Εάν ανοιχτεί το πρόγραμμα SimulationTutorial1 χρησιμοποιώντας τοVisual Studio, φαίνεται ο κώδικας που χρησιμοποιείται για να δημιουργηθεί το βασικό περιβάλλον προσομοίωσης. Το πρώτο πράγμα που παρατηρείται είναι η μέθοδος έναρξης, η οποία καλείται αυτόματα όταν αρχίζει μια υπηρεσία:

```
protected override void Start()
{
    base.Start();
    // Αρχικό σημείο που κοιτάει η προσομοιωμένη κάμερα
    SetupCamera();
    // Προσθέτει οντότητες στο κόσμο προσομοίωσης
    PopulateWorld();
}
```

Η μέθοδος έναρξης βρίσκεται εκεί που έχει προστεθεί ο κώδικας για τον καθορισμό του περιβάλλοντος προσομοίωσης. Για το πρόγραμμα SimulationTutorial1, αυτό περιλαμβάνει το στήσιμο της κάμερας και τον καθορισμό της εικόνας προσομοίωσης με τα αντικείμενα οντότητας.

Εκτός από την κάμερα, το βασικό περιβάλλον προσομοίωσης περιέχει τις οντότητες που χρησιμοποιούνται για να αντιπροσωπεύσουν τον ουρανό, το έδαφος, το κουτί, και την παγκόσμια σφαίρα. Ο κώδικας για να εισαχθεί η παγκόσμια σφαίρα, ή η κατασκευασμένη σφαίρα, παρατίθεται **στην Εικόνα6.3.4**.

Εικόνα 6.3.4:Εισαγωγή της σφαίρας(Insert the World Globe)

```
void AddTexturedSphere(Vector3 position)
{
   SingleShapeEntity entity = new SingleShapeEntity(
      new SphereShape(
         new SphereShapeProperties(10, // μάζα σε  kg
         new Pose(), // τίθεται η μορφή μέσα στην οντότητα
         1)), //η προκαθορισμένη ακτίνα
      position);

   entity.State.Assets.Mesh = "earth.obj";
```

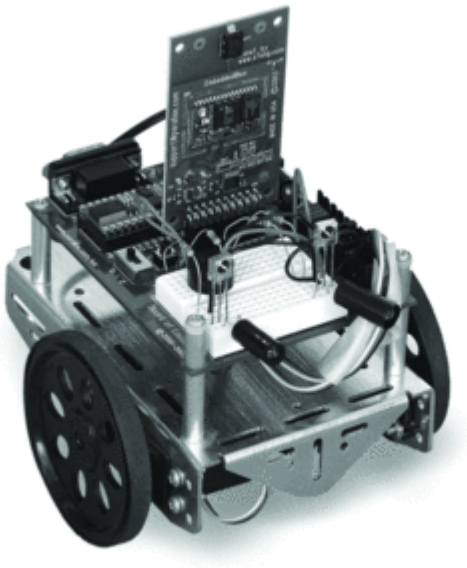
```
    entity.SphereShape.SphereState.Material =
      new MaterialProperties("sphereMaterial", 0.5f, 0.4f, 0.5f);

    // ονομασία της οντότητας
    entity.State.Name = "detailed sphere";

    // προσθήκη οντότητας στη προσομοίωση
    SimulationEngine.GlobalInstancePort.Insert(entity);
}
```

Η πρώτη γραμμή κώδικα στη μέθοδο AddTexturedSphere δημιουργεί μια περίπτωση του SingleShapeEntity. Αυτός ο τύπος αντιπροσωπεύει μια οντότητα με μια ενιαία γεωμετρική μορφή, όπως μια σφαίρα, και είναι χρήσιμος όταν πρέπει να προστεθεί μια οντότητα με μια πολύ απλή φυσική γεωμετρία. Σε αυτήν την περίπτωση, δημιουργείται μια οντότητα με μια μάζα 10 χιλιόγραμμων, ή περίπου τεσσεράμισι λίβρες.

Το πλέγμα που ορίζεται σε αυτήν την οντότητα είναι ένα αρχείο αντικειμένου, το οποίο έχει μια επέκταση αρχείων .obj. Το αρχείο αντικειμένου δημιουργείται με ένα τρισδιάστατο γραφικό εργαλείο έκδοσης και εξάγεται σε άλλο τύπο αντικειμένου. Το MSRS απαιτεί το αρχείο αντικειμένου του πλέγματος να είναι αυτής της μορφής. Το τελευταίο πράγμα που πρέπει να γίνει στη μέθοδο AddTexturedSphere είναι να παρεμβληθεί η οντότητα σφαιρών στο περιβάλλον προσομοίωσης.

Καθορισμός μίας νέας οντότητας στο VSE

Στη συνέχεια παρουσιάζεται ο τρόπος με τον οποίο δημιουργείται μια νέα οντότητα ρομπότ η οποία αντιπροσωπεύει μια παραλλαγή του Boe-Bot. Το Boe-Bot είναι ένα μικρό τροχοφόρο ρομπότ που υποστηρίζει ένα διαφορικό σύστημα κίνησης δύο-τροχών (δείτε **την Εικόνα 6.3.5**).



Εικόνα6.3.5: Το Boe-Bot by Parallax

Το Boe-Bot είναι μια από τις διάφορες πλατφόρμες ρομπότ που υποστηρίζονται από πακέτα εκτός του MSRS. Αυτό σημαίνει ότι η εγκατάσταση του MSRS περιλαμβάνει τις βασικές υπηρεσίες που χρησιμοποιούνται για να ενεργοποιήσουν το σύστημα κίνησης του Boe-Bot και τους ενσωματωμένους αισθητήρες επαφών. Ενώ το MSRS περιλαμβάνει τις υπηρεσίες πλατφορμών για το Boe-Bot, δεν περιλαμβάνει πραγματικά έναν τύπο οντοτήτων Boe-Bot.

Δημιουργία μίας κλάσης της οντότητας

Για τη δημιουργία ενός νέου τύπου οντοτήτων για το Boe-Bot, πρέπει να προστεθεί μια κατηγορία που προέρχεται από το DifferentialDriveEntity. Μέσω της εξαγωγής από την κατηγορία DifferentialDriveEntity, παρέχεται η δυνατότητα της επαναχρησιμοποίησης του κώδικα που καθορίζει πώς το Boe-Bot, πρέπει να συμπεριφερθεί όταν κινείται μέσω μιας προσομοίωσης. Ο κώδικας που χρησιμοποιείται για να δημιουργήσει τον τύπο οντοτήτων BoeBot παρουσιάζεται **στην Εικόνα6.3.6**.

Εικόνα6.3.6 : Δημιουργία οντότητας τύπου BoeBot.

```
[DataContract]
public class BoeBot : DifferentialDriveEntity
{
   Port<EntityContactNotification> _notifications =
         new Port<EntityContactNotification>();

   // προκαθορισμένος κατασκευαστής ,χρησιμοποιείται για την δημιουργία οντότητας από
XML
   public BoeBot() { }

   // Ο προσαρμοσμένος κατασκευαστής για τη δημιουργία του μοντέλου από τις
προκαθορισμένες τιμές.
   //χρησιμοποιήθηκε για την δημιουργία οντότητας προγραμματιστικά
   public BoeBot(Vector3 initialPos)
   {
      MASS = 0.454f; //σε kilograms  (γύρω στο  1 κιλό)
      //οι προκαθορισμένες τιμές των παραμέτρων προσεγγίζουν τα πλαίσια BoeBot
      CHASSIS_DIMENSIONS = new Vector3(0.09f, //μέτρα σε εύρος
                                       0.09f,  //μέτρα σε ύψος
                                       0.13f); //μέτρα σε μήκος
      FRONT_WHEEL_MASS = 0.01f;
      CHASSIS_CLEARANCE = 0.015f;
      FRONT_WHEEL_RADIUS = 0.025f;
      CASTER_WHEEL_RADIUS = 0.0125f;
      FRONT_WHEEL_WIDTH = 0.01f;
      CASTER_WHEEL_WIDTH = 0.008f;
      FRONT_AXLE_DEPTH_OFFSET = 0.01f; //απόσταση από το κέντρο του robot

      base.State.Name = "BoeBot";
      base.State.MassDensity.Mass = MASS;
      base.State.Pose.Position = initialPos;
```

```
    //θέση πλαισίων
    BoxShapeProperties motorBaseDesc =
       new BoxShapeProperties("BoeBot Body", MASS,
          new Pose(new Vector3(
          0, // χρησιμοποίησετο0 για το offset του άξονα X
         CHASSIS_CLEARANCE + CHASSIS_DIMENSIONS.Y / 2,
          0.03f)),//μικρόoffset(αντιστάθμιση)στοβάθοςτουάξοναZ
CHASSIS_DIMENSIONS);

    motorBaseDesc.Material =
        new MaterialProperties("high friction", 0.0f, 1.0f, 20.0f);
    motorBaseDesc.Name = "Chassis";
    ChassisShape = new BoxShape(motorBaseDesc);

    // ο πίσω τροχός ονομάζεται και castor
    CASTER_WHEEL_POSITION = new Vector3(0, // κέντρο του πλαισίου
        CASTER_WHEEL_RADIUS,            //απόσταση από το έδαφος
        CHASSIS_DIMENSIONS.Z / 2);      // στο πίσω μέρος του ρομπότ
    RIGHT_FRONT_WHEEL_POSITION = new Vector3(
        +CHASSIS_DIMENSIONS.X / 2,//αριστερά από το κέντρο
        FRONT_WHEEL_RADIUS,             // απόσταση από το έδαφος του άξονα
FRONT_AXLE_DEPTH_OFFSET); // απόσταση από το κέντρο ,στον άξονα z

    LEFT_FRONT_WHEEL_POSITION = new Vector3(
        -CHASSIS_DIMENSIONS.X / 2,// δεξιά από το κέντρο
        FRONT_WHEEL_RADIUS,      // απόσταση από το έδαφος του άξονα
        FRONT_AXLE_DEPTH_OFFSET); // απόσταση από το κέντρο ,στον άξονα z
    MotorTorqueScaling = 30;

    //καθορίσετε ένα προεπιλεγμένο mesh
    State.Assets.Mesh = "boe-bot.bos";
}
```

Ο κατασκευαστής για την κατηγορία του BoeBot χρησιμοποιείται για να θέσει τιμές για διάφορες μεταβλητές που καθορίζονται στην κατηγορία του DifferentialDriveEntity. Παραδείγματος χάριν, η μάζα θεωρείτε ότι είναι 0,454, και αντιπροσωπεύει τη μάζα σε χιλιόγραμμα. Επιπλέον, το σασί του BoeBot καθορίζεται σύμφωνα με το πλάτος το μήκος και το ύψος. Αυτές οι μετρήσεις λήφθηκαν με το ζύγισμα του πραγματικού ρομπότ και τη μέτρησή του χρησιμοποιώντας μία μεζούρα.

Η θέση του BoeBot καθορίζεται μέσω ενός ζεύγους συντεταγμένων που εισάγονται όταν δημιουργείται η οντότητα. Αυτές οι συντεταγμένες αντιπροσωπεύουν τα σημεία στους άξονες X, Y, και Z. Η μηχανή προσομοίωσης MSRS χρησιμοποιεί ένα δεξιόστροφο σύστημα συντεταγμένων, το οποίο επηρεάζει την κατεύθυνση του άξονα Z.

Ο κατασκευαστής BoeBot καθορίζει επίσης τη θέση του σασί και των τροχών μέσα στην οντότητα. Η κατηγορία DifferentialDriveSystem υποθέτει ότι το ρομπότ θα έχει δύο κύριους τροχούς και ένα πίσω τροχό ο οποίος θα χρησιμοποιείται συνήθως για εξισορρόπηση. Η δύναμη θα οριστεί στις αριστερές και δεξιές μηχανές, οι οποίες ελέγχουν τους κύριους τροχούς. Η διαφορά μεταξύ των επιπέδων δύναμης που ορίζονται σε κάθε τροχό καθορίζει εάν θα κινηθεί μπροστά, πίσω, αριστερά, ή δεξιά. Αυτή είναι η ίδια μέθοδος που χρησιμοποιείται για να κινηθεί το φυσικό ρομπότ.

Αυτό που καθιστά την προσομοίωση τόσο ελκυστική είναι ότι, θεωρητικά, δεν πειράζει εάν το ρομπότ είναι εικονικό ή πραγματικό- ο κώδικας που χρησιμοποιείται για να τροφοδοτήσει το ρομπότ και να λάβει τα στοιχεία από τους αισθητήρες θα είναι ο ίδιος. Ένα μέρος του κώδικα που χρησιμοποιείται στο πρόγραμμα προσομοίωσης μπορεί να επαναχρησιμοποιηθεί κατά την εργασία με το πραγματικό ρομπότ. Μέχρι τώρα αναφέρονται πράγματα σε θεωρητικό επίπεδο κι αυτό επειδή η προσομοίωση δεν μπορεί εντελώς να μιμηθεί ένα πραγματικό περιβάλλον. Η προσομοίωση δεν μπορεί να λογαριάσει τους θορύβους - πράγματα που δεν αναμένονται, όπως τα εμπόδια που είναι στη λανθασμένη θέση.

Αυτό που η προσομοίωση μπορεί να κάνει είναι να δώσει μια αρκετά ρεαλιστική ευκαιρία για πειραματισμό με ένα νέο ρομπότ ή για προσομοίωση της αλληλεπίδρασης πολλαπλών ρομπότ. Αυτό μπορεί να είναι πολύ χρήσιμο σε ακαδημαϊκό επίπεδο όπου οι πόροι είναι περιορισμένοι και ο αριθμός των σπουδαστών είναι υψηλός.

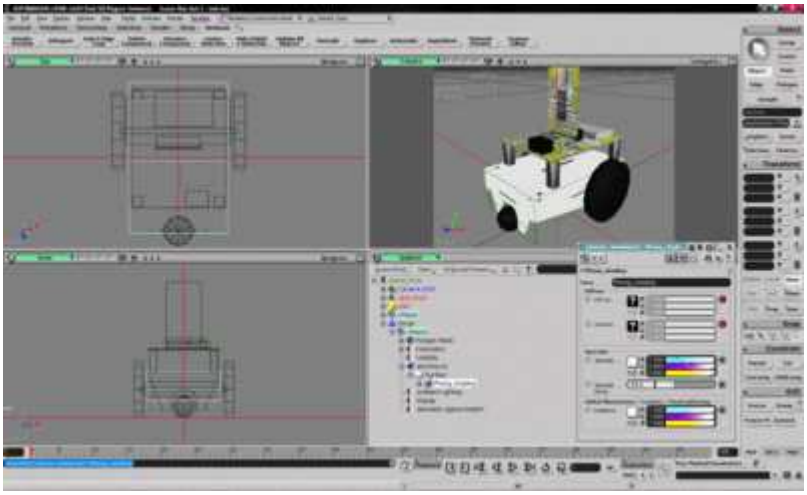
6.4 Δημιουργία αρχείου Mesh

Κάθε οντότητα μπορεί να συνδεθεί με ένα πλέγμα, το οποίο είναι αυτό που κάνει την οντότητα να εμφανιστεί ρεαλιστική. Παραδείγματος χάριν, στην περίπτωση της σφαίρας, το πλέγμα είναι αυτό που κάνει την οντότητα της σφαίρας να εμφανιστεί όπως τον πλανήτη γη. Δεν είναι απαραίτητο να συνδεθεί μια οντότητα με ένα πλέγμα, αλλά στην περίπτωση των σύνθετων οντοτήτων, όπως τα ρομπότ, ένα αντικείμενο πλέγματος είναι προτιμητέο.

Οποιοδήποτε τρισδιάστατο γραφικό εργαλείο έκδοσης μπορεί να χρησιμοποιηθεί για να δημιουργήσει το πλέγμα. Ο Steve Grand, δημιούργησε ένα πλέγμα Boe-Bot χρησιμοποιώντας ένα τρισδιάστατο πακέτο αποκαλούμενο ως SoftImage.

Το SoftImage είναι σε θέση να εξάγει την εικόνα στη μορφή .obj. Δεν μπορούν όλα τα πακέτα να το κάνουν αυτό. Αυτό πρέπει να ληφθεί υπόψη κατά την επιλογή ενός πακέτου γραφικών για να δημιουργηθεί το αρχείο πλέγματος.

Το πλέγμα Boe-Bot δημιουργήθηκε με την ένωση πολλαπλών πλεγμάτων σε πολυγωνικό σχήμα. Παραδείγματος χάριν, τα μεταλλικά πλαίσια άρχισαν ως μορφή κύβων που τροποποιήθηκαν για να αντιπροσωπεύσουν το μέγεθος και τη μορφή του Boe-Bot. Πρόσθετες μορφές κυλίνδρων προστέθηκαν έπειτα για να αντιπροσωπεύσουν τους τροχούς Μια ιεραρχία καθιερώθηκε για να δεσμεύσει τους τροχούς στα πλαίσια και να επιτρέψει σε ολόκληρο το αντικείμενο να λειτουργήσει ως ενιαία μορφή. Το αποτέλεσμα (όπως φαίνεται **στην Εικόνα6.4.1**) είναι μια εικόνα που αντιπροσωπεύει το φυσικό Boe-Bot.



Εικόνα6.4.1: Χρησιμοποίηση του Softlmageγια την δημιουργία του Boe-Bot Mesh File

Μετατροπή του αρχείου Mesh σε δυαδική μορφή

Το MSRS προσφέρει ένα εργαλείο γραμμής εντολών που είναι σε θέση να μετατρέπει ένα αρχείο .obj σε ένα βελτιστοποιημένο δυαδικό αρχείο με μια επέκταση .bos. Το όφελος αυτού του τύπου αρχείου είναι ότι φορτώνει γρηγορότερα από το αρχείο .obj. Το εργαλείο γραμμής εντολών, αποκαλούμενο ως Obj2bos.exe, μπορεί να προσεγγιστεί χρησιμοποιώντας την εντολή prompt από το αντίστοιχο μενού που περιλαμβάνεται στην εγκατάσταση του MSRS.

Όταν η δουλειά γίνεται σε ένα πλέγμα είναι σημαντικό να αναφερθεί ότι τα υλικά χρησιμοποιούνται για να χρωματίσουν τα φυσικά αντικείμενα και κάθε οντότητα μπορεί να συνδεθεί με ένα ή περισσότερα υλικά. Επομένως, όταν δημιουργείται το .obj αρχείο, θα εμφανιστούν πιθανότατα πρόσθετα αρχεία υλικών με τις επεκτάσεις .mtl. Μπορεί επίσης να συμπεριληφθούν αρχεία εικόνας που χρησιμοποιούνται για να δημιουργήσουν textures. Αυτά τα αρχεία πρέπει να αντιγραφούν στην ίδια θέση όπως το αρχείο .obj πριν χρησιμοποιηθεί το εργαλείο μετατροπής πλέγματος.

Το αρχείο.obj και όλα τα σχετικά αρχεία πρέπει να τοποθετηθούν στο κατάλογο /store/media που συνδέονται με την τοπική εγκατάσταση του MSRS. Τα αρχεία που συνδέονται με το πλέγμα Boe-Bot, και περιέχονται στον κώδικα, είναι: Aluminum.png, Boe- Bot.mtl, Boe- Bot.obj, BoebotWheel.png, και Pcb.png. Αφού τοποθετηθούν στον κατάλογο, πρέπει να εφαρμοστεί η εντολή prompt του MSRS και να δακτυλογραφηθούν τα εξής:

```
Obj2bos.exe /i:"store\media\Boe-bot.obj"
```

Το εργαλείο μετατροπής θα δημιουργήσει ένα αρχείο με όνομα Boe-bot.bos, και θα τοποθετηθεί μέσα στον κατάλογο.

Δημιουργία μιας υπηρεσίας προσομοίωσης

Στην ενότητα αυτή θα παρουσιαστεί ο τρόπος με τον οποίο δημιουργείται μια υπηρεσία προσομοίωσης Boe-Bot.

Εάν το πρόγραμμα δημιουργείται από την αρχή, θα πρέπει να δημιουργηθεί πρώτα μια νέα DSS υπηρεσία χρησιμοποιώντας το οπτικό Visual Studio. Το όνομα που μπορεί να δοθεί στο πρόγραμμα είναι SimulatedBoeBot. Στη συνέχεια θα πρέπει να προστεθεί μια αναφορά στο Microsoft.Xna.Framework και οι ακόλουθες αναφορές namespace στην κορυφή του αρχείου κατηγορίας SimulatedBoeBot.cs:

```
using xna = Microsoft.Xna.Framework;
using xnagrfx = Microsoft.Xna.Framework.Graphics;
using xnaprof = Microsoft.Robotics.Simulation.MeshLoader;
```

Το Xna framework χρησιμοποιείται από το MSRS για να δώσει τις οντότητες. Οι αναφορές Xna θα χρησιμοποιηθούν στον κώδικα που καθορίζει τον τύπο οντοτήτων Boe-Bot. Όλοι οι τύποι οντοτήτων που περιλαμβάνονται στο MSRS περιέχονται και στο Microsoft.Robotics.Simulation.Engine namespace.

Populating the Simulation Scene

Ένας από τους αρχικούς στόχους για κάθε υπηρεσία προσομοίωσης είναι να παρεμβληθούν οι οντότητες στην εικόνα προσομοίωσης. Αυτό πρέπει να γίνει όταν προστίθενται οι ενάρξεις υπηρεσιών, για αυτό και μια κλήση σε μια μέθοδο που ονομάζεται PopulateWorld προστίθενται στη μέθοδο έναρξης. Για την προσομοίωση του Boe-Bot, η μέθοδος PopulateWorld θα μοιάσει με αυτό:

```
private void PopulateWorld()
{
    AddSky();
    AddGround();
    AddBoeBot(new Vector3(1, 0, 1));
    AddBox(new Vector3(2, 1, 1));
}
```

Οι μέθοδοι AddSky και AddGround χρησιμοποιούνται για να οργανώσουν το περιβάλλον όπου το BoeBot θα ανήκει. Για τη μέθοδο AddSky, ένας νέος τύπος SkyDomeEntity δημιουργείται και παρεμβάλλεται στην προσομοίωση. Ένα κατευθυντικό φως, που προορίζεται να αντιπροσωπεύσει τον ήλιο, παρεμβάλλεται επίσης στην προσομοίωση (ο κώδικας για αυτήν την μέθοδο παρουσιάζεται **στην Εικόνα6.4.2**).

Εικόνα6.4.2 :Προσθήκη ουρανού (Add Sky)

```
void AddSky()  {
   // Προσθέστε έναν ουρανό με ένα στατικό texture. Θα χρησιμοποιήσουμε το texture του
ουρανού
   // να κάνει φωτισμό ανά pixel σε κάθε προσομοίωση οπτικής οντότητας
   SkyDomeEntity sky = new SkyDomeEntity("skydome.dds",
                                 "sky_diff.dds");
   SimulationEngine.GlobalInstancePort.Insert(sky);

   // Προσθέστε κατεύθυνόμενο φως για την προσομοίωση του ήλιου.
   LightSourceEntity sun = new LightSourceEntity();
   sun.State.Name = "Sun";
   sun.Type = LightSourceEntityType.Directional;
   sun.Color = new Vector4(0.8f, 0.8f, 0.8f, 1);
   sun.Direction = new Vector3(0.5f, -.75f, 0.5f);
   SimulationEngine.GlobalInstancePort.Insert(sun);
}
```

Η μέθοδος AddGround χρησιμοποιεί τον τύπο HeightFieldEntity για να δημιουργήσει ένα μεγάλο οριζόντιο επίπεδο σε μηδέν ανύψωση. Η εικόνα της βασικής δομής, που παρέχεται με την εγκατάσταση του MSRS, χρησιμοποιείται για να αντιπροσωπεύσει το έδαφος. Ο κώδικας για αυτήν την μέθοδο φαίνεται στη συνέχεια:

```
void AddGround()
{
   // δημιουργεί ένα μεγάλο οριζόντιο επίπεδο, με μηδενικό υψόμετρο.
   HeightFieldEntity ground = new HeightFieldEntity(
      "simple ground", // όνομα
      "03RamieSc.dds", // υφή της εικόνας
      new MaterialProperties("ground",
         0.2f, // ανταπόδοση, επιστροφή (όποιο ταιριάζει)
         0.5f, // δυναμική τριβή
         0.5f) // στατική τριβή
  );
   SimulationEngine.GlobalInstancePort.Insert(ground);
}
```

Το επόμενο πράγμα που προστίθεται είναι η οντότητα Boe-Bot . Η μέθοδος AddBoeBot δέχεται μια εισερχόμενη παράμετρο Vector3 που χρησιμοποιείται για να αντιπροσωπεύσει τη θέση του ρομπότ. Αυτήν η θέση θα περαστεί από τον κατασκευαστή οντοτήτων BoeBot, η οποία δημιουργήθηκε νωρίτερα. Η μέθοδος AddBoeBot θα αρχίσει επίσης τη μιμούμενη υπηρεσία κίνησης Boe-Bot ως συνεργάτη οντοτήτων. Ο κώδικας για αυτήν την μέθοδο παρουσιάζεται εδώ:

```
void AddBoeBot(Vector3 pos)
{
    BoeBot boeBot = new BoeBot(pos);
    boeBot.State.Name = "SimulatedBoeBot";
```

```
    // ξεκινά τη προσομοιωμένη υπηρεσία οδήγησης του Boe-Bot Drive
    CreateService(
      drive.Contract.Identifier,
        Microsoft.Robotics.Simulation.Partners.CreateEntityPartner(
         "http://localhost/" + boeBot.State.Name));

    SimulationEngine.GlobalInstancePort.Insert(boeBot);
}
```

Η τελευταία οντότητα που προστίθεται είναι ένα στερεό κιβώτιο που χρησιμοποιείται για να αντιπροσωπεύσει ένα εμπόδιο. Η μέθοδος AddBox θα δημιουργήσει μια οντότητα σε σχήμα κουτιού από τον τύπο SingleShapeEntity. Στην **Εικόνα6.4.3** φαίνεται ο κώδικας για αυτήν την μέθοδο.

Εικόνα6.4.3 : Δημιουργία οντότητας σε σχήμα κουτιού

```
void AddBox(Vector3 position)
{
   Vector3 dimensions =
      new Vector3(0.2f, 0.2f, 0.2f); //μέτρα

   // δημιουργία απλής κινούμενης οντότητας, με ένα ξεχωριστό σχήμα
   SingleShapeEntity box = new SingleShapeEntity(
      new BoxShape(
         new BoxShapeProperties(
           100, // μάζα σε kilograms.
           new Pose(), // σχετική παρουσίαση
           dimensions)), // διαστάσεις
           position);

   box.State.MassDensity.Mass = 0;
   box.State.MassDensity.Density = 0;

   //ονομάζει την οντότητα. Όλες οι οντότητες πρέπει να έχουν μοναδικά ονόματα
   box.State.Name = "box";

   //προσθήκη οντότητας στη προσομοίωση
   SimulationEngine.GlobalInstancePort.Insert(box);
}
```

6.5 Δημιουργία αρχείου Manifest

Οι περισσότερες υπηρεσίες συνεργάζονται με άλλες υπηρεσίες για να εισάγουν δεδομένα και συναρτήσεις που απαιτούνται από την υπηρεσία. Αυτές οι υπηρεσίες που συνεργάζονται αναφέρονται στην κορυφή της κατηγορίας που υλοποιείται χρησιμοποιώντας τις ιδιότητες των συνεργατών. Παραδείγματος χάριν, η υπηρεσία προσομοίωσης Boe-Bot θα συνεργαστεί με την υπηρεσία μηχανών προσομοίωσης. Ο κώδικας για να συμπεριλάβει αυτήν την δήλωση συνεργατών πρέπει να έχει την ακόλουθη μορφή:

```
//θύρα της μηχανής προσομοίωσης
```

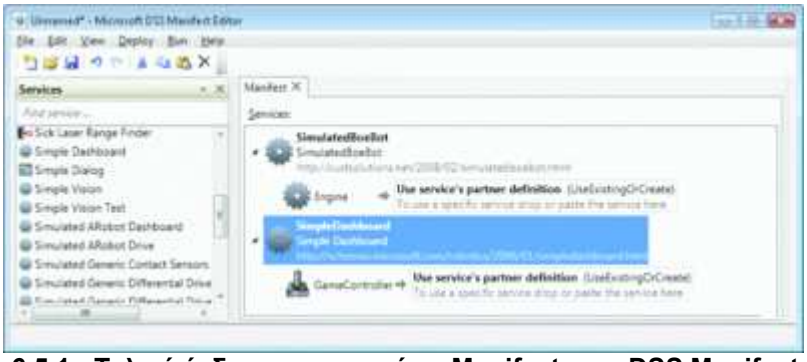
```
[Partner("Engine",
    Contract = engineproxy.Contract.Identifier,
    CreationPolicy = PartnerCreationPolicy.UseExistingOrCreate)]
private engineproxy.SimulationEnginePort _engineStub =
    new engineproxy.SimulationEnginePort();
```

Οι DSS υπηρεσίες χρησιμοποιούν ένα XML αρχείο γνωστό ως manifest για να απαριθμήσει τις υπηρεσίες που συνεργάζονται και συνδέονται με μια βασική υπηρεσία. Αυτό λέει στο χρόνο εκτέλεσης του MSRS πώς η βασική υπηρεσία πρέπει να φορτώσει και να αλληλεπιδράσει με αυτές τις υπηρεσίες. Ακόμα κι αν το manifest είναι ένα XML αρχείο και μπορεί να εκδοθεί με έναν συντάκτη κειμένου, είναι καλύτερο να χρησιμοποιηθεί ο DSS Manifest συντάκτης που συμπεριλαμβάνεται στο MSRS. Ο DSS Manifest συντάκτης είναι ένα οπτικό εργαλείο που επιτρέπει με σύρσιμο να μεταφερθούν υπηρεσίες που συνεργάζονται πάνω σε μια επιφάνεια σχεδίου. Πριν τη χρήση του DSS Manifest συντάκτη, θα πρέπει να συνταχθεί επιτυχώς η βασική υπηρεσία. Αυτό θα δημιουργήσει τα συναρμολογημένα αρχεία που χρησιμοποιούνται από το DSS Manifest συντάκτη.

Μόλις φορτωθεί ο DSS Manifest συντάκτης, ένας κατάλογος με όλες τις διαθέσιμες υπηρεσίες θα εμφανιστεί στο αριστερό μέρος της οθόνης. Αρχικά θα πρέπει να τοποθετηθεί η βασική υπηρεσία στον κατάλογο των διαθέσιμων υπηρεσιών και με σύρσιμο να μεταφερθεί ένα παράδειγμα στην επιφάνεια σχεδίου. Μετά από αυτή την ενέργεια για την υπηρεσία προσομοίωσης Boe-Bot, η επιφάνεια του σχεδίου θα συμπεριλάβει έναν κόμβο για την υπηρεσία SimulatedBoeBot και έναν υπό- κόμβο για την υπηρεσία μηχανών προσομοίωσης.

Η εγκατάσταση του MSRS περιλαμβάνει διάφορες υπηρεσίες στο φάκελο δειγμάτων. Μια τέτοια υπηρεσία, το SimpleDashboard, μπορεί να χρησιμοποιηθεί ως επιτροπή ελέγχου για τα μιμούμενα ρομπότ. Το SimpleDashboard επιτρέπει το ξεκίνημα κι άλλων υπηρεσιών οι οποίες χρησιμοποιούνται για να ελέγξουν τη συγκεκριμένη λειτουργία για το ρομπότ. Παραδείγματος χάριν, ο κώδικας στην υπηρεσία SimulatedBoeBot απλά εμφανίζεται στην εικόνα προσομοίωσης με τις οντότητες, μια από τις οποίες είναι το Boe-Bot. Δεν υπάρχει κανένας κώδικας στην υπηρεσία που να κινεί πραγματικά το ρομπότ. Ο κώδικας αυτός λαμβάνεται από την υπηρεσία Drive.

Προκειμένου να φορτωθεί και να χρησιμοποιηθεί το απλό ταμπλό, πρέπει να προστεθεί στο manifest τοποθετώντας με ένα παράδειγμα του SimpleDashboard στον κατάλογο υπηρεσιών και με σύρσιμο στο κάτω μέρος της επιφάνειας σχεδίου. Το τελικό αποτέλεσμα πρέπει να έχει τη μορφή που φαίνεται στην **Εικόνα6.5.1**. Η τελευταία ενέργεια που πρέπει να γίνει είναι να σωθεί το manifest στον κατάλογο προγράμματος της υπηρεσίας SimulatedBoeBot και να γραφτεί πάνω στην επιφάνεια το manifest που ονομάζεται SimulatedBoeBot.manifest.xml.

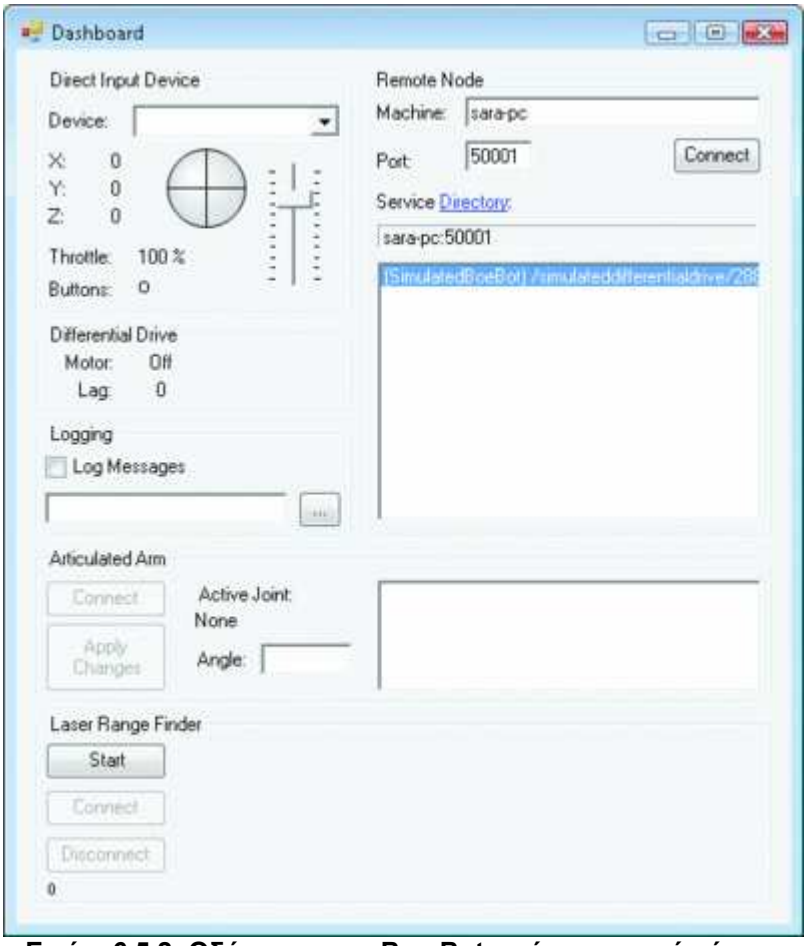


Εικόνα6.5.1 : Τελική έκδοση του αρχείου Manifest στο DSS Manifest Editor

Εκτέλεση της προσομοίωσης

Για να τρέξει η υπηρεσία SimulatedBoeBot πρέπει να γίνει κλικ στο Debug and Run από το Visual Studio μενού. Πρώτα, ένα command prompt παράθυρο πρέπει να εμφανιστεί.

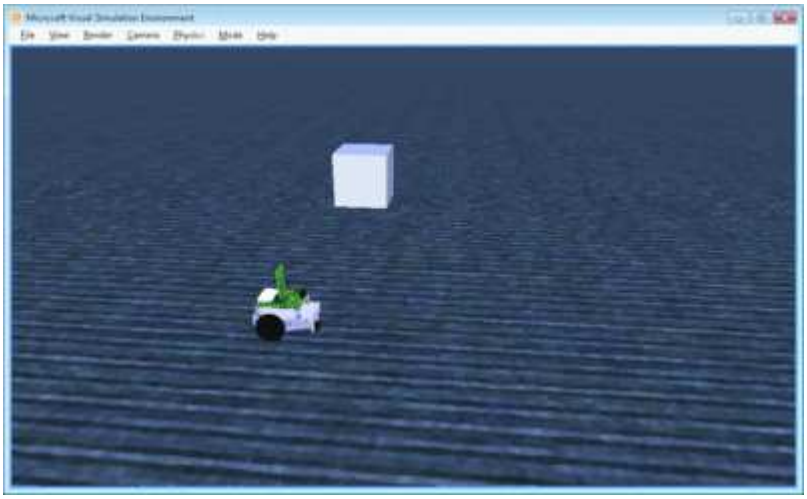
Μόλις φορτώσει η υπηρεσία, δύο πρόσθετα παράθυρα πρέπει να εμφανιστούν. Η διεπαφή για το απλό ταμπλό είναι μια μορφή παραθύρων που περιλαμβάνει διάφορες ομάδες κιβώτιων. Η δεξιά πλευρά της εικόνας (**Εικόνα6.5.2**) περιλαμβάνει μια ομάδα κιβωτίων με τον τίτλο Remote Node. Αρχικά πρέπει να εισαχθεί το όνομα της τοπικής μηχανής (localhost), στη μηχανή textbox και έπειτα να πατηθεί κλικ στο Connect. Αυτό θα αρχίσει μια σύνδεση με την υπηρεσία SimulatedBoeBot και θα φορτώσει τις συνεταιρικές υπηρεσίες οντοτήτων, όπως η υπηρεσία κίνησης που παρουσιάζεται στον κατάλογο listbox.



Εικόνα6.5.2: Οδήγηση του Boe-Bot σε ένα εικονικό κόσμο.

Η υπηρεσία κίνησης χρησιμοποιείται για να στείλει τη δύναμη στους τροχούς που ελέγχουν το Boe-Bot. Αυτό επιτρέπει την κίνηση του Boe-Bot γύρω από την προσομοίωση χρησιμοποιώντας είτε τον εικονικό έλεγχο πηδαλίων είτε ένα πραγματικό πηδάλιο που συνδέεται με τον υπολογιστή σας. Για να ξεκινήσει η διαδικασία θα πρέπει να πατηθεί διπλό κλικ στο στοιχείο SimulatedBoeBot που εμφανίζεται στο listbox. Στη συνέχεια το στρογγυλό εικονικό έλεγχο των πηδαλίων που βρίσκεται στην κορυφαία αριστερή γωνία της φόρμας Simple Dashboard σέρνεται με το ποντίκι.

Θα χρειαστούν αρχικά κάποιες δοκιμές για εξοικείωση με τον εικονικό έλεγχο πηδαλίων (**Εικόνα6.5.3** φαίνεται πώς πρέπει να μοιάζει. η προσομοίωση). Για εξάσκηση πρέπει να κινηθεί το Boe-Bot γύρω και στην οθόνη θα εμφανίζεται το πώς αντιδρά όταν συγκρούεται με το εμπόδιο κιβωτίων. Από το edit mode, δίνεται η δυνατότητα επιλογής μίας από τις οντότητες, και η ρύθμιση μερικών ιδιοτήτων ώστε να γίνει αντιληπτό το πώς επηρεάζεται η οντότητα.



Εικόνα6.5.3 :Λειτουργία ενός Simulated Boe-Bot σε ένα εικονικό κόσμο.

6.6 Δημιουργία οντότητας χωρίς διαφορική κίνηση

Σε αυτή την ενότητα περιγράφεται το πώς δημιουργήθηκε μια νέα οντότητα ρομπότ που ονομάστηκε Boe-Bot. Επειδή το Boe-Bot χρησιμοποιεί ένα διαφορικό σύστημα κίνησης για να ελέγξει τους τροχούς του ρομπότ, χρησιμοποιείται ένας τύπος οντοτήτων που προήλθε από την κατηγορία DifferentialDriveEntity. Έτσι επαναχρησιμοποιείται ο κώδικας που προήλθε από το MSRS για τον έλεγχο του ρομπότ το οποίο χρησιμοποιεί αυτόν τον τύπο συστήματος κίνησης.

Για την προσομοίωση ενός ρομπότ το οποίο χρησιμοποίει ένα διαφορετικό σύστημα κίνησης, αρχικά πρέπει να προστεθεί μια κατηγορία η οποία θα αντιπροσωπεύει το σύστημα κίνησης. Παραδείγματος χάριν, μερικά ρομπότ χρησιμοποιούν αυτό που είναι γνωστό ως τρίκυκλο σύστημα κίνησης. Σε αυτήν την περίπτωση, ένας ενιαίος μπροστινός τροχός χρησιμοποιείται για να κινήσει το ρομπότ, και δύο δευτερεύοντες τροχοί που συνδέονται με μια χωριστή μηχανή χρησιμοποιούνται για να οδηγήσουν το ρομπότ σε καθεμία κατεύθυνση.

Για την αποδοχή αυτού του τύπου συστήματος είτε στον εικονικό είτε στο φυσικό κόσμο, θα πρέπει να δημιουργηθεί μια κατηγορία η οποία θα αποτελεί τον τύπο για αυτό το σύστημα κίνησης.

Κεφάλαιο 7

Κατασκευή ενός κινούμενου ρομπότ με διαφορική κίνηση από απλά υλικά

7.1 Εισαγωγή

Ένα από τα δυσκολότερα προβλήματα σχετικά με τα κινούμενα ρομπότ είναι η κατασκευή τους. Στη συνέχεια ακολουθεί μια περιγραφή για το πώς ένα τέτοιο κινούμενο ρομπότ κίνησης μπορεί να κατασκευαστεί από συνηθισμένα διαθέσιμα υλικά.

7.1.1Θεωρία της διαφορικής κίνησης

Μια έννοια του διαφορικού είναι απλά η κίνηση σε διαφορετικά επίπεδα. Για παράδειγμα, η ακόλουθη **Εικόνα7.1.1.1** δείχνει ένα πολύ απλό όχημα με διαφορική κίνηση.

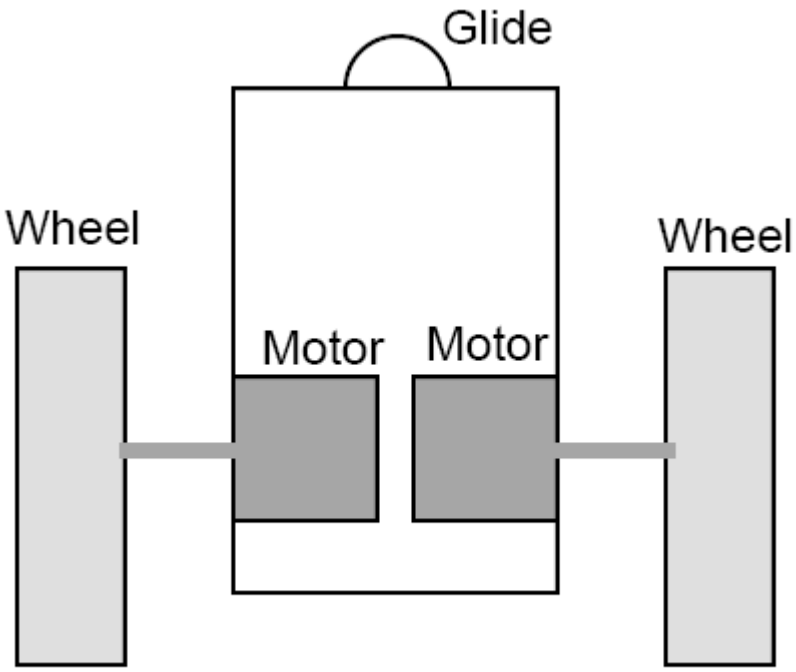


Figure 1 Simple Differential Drive

Εικόνα7.1.1.1 :Όχημα με διαφορική κίνηση.

Τροχός
Μηχανή
Ροδέλα κύλισης
Μηχανή
Τροχός

Κάθε τροχός συνδέεται με τη μηχανή της. Μια ροδέλα κύλισης παρέχει ισορροπία από τη μία πλευρά του οχήματος. Εάν και οι δύο μηχανές περιστρέφονται στον ίδιο ρυθμό και όλα τα συστατικά είναι ίδια τότε το όχημα θα κινηθεί σε μια ευθεία γραμμή. Εάν ο ένας τροχός περιστρέφεται γρηγορότερα από τον άλλο, το όχημα θα έχει μια τάση να γυρίσει προς την αντίθετη κατεύθυνση από αυτή του γρήγορου τροχού. Εάν κάθε τροχός περιστρέφεται σε αντίθετη κατεύθυνση από τον άλλο, το όχημα θα γυρίσει επιτόπου. Αυτή η μορφή οδήγησης βρίσκεται συνήθως στα οχήματα όπως τα τρακτέρ και τα τανκ. Η κίνηση έχει το πλεονέκτημα της απλότητας

7.2 Δημιουργία ενός οχήματος διαφορικής κίνησης

Μια απλή διαφορική κίνηση μπορεί να κατασκευαστεί με διάφορους τρόπους. Ένας οικονομικός αποδοτικός τρόπος είναι να χρησιμοποιήσεις απλά υλικά που μπορείς να τα βρεις οπουδήποτε.

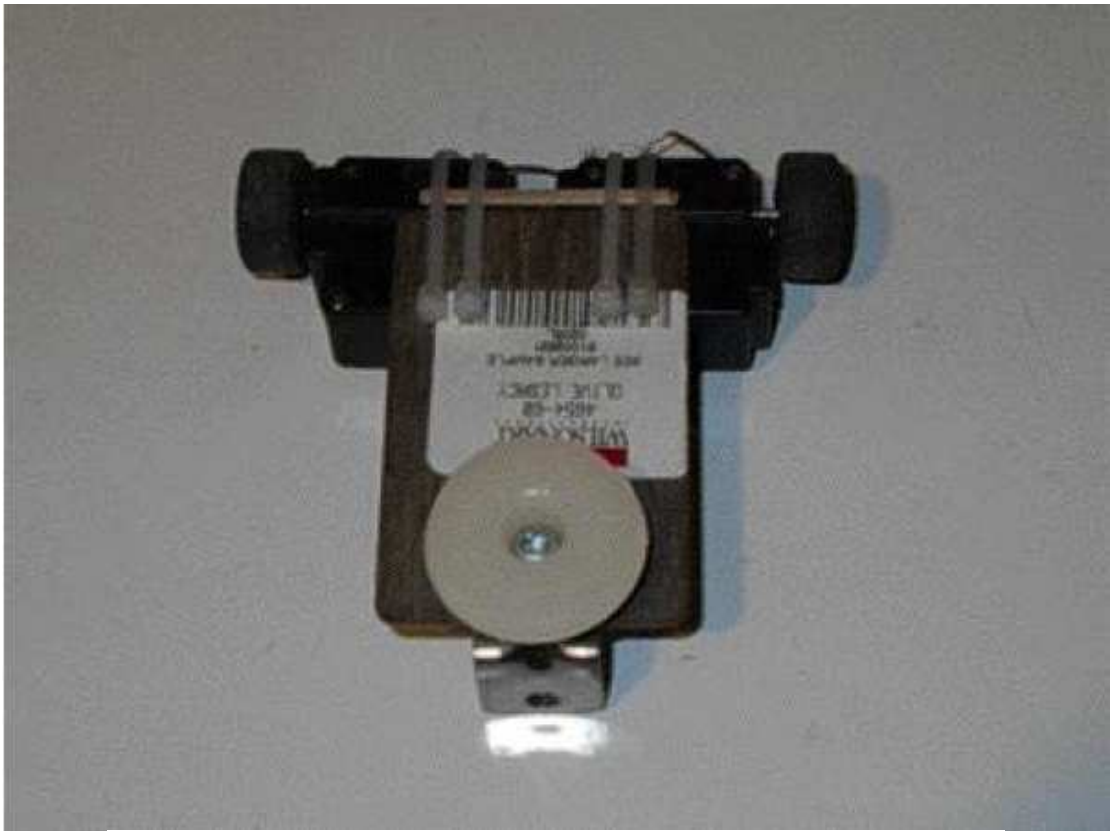
Εικόνα 7.2.1: Υλικά κατασκευής ρομπότ διαφορικής κίνησης.

Για να κατασκευαστεί ένα όχημα σαν αυτό που παρουσιάζεται στην **Εικόνα7.2.1** χρειάζονται διάφορα υλικά όπως:

1. Ένα παλαιό ραδιόφωνο ή ένα τηλεκατευθυνόμενο αυτοκίνητο
2. Ένα προστατευτικό πατάκι που χρησιμοποιούνται για τα πόδια της καρέκλας
3. Διάφορες μικρές βίδες
4. Δύο γωνιές νούμερο 1
5. Μερικά συρματάκια που χρησιμοποιούνται για σφίξιμο καλωδίων
6. Μία θήκη για μπαταρίες μεγέθους AA
7. Ένα ξυλάκι από σουβλάκι καλαμάκι
8. Μία ταινία Hook-and-loop tape (Velcro)
9. Ένα κομμάτι διακλαδωτή ανόδου ή Formica

The Drive

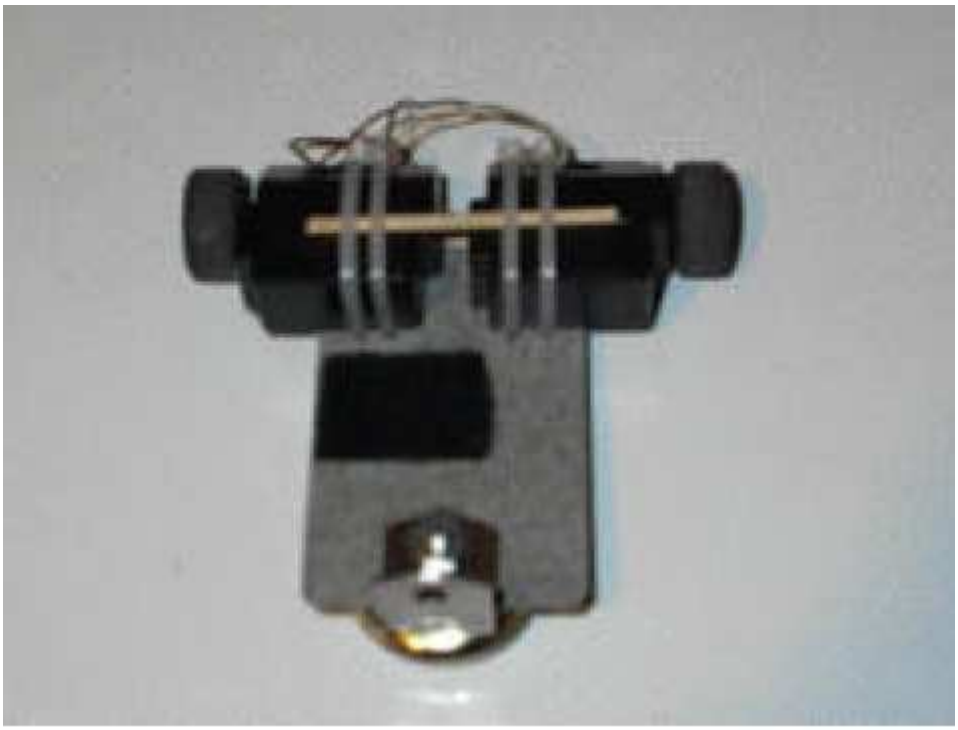
Αρχικά βγαίνουν οι οδηγοί από το τηλεκατευθυνόμενο αυτοκίνητο. Αυτά τα αυτοκίνητα έχουν μηχανές πολύ χαμηλής τάσης (3-9 συνεχές ρεύμα βολτ) μεταδίδοντας την παρεχόμενη ενέργεια από τα γρανάζια στους τροχούς. Αυτή η σχέση μπορεί να αλλάξει με την κατάλληλη ρύθμιση κατάλληλων βιδών. Ο στόχος είναι να αναιρεθούν η μηχανή και τα γρανάζια ώστε να μην είναι σε επαφή. Θα χρειαστούν δύο μηχανές και τα αντίστοιχα ζευγάρια γραναζιών. Τα καλύτερα αποτελέσματα επιτυγχάνονται όταν οι οδηγοί είναι παρόμοιοι ή ίδιοι κι από τις δύο πλευρές του οχήματος.



Εικόνα 7.2.2: Αναίρεση υλικών από το τηλεκατευθυνόμενο αυτοκίνητο.

The Vehicle Body

Ανοίγοντας τρύπες στο Formica με ένα μικρό τρυπάνι μπροστά από το σημείο όπου οι οδηγοί πρόκειται να τοποθετηθούν δημιουργούνται υποδοχές για τα καλώδια που θα κρατήσουν τις μηχανές στη σωστή θέση. Η παραπάνω **Εικόνα7.2.2** δείχνει την κάτοψη του οχήματος. Παρατηρείτε το τρόπο που τα καλώδια κρατούν τις μηχανές στη σωστή θέση. Το ξυλάκι από το καλαμάκι σπάει και τοποθετείται κάτω από τους σφικτήρες για να τους στερεώσει και να κρατήσει τις μηχανές ακίνητες. Όπως φαίνεται και στην **Εικόνα7.2.3** η ροδέλα κύλισης η οποία κατασκευάστηκε από το πατάκι για την κάλυψη των ποδιών της καρέκλας,



Εικόνα 7.2.3: Προσθέτουμε ένα καλαμάκι για καλύτερη στερέωση των μηχανών .

Η παραπάνω **Εικόνα7.2.3** δείχνει την πάνω όψη του οχήματος. Όπως και στην κάτοψη ένα ξυλάκι τοποθετείται κάτω από τους σφικτήρες. Το μαύρο κομμάτι στη μέση του οχήματος είναι ένα μέρος της ταινίας (the hook and loop tape). Το άλλο μέρος τοποθετείται στο κατώτατο σημείο της θήκης μπαταριών. Ένα άγκιστρο κρατείται σταθερό με τη βίδα που κρατά τη ροδέλα κύλισης ακίνητη κάτω από το όχημα.

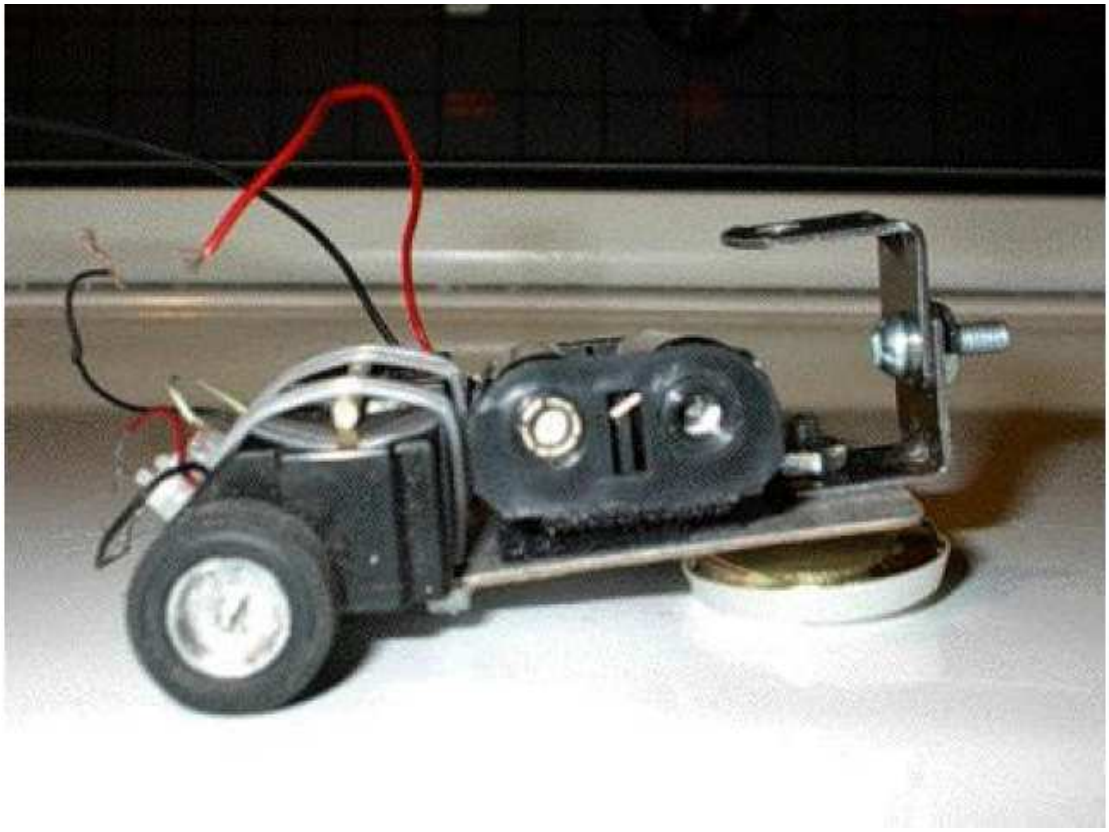
Παροχή Ισορροπίας

Μια πλάγια όψη του οχήματος παρουσιάζεται στη συνέχεια στην **Εικόνα7.2.4**. Παρατηρήστε πώς η ροδέλα κύλισης ισορροπεί τους τροχούς. Για καλύτερη απόδοση, το μεγαλύτερο μέρος του βάρους του οχήματος πρέπει να μεταφερθεί στους τροχούς κίνησης. Αυτό μπορεί να ολοκληρωθεί με την τοποθέτηση της θήκης μπαταριών πάνω ή κοντά στους πίσω τροχούς.

Εικόνα 7.2.4 :Παρουσία πλάγιας όψης του οχήματος.

Επιπρόσθετα στοιχεία οχήματος

Στην παρακάτω **Εικόνα7.2.5**, παρουσιάζεται ο τρόπος που η θήκη μπαταριών τοποθετείται πάνω στην ταινία (on the hook and loop tape). Για να επιτραπεί το μοντάρισμα των πρόσθετων αντικειμένων (που τοποθετούνται ενδεχομένως σε πρόσθετα δείγματα Formica), ένα δεύτερο υποστήριγμα είναι συνδεμένο με το πρώτο με τον τρόπο που παρουσιάζεται. Τα απομένοντα συστατικά θα σταθεροποιηθούν από το δεύτερο υποστήριγμα.



Εικόνα7.2.5 :Τοποθέτηση θήκης μπαταριών.

Σημείωση

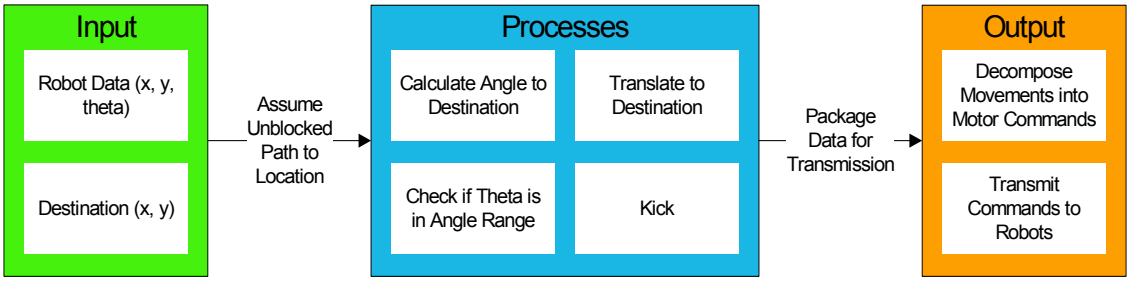
Το όχημα πρέπει ετοιμαστεί για την τελική καλωδίωση σε κάποια μορφή ελεγκτή. Αυτό δεν φαίνεται αλλά μπορεί να περιλαμβάνει κάτι τόσο απλό όσο ένα πηδάλιο ή κάτι πιο σύνθετο όπως ένας μικροελεγκτής ή υπολογιστής. Είναι σωστό να καθοριστεί πόσο ρεύμα θα τραβήξουν οι μηχανές από τις μπαταρίες πριν την απόπειρα να τις αγκιστρώσουμε σε κάποιο ηλεκτρονικό ελεγκτή αλλά, πέρα από αυτό το όχημα πρέπει να είναι έτοιμο να προχωρήσει.

7.3 Σχέδιο κίνησης Ρομπότ

Εισαγωγή

Ο σκοπός του προγράμματος ελέγχου του ρομπότ είναι να υπάρχει η ελευθερία να μετακινηθεί το ρομπότ σε οποιαδήποτε κατεύθυνση, και να περιστραφεί σε δεδομένες γωνίες. Επειδή το ρομπότ έχει παγκατευθυντικές ρόδες και έχει ένα σχεδόν συμμετρικό σκελετό στη διεύθυνση της ακτίνας, πρέπει να προγραμματιστούν κάποιες περιστροφικές ρυθμίσεις. Αντ' αυτού εφαρμόζεται μια δημιουργική μέθοδος των τεσσάρων ισοδύναμων κατευθύνσεων των τροχών με τις κατάλληλες ρυθμίσεις ενός μηχανισμού αντίστασης της κίνησης στο μπροστά τμήμα του ρομπότ.

Interface



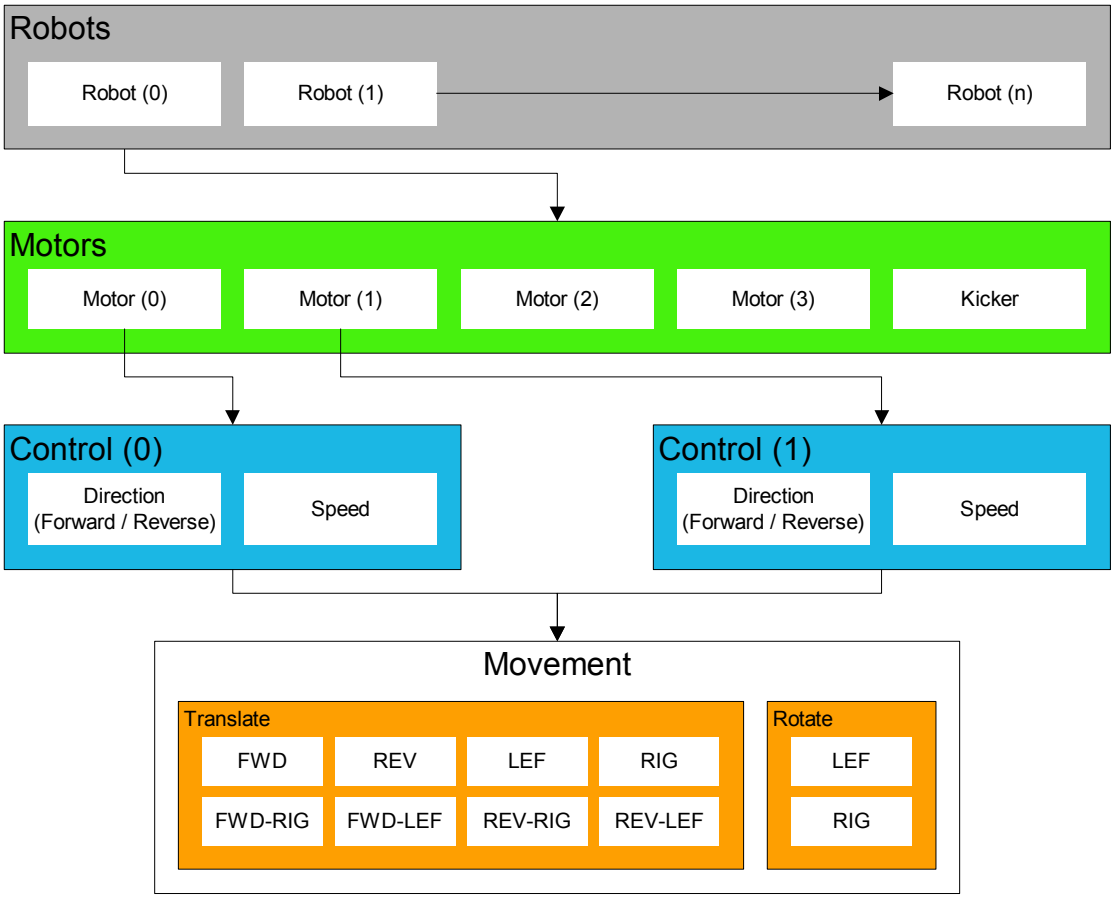
Εικόνα7.3.1 : Σχεδιάγραμμα κίνησης ρομπότ

Μετά από την επεξεργασία της **Εικόνας7.3.1**, μια θέση για το ρομπότ πρέπει να είναι διαθέσιμη στις συντεταγμένες x,y. Επιπλέον, χρησιμοποιώντας τις στοιχειώδεις λειτουργίες στρατηγικής, το κομμάτι της "συμπεριφοράς" του προγράμματος πρέπει να δώσει μια γενική ιδέα για το που πρέπει να επανατοποθετηθεί το ρομπότ. Χρησιμοποιώντας αυτές τις δύο παραμέτρους ως δεδομένα εισόδου, η μετακίνηση είναι μόνο μια πορεία "ευθειών γραμμών" στον προορισμό (που αγνοεί τους πολυάριθμους παράγοντες που θα συζητηθούν αργότερα). Οι βασικοί αλγόριθμοι μετακίνησης θεωρούν ότι η στρατηγική είναι υπεύθυνη για την αποφυγή σύγκρουσης και για τον προορισμό για κάθε ρομπότ.

Μετά από την αξιολόγηση του κύριου προγράμματος για τη μετακίνηση, οι οδηγίες πρέπει να μεταφραστούν στις απλές εντολές μηχανών και να σταλούν στα ρομπότ χρησιμοποιώντας κάποιο είδος κατάτμησης επικοινωνίας.

Αφαίρεση

Κάθε ρομπότ θεωρείται ότι είναι μέλος ενός πίνακα. Έχοντας αυτό υπόψη, κάθε ρομπότ έχει ένα πίνακα τεσσάρων μηχανών. Μια μεμονωμένη μηχανή μπορεί να αποσυντεθεί σε μια ενιαία δομή δεδομένων ευθεία και αντίστροφη (δυαδική), και είναι η συλλογή αυτών των καταστάσεων που πολλαπλασιάζονται στις 4 μηχανές δίνοντας στο ρομπότ περισσότερους βαθμούς ελευθερίας (αυτό φαίνεται στη παρακάτω **Εικόνα7.3.2**). Με το χειρισμό των στοιχείων στη δομή, οι πίνακες των μηχανών μπορούν να εξαχθούν ως συσκευασίες και ο μοναδικός συνδυασμός κατεύθυνσης και ταχύτητας για κάθε μηχανή μπορεί να υπαγορεύσει τις μετακινήσεις του ρομπότ. Επιπλέον, επειδή τα ρομπότ έχουν τις παγκατευθυντικές ρόδες μπορούν να μεταφράσουν συνολικά 8 κατευθύνσεις.



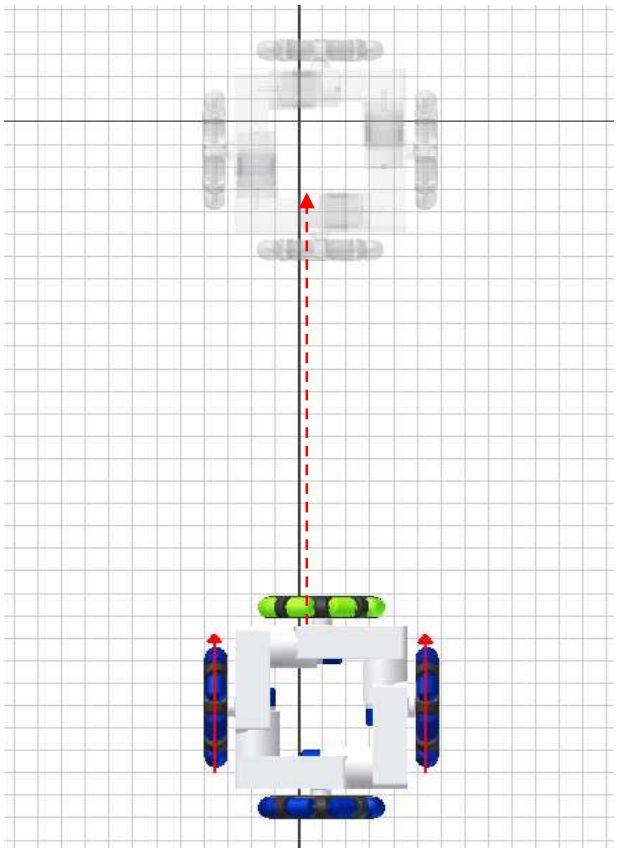
Εικόνα7.3.2: Πίνακας παρουσίασης μηχανών.

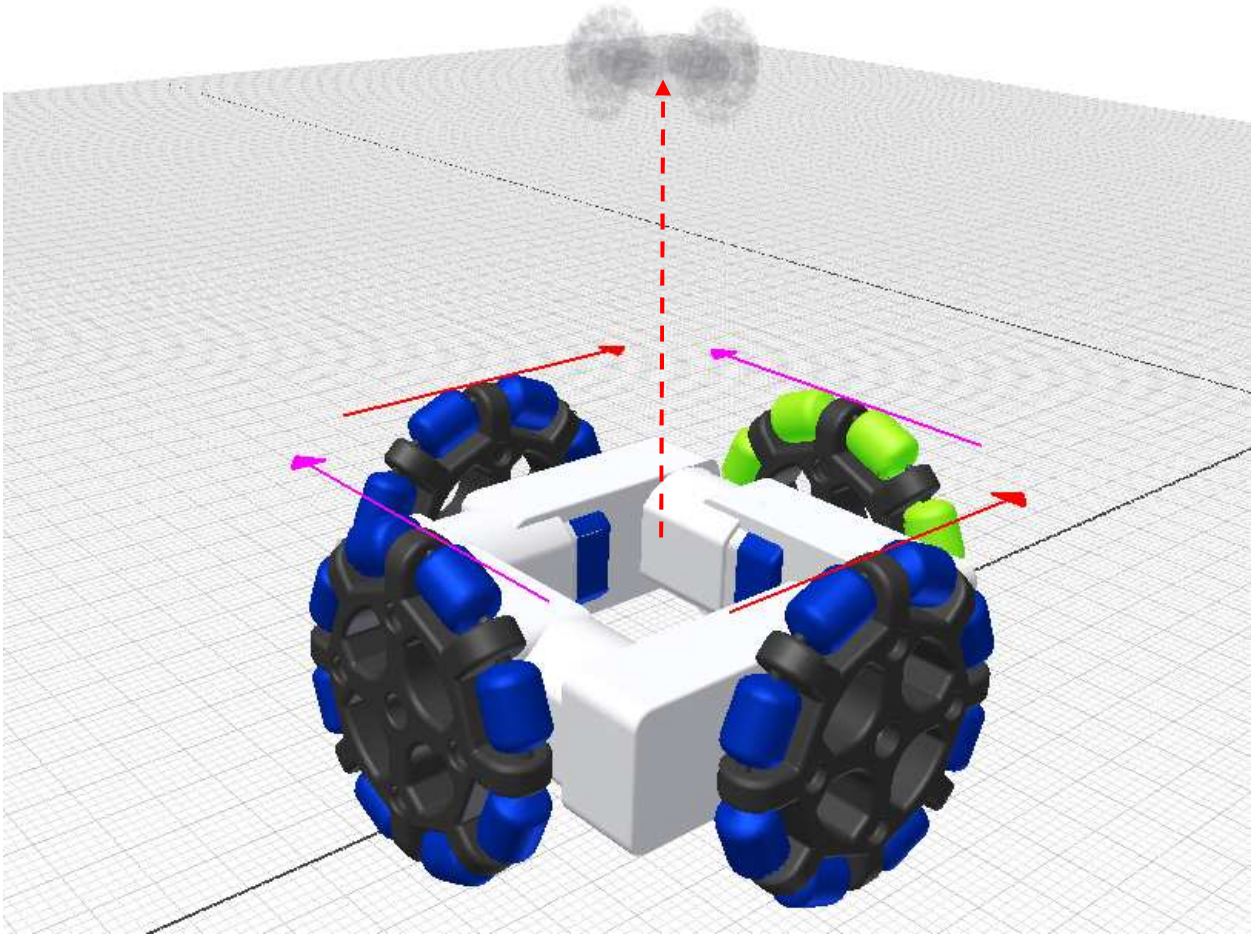
Συμπληρωματικά

Η ευκολότερη μέθοδος προγραμματισμού των μεταφράσεων είναι να τεθεί κάθε μία χειροκίνητα σε λειτουργία. Εντούτοις, αυτή η μέθοδος είναι μάλλον άχρηστη σε ένα πρόγραμμα με 8 ευδιάκριτες λειτουργίες. Αντ' αυτού προτείνεται η στοιχειώδης διανυσματική αποσύνθεση για τη μετακίνηση. Εάν κάθε ζευγάρι μηχανών ορίζει μια κατεύθυνση βασισμένη στη θέση του ρομπότ, τότε χρησιμοποιώντας τα δύο ζευγάρια συντεταγμένων "X" και "Y" του διανύσματος, η συνισταμένη θα δώσει το δεδομένο προορισμό.

Στη διπλανή **Εικόνα7.3.3 και Εικόνα7.3.4** παρακάτω , υπάρχει μόνο η συνιστώσα Y της κατεύθυνσης. Έτσι, το ρομπότ κινεί κατά τη συνιστώσα Y τις μηχανές και μεταφέρεται όπως δείχνει η σκιασμένη περιοχή.

Εικόνα7.3.3:Κίνηση ρομπότ κατά την συνιστώσα Y.





Εικόνα 7.3.4:Κίνηση του ρομπότ κατά την συνιστώσα Y.

Για να μετακινηθεί διαγώνια το ρομπότ πρέπει να χρησιμοποιηθούν και η x και η y συνιστώσα . Για να υπάρχει ποικιλία στις γωνίες τις οποίες το ρομπότ θα χρειαστεί να μετακινηθεί στον προορισμό, θα πρέπει να αλλάξει η ταχύτητα του ζευγαριού μηχανών κατά X ή του ζευγαριού μηχανών κατά Y.

Εναλλακτικά μια πιο κλασική μέθοδος που οδηγεί στον προορισμό και έπειτα μετακινεί με μία μόνο συνιστώσα της μηχανής είναι εφικτή και πιθανώς ευκολότερη (λιγότερη βαθμολόγηση).

Οι ιδιωτικές τιμές μιας κατηγορίας ρομπότ περιλαμβάνουν τη γωνία του προσανατολισμού και τη x-y θέση .

Public functions

SetMotorX (velocity) //θέτει την ταχύτητα μηχανής για το ζευγάρι στη x συνιστώσα (+,-)

SetMotorY (velocity) // θέτει την ταχύτητα μηχανής για το ζευγάρι στη y συνιστώσα (+,-)

MoveToLoc(x, y) // υπολογίζει τις σωστές ταχύτητες σε μια κατεύθυνση για το ζευγάρι μηχανών

Data Out () // εξάγει τις εντολές της μηχανής από το software στο hardware

[εισάγει γενικές εντολές κίνησης μιας μηχανής για απομάκρυνση των λαθών]

Error Checking

Επειδή η ζωή δεν είναι τέλεια, οι αλγόριθμοι μετακίνησης θα απαιτήσουν πιθανώς πολύ κουραστικές δοκιμές για τις δευτερεύουσες ρυθμίσεις. Η χρησιμοποίηση της ταχύτητας ως καθοριστικός παράγοντας για την κατεύθυνση του ταξιδιού δεν είναι μια σοφή ιδέα εξετάζοντας τους παράγοντες ολίσθησης, την υπέρβαση μηχανών, τις γενικές ασυνέπειες στις μηχανές, και την εξάρτηση στον έλεγχο ταχύτητας για το μέρος ΕΕ των ρομπότ. Θεωρητικά ίσως να ήταν ένα καλό κόλπο η περιστροφή σε ένα σημείο και ρυθμίζοντας την κατεύθυνση να ταίριαζε η γωνία κατά την οποία θα μετακινιόταν το ρομπότ .

7.4 ΘΕΩΡΙΑ ΚΙΝΗΣΕΩΝ ΤΩΝ ΡΟΜΠΟΤ

7.4.1 Κατηγορίες ρομπότ

Τα ρομπότ χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία αναφέρεται στα ρομπότ τα οποία μετακινούνται (βαδίζουνε) και έτσι αλλάζει η γεωγραφική τους θέση. Τα ρομπότ αυτά ονομάζονται βαδίζοντα ρομπότ. Η δεύτερη κατηγορία αναφέρεται σε ρομπότ τα οποία είναι ακίνητα στον χώρο, τα οποία είναι τοποθετημένα σε μία ακίνητη βάση. Αυτά τα ρομπότ, συνήθως μετακινούνε άλλα αντικείμενα και κάνουν επαναλαμβανόμενες κινήσεις. Στις μέρες μας αυτά τα ρομπότ χρησιμοποιούνται πολύ κυρίως στην βιομηχανία και κάνουνε δουλειές που είναι δύσκολες ή ασύμφορες να γίνουνε με το ανθρώπινο σώμα. Τέτοιο ρομπότ είναι για παράδειγμα ένας βαφέας αυτοκινήτων.

Τα ρομπότ ακίνητης βάσης αριθμούνται περισσότερα απ’ ότι τα βαδίζοντα ρομπότ. Αυτό συμβαίνει επειδή τα βαδίζοντα ρομπότ είναι ακριβά, και δεν προσφέρουνε κέρδος στους ιδιοκτήτες τους. Τα επόμενα χρόνια, καθώς βελτιώνεται η τεχνολογία αυτό θα αλλάξει, αφού θα δημιουργηθούνε «έξυπνα» βαδίζοντα ρομπότ, χρήσιμα για τον άνθρωπο με προσιτή τιμή.

7.4.2 Τρόποι κινήσεως των ρομπότ

Για την κίνηση στα βαδίζοντα ρομπότ, οι επιστήμονες λαμβάνουν πολλά υπ’ όψιν τους. Θα πρέπει ο μηχανικός του εξοπλισμός να είναι το δυνατόν μικρότερος (για πρακτικούς λόγους) και ελαφρύτερος (αλλιώς το ρομπότ θα έχει επιπλέον βάρος, με αποτέλεσμα να απαιτεί περισσότερη ενέργεια για να γίνει κάποια κίνηση). Επιπλέον πολύ σημαντικό ρόλο παίζει η

ακρίβεια. Εάν για παράδειγμα το ρομπότ πρέπει να φτάσει σε κάποιο σημείο, τότε πρέπει να φτάσει στον προορισμό χωρίς απόκλιση. Πρέπει λοιπόν να υπάρχουν μηχανισμοί διόρθωσης λαθών για την κίνηση. Επιθυμητό επίσης θα ήτανε το ρομπότ όταν βαδίζει να έχει μια ικανοποιητική ταχύτητα (διαφορετική σε κάθε περίπτωση). Είναι συμφέρον τα ρομπότ να είναι γρήγορα. Η μορφολογία του εδάφους δεν είναι ποτέ δεδομένη. Πρέπει να υπάρχουν τρόποι το ρομπότ να μπορεί να κινηθεί ακόμη και σε δύσβατες περιοχές με τραχύ έδαφος. Όταν βρεθούνε περιοχές που το ρομπότ δεν μπορεί να προσπελάσει, τότε δεν πρέπει να τις πλησιάζει. Αυτό μπορεί να γίνει με μηχανισμούς που παίζουν ρόλο ματιών. Σε κάθε χρονική στιγμή το ρομπότ θα πρέπει να ξέρει πού βρίσκεται. Άρα πρέπει να υπάρχει τρόπος αυτοεντοπισμού.

Ένα ρομπότ μπορεί να κινηθεί με πολλούς τρόπους. Αυτός είναι ένας λόγος που υπάρχουν τόσα πολλά διαφορετικά στην όψη και την λειτουργία ρομπότ. Οι συνηθέστεροι και κύριοι τρόποι κίνησης των ρομπότ είναι οι επίγειοι τρόποι. Επίγεια κίνηση μπορεί να γίνει είτε με τροχούς είτε με πόδια. Το ρομπότ με τροχούς μετακινείται όπως ένα αυτοκίνητο ή ένα ποδήλατο ή ακόμη και ένα τανκ. Το ρομπότ με πόδια μετακινείται όπως μερικά ζώα. Τα τελευταία χρόνια παρατηρούνται κινήσεις ρομπότ που δεν είναι προφανείς. Για παράδειγμα, ρομπότ που κολυμπάνε (Υποβρύχια ρομπότ), ρομπότ που πετάνε (Ιπτάμενα ρομπότ), ρομπότ που αναρριχούνται (για παράδειγμα κατακόρυφη αναρρίχηση με βεντούζες σε τοίχους και άλλες επίπεδες επιφάνειες), ρομπότ που στηρίζονται και κινούνται με μία σφαίρα, ρομπότ που σέρνονται όπως το φίδι (το λεγόμενο ζικ-ζακ), και όπως η κάμπια (μαζεύεται και αμέσως μετά τεντώνεται) κ.α.

Στην συνέχεια περιγράφονται οι κύριοι τρόποι κίνησης των ρομπότ:

1) Τροχοφόρα ρομπότ: Η κίνηση με τροχούς είναι πιο κατάλληλη μέθοδος για πολλές εφαρμογές, αφού δεν είναι τόσο πολύπλοκη σε σχέση με την κίνηση με πόδια. Όταν υπάρχουν τρεις τροχοί, τότε εγγυάται η ισορροπία του ρομπότ. Όταν οι τροχοί είναι περισσότεροι, τότε απαιτείται να υπάρχει ευκαμψία (ανάρτηση).

Τα πλεονεκτήματα των τροχοφόρων ρομπότ είναι:

- Ευκολία στον έλεγχο κίνησης
- Ευκολία στον σχεδιασμό του ρομπότ
- Χρειάζεται λιγότερη ενέργεια σε σχέση με τα ρομπότ με πόδια, για να διανυθεί η ίδια απόσταση.

Τα μειονεκτήματα αυτής της μεθόδου είναι:

- Περιορίζονται οι κινήσεις σε λείο έδαφος
- Καθώς η τραχύτητα του εδάφους αυξάνεται, απαιτούνται μεγαλύτερες ρόδες, και άρα περισσότερη ενέργεια για να γίνει η κίνηση.

2) Ρομπότ με πόδια:
Για βήμα με πόδι απαιτούνται τουλάχιστον δύο βαθμοί ελευθερίας για να γίνουνε δύο περιστροφικές κινήσεις. Στα βαδίζοντα ρομπότ συνήθως χρησιμοποιούνται τρεις βαθμοί ελευθερίας. Έτσι ίσως βελτιώνεται λίγο η κίνηση, αλλά αυξάνεται η πολυπλοκότητα του σχεδιασμού και ιδιαίτερα του ελέγχου της κίνησης.

Τα πλεονεκτήματα αυτής της μεθόδου είναι:

- Μπορούνε να κινηθούνε ακόμη και σε μέρη που δεν μπορούνε να κινηθούνε ρομπότ με ρόδες.
- Μπορούνε να κινηθούνε σε επίπεδα με κλίση ή ακόμη και να ανέβουνε σκαλοπάτια.

Τα μειονεκτήματα αυτής της μεθόδου είναι:

- Καταναλώνουνε πολύ ενέργεια
- Δύσκολη κατασκευή
- Δύσκολος έλεγχος κίνησης

7.4.3 Σταθερότητα των ρομπότ

Ενώ τα περισσότερα ζώα στην γη κινούνται με πόδια, στον κλάδο της ρομποτικής, αυτό είναι πολύ δύσκολη υπόθεση, αφού τα ρομπότ θα πρέπει να κινούνται χωρίς να πέφτουνε στο έδαφος. Υπάρχουν πολλά στοιχεία που επηρεάζουνε την ισορροπία των ρομπότ όπως: Ο αριθμός, η επιφάνεια και η γωνία των άκρων του ρομπότ που ακουμπάει το έδαφος, το σημείο που βρίσκεται το κέντρο ισορροπίας, η δομή του εδάφους (νερό, λάσπη, άμμο κ.τ.λ.), η τριβή του σημείου επαφής κ.α.

Σταθερότητα σημαίνει η δυνατότητα να διατηρήσει το ρομπότ όρθια θέση με τα πόδια του. Υπάρχουνε δύο μορφές σταθερότητας στα ρομπότ, η στατική και η δυναμική σταθερότητα:

1) Στατική σταθερότητα: Στην στατική σταθερότητα, τα ρομπότ μπορούν να στέκονται ακίνητα χωρίς να πέφτουν στο έδαφος. Αυτή είναι μια χρήσιμη μέθοδος που όμως είναι δύσκολο να επιτευχθεί, αφού έχει προϋπόθεση να υπάρχουν αρκετά πόδια ώστε να παρέχεται στο ρομπότ ξεκάθαρα η σταθερότητά του κατά την διάρκεια της κίνησης. Στην στατική σταθερότητα ανήκουνε ακίνητα ρομπότ με τουλάχιστον τρία πόδια, τα οποία πρέπει να είναι σωστά τοποθετημένα ώστε το κέντρο βάρος του ρομπότ να βρίσκεται μέσα σε κάποια συγκεκριμένα όρια. Τι γίνεται όμως όταν κάποιο πόδι μετακινείται για να γίνει βήμα; Τότε το ρομπότ πέφτει;

Στατική σταθερότητα βαδίσματος σημαίνει ότι τα πόδια του ρομπότ μπορεί να «παγώσουν» οποιαδήποτε χρονική στιγμή κατά την διάρκεια της κίνησης, χωρίς το ρομπότ να χάσει την ισορροπία του. Για να υπάρχει στατική σταθερότητα κατά την διάρκεια της κίνησης, απαιτούνται τουλάχιστον έξη πόδια για να παρέχεται στο ρομπότ η σωστή ισορροπία αφού τότε πάντα υπάρχουνε τρία πόδια στο έδαφος. Για παράδειγμα, ο άνθρωπος δεν είναι στατικά σταθερός. Για να σταθεί όρθιος (κάτι το οποίο χρειάζεται προσπάθεια)

στην πραγματικότητα χρησιμοποιεί μυς, τένοντες και νεύρα. Αυτή η ισορροπία του ανθρώπου είναι υποσυνείδητη και μαθαίνεται από την βρεφική ηλικία.

2) Δυναμική σταθερότητα: Στην δυναμική σταθερότητα τα ρομπότ ισορροπούνε λαμβάνοντας υπ' όψιν και την στάση του σώματός τους. Για παράδειγμα ένα ρομπότ που έχει μόνο ένα πόδι ισορροπεί με δυναμική σταθερότητα. Στην περίπτωση αυτή όμως το ρομπότ δεν πρέπει να σταματήσει ποτέ να πηδάει, για να συνεχίσει να ισορροπεί. Στην περίπτωση των ανθρωποειδών ρομπότ, εάν για παράδειγμα το ρομπότ κινήσει το πόδι του μπροστά, τότε επηρεάζεται το κέντρο βάρους και η ισορροπία διαταράσσεται. Σε αυτήν την περίπτωση στην στατική σταθερότητα το ρομπότ θα έπεφτε. Στην δυναμική σταθερότητα όμως το ρομπότ θα προσαρμόσει την κίνηση (π.χ. με την κατάλληλη κλήση στην μέση) ώστε το κέντρο βάρους να βρίσκεται πάντα πάνω από το πόδι που στηρίζεται στο έδαφος. Δυναμική σταθερότητα βαδίσματος σημαίνει ότι η σταθερότητα μπορεί να διατηρηθεί μόνο κατά την διάρκεια της κίνησης.

7.4.4 Βηματισμός των ρομπότ

Ο βηματισμός καθορίζει την συχνότητα με την οποία εναλλάσσεται η υποστήριξη ποδιού στο έδαφος κατά την διάρκεια της κίνησης. Ο βηματισμός χωρίζεται σε δύο κατηγορίες:

- Περιοδικός βηματισμός, όπου επαναλαμβάνεται η ίδια συχνότητα με την οποία εναλλάσσεται η υποστήριξη ποδιού στο έδαφος
- Μη περιοδικός ή ελεύθερος βηματισμός, όπου δεν υπάρχει περιοδικότητα στο βήμα.

Ο ρυθμός των βημάτων εξαρτάται από τον αριθμό των ποδιών. Η κίνηση ενός ρομπότ με πόδια ονομάζεται βάδισμα μόνο εάν κατά την διάρκεια της κίνησης υπάρχει πάντα ένα τουλάχιστον πόδι που στηρίζει το σώμα. Εάν υπάρχει κάποια στιγμή όπου κανένα πόδι δεν στηρίζει το ρομπότ, τότε η κίνηση λέγεται τρέξιμο. Ενώ το βάδισμα μπορεί να είναι είτε στατικό είτε δυναμικό, το τρέξιμο μπορεί να είναι μόνο δυναμικό.

7.4.4.1 Παθητικός Βηματισμός

Για να κινηθούνε τα βαδίζοντα ρομπότ απαιτείται πολύ ενέργεια. Πολλοί μελετητές υποστηρίζουνε ότι είναι ασύμφορο να δημιουργηθούνε ρομπότ με πόδια που περπατάνε. Αυτό το πρόβλημα κλήθηκαν να διορθώσουνε άλλοι επιστήμονες δημιουργώντας τα βαδίζοντα ρομπότ με παθητικό βηματισμό.

Στον παθητικό βηματισμό το ρομπότ εκμεταλλεύεται και τους νόμους της φυσικής. Αντί να γίνεται χρήση μόνο υψηλής δυναμικής ηλεκτρονικών μόνιτορ, γίνεται επιπλέον, εκμετάλλευση της συχνότητα κίνησης και του βάρους των ποδιών. Όταν εφαρμόζεται αυτή η μέθοδος βηματισμού, τα προβλήματα ισορροπίας και ελέγχου της κίνησης αυξάνονται. Γι' αυτό το λόγο οι επιστήμονες προσπαθούν να την συνδυάσουν με την δυναμική

σταθερότητα για να κατασκευαστεί ρομπότ που καταναλώνει λιγότερη ενέργεια. Παθητικός βηματισμός χρησιμοποιείται περισσότερο στα δίποδα ρομπότ, γιατί αυτή η μέθοδος βηματισμού είναι πιο αποτελεσματική.

7.5 ΠΕΡΙΓΡΑΦΗ ΡΟΜΠΟΤ ΑΝΑΛΟΓΑ ΜΕ ΤΟΝ ΑΡΙΘΜΟ ΤΩΝ ΑΚΡΩΝ.

Γενικά ισχύει ότι όσο λιγότερα είναι τα πόδια σε ένα ρομπότ τόσο πιο πολύπλοκη και δύσκολη είναι η κίνησή του.

7.5.1 Δίτροχα ρομπότ

Τα τελευταία χρόνια τα δίτροχα ρομπότ γίνονται όλο και περισσότερο διαδεδομένα. Όπως είναι αναμενόμενο, τα δίτροχα ρομπότ για να μην πέσουν χρησιμοποιούν δυναμική σταθερότητα. Συγκεκριμένα, ελέγχουν την ισορροπία τους κάνοντας συνεχείς μετακινήσεις μπρος-πίσω. Πλέον τα δίτροχα ρομπότ μπορούν να μένουνε ακίνητα, ακόμη και σε απότομη κλίση εδάφους.

7.5.2 Κίνηση ρομπότ με ένα πόδι

Η κίνηση αυτή κεντρίζει το ενδιαφέρον πολλών ανθρώπων, και λίγοι είναι αυτοί που θα την πιστέψουν πριν την δούνε με τα ίδια τους τα μάτια! Ο λόγος είναι ότι δεν υπάρχει τέτοια κίνηση στην φύση και φαίνεται πρωτόγνωρη σε όλους. Επιπλέον, ο κλάδος αυτός της κίνησης των ρομπότ με ένα πόδι είναι πολύ πρόσφατος, με αποτέλεσμα να μην ξέρουνε οι άνθρωποι την μέχρι τώρα τεχνολογία.

Τα ρομπότ που έχουν μόνο ένα πόδι ισορροπούνε πολύ δύσκολα. Συγκεκριμένα το ρομπότ κάνει συνεχείς μετατροπές στην γωνία του ποδιού του έτσι ώστε να ελέγχει την συμπεριφορά του σώματος του. Αυτή η μέθοδος κίνησης είναι πολύ δύσκολη και ασύμφορη, αφού η κατασκευή είναι πολύπλοκη και καταναλώνεται πολύ ενέργεια. Επιπρόσθετα, για να κινηθεί το ρομπότ προς κάποια κατεύθυνση θα πρέπει απαραιτήτως το δάπεδο να είναι λείο. Σε αντίθετη περίπτωση το ρομπότ δεν φτάνει ποτέ στον προορισμό του, αφού η τραχύτητα του εδάφους το εξαναγκάζει να αποκλίνει συνεχώς από την πορεία του για να διατηρήσει την ισορροπία του.

7.5.3 Κίνηση ρομπότ με δύο πόδια

Για να γίνει το ρομπότ «ο καλύτερος φίλος του ανθρώπου» πρέπει να μάθει να βαδίζει στο πλάι του. Αυτό το γνωρίζουν καλά οι κατασκευαστές ρομπότ και δεκαετίες τώρα προσπαθούν να διδάξουν όσο καλύτερα μπορούν το βάδισμα στα δίποδα ρομπότ. Το δίποδο ρομπότ όμως καταναλώνει βαδίζοντας δέκα φορές περισσότερη ενέργεια από ότι ένας άνθρωπος κάτι το οποίο είναι απογοητευτικά ασύμφορο. Παρ’ όλα αυτά, αυτή η κίνηση είναι ιδιαίτερα διαδεδομένη.

Τα ρομπότ με δύο πόδια έχουνε στατική σταθερότητα μόνο σε μικρές χρονικές στιγμές. Όταν το κέντρο βάρους του σώματός τους βρίσκεται ακριβώς πάνω από το πόδι του ρομπότ. Περισσότερο χρησιμοποιείται η δυναμική σταθερότητα. Επειδή ισορροπούνε δύσκολα η ταχύτητα κίνησής τους είναι αργή.

Στον παθητικό βηματισμό η κίνηση του δίποδου ρομπότ (όπως και ο τρόπος που βαδίζουν οι άνθρωποι) μοιάζει κατά κάποιον τρόπο με την κίνηση του τροχού, όπου η ακτίνα του τροχού παίζει τον ρόλο των ποδιών. Γίνονται προσπάθειες από επιστήμονες να δημιουργήσουνε δίποδο ρομπότ που βαδίζει σύμφωνα με την κίνηση ενός τροχού, έτσι ώστε να αυξηθεί η ταχύτητα κίνησής του και να μειωθεί η ενέργεια που καταναλώνει. Εδώ πρέπει να συμπληρωθεί ότι στα δίποδα ρομπότ όπως και στα ρομπότ που κινούνται με ένα πόδι, υπάρχει μια έξτρα απώλεια ενέργειας, που οφείλεται στην κίνηση πάνω-κάτω που κάνει το κέντρο βάρους του σώματος.

7.5.4 Κίνηση ρομπότ με τέσσερα πόδια

Το βάδισμα με τέσσερα πόδια θυμίζει σε όλους κινήσεις τετράποδων ζώων. Τα περισσότερα εμπορικά τετράποδα ρομπότ μοιάζουν με σκύλους και απευθύνονται σε ζωόφιλο κοινό ή σε μικρά παιδιά. Είναι δύσκολο να διατηρηθεί η ισορροπία καθ' όλη την διάρκεια της κίνησης στα τετράποδα ρομπότ. Χρειάζεται κάθε φορά να προσαρμοστούν τα πόδια, έτσι ώστε το κέντρο βάρους του ρομπότ να βρίσκεται σε συγκεκριμένα όρια. Σε περίπτωση όμως που το έδαφος είναι τραχύ, εύκολα μπορεί να ακολουθήσει πτώση. Στα τετράποδα ρομπότ είναι εφικτή η στατική σταθερότητα, αλλά όχι πάντα. Παίζει ρόλο πού βρίσκεται το κέντρο βάρους κάθε φορά. Παρ' όλα αυτά έχει πρόσφατα δημιουργηθεί τετράποδο ρομπότ το οποίο δεν χάνει την ισορροπία του κατά την διάρκεια του βαδίσματος ακόμη κι όταν δέχεται σπρώξιμο.

7.5.5 Κίνηση ρομπότ με έξη πόδια

Τα ρομπότ με έξη πόδια κινούνται έχοντας πάντα τρία πόδια στο έδαφος, ενώ τρία πόδια μετατοπίζονται ταυτόχρονα για να γίνει η μετακίνηση. Αυτό έχει ως αποτέλεσμα να έχουνε πολύ καλή ισορροπία. Αυτός είναι και ο λόγος που είναι τα αγαπημένα των κατασκευαστών. Λόγω των πολλών ποδιών, χρησιμοποιούνε στατική σταθερότητα, αλλά αυτό τα κάνει αργά και ασύμφορα, αφού απαιτούνε πολύ ενέργεια. Υπάρχει βέβαια και η περίπτωση να αυξηθεί η ταχύτητά τους, εάν εφαρμοστεί δυναμική σταθερότητα. Στα εξάποδα ρομπότ, η ισορροπία κατά την διάρκεια της κίνησης είναι δεδομένη, αφού πάντα τρία πόδια βρίσκονται στο έδαφος. Από την οικογένεια των ρομπότ με πόδια, τα εξάποδα θεωρούνται από πολλούς τα καταλληλότερα για πλήθος εφαρμογών, και ιδιαίτερα όταν απαιτείται πολύ καλή ισορροπία και υπάρχει πολύ τραχύ έδαφος.

7.5.6 Κίνηση ρομπότ με περισσότερα από έξι πόδια

Τα ρομπότ με πολλά πόδια είναι γενικά ασύμφορα και πολύπλοκα. Έχουνε κατασκευαστεί πολλά ρομπότ με περισσότερα από 6 πόδια. Υποστηρίζεται ότι τα επιπλέον πόδια μπορούν να βοηθήσουν επιπλέον την κίνηση, για να γίνεται με πιο σωστό τρόπο. Τα πολλά πόδια όμως απαιτούνε επιπλέον ενέργεια και προσθέτουνε στο ρομπότ βάρος. Άρα προτιμότερο είναι να χρησιμοποιούνται έξη πόδια, αφού και έτσι η ισορροπία είναι δεδομένη. Η ενέργεια που καταναλώνουνε τα ρομπότ που έχουνε πάνω από έξη πόδια είναι τόσο μεγαλύτερη, που δεν συμφέρει από άποψη ενέργειας να βελτιωθεί η κίνηση με αυτόν τον τρόπο.

Κεφάλαιο 8

Αισθητήρες

8.1 Σκοπός

Εισαγωγή στους απλούς και σύνθετους αισθητήρες, στο πώς λειτουργούν, ο τύπος στοιχείων που παράγουν, και πώς μπορούν να χρησιμοποιηθούν στη ρομποτική.

Ένας αισθητήρας είναι μια συσκευή που μετρά μια φυσική ποσότητα και την μετατρέπει σε ένα σήμα που μπορεί να διαβαστεί από έναν παρατηρητή ή από ένα όργανο.

Τι είναι οι αισθητήρες

Οι αισθητήρες είναι συσκευές που μετατρέπουν μια φυσική παράμετρο όπως η θερμοκρασία δωματίου, η πίεση του αίματος ή η ταχύτητα του αέρα σε ένα σήμα που μπορεί να μετρηθεί ηλεκτρικά. Κι άλλοι ακροδέκτες εξόδου είναι εξίσου αποτελεσματικοί, αλλά εδώ θα γίνει αναφορά στους ηλεκτρικούς αισθητήρες παραγωγής δεδομένων γιατί είναι συμβατότεροι με τις ηλεκτρονικές συσκευές μέτρησης.

Μόλις μετατραπεί η φυσική παράμετρος σε μια ισοδύναμη ηλεκτρική, είναι εύκολο να εισαχθεί σε ένα υπολογιστή ή έναν μικροεπεξεργαστή για το χειρισμό, την ανάλυση και την εμφάνιση στην οθόνη.

Η πιο κοινή παράμετρος που μετριέται είναι η θερμοκρασία. Αυτή τη στιγμή η θερμοκρασία είναι η μόνη παράμετρος που καλύπτεται λεπτομερώς. Κάθε έτος εκατομμύρια εκατοντάδων αισθητήρες κατασκευάζονται. Είναι στις εσωτερικές συσκευές, τον ιατρικό εξοπλισμό, τα βιομηχανικά συστήματα ελέγχου, τα συστήματα κλιματισμού, τα αεροσκάφη, τους δορυφόρους και τα παιχνίδια.

Οι αισθητήρες γίνονται εξυπνότεροι, πιο ακριβείς και φτηνότεροι. Θα παίξουν σημαντικό ρόλο για κάθε τομέα .

Χρήση

Οι αισθητήρες χρησιμοποιούνται σε αντικείμενα της καθημερινότητας όπως είναι τα ευαίσθητα κουμπιά επαφής των ανελκυστήρων και οι λαμπτήρες που εξασθενίζουν ή φωτίζουν πιο πολύ με το άγγιγμα της βάσης. Υπάρχουν επίσης αναρίθμητες εφαρμογές για τους αισθητήρες των οποίων οι περισσότεροι άνθρωποι δεν γνωρίζουν ποτέ. Οι εφαρμογές περιλαμβάνουν τα αυτοκίνητα, τις μηχανές, το αεροδιάστημα, την ιατρική, την κατασκευή και τη ρομποτική.

8.2 Αισθητήρες Ρομπότ

Οι αισθητήρες είναι αυτό που τα ρομπότ χρειάζονται για να αντιλαμβάνονται τον κόσμο. Υπάρχουν διάφορα είδη αισθητήρων όπως υπερηχητικοί, υπέρυθρων ακτίνων, GPS, κάμερες, επιταχυντές, κωδικοποιητές και άλλοι.

Οι αισθητήρες ρομπότ είναι ουσιαστικά στοιχεία που δημιουργούν αυτόνομα ρομπότ δεδομένου ότι είναι τα μόνα μέσα για ένα ρομπότ που μπορούν να ανιχνεύσουν πληροφορίες για το περιβάλλον του.

Υπάρχει μια μεγάλη ποικιλία διαθέσιμων αισθητήρων που είναι σε θέση να μετρήσουν σχεδόν τα πάντα, από περιβαλλοντικές συνθήκες (απόσταση, φως, ήχος, θερμοκρασία) μέχρι γωνιακή και γραμμική επιτάχυνση, δυνάμεις και αποστάσεις. Ο πρώτος αισθητήρας που ενσωματώνεται συχνά σε ένα κινητό ρομπότ είναι ένας αισθητήρας απόστασης, ο οποίος είναι συνήθως υπό μορφή υπέρυθρου ή υπερηχητικού αισθητήρα (infrared or ultrasonic sensor). Και στις δύο περιπτώσεις, ένας παλμός (του φωτός ή του ήχου) στέλνεται και η αντανάκλασή του είναι χρονομετρημένη για να πάρει μια εικόνα της απόστασης. Συνήθως αυτές οι τιμές στέλνονται στον ελεγκτή πολλές φορές κάθε δευτερόλεπτο.

Επιλογή αισθητήρων για το ρομπότ

Οι αισθητήρες είναι συσκευές που μετατρέπουν τα κοσμικά γεγονότα σε ηλεκτρικά σήματα. Η επιλογή των αισθητήρων για ένα ρομπότ μπορεί να είναι ένας από τους δυσκολότερους στόχους σε ολόκληρη τη διαδικασία κατασκευής.

Ένα κλασσικό ρομπότ χρειάζεται περισσότερους από έναν αισθητήρες. Ακόμη και με τους καλύτερους μηχανικούς και προγραμματιστές παγκοσμίως, ένα ρομπότ με μια φτωχή ακολουθία αισθητήρων δε θα λειτουργεί σωστά. Οι αισθητήρες είναι το σημαντικότερο μέρος ενός αυτόνομου ρομπότ, και επίσης το ακριβότερο.

Οι βασικότερες κατηγορίες των αισθητήρων

Ο κόσμος των αισθητήρων χωρίζεται σε διάφορες κατηγορίες:

Οι αισθητήρες Proximity: μοιάζουν με τους αισθητήρες στους προφυλακτήρες (bumpers) και προειδοποιούν εάν κάτι χτυπά το ρομπότ. Δίνουν μια απλή ναι/όχι απάντηση. Οι αισθητήρες Range, μοιάζουν με τους ανιχνευτές IR και τους sonar οι οποίοι λένε το πόσο μακριά είναι ένα αντικείμενο.

Οι αισθητήρες Global navigation, είναι όπως οι πυξίδες, τα GPS, και οι κωδικοποιητές. Οι αισθητήρες Proprioceptive που μετρούν τα εσωτερικά συστήματα. Άλλοι αισθητήρες όπως οι κάμερες, οι UV αισθητήρες, και οι ανιχνευτές θερμότητας.

Ένα σημαντικό μέρος της κατασκευής ενός ρομπότ είναι η ενσωμάτωση των αισθητήρων. Οι αισθητήρες μεταφράζουν αυτά που

λαμβάνονται από το φυσικό κόσμο και τον θεωρητικό κόσμο των μικροελεγκτών. Οι αισθητήρες βοηθούν στη μετάφραση των ιδιοτήτων του φυσικού κόσμου σε τιμές που ο υπολογιστής μπορεί να χρησιμοποιήσει σε ένα ρομπότ. Η μετάφραση παράγει κάτι σαν την τιμή εξόδου που ο μικροελεγκτής μπορεί να χρησιμοποιήσει. Γενικά, οι περισσότεροι αισθητήρες εμπίπτουν στη μια από τις δύο ακόλουθες κατηγορίες:

1. αναλογικοί αισθητήρες
2. ψηφιακοί αισθητήρες

Ένας αναλογικός αισθητήρας, όπως ένα κύτταρο CdS (τα κύτταρα σουλφιδίου καδμίου μετρούν την ελαφριά ένταση), μπορεί να συνδεθεί με καλώδιο σε ένα κύκλωμα με τέτοιο τρόπο έτσι ώστε να έχει μια έξοδο που κυμαίνεται από 0 βολτ ως 5 βολτ. Η τιμή μπορεί να έχει οποιαδήποτε πιθανή τιμή μεταξύ 0 και 5 βολτ. Ένα "αναλογικό σήμα" είναι αυτό που μπορεί να θέσει οποιαδήποτε τιμή σε μια κλίμακα. Για παράδειγμα ένα αναλογικό σήμα λειτουργεί όπως ένα δέκτης σε ένα παλιό ραδιόφωνο. Ο χρήστης μπορεί να τον γυρίζει πάνω ή κάτω σε μια συνεχή κίνηση.

Οι ψηφιακοί αισθητήρες παράγουν αυτό που καλείται "Discrete Signal". Αυτό σημαίνει ότι υπάρχει μια κλίμακα τιμών που ο αισθητήρας μπορεί να έχει ως έξοδο, αλλά η τιμή πρέπει να αυξηθεί σε βήματα. Υπάρχει μια γνωστή σχέση μεταξύ οποιασδήποτε τιμής και τόσο για αυτές που προηγούνται όσο και για αυτές που ακολουθούν. Τα "Discrete Signals" τυπικά έχουν μια εμφάνιση σκαλοπατιών όταν δίνονται με γραφική παράσταση στο διάγραμμα.

Χαρακτηριστικά αισθητήρων

Δεν κατασκευάζονται όλοι οι αισθητήρες το ίδιο. Η κατανόηση για το πώς ο αισθητήρας λειτουργεί, και τι λέει, είναι σημαντική ώστε ένας αισθητήρας να χρησιμοποιηθεί σωστά. Υπάρχουν διάφορα βασικά χαρακτηριστικά κλειδιά που αξιολογούν την επιλογή των αισθητήρων:

- Το πώς διασυνδέεται με τον ελεγκτή
- Ποια είναι η ανάλυσή του, η επαναληπτικότητα και η γενική ακρίβεια του;
- Ποια είναι η κλίμακά του ;
- Ποιο είναι το επίπεδο τάσης και η κατανάλωση ισχύος;

Υπάρχει ένα πλήθος τρόπων να διασυνδεθούν οι αισθητήρες με έναν ελεγκτή. Ο πιο κοινός θα ήταν:

- αναλογικοί αισθητήρες, βγάζουν μια αναλογική τάση σχετική με κάποια μέτρηση που κάνουν. Απαιτεί έναν αναλογικό σε ψηφιακό μετατροπέα.
- οι ψηφιακοί αισθητήρες μπορούν να είναι είτε TTL (5 βολτ) είτε χαμηλής τάσης (3,3 βολτ)
- οι σειριακές συνδέσεις είναι αρκετά κοινές
- ο I2C είναι μια πολυσυσκευή διαύλων που συχνά εμφανίζεται στους αισθητήρες

- RS- 485 ή τύπου σειριακών διαύλων συνδέσεις μπορούν επίσης να βρεθούν στους υψηλότερους αισθητήρες.

Το πρώτο χαρακτηριστικό που πρέπει να εξετάζεται για έναν αισθητήρα, είναι η ανάλυσή του. Οι ψηφιακοί αισθητήρες έχουν μόνο δύο λειτουργίες: on και off. Ένας αναλογικός, σειριακός ή με διαύλους συνδεδεμένος αισθητήρας μπορεί να έχει πολλές τιμές. Παραδείγματος χάριν, υπάρχουν πυξίδες διαθέσιμες που δίνουν τη μια από τις τέσσερις κατευθύνσεις N,W,S,E αλλά υπάρχουν επίσης πυξίδες που έχουν μια σειριακή σύνδεση και δίνουν ερμηνείες και ενδείξεις κάτω των 0.1 βαθμών.

Υπάρχει επίσης ένα θέμα για την ακρίβεια και την επανάληψη. Ενώ η πυξίδα μας μπορεί να ερμηνεύει κάτω από 0.1 βαθμούς, μπορεί να είναι ακριβής μόνο +/- 2 βαθμούς. Παρόμοια προβλήματα εμφανίζονται με το GPS, καθώς πολλοί αισθητήρες των GPS δίνουν μια ερμηνεία σε κλίμακα εκατονταβάθμια, αλλά μόνο με ακρίβεια +/- 2 μέτρων, προκαλώντας απότομα μέγιστα τιμών πράγμα που αναγκάζει το χρήστη να φιλτράρει τα εισερχόμενα σήματα. Για τη κατασκευή ενός ρομπότ είναι σημαντική η κατανόηση του πώς ένας συγκεκριμένος αισθητήρας λειτουργεί, και πότε τα στοιχεία του χρειάζονται φιλτράρισμα.

Μια τελική σημαντική εκτίμηση κατά την επιλογή των αισθητήρων είναι το επίπεδο των τάσεών τους και του ρεύματος εξόδου. Το ταίριασμα του επιπέδου τάσης ενός αισθητήρα στην μπαταρία ή στους ήδη υπάρχοντες ρυθμιστές μπορεί να βοηθήσει στην ενεργειακή απώλεια. Χαμηλό ρεύμα εξόδου (current draw) είναι ιδιαίτερα σημαντικό για τα κινούμενα ρομπότ.

Ποντεσιόμετρα(Potentiometers)

Ένας συχνά αγνοημένος αλλά εξαιρετικά χρήσιμος αισθητήρας είναι ο παλιός καλός POT. Οι αισθητήρες POT είναι ιδιαίτερα χρήσιμοι για τις γωνιακές μετρήσεις, δεδομένου ότι τα περισσότερα ποτενσιόμετρα γυρίζουν κατά μέσο όρο στους 270 βαθμούς. Είναι για παράδειγμα οι καταλληλότεροι για τον καθορισμό των γωνιών ενός βραχίονα ρομπότ.

Διακόπτες (Switches)

Ένας από τους πιο βασικούς όλων των αισθητήρων είναι ένας απλός διακόπτης. Οι διακόπτες χρησιμοποιούνται στους αισθητήρες των προφυλακτήρων (bumper sensors), για να ανιχνεύσουν τα όρια της κίνησης, για την είσοδο χρηστών, και για πλήθος άλλων πραγμάτων.

Οι διακόπτες χωρίζονται σε δύο τύπους: κανονικά ανοικτό (NO) και κανονικά κλειστό (NC). Πολλά σχέδια μικροδιακοπτών έχουν στην πραγματικότητα ένα κοινό τερματικό, και μαζί ένα τερματικό NO και ένα τερματικό NC.

Υπέρυθροι ανιχνευτές (Infrared Detectors)

Η υπέρυθρη ανίχνευση είναι ένα κοινό πράγμα για να προσθέσεις σε ένα ρομπότ. Επιτρέπει στο ρομπότ να καθορίσει πότε έχει έρθει σε αμεσότητα ως προς ένα αντικείμενο χωρίς να έρθει σε φυσική επαφή.

8.3 Διάφοροι τύποι αισθητήρων

Οι αισθητήρες συνήθως εξάγουν δύο τύπους σημάτων. Ένα αναλογικό σήμα ή ένα Discrete σήμα. Οι μικροελεγκτές εξετάζουν συνήθως τα Discrete ή ψηφιακά σήματα. Ένας αναλογικός συγκριτικά με ένα ψηφιακό μετατροπέα δίνει τη δυνατότητα η έξοδος μιας αναλογικής συσκευής να χρησιμοποιηθεί από έναν μικροελεγκτή. Πολλοί μικροελεγκτές έχουν ενσωματωμένους μετατροπείς A/D.

Η διασύνδεση των αισθητήρων είναι αρκετά απλή. Απαραίτητη είναι η γνώση του τύπου εξόδου που παρέχει ένας αισθητήρας. Επίσης πρέπει να δοθεί ιδιαίτερη προσοχή στο να μην δημιουργηθεί μια πορεία που να επιτρέπει μεγάλη ροή ρεύματος. Οι αντιστάσεις για τη ρύθμιση του ρεύματος είναι σημαντικοί σε ότι αφορά τη διασύνδεση των μικροελεγκτών στους αισθητήρες.

Οι αισθητήρες παρέχουν σε ένα ρομπότ τα μέσα για να παρατηρεί χαρακτηριστικά του εξωτερικού περιβάλλοντος, και της εσωτερικής κατάστασης. Χωρίς αισθητήρες το ρομπότ θα ήταν ανίκανο να καθορίσει την επικρατούσα κατάστασή του και επομένως ανίκανο να ενεργήσει. Οι τύποι των διαθέσιμων αισθητήρων ποικίλλουν σε κόστος, σε πολυπλοκότητα της χρήσης, και σε ακρίβεια. Δεν είναι πάντα απαραίτητο να χρησιμοποιηθεί ο μεγαλύτερος και ο καλύτερος αισθητήρας για κάθε στόχο. Απλοί αισθητήρες για παράδειγμα είναι οι υπέρυθρες ακτίνες, και πιο σύνθετοι αισθητήρες είναι τα λέιζερ και οι κάμερες.

Υπάρχει μια ευρεία ποικιλία αισθητήρων που ένα ρομπότ θα μπορούσε να χρησιμοποιήσει. Μερικοί είναι απλοί, όπως οι αισθητήρες πρόσκρουσης, ενώ άλλοι είναι πολύ σύνθετοι, όπως οι αισθητήρες όρασης.

Ο τύπος των αισθητήρων που χρησιμοποιούνται σε ένα ρομπότ εξαρτάται από τους στόχους που το ρομπότ πρέπει να εκτελέσει, τη διαθέσιμη ηλεκτρική και υπολογιστική δύναμη, και την ικανότητα μεταφοράς.

Οι υπέρυθροι αισθητήρες είναι κοινοί, και μπορούν να διαμορφωθούν με πολυάριθμους τρόπους. Μια κοινή συνδεσμολογία είναι ένας αισθητήρας break beam (σπασμένης δοκού), όπου ένας πομπός IR και ένας αισθητήρας ανίχνευσης επιφανειών χωρίζονται μεταξύ τους από ένα κενό. Όταν ένα αντικείμενο βρεθεί μεταξύ του αισθητήρα και του πομπού, ένα σήμα παράγεται. Μια άλλη τυπική συνδεσμολογία έχει τον αισθητήρα και τον πομπό προς την ίδια κατεύθυνση. Το IR που έχει παραχθεί από τον πομπό όταν συναντάται κάποιο αντικείμενο, ανιχνεύεται από τον αισθητήρα. Η χρονική διαφορά μεταξύ της εκπομπής και της ανίχνευσης χρησιμοποιείται για να υπολογίσει την απόσταση ενός αντικειμένου που προκαλεί αντανάκλαση.

Τα Sonar είναι άλλος ένας τύπος αισθητήρα που συναντάται συνήθως στα ρομπότ. Οι ίδιες αρχές που χρησιμοποιούνται σε έναν αισθητήρα σειράς IR ισχύουν για έναν sonar αισθητήρα. Τα ηχητικά κύματα εκπέμπονται από το sonar, και ανιχνεύονται αφού έρθουν σε επαφή με αντικείμενα που βρίσκονται στο περιβάλλον. Η χρονική διαφορά μεταξύ της εκπομπής και της ανίχνευσης χρησιμοποιείται για να καθορίσει την απόσταση ενός αντικειμένου. Οι αισθητήρες Sonar έχουν μεγαλύτερη κλίμακα και ευρύτερο οπτικό πεδίο από το Glossary#IR, αλλά υστερούν στην κατοπτρική αντανάκλαση.

Η ακολουθία τοίχων είναι μια συνήθης συμπεριφορά ενός ρομπότ, και σχετικά εύκολη στο να εφαρμοστεί. Όπως μαρτυρά το όνομα, ο στόχος είναι να κινηθεί το ρομπότ παράλληλα σε έναν τοίχο. Για το σκοπό αυτό χρησιμοποιείται ένας αισθητήρας υπέρυθρων ακτίνων και ένα sonar που τοποθετείται στο ρομπότ για να διατηρήσει μια ορισμένη απόσταση από έναν τοίχο.

Οι αισθητήρες λέιζερ παρέχουν σημαντικά μεγαλύτερη ακρίβεια. Επίσης, αυτός ο τύπος λέιζερ δεν υστερεί όσον αφορά την κατοπτρική αντανάκλαση και έχει γενικά μια καλύτερη κλίμακα συγκριτικά με τους sonar και τους IR. Αυτές οι ιδιότητες έχουν κάνει τα λέιζερ από τους πιο κοινούς τύπους αισθητήρων που χρησιμοποιούνται στην ερευνητική κοινότητα. Οι αισθητήρες λέιζερ είναι πολύ ακριβές συσκευές. Ζυγίζουν επίσης περισσότερο από ότι τα ρομπότ μπορούν να υποστηρίξουν και έχουν σημαντικές απαιτήσεις δύναμης.

Η κάμερα προσφέρει μια σημαντική πηγή δεδομένων για ένα ρομπότ. Οι εικόνες είναι γεμάτες με χρήσιμες πληροφορίες όπως το χρώμα, τη σύσταση, την κίνηση και τα τρισδιάστατα στοιχεία όταν χρησιμοποιούνται στερεοφωνικά συστήματα όρασης. Εντούτοις, η εξαγωγή αυτών των πληροφοριών είναι υπολογιστικά ακριβή.

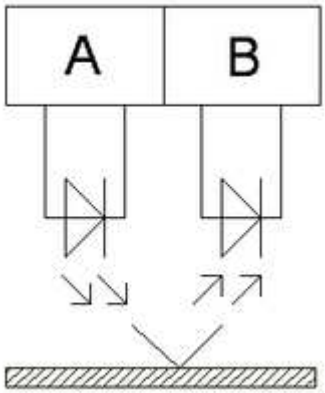
Μία κάμερα CMU Cam 2, παρέχει hardware κατανομής σημαδιών από χρώματα (color blob tracking). Αυτό σημαίνει ότι η κάμερα θα επεξεργαστεί τις εικόνες αυτόματα και θα παρέχει στο ρομπότ τις πληροφορίες για σημάδια χρωμάτων που εμφανίζονται στην εικόνα. Αυτό το στοιχείο θα χρησιμοποιηθεί από το ρομπότ για κατανομή των χρωματιστών αντικειμένων.

8.4 Αισθητήρες ρομπότ /Αισθητήρες απόστασης

IR

Οι αισθητήρες απόστασης IR είναι από τους πιο χρησιμοποιούμενους αισθητήρες απόστασης. Υπάρχουν 3 κατηγορίες:

Αντανάκλαση



Εικόνα 8.4.1: Αντανάκλαση φωτός που εκπέμπεται από LED.

Αισθητήρας IR αντανάκλασης

Αυτός ο τύπος χρησιμοποιεί ένα IR-LED και μια IR-diode (ή ένα φωτοτρανζίστορ). Όταν ένα αντικείμενο είναι κοντά στον αισθητήρα αντανακλάει το φως που εκπέμπεται από το LED στην IR-diode(**Εικόνα 8.4.1**). Η χρήση του διαμορφωμένου IR-light (A) και το φιλτράρισμα του σήματος που εισάγεται από το IR-light στο (B) οδηγεί σε καλύτερα αποτελέσματα καθώς αυτό εμποδίζει τα άλλα IR-light. Υπάρχουν ICs που περιέχουν και διαμορφωτή, και IR-diode και το κύκλωμα φιλτραρίσματος.

Αισθητήρας ανίχνευσης γραμμής

Παρόμοιες συνδεσμολογίες με τον αισθητήρα αντανάκλασης IR, στοχεύουν προς τα κάτω για να ανιχνεύσουν τις γραμμές στο έδαφος. Αυτός ο αισθητήρας χρησιμοποιεί τη διαφορά της αντανάκλασης μεταξύ ενός άσπρου φόντου και μιας μαύρης γραμμής.

Τα πιο προηγμένα ρομπότ χρησιμοποιούν τους πολλαπλούς αισθητήρες ανίχνευσης γραμμών για να ακολουθήσουν μια γραμμή πιο καθαρά.

Ultrasonic

Οι αισθητήρες απόστασης Ultrasonic χρησιμοποιούν τον ήχο για να μετρήσουν την απόσταση. Αυτοί οι αισθητήρες εκπέμπουν μια υπερηχητική παλμική ακολουθία και στη συνέχεια μετρούν το χρόνο μέχρι να ανακλαστεί ο παλμός. Δεδομένου ότι η ταχύτητα του ήχου είναι σταθερή στον αέρα (περίπου 344m/s) είναι εύκολο να υπολογιστεί η απόσταση στο ανακλώμενο αντικείμενο. Αυτοί χρησιμοποιούνται για την απόσταση που κυμαίνεται από μερικά εκατοστόμετρα μέχρι μερικά μέτρα.

Laser

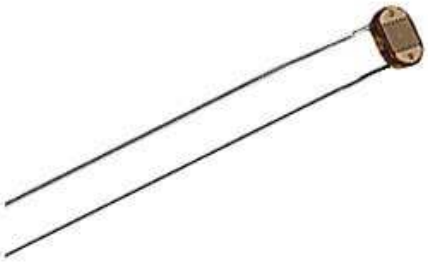
Οι αισθητήρες απόστασης που βασίζονται στα Laser χρησιμοποιούν την ίδια αρχή με τους υπερηχητικούς αισθητήρες απόστασης, αλλά χρησιμοποιούν το φως αντί του ήχου. Αυτοί οι αισθητήρες είναι χρήσιμοι μόνο για μεγαλύτερες αποστάσεις (τουλάχιστον μερικά μέτρα).

Bumper Switches

Εικόνα 8.4.2: Διακόπτες bumper

Οι διακόπτες είναι πιθανώς οι απλούστεροι αισθητήρες που χρησιμοποιούνται σε ένα ρομπότ. Μπορούν να ανιχνεύσουν τις συγκρούσεις, να περιορίσουν την απόσταση πορείας, να ανιχνεύσουν ακόμη και τις καθόδους(**Εικόνα 8.4.2**). Η χρήση τους έχει ως απαίτηση την υποστήριξη άλλων αισθητήρων που μπορούν να αποτύχουν. Οι κατασκευαστές πρόσθεσαν μήκος στον ήδη υπάρχον μοχλό. Άλλοι πάλι τοποθέτησαν ένα δακτύλιο σε έναν κυκλικό προφυλακτήρα γύρω από το ρομπότ. Οι ανιχνευτές πτώσης θα είχαν ανιχνευτές κάτω από το ρομπότ που θα γλιστρούσαν πάνω στο πάτωμα. Πολύ απλή λειτουργία που επιτρέπει την αξιόπιστη ανίχνευση, είναι αυτό που κάθε ρομπότ χρειάζεται.

CdS Cells

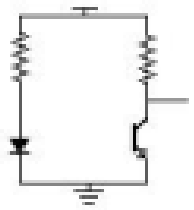


Εικόνα 8.4.3 : cds cells

Ένας άλλος απλός αισθητήρας για τα ρομπότ χρησιμοποιεί την ευαισθησία του σουλφιδίου καδμίου ή των κυττάρων CdS (**Εικόνα 8.4.3**).

Αυτοί αλλάζουν την αντίσταση βασισμένη στην ένταση του φωτός που ακτινοβολεί πάνω τους. Στο σκοτάδι, μπορούν να είναι 100s ή μερικά kΩ (kohms). Στο φως, μπορούν να είναι λιγότερο από 1k. Χρησιμοποιούνται σε έναν ανθεκτικό διαιρέτη μεταξύ 5 βολτ και του εδάφους, είτε ως κορυφή είτε ως κατώτατο σημείο με την έξοδο μεταξύ του κυττάρου CdS και του τρέχοντος αντιστάτη περιορισμού. Αυτό παρέχει ένα ιδιαίτερα μεταβλητό αναλογικό σήμα σε έναν ελεγκτή, το οποίο μπορεί να προκαλέσει ανώμαλη μετατροπή. Εάν ο ελεγκτής το υποστηρίζει, μια αναλογική είσοδος μπορεί να χρησιμοποιηθεί για να τους διαβάσει. Διαφορετικά ένα αναλογικό σε ψηφιακό μετατροπέα (ADC) ή μια συσκευή σύγκρισης μπορεί να κάνει την είσοδο πιο χρησιμοποιήσιμη. Αυτοί λειτουργούν καλά για την εύρεση των φωτεινών και σκοτεινών περιοχών ή στην ανίχνευση γραμμών στη πορεία που ακολουθούν τα ρομπότ.

Αισθητήρες Infra Red

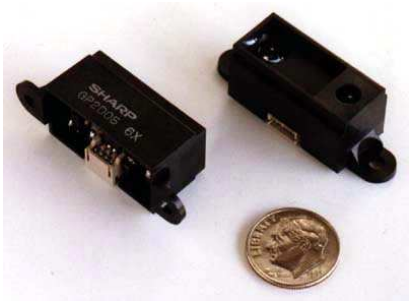


Εικόνα 8.4.4 : κύκλωμα αισθητήρα infra red

Τα κυκλώματα αισθητήρων InfraRed (**Εικόνα 8.4.4**)έχουν πολλά είδη. Η πιο απλή χρησιμοποιεί ένα IR LED για να διανείμει το φως και ένα ευαίσθητο φωτοτρανζίστορ IR για να το δει. Το φωτοτρανζίστορ ανοίγει όταν το φως IR το χτυπά, είτε αυτό προέρχεται από το LED είτε από περιβαλλοντικές πηγές είτε λόγω αντανάκλασης. Το κύκλωμα που παρουσιάζεται πρέπει να αποδίδει στην έξοδο οποιοδήποτε IR προσπίπτει σε αυτό. Οι χαρακτηριστικές τιμές για τον αντιστάτη που πηγαίνει στο συλλέκτη φωτοτρανζίστορ είναι περίπου 10KOhms. Τα LEDs συνήθως μεταβαίνουν από 200 σε 1Kohm στους αντιστάτες. Όταν κανένα φως IR δεν μπορεί να φανεί από τον ανιχνευτή, τότε θα εμφανίζεται μία καλή υψηλή τιμή (5 βολτ) από την έξοδο που συνδέεται με το συλλέκτη. Όταν το φως IR είναι παρόν, το φωτοτρανζίστορ δρα ως μέσο μεταφοράς και εμφανίζει μία καλή χαμηλή τιμή, γύρω στα 0,7 βολτ. Μια απλή χρήση περιλαμβάνει την ανίχνευση γραμμών με αισθητήρες σε κάθε πλευρά και σε μια γραμμή που ακολουθείται από το ρομπότ.

Η περιπλοκότερη χρήση IR περιλαμβάνει παλμικό φως IR σε μια γνωστή συχνότητα, και χρησιμοποιεί ένα ανιχνευτή που είναι προσαρμοσμένος σε αυτή την συχνότητα. Τα στοιχεία κυκλώματος μέσα στην σειρά Sharp GP1U58Y συντονίζουν τον ανιχνευτή σε μια συγκεκριμένη συχνότητα. Κάθε ένα είναι ευαίσθητο στο IR το οποίο έχει ρυθμιστεί σε παλμικό σε αυτή τη δεδομένη συχνότητας για μικρά ξεσπάσματα. Αυτή η κωδικοποίηση του IR βοηθά στην κάλυψη των εξωτερικών πηγών που μπορεί

να εμφανιστούν στην πορεία των πληροφοριών που χρειάζονται, ειδικά για το πόσο κοντά είναι κάτι ή για τη λήψη των κωδικοποιημένων στοιχείων. Αυτό που απαιτείται για την εγγύτητα της αναζήτησης είναι η πηγή IR που εξάγει φως σε διευκρινισμένη συχνότητα και εγκαίρως για αυτή τη συχνότητα να ταιριάξει τα χαρακτηριστικά των ανιχνευτών λήψης. Ποικίλες κατηγορίες κυκλωμάτων εργάζονται, από 555 χρονοδιακόπτες που οδηγούν τους IR LEDs στο να γαντζωθούν απλά από τον ελεγκτή με αυτούς σε σειρά και να προγραμματίσουν την κατάλληλη ακολουθία οδήγησης. Αυτό το κύκλωμα χρησιμοποιεί ένα PIC για να παραγάγει την κατάλληλη συχνότητα και τον παλμό για το αριστερό και το δεξί IRs, μετρά τον αριθμό των ανιχνεύσεων για κάθε πλευρά και τελικά εξάγει μία καλή ανίχνευση τόσο για την αριστερή όσο και για τη δεξιά πλευρά. Το να κάνεις όλο αυτό σε ένα ξεχωριστό επεξεργαστή ελευθερώνει τον ελεγκτή για άλλους στόχους στην οδήγηση ενός ρομπότ. Μπορεί να χρησιμοποιήσει τον ανιχνευτή Sharp, έναν ανιχνευτή Radio Shack LiteOn, ή μια σειρά Panasonic PNA4601M(**Εικόνα 8.4.5**). Ο πίνακας παρέχει άριστες δεξιές και αριστερές ανιχνεύσεις, και είναι συγκεχυμένος μόνο στα δωμάτια με τον υψηλό κυματοειδή (fluorescent) φωτισμό.



Εικόνα 8.4.5:ανιχνευτές sharp

Οι προηγμένοι υπέρυθροι αισθητήρες έρχονται σε πακέτα, όπως η σειρά Sharps GP2Dxx(**Εικόνα8.4.6**). Αυτοί οι αισθητήρες μπορούν να παρέχουν πληροφορίες κλίμακας, είτε με αναλογική είτε με ψηφιακή μορφή, είτε να είναι απλά ένα σημείο ώθησης όταν είναι κάτι μέσα σε μια διευθετήσιμη κλίμακα. Το GP2D02 δίνει μια ψηφιακή ανάγνωση 8 bit όταν προκαλείται. Τα GP2D05s δίνουν ένα υψηλό ή χαμηλό σήμα όταν μια συγκεκριμένη προετοιμασμένη κλίμακα επιτυγχάνεται. Το GP2D12s εξάγει μια αναλογική τάση που αντιστοιχεί στη συνεχή κλίμακα. Το GP2D15s εξάγει ένα υψηλό ή χαμηλό όταν ξεπερνά απόσταση 9,5’’.

Εικόνα 8.4.6 :υπέρυθροι αισθητήρες sharps GP2Dxx

Τα ζευγάρια των φωτοτρανζίστορ IRLED μπορούν να χρησιμοποιηθούν στους κωδικοποιητές. Κατάλληλα είναι τόσο ένα ζευγάρι αντανάκλασης το οποίο μπορεί να μετρήσει λωρίδες σε ένα τροχό όσο και ένας τύπος για να μετρήσει τις σχισμές σε ένα τροχό. Μερικοί διαφορετικοί τύποι υπάρχουν, συνήθως υπό μορφή κωδικοποιητών τετραγωνισμού. Οι κωδικοποιητές τετραγωνισμού χρησιμοποιούν 2 ζευγάρια αισθητήρων που τοποθετούνται σε ακολουθία το ένα με το άλλο για να επιτρέψουν την αναγνώριση της κατεύθυνσης μιας μηχανής που κινείται, καθώς επίσης και πόσο μακριά ή πόσο γρήγορα πηγαίνει. Μερικοί άνθρωποι δεν έχουν καταφέρει ούτε να τα χαράξουν μέσω του ποντικιού του υπολογιστή.

Μια περαιτέρω προηγμένη χρήση του IR είναι η βασική του χρήση στον τηλεχειρισμό. Πάλι η κατάλληλη ακολουθία απαιτείται, αλλά οι πληροφορίες μπορούν να διαβιβαστούν και να παραληφθούν, ακριβώς όπως όταν ανοίγει και κλείνει η τηλεόραση.

Sonar

Εικόνα 8.4.7: Διάφορα sonar

Το Sonar (**Εικόνα 8.4.7**)παρέχει μια άλλη μέθοδο όπου λέει στο ρομπότ που είναι τα πράγματα. Η σειρά των sonar της Polaroid είναι καλά στην παροχή της απόστασης ενός αντικειμένου μέσω ενός μετατροπέα και της αναμονής μιας επιστροφής ήχου. Ο ελεγκτής μετρά την περίοδο μεταξύ του αρχικού παλμού και της επιστροφής ήχου. Μέσω ενός γρήγορου υπολογισμού υπολογίζεται η απόσταση, βασισμένη στον ήχο που ταξιδεύει περίπου 1 ίντσα σε 74 μικροδευτερόλεπτα. Οι μονάδες της Polaroid μπορούν να κυμανθούν από περίπου 6 ίντσες ως 35 πόδια. Κάποιοι έχουν χρησιμοποιήσει μονάδες που λαμβάνονται από τις κάμερες και που δεσμεύονται για την παροχή ενός σήματος. Αυτό αποτελεί μια φτηνότερη λύση από την αγορά μιας πλήρως εξαρτημένης Polaroid. Μερικές sonar κάμερες (**Εικόνα 8.4.8**) απαιτούν λίγα χρήματα και μερικά πρόσθετα φτηνά μέρη για να λειτουργήσουν.

Εικόνα8.4.8 :κάμερα sonar

Μια νέα εναλλακτική λύση είναι η μονάδα sonar τοποθετημένη μαζί με Devantech . Η σειρά μονάδων Devantech κυμαίνεται από 1 ίντσα ως 10 πόδια. Η διεπαφή είναι σχετικά απλή για τους περισσότερους ελεγκτές.

Άλλοι αισθητήρες

Εικόνα 8.4.9: Είδος αισθητήρα

Πολλοί άλλοι αισθητήρες υπάρχουν για να βοηθήσουν ένα ρομπότ να δει που είναι τι ή ποιες είναι οι συνθήκες γύρω από αυτό (**Εικόνα 8.4.9**). Τα μικρόφωνα επιτρέπουν στο ρομπότ να ανιχνεύσει το θόρυβο ή τη μετακίνηση εδώ κοντά. Οι αντιστάτες αυτόματης ανίχνευσης δύναμης ή FSRs δείχνουν πόσο κάμπτεται κάποιο μέρος του ρομπότ ή το ποσό της πίεσης σε εκείνο το μέρος. Οι αισθητήρες υγρασίας και θερμοκρασίας δείχνουν τις συνθήκες της περιοχής που βρίσκεται ένα ρομπότ ή ακόμα και προβλήματα με τα εσωτερικά συστατικά. Υπάρχουν πολύ μικρά επιταχυνσιόμετρα τα οποία παράγουν στην έξοδο ένα σήμα PWM βασιζόμενα στο πόσο μεταβάλλεται η δύναμη που νιώθουν. Αυτοί θα μπορούσαν να χρησιμοποιηθούν για να πουν εάν ένα ρομπότ είναι σε μια κλίση ή έχει ανατραπεί ή εάν έχει χτυπήσει σε κάτι. Τα πιεζογυροσκόπια ανιχνεύουν απλά τις αλλαγές στον προσανατολισμό. Οι ηλεκτρονικές πυξίδες ανιχνεύουν τις αλλαγές του προσανατολισμού σε ένα μαγνητικό πεδίο, είτε είναι της γης είτε αυτό που παράγεται από μια μαγνητική συσκευή εδώ κοντά. Τα αποστασιόμετρα λέιζερ παρέχουν πληροφορίες απόστασης. Υπάρχουν πολλοί τύποι αισθητήρων για κάμερες και τηλεοράσεις που θα μπορούσαν να χρησιμοποιηθούν για να επιτρέψουν στο ρομπότ να δουν. Τα απλά τσιπ ανιχνευτών από τις φωτογραφικές μηχανές Gameboy έχουν διασυνδεθεί με τους προηγμένους ελεγκτές για την ανίχνευση γραμμών και την καθοδήγηση ενός ρομπότ. Έχουν επίσης χρησιμοποιηθεί οι δικτυακές κάμερες όπως οι γρήγορες κάμερες (quickcam). Το κυριότερο πρόβλημα είναι το να αναγνωρίσει ένα ρομπότ τι βλέπει. Οι αλγόριθμοι ανίχνευσης ακρών χρησιμοποιούνται για να καθορίσουν μια πορεία στην οποία θα κατευθυνθεί το ρομπότ ώστε να υπάρξει ανταπόκριση. Η χρήση 2 καμερών μπορεί να δώσει πληροφορίες για την απόστασης μέσω τριγωνισμού των παρεμβαλλόμενων δεδομένων. Η επεξεργασία της όρασης απαιτεί ουσιαστικές ικανότητες επεξεργασίας, λόγω της εμφάνισης όλων των δεδομένων. Υπάρχουν πολλοί αισθητήρες για να λειτουργήσει ένα ρομπότ, κι αυτό αποτελεί μία πραγματική πρόκληση.

Αισθητήρες εικόνας

Ένας αισθητήρας εικόνας είναι μια συσκευή που μετατρέπει μια οπτική εικόνα σε ένα ηλεκτρικό σήμα. Χρησιμοποιείται συνήθως στις ψηφιακές κάμερες και σε άλλες συσκευές απεικόνισης. Ένας αισθητήρας εικόνας είναι χαρακτηριστικά μια charge-coupled συσκευή (CCD) ή ένας συμπληρωματικός αισθητήρας ενεργού εικονοστοιχείου (active-pixel sensor) (CMOS).

8.5 CCD εναντίον CMOS

Σήμερα, οι περισσότερες ψηφιακές κάμερες χρησιμοποιούν είτε έναν αισθητήρα εικόνας CCD είτε ένα αισθητήρα CMOS. Και οι δύο τύποι αισθητήρων μετατρέπουν σε ηλεκτρικά σήματα το φως που λαμβάνουν.

Ένας CCD είναι μια αναλογική συσκευή. Όταν το φως πέφτει πάνω στο τσιπ κρατιέται ως ένα μικρό ηλεκτρικό φορτίο για κάθε αισθητήρα εικόνας. Οι φορτίσεις μετατρέπονται σε τάση ενός εικονοστοιχείου τη φορά καθώς διαβάζονται από το τσιπ. Τα πρόσθετα στοιχεία του κυκλώματος στην κάμερα μετατρέπουν την τάση σε ψηφιακές πληροφορίες.

Ένα τσιπ CMOS είναι ένας τύπος ενεργού αισθητήρα εικονοστοιχείου που κατασκευάζεται χρησιμοποιώντας τη διαδικασία ημιαγωγών CMOS. Τα πρόσθετα στοιχεία του κυκλώματος δίπλα σε κάθε αισθητήρα εικόνας μετατρέπουν την ενέργεια του φωτός σε τάση. Τα πρόσθετα στοιχεία του κυκλώματος στο τσιπ μπορεί να συμπεριλαμβάνονται στη μετατροπή της τάσης σε ψηφιακά δεδομένα.

Καμία τεχνολογία δεν έχει ένα σαφές πλεονέκτημα στην ποιότητα εικόνας. Το CMOS μπορεί να εφαρμοστεί χρησιμοποιώντας λιγότερα συστατικά, λιγότερη δύναμη ή/και να παρέχει γρηγορότερη ανάγνωση από CCDs. Το CCD είναι μια πλήρως ανεπτυγμένη τεχνολογία και θεωρείται τόσο κατανοητή όσο και το CMOS.

Υπάρχουν πολλές παράμετροι που μπορούν να χρησιμοποιηθούν για να αξιολογήσουν την απόδοση ενός αισθητήρα εικόνας. Μεταξύ αυτών είναι και το δυναμικό πεδίο τους, η αναλογία σήματος και θορύβου, η ευαισθησία στο χαμηλό φως κ.α.. Για τους αισθητήρες των συγκρίσιμων τύπων, η αναλογία σήματος και θορύβου και το δυναμικό πεδίο βελτιώνονται καθώς αυξάνεται το μέγεθος.

Αισθητήρες χρώματος

Υπάρχουν διάφοροι κύριοι τύποι αισθητήρων έγχρωμης εικόνας, διαφέροντας ως προς τα μέσα των μηχανισμών διαχωρισμού χρωμάτων:

- **αισθητήρας Bayer**, χαμηλού κόστους και ο πιο κοινός. Χρησιμοποιεί μια σειρά φίλτρων χρώματος όπως ένα φίλτρο Bayer που περνά το κόκκινο, το πράσινο, ή το μπλε φως σε επιλεγμένα αισθητήρια ή εικονοστοιχεία, που διαμορφώνουν τα πλέγματα ως προς την

ευαισθησία τους στο κόκκινο, το πράσινο, και το μπλε. Η εικόνα παρεμβάλλεται στη συνέχεια, χρησιμοποιώντας έναν αλγόριθμο.

- **αισθητήρας FoveonX3**, χρησιμοποιεί μια σειρά στρώσεων αισθητήρων όπου κάθε εικονοστοιχείο περιέχει τρεις συσσωρευμένους αισθητήρες, ευαίσθητους στα μεμονωμένα χρώματα.
- αισθητήρας Backside Illuminated (BSI) CMOS, όπου το φως διασχίζει το τσιπ πυριτίου για να φθάσει στη φωτοευαίσθητη πλευρά. Για αυτό, οι μεταλλικές συνδέσεις των συσκευών δεν αποτελούν εμπόδιο για το φως, και η αποδοτικότητα είναι υψηλότερη. Η Sony έχει καινοτομήσει στην παραγωγή ενός τέτοιου αισθητήρα, ακολουθούμενη από την OmniVision
- **3CCD,** χρησιμοποιεί τρεις μεμονωμένους αισθητήρες εικόνας και ο χωρισμός του χρώματος γίνεται από ένα διχροϊκό πρίσμα. Εξετάζει την καλύτερη ποιότητα, και γενικά είναι ακριβότερος από τους αισθητήρες ccd.

Ειδικοί αισθητήρες

Οι ειδικοί αισθητήρες χρησιμοποιούνται για διάφορες εφαρμογές όπως είναι η θερμική απεικόνιση, η δημιουργία των πολυφασματικών εικόνων, των καμερών γάμμα, των σειρών αισθητήρων για τις ακτίνες X, και άλλων ιδιαίτερα ευαίσθητων σειρών για την αστρονομία.

8.6 Lego Mindstorms

Τα Lego Mindstorms είναι μια γραμμή παραγωγής της Lego που συνδυάζει προγραμματίσημα τούβλα με ηλεκτρικές μηχανές, αισθητήρες, τούβλα Lego, και τεχνικά κομμάτια Lego (όπως εργαλεία, άξονες, ακτίνες, και υδραυλικά μέρη)κατάλληλα για να χτίσει ο χρήστης ρομπότ και άλλα αυτοματοποιημένα ή διαλογικά συστήματα. Η πρώτη λιανική έκδοση Lego Mindstorms κυκλοφόρησε το 1998 και πωλήθηκε εμπορικά με την επωνυμία Robotics Invention System (RIS). Η τρέχουσα έκδοση κυκλοφόρησε το 2006 ως Lego Mindstorms NXT. Η αρχική Mindstorms Robotics Invention System περιείχε δύο μηχανές, δύο αισθητήρες αφής, και έναν ελαφρύ αισθητήρα. Η έκδοση NXT έχει τρεις σερβομηχανές και τέσσερις αισθητήρες για την αφή, το φως, τον ήχο, και την απόσταση. Τα Lego Mindstorms μπορούν να χρησιμοποιηθούν για να κατασκευαστεί ένα μοντέλο ενσωματωμένου συστήματος με ηλεκτρομηχανικά μέρη ελεγχόμενα από υπολογιστή . Πολλά είδη πραγματικών ενσωματωμένων συστημάτων, από ελεγκτές ανελκυστήρων έως βιομηχανικά ρομπότ, μπορούν να διαμορφωθούν χρησιμοποιώντας τα Mindstorms.

Τα Mindstorms kits πωλούνται επίσης και χρησιμοποιούνται ως εκπαιδευτικό εργαλείο, αρχικά μέσω μιας συνεργασίας μεταξύ της Lego και του MIT Media Laboratory. Η εκπαιδευτική έκδοση των προϊόντων καλείται Lego Mindstorms for Schools, και έρχεται με το βασισμένο σε γραφική διεπαφή λογισμικό προγραμματισμού ROBOLAB, που αναπτύχθηκε στο

Πανεπιστήμιο Tufts χρησιμοποιώντας ως μηχανή το National Instruments LabVIEW.

8.6.1 Robotics Invention System - RCX brick

Εικόνα8.6.1.1 : RCX brick

RCX Brick πρώτης γενιάς προγραμματιζόμενο από το χρήστη **(Εικόνα 8.6.1.1)**.

Η πρώτη γενιά LEGO Mindstorms χτίστηκε γύρω από το κομμάτι της LEGO με τη μορφή τούβλου γνωστό ως RCX. Περιέχει έναν μικροελεγκτή Renesas H8/300 ως εσωτερική ΚΜΕ του. Το τούβλο προγραμματίζεται με τη μεταφόρτωση ενός προγράμματος (που γράφεται σε μια από διάφορες διαθέσιμες γλώσσες προγραμματισμού) από ένα PC ή MAC στη RAM του τούβλου μέσω μιας ειδικής υπέρυθρης διεπαφής (IR).

Αφότου αρχίσει ο χρήστης ένα πρόγραμμα, μια δημιουργία RCX Mindstorms μπορεί να λειτουργήσει από μόνη της, ενεργώντας στα εσωτερικά και εξωτερικά ερεθίσματα σύμφωνα με τις προγραμματισμένες οδηγίες. Επίσης, δύο ή περισσότερα τούβλα RCX μπορούν να επικοινωνήσουν το ένα με το άλλο μέσω της διεπαφής IR, επιτρέποντας τη συνεργασία ή τον ανταγωνισμό μεταξύ των τούβλων. Εκτός από τη θύρα IR, υπάρχουν τρεις θύρες εισαγωγής αισθητήρων και τρεις θύρες σύνδεσης μηχανών (επίσης χρησιμοποιήσιμοι για τους λαμπτήρες, κ.λπ.). Υπάρχει επίσης μία LCD που μπορεί να επιδείξει το επίπεδο φόρτισης των μπαταριών, την κατάσταση των θυρών εισόδου-εξόδου, το ποιο πρόγραμμα εκτελείται, καθώς και άλλες πληροφορίες. Τα τούβλα RCX έκδοσης 1.0 διαθέτουν μία παροχή ρεύματος για να επιτρέπουν τη συνεχή λειτουργία αντί της λειτουργίας περιορισμένου χρόνου κατά τη χρησιμοποίηση μπαταριών. Στην έκδοση RCX 2.0, η παροχή ρεύματος αναιρέθηκε. Τα τούβλα RCX με παροχή ρεύματος είναι δημοφιλή για τα στατικά προγράμματα ρομποτικής (όπως τα ρομπότ βραχίονες) ή για τα πρότυπα μοντέλα τρένων Lego.

Κατασκευή του RCX

Το RCX (Robotic Control X), είναι ένας αυτόνομος μικροελεγκτής που μπορεί να προγραμματιστεί χρησιμοποιώντας ένα PC. Το RCX χρησιμεύει ως ο εγκέφαλος των εφευρέσεων LEGO MINDSTORMS. Χρησιμοποιεί τους αισθητήρες που μας παρέχουν τα Mindstorms ως είσοδο από το περιβάλλον του, επεξεργάζεται αυτά τα στοιχεία, και με βάση αυτά δίνει κίνηση στους κινητήρες.

Προγραμματίζοντας το RCX

Οι χρήστες χτίζουν αρχικά το ρομπότ τους χρησιμοποιώντας τα κομμάτια LEGO και το RCX. Κατόπιν δημιουργούν ένα πρόγραμμα της αρεσκείας τους χρησιμοποιώντας όποια διαθέσιμη γλώσσα θέλουν (Robolab, NQC ή LEJOS) και το φορτώνουν στο RCX χρησιμοποιώντας μια ειδική υπέρυθρη συσκευή αποστολής σημάτων. Η δημιουργία τους μπορεί πλέον να αλληλεπιδράσει με το περιβάλλον, πλήρως αυτόνομα. Η επικοινωνία γίνεται με τη βοήθεια του υπέρυθρου φωτός. Ένας υπέρυθρος αισθητήρας συνδέεται σε σειριακή θύρα ή σε θύρα USB. Μια ασύρματη σύνδεση με το ίδιο το RCX του επιτρέπει να κινηθεί ελεύθερα, ειδικά ως τμήμα της κίνησης των οχημάτων ρομπότ.

Η επικοινωνία μεταξύ PC και RCX

Το RCX έχει τρεις θύρες εισόδου για αισθητήρες (π.χ. αισθητήρα αφής ή αισθητήρα φωτός) και τρεις θύρες εξόδου (π.χ. για τους κινητήρες ή για τα λαμπάκια).

Η γλώσσα LEJOS

Το LEGO Java Operating System (LeJos) είναι μια γλώσσα προγραμματισμού βασισμένη σε JAVA που χρησιμοποιείται για να προγραμματίζουμε το RCX. Από τη στιγμή που η JAVA είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, έτσι και η LeJos μας προσφέρει όλα τα πλεονεκτήματα του αντικειμενοστραφούς προγραμματισμού. Το LeJOS αναπτύχθηκε εξολοκλήρου από τον Jose Solorzano, αλλά τώρα συντηρείται από τους Paul Andrews και Jürgen Stuber. Το γεγονός ότι το leJOS είναι ένα καινούργιο firmware για το RCX σημαίνει ότι μπορούμε να κάνουμε τα προγράμματα μας πιο εξελιγμένα από ότι ήταν δυνατό χρησιμοποιώντας είτε το προγραμματιστικό περιβάλλον της LEGO (Robolab) είτε το NQC (Not Quite C).

8.6.2 Lego Mindstorms NXT

Η ανανέωση των Ρομπότ

Με τη νέα γενιά NXT, , η Lego προχωρεί ένα βήμα πιο πέρα από την επανάσταση των «οικιακής κατασκευής» ρομπότ που η ίδια είχε ξεκινήσει πριν από οκτώ χρόνια με τα Lego Mindstorms. Πολύ πιο εύκολα και γρήγορα στην κατασκευή τους, τα κιτ Mindstorms NXT δίνουν τη δυνατότητα στους ερασιτέχνες λάτρεις της ρομποτικής κάθε ηλικίας να φτιάξουν και να προγραμματίσουν το δικό τους μίνι ρομπότ μέσα σε μόλις 30 λεπτά της ώρας. Η αναπροσαρμογή λογισμικού για το δημοφιλές σύστημα εφευρέσεων ρομποτικής LEGO MINDSTORMS NXT απελευθερώνεται. Η νέα έκδοση λογισμικού LEGO MINDSTORMS NXT1.1 τώρα παρέχει την υποστήριξη για Vista και Macintosh Windows. Με τη βελτιωμένη χρήση μνήμης του λογισμικού, το LEGO MINDSTORMS NXT περιλαμβάνει μικρότερα συνταγμένα προγράμματα και συμπιεσμένα αρχεία.

Το NXT βασίζεται στο επιτυχημένο Robotics System Invention της εταιρείας, το οποίο έχει βελτιωθεί με την πρόσθεση νέων τεχνολογιών και αισθητήρων αυξημένων ικανοτήτων. Το «τουβλάκι» NXT που αποτελεί τον εγκέφαλο του ρομπότ είναι ένας αυτόνομος μικροεπεξεργαστής των 32 bit (σε αντίθεση με τα 16 bit της πρώτης γενιάς), ο οποίος μπορεί να προγραμματιστεί μέσω ηλεκτρονικού υπολογιστή PC ή - άλλη καινοτομία - Mac. Αφού κατασκευάσει το ρομπότ του, ο χρήστης δημιουργεί ένα δικό του πρόγραμμα χρησιμοποιώντας ένα εύχρηστο αλλά πλούσιο σε χαρακτηριστικά λογισμικό LabVIEW, το οποίο έχει σχεδιαστεί από τη National Instruments

Λογισμικό LABVIEW

Το λογισμικό LEGO MINDSTORMS NXT τροφοδοτείται από το LabVIEW, ένα διαισθητικό γραφικό λογισμικό προγραμματισμού που χρησιμοποιείται από τους επιστήμονες και τους μηχανικούς παγκοσμίως στα καταναλωτικά προϊόντα σχεδίου, ελέγχου και δοκιμής και τα συστήματα όπως MP3 και DVD, τα κινητά τηλέφωνα, και τα συστήματα ασφάλειας αερόσακων οχημάτων. Οι εφαρμογές περιλαμβάνουν την εξερεύνηση ανιχνευτών της NASA στον Άρη, αλλά και τη δοκιμή του Microsoft Xbox.

Απεικόνιση του NXT – Ανάλυση των μερών που το αποτελούν

Το NXT είναι ο εγκέφαλος ενός ρομπότ MINDSTORMS®. Είναι ένα ευφυές, ελεγχόμενο από υπολογιστή τούβλο LEGO® που δίνει τη δυνατότητα σε ένα ρομπότ MINDSTORMS να ζωντανέψει και να εκτελέσει τις διαφορετικές διαδικασίες.

Θύρες μηχανών

Το NXT έχει τρεις θύρες παραγωγής για την ένωση των μηχανών – θύρες Α, Β και Γ θύρες αισθητήρων

Το NXT έχει τέσσερις εισαγμένες θύρες για την ένωση των αισθητήρων - θύρες 1, 2, 3 και 4.

Θύρα USB συνδέεται με ένα καλώδιο USB με το λιμένα USB και μεταφορτώνονται τα προγράμματα από τον υπολογιστή στο NXT (ή φορτώστε τα στοιχεία από το ρομπότ στον υπολογιστή σας). Χρησιμοποιείται επίσης η ασύρματη σύνδεση Bluetooth για το φόρτωμα και τη μεταφόρτωση.

Μεγάφωνο Ένα πρόγραμμα μπορεί να δημιουργηθεί με τους πραγματικούς ήχους και να ακούγονται όταν τρέχει το πρόγραμμα .

Αισθητήρας αφής

Ο αισθητήρας αφής δίνει στο ρομπότ μια αίσθηση αφής. Ο αισθητήρας αφής ανιχνεύει πότε πιέζεται από κάτι και πότε απελευθερώνεται πάλι.

Προτάσεις χρήσης ο αισθητήρας αφής χρησιμοποιείται για να κάνει το ρομπότ να πάρει τα πράγματα: ένας ρομποτικός βραχίονας που εξοπλίζεται με έναν αισθητήρα αφής ενημερώνει το ρομπότ εάν υπάρχει ή όχι κάτι στο βραχίονά του που αρπάζει. Εναλλακτικά ένας αισθητήρας αφής μπορεί να χρησιμοποιηθεί για να εφαρμόσει το ρομπότ μια εντολή. Παραδείγματος χάριν, με τη συμπίεση του αισθητήρα αφής το ρομπότ μπορεί να περπατήσει, να μιλήσει, να κλείσει μια πόρτα, ή να ανοίξει τη TV.

Αισθητήρας ήχου

Ο υγιής αισθητήρας μπορεί να ανιχνεύσει και τα δύο decibels [DB] και ρυθμισμένο decibel [DBA]. Decibel είναι μια μέτρηση της υγιούς πίεσης. DBA: στην ανίχνευση ρυθμισμένων decibels, η ευαισθησία του αισθητήρα προσαρμόζεται στην ευαισθησία του ανθρώπινου αυτιού. Με άλλα λόγια, αυτοί είναι οι ήχοι ότι τα αυτιά σας είναι σε θέση να ακούσουν. DB: στην ανίχνευση τυποποιημένων [χωρίς διόρθωση] decibels, όλοι οι ήχοι μετρούνται με την ίση ευαισθησία. Κατά συνέπεια, αυτοί οι ήχοι μπορούν να περιλάβουν μερικοί που είναι πάρα πολύ υψηλοί ή πάρα πολύ χαμηλοί για το ανθρώπινο αυτί για να ακούσουν. Ο υγιής αισθητήρας μπορεί να μετρήσει τα επίπεδα υγιούς πίεσης μέχρι 90 DB - για το επίπεδο ενός χαρτοκόπτη. Τα επίπεδα υγιούς πίεσης είναι εξαιρετικά περίπλοκα, έτσι οι υγιείς αναγνώσεις αισθητήρων στο MINDSTORMS NXT επιδεικνύονται σε ποσοστό [%]. Όσο χαμηλότερα τα τοις εκατό τόσο πιο ήρεμο το παράδειγμα soundFor: • 4-5% είναι όπως ένα σιωπηλό καθιστικό • 5-10% θα ήταν κάποιος που μιλά κάποια απόσταση μακριά • 10-30% είναι κανονική συνομιλία κοντά στον αισθητήρα ή τη μουσική που παίζεται σε κανονικό επίπεδο • 30-100% είναι να φωνάξει ή μουσική άνθρωποι που παίζεται σε μια μεγάλη ποσότητα

Αισθητήρας φωτός

Ο αισθητήρας φωτός είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ σας [ο υπερηχητικός αισθητήρας είναι άλλος]. Ο ελαφρύς αισθητήρας επιτρέπει στο ρομπότ σας να διακρίνει μεταξύ του φωτός και του

σκοταδιού. Μπορεί να διαβάσει την ελαφριά ένταση σε ένα δωμάτιο και να μετρήσει την ελαφριά ένταση των χρωματισμένων επιφανειών.

Προτάσεις χρήσης ένας ελαφρύς αισθητήρας μπορεί να χρησιμοποιηθεί για να μετατραπεί ένα ρομπότ σε συναγερμό διαρρηκτών: όταν ένας εισβολέας ανοίγει το φως στο δωμάτιο το ρομπότ μπορεί να αντιδράσει για να υπερασπίσει την ιδιοκτησία. Ένας ελαφρύς αισθητήρας μπορεί να χρησιμοποιηθεί για να κάνει ένα ρομπότ να ταξινομήσει πράγματα ανάλογα με το χρώμα.

Υπάρχει επίσης η δυνατότητα χρήσης του ελαφριού αισθητήρα για να διαβάσει το περιβαλλοντικό φως με τη μέτρηση του ελαφριού επιπέδου στις διαφορετικές θέσεις του δωματίου. Παραδείγματος χάριν, κρατώντας αρχικά τον αισθητήρα ενάντια στο παράθυρο και στη συνέχεια στο πλαίσιο ενός πίνακα.

Αισθητήρας Υπέρηχων

Ο αισθητήρας υπέρηχων είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ [ο αισθητήρας φωτός είναι άλλος]. Ο υπερηχητικός αισθητήρας επιτρέπει στο ρομπότ να δει και να ανιχνεύσει τα αντικείμενα. Μπορεί επίσης να χρησιμοποιηθεί για να αποφεύγει το ρομπότ τα εμπόδια, την απόσταση αίσθησης και μέτρου και να ανιχνεύει την μετακίνησης. Ο υπερηχητικός αισθητήρας μετρά την απόσταση στα εκατοστόμετρα και στις ίντσες. Είναι σε θέση να μετρήσει τις αποστάσεις από 0 έως 255 εκατοστόμετρα με μια ακρίβεια +/- 3 εκατ. Ο υπερηχητικός αισθητήρας χρησιμοποιεί την ίδια επιστημονική αρχή με τα ρόπαλα: μετρά την απόσταση με τον υπολογισμό του χρόνου που παίρνει ένα κύμα για να χτυπήσει ένα αντικείμενο και να επιστρέψει - ακριβώς όπως μια ηχώ. Τα μεγάλου μεγέθους αντικείμενα με τις σκληρές επιφάνειες επιστρέφουν τις καλύτερες αναγνώσεις. Τα αντικείμενα φτιαγμένα από μαλακό ύφασμα ή τα κυρτά [όπως μια σφαίρα] ή τα πολύ λεπτά ή μικρά μπορεί να είναι δύσκολα για τον αισθητήρα να ανιχνευθούν.

Σερβομηχανές

Οι τρεις σερβομηχανές δίνουν στο ρομπότ σας τη δυνατότητα να κινηθούν. Εάν χρησιμοποιήσετε το φραγμό κίνησης στο λογισμικό LEGO MINDSTORMS NXT για να προγραμματίσετε τις μηχανές σας, οι δύο μηχανές θα συγχρονίσουν αυτόματα, έτσι ώστε το ρομπότ να κινηθεί σε μια ευθεία γραμμή.

Αισθητήρας περιστροφής

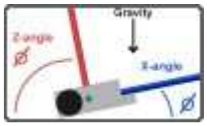
Κάθε μηχανή έχει έναν ενσωματωμένο αισθητήρα περιστροφής. Αυτό αφήνει τον έλεγχο των μετακινήσεων του ρομπότ ακριβώς. Ο αισθητήρας περιστροφής μετρά τις περιστροφές μηχανών στους βαθμούς ή τις πλήρεις περιστροφές [ακρίβεια +/- ενός βαθμού]. Μια περιστροφή είναι ίση με 360 βαθμούς, έτσι εάν τεθεί μια μηχανή στη στροφή 180 βαθμοί, ο άξονας παραγωγής του θα κάνει μισή στροφή. Ο ενσωματωμένος αισθητήρας περιστροφής σε κάθε μηχανή αφήνει στο χρήστη τη ρύθμιση των

διαφορετικών ταχυτήτων των μηχανών [με τον καθορισμό των διαφορετικών παραμέτρων δύναμης στο λογισμικό].

8.6.3 Λίστα αισθητήρων

Κατάλογος αισθητήρων ρομπότ, με τις αναλυτικές πληροφορίες και τις χρήσεις.

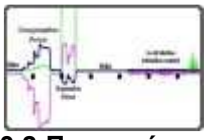
Accelerometer



Εικόνα 8.6.3.1:accelerometer

Ανιχνεύει την κίνηση, τη δόνηση, και τη γωνία όσον αφορά τη βαρύτητα**(Εικόνα 8.6.3.1)**.

Advanced Sensor Interpretation



Εικόνα 8.6.3.2:Προηγμένοι αισθητήρες

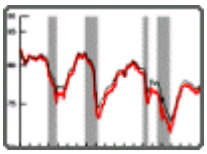
Αναπαριστά την μαθηματική ακρίβεια οποιουδήποτε αισθητήρα**(Εικόνα 8.6.3.2)**.

ColorSensors

Εικόνα 8.6.3.3:Αισθητήρας χρώματος

Δείχνει τον τρόπο που αντιλαμβάνεται τα χρώματα ένα ρομπότ**(Εικόνα 8.6.3.3)**.

Current Sensor



Εικόνα 8.6.3.4: Αισθητήρας ρεύματος

Με ένα αισθητήρα ρεύματος, μετράτε το ρεύμα και η χρησιμοποιούμενη ενέργεια με ακρίβεια**(Εικόνα 8.6.3.4)**.

Digital Compass

Εικόνα 8.6.3.5:Ψηφιακή πυξίδα

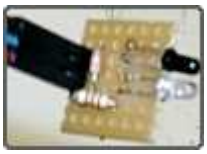
Ανιχνεύει τον προσανατολισμό όσον αφορά το γήινο μαγνητικό πεδίο**(Εικόνα 8.6.3.5).**

Encoder (Slot, Rotary, Linear)

Εικόνα 8.6.3.6: Encoder (Slot, Rotary, Linear)

Χρησιμοποιείται για τον καθορισμό της μεταδιδόμενης απόστασης, της περιστροφικής ταχύτητας και / ή της γωνίας ενός κινούμενου μέρους του ρομπότ**(Εικόνα 8.6.3.6)**.

Infrared Emitter/Detector



Εικόνα8.6.3.7 : Infrared Emitter/Detector

Εκπέμπει και ανιχνεύει το IR. Μπορεί να χρησιμοποιηθεί για τη σηματοδότηση, την αποφυγή εμποδίων, και την ανίχνευση χρώματος**(Εικόνα 8.6.3.7)**.

Load and Torque Sensors

Εικόνα8.6.3.8 : Αισθητήρες Load και Torque

Μετράει ροπές και άλλες δυνάμεις απευθείας από το σύστημα του ρομπότ**(Εικόνα 8.6.3.8)**.

Mercury Tilt Switch

Εικόνα 8.6.3.9: Διακόπτης Mercury tilt

Ένας διακόπτης δίνει μια δυαδική έξοδο χρησιμοποιώντας υδράργυρο, ανάλογα με τη γωνία όσον αφορά τη βαρύτητα**(Εικόνα 8.6.3.9)**.

Photoresistor

Εικόνα 8.6.3.9.α: Photoresistor

Αλλάζει την αντίσταση με το ανιχνευόμενο φως. Μπορούν να χρησιμοποιηθούν για τις απλές συμπεριφορές ενός ρομπότ, όπως η αποφυγή εμποδίων **(Εικόνα 8.6.3.9.α)**.

Robot Computer Vision

Εικόνα 8.6.3.9.β: Robot Computer Vision

Χρησιμοποιεί μια ψηφιακή τηλεοπτική κάμερα για επεξεργασία υψηλής ταχύτητας για την ανίχνευση του "πραγματικού κόσμου."**(Εικόνα 8.6.3.9.β)**

SharpIR Rangefinder

Εικόνα 8.6.3.9.γ: SharpIR Rangefinder

Ανιχνεύει μια σειρά εμποδίων από ίντσες μέχρι μερικά μέτρα**(Εικόνα 8.6.3.9.γ)**.

Sonar

Εικόνα 8.6.3.9.δ: Sonar

Ανιχνεύει τα εμπόδια και μπορεί να καθορίσει τη μαλακότητα ενός αντικειμένου / τη σκληρότητά του μέσω εντοπισμού με ήχο **(Εικόνα 8.6.3.9.δ)**.

Tactile Bumper Switch

Εικόνα 8.6.3.9.ε: Ο διακόπτης tactile bumper

Ανιχνεύει τη φυσική επαφή, όπως είναι η ανίχνευση της σύγκρουσης ενός ρομπότ**(Εικόνα 8.6.3.9.ε)** .

Κεφάλαιο 9

Παρουσίαση προγραμμάτων σε γλώσσα VPL του Microsoft Robotics Developer Studio

9.1 Εφαρμογή Text-to-Speech

Απαιτήσεις Υλικού

Η εφαρμογή αυτή δεν απαιτεί ειδικό υπολογιστικό υλικό

Λογισμικό

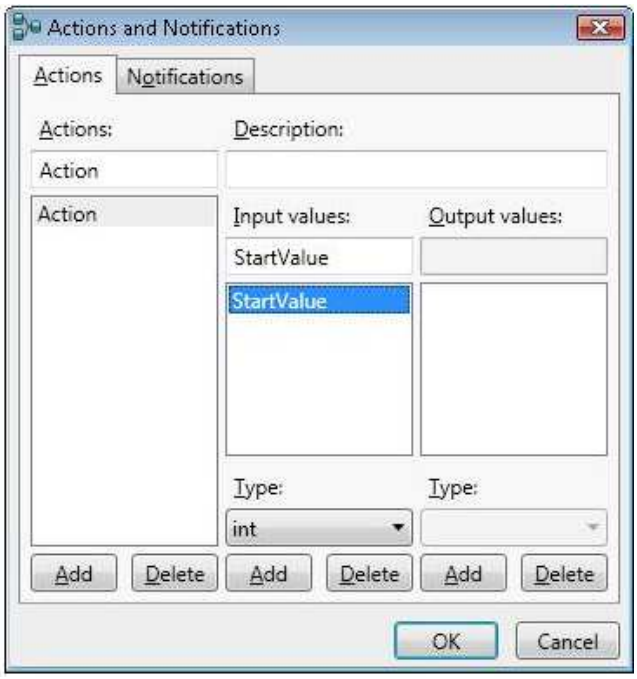
Η εφαρμογή αυτή σχεδιάστηκε για να χρησιμοποιηθεί με την Microsoft Visual Programming Language (VPL), η οποία συμπεριλαμβάνεται σαν μέρος του Microsoft Robotics Developer Studio

Δημιουργία και ορισμός των μεταβλητών

Αρχικά πρέπει να δημιουργηθεί ένα νέο project και στη συνέχεια να τοποθετηθεί μια δραστηριότητα **Data**. Η τιμή της δραστηριότητας τίθεται **ίσον με 1**. Για τη δημιουργία μιας νέας δραστηριότητας τοποθετείται το Activity block στο διάγραμμα. Με κλικ στο τίτλο του ο χρήστης του δίνει το όνομα **Count To10.** στη συνέχεια ανοίγει με διπλό κλικ. Μια νέα σελίδα θα εμφανιστεί.

Χρησιμοποιώντας την κυλιόμενη λίστα που βρίσκεται πάνω πάνω στην σελίδα, και κάνοντας κλικ στο Action, αυτό μεταπηδά στη σελίδα του action handler(χειριστή). Ενώ η σελίδα του διαγράμματος φαίνεται παρόμοια με την αρχική σελίδα ροής πληροφοριών, παρατηρήστε ότι έχει σύνορα σύνδεσης σε κάθε πλευρά όπου μπορεί να γίνει σύνδεση της εσωτερικής ροής πληροφοριών με μια εξωτερική ροή πληροφοριών. Χρησιμοποιώντας την εντολή **Action and Notifications** στο **Edit** μενού ή κάνοντας κλικ στο κουμπί δίπλα στην κυλιόμενη λίστα (drop-down list) του **Action** που βρίσκεται πάνω πάνω στη σελίδα, θα εμφανιστεί το παράθυρο του **Action and Notifications(Εικόνα 9.1.1)**

Στη συνέχεια από τη λίστα επιλέγεται το **Action,** στο οποίο εισάγεται μια τιμή. Αυτή είναι η τιμή έναρξης που θα χρησιμοποιηθεί αργότερα για τον υπολογισμό. Το όνομα της τιμής εισαγωγής αλλάζει σε **Start Value** και επιλέξτε το **INT** ως τύπος του. Με κλικ στο ok (εντάξει) ολοκληρώνεται η διαδικασία.



Εικόνα 9.1.1 :Παράθυρο του Actions and Notifications.

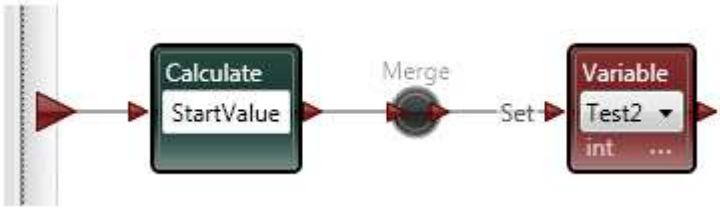
Τοποθέτηση των δραστηριοτήτων **Merge**, **If**, **Calculate**, **Variable**, και **Data**.

Αρχικά τοποθετείστε ένα **Calculate** block δραστηριότητας (**Εικόνα 9.1.2**) κοντά στα αριστερά σύνορα και συνδέστε με την ακίδα εισόδου του action. Πληκτρολογείστε **Start Value στο Calculate** textbox εισάγοντας έτσι την τιμή έναρξης από το μήνυμα εισόδου του action.

Στη συνέχεια τοποθετείστε ένα Merge block δίπλα στο **Calculate** block και συνδέστε με αυτό.

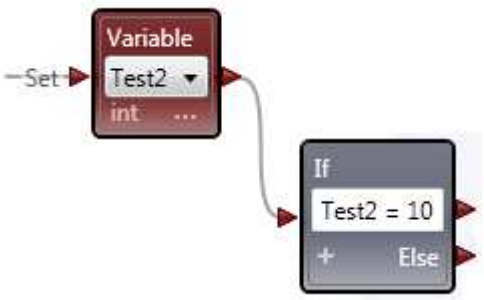
Το **Variable block** τοποθετείστε το δεξιά του Merge block. Χρησιμοποιώντας την εισερχόμενη τιμή που μπαίνει στη δραστηριότητα που καθοδηγείται μέσω του Merge block τίθεται η επιθυμητή μεταβλητή που θα χρησιμοποιηθεί.

Στη συνέχεια δημιουργείστε μια νέα μεταβλητή με όνομα **Test2**. Για να δημιουργηθεί μια μεταβλητή, εμφανίστε πρώτα τη κυλιόμενη λίστα του **Variable block** και κάντε κλικ **στο Define Variables.** Στο προκύπτον dialog box, εισάγετε το όνομα και το τύπο για τη μεταβλητή. Ο τύπος αυτός ορίζεται και **στο INT**. Η έξοδος του Merge block συνδέεται με το **Variable block**. Αυτό έχει ως αποτέλεσμα η αξία που έρχεται μέσω του Merge block να θέτει την αξία του **Variable block σε Test2**.



Εικόνα 9.1.2 :Τοποθέτηση των δραστηριοτήτων Calculate,Variable και Merge στο διάγραμμά μας.

Η έξοδος του **Variable** block συνδέεται στο **If** block όπως φαίνεται στην εικόνα που ακολουθεί(**Εικόνα 6.1.3**) .Στο text box του **If**, η τιμή του **Test2** ορίζεται ίση με 10 γράφοντας **Test2 = 10**. Αυτή η δοκιμή ελέγχει το όριο του αριθμού εκτέλεσης.



Εικόνα 9.1.3 :Τοποθέτηση της δραστηριότητας If στο διάγραμμά μας και σύνδεση με το Variable.

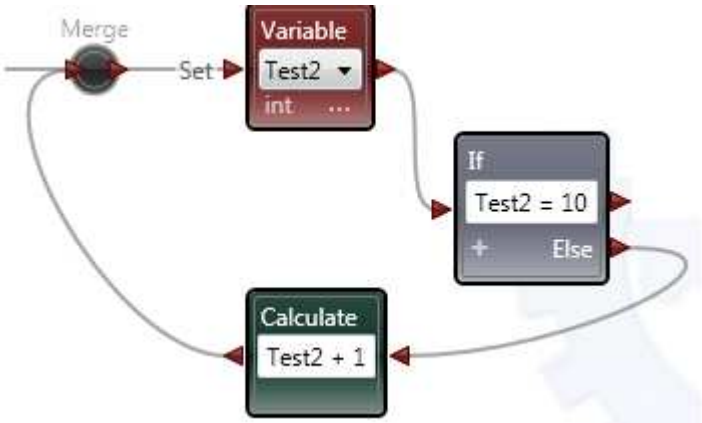
Δημιουργία μιας νέας **Calculate (Εικόνα 9.1.4)** δραστηριότητας.

Χρησιμοποιούνται οι υπερεμφανιζόμενες επιλογές πλαισίου του **Calculate** για να επιλεγούν οι συνδέσεις έτσι ώστε η είσοδος να εισάγεται στα δεξιά αντί στα αριστερά.

Σύνδεση της εξόδου της **Else** σύνδεσης της **If** δραστηριότητας με την είσοδο ενός **Calculate** block δραστηριότητας.

Στο **Calculate** text box, για να προστεθεί ένα στην τρέχουσα τιμή της **Test2** πληκτρολογήστε **Test2 + 1**. Στη συνέχεια η έξοδος του **Calculate** block της δραστηριότητας συνδέεται με την είσοδο του **Merge** block της δραστηριότητας. (Ένα **Merge** δέχεται τις πολλαπλάσιες εισερχόμενες συνδέσεις).

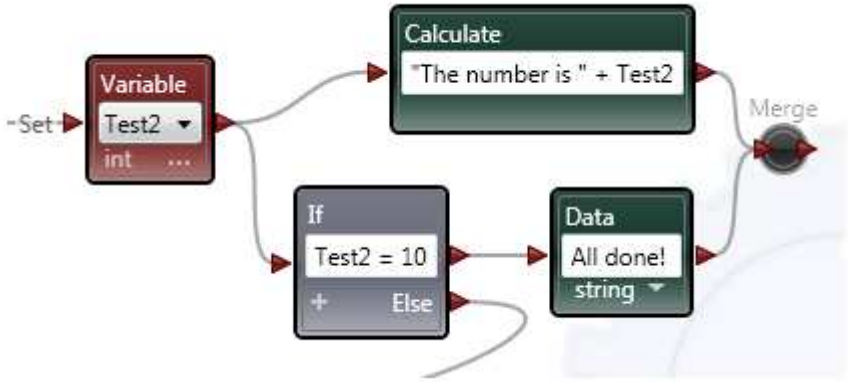
Αυτό ενημερώνει την τιμή της μεταβλητής όταν ρέει κατευθείαν το μήνυμα στη **Variable** δραστηριότητας. Παρατηρείστε ότι έτσι δημιουργείται ένας βρόχος.



Εικόνα 9.1.4 :Δημιουργία νέας δραστηριότητας Calculate και σύνδεσή της με την Merge.

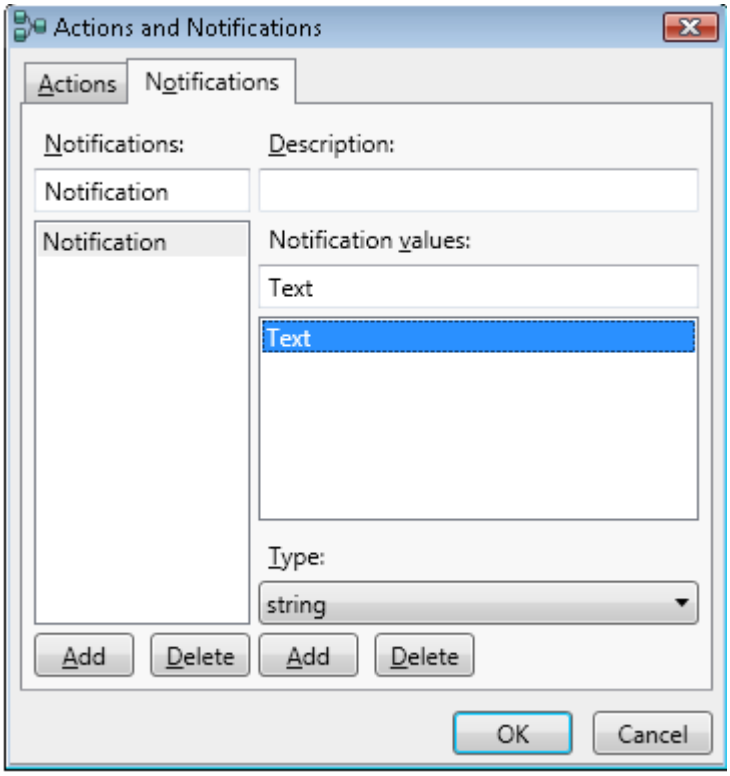
Σύνδεση ενός **Calculate** block της δραστηριότητας (**Εικόνα 9.1.5**) με

την έξοδο της **Variable** block δραστηριότητας. Πληκτρολογήστε **"The number is " + Test2** στο **Calculate** text box του block δραστηριότητας. Στη συνέχεια τοποθετείστε και συνδέστε ένα block δραστηριότητας **Data** στην πρώτη από πάνω έξοδο του **If** block της δραστηριότητας. Επιλέξτε το **string** από την κυλιόμενη λίστα, και πληκτρολογήστε **All done** στο text box. Χρησιμοποιείστε ένα δεύτερο **Merge** block δραστηριότητας για να συνδεθεί η έξοδος του **Calculate** block δραστηριότητας και του block δραστηριότητας **Data**. Αυτό απλοποιεί τη σύνδεση με άλλες ροές πληροφοριών έξω από αυτήν την δράση.



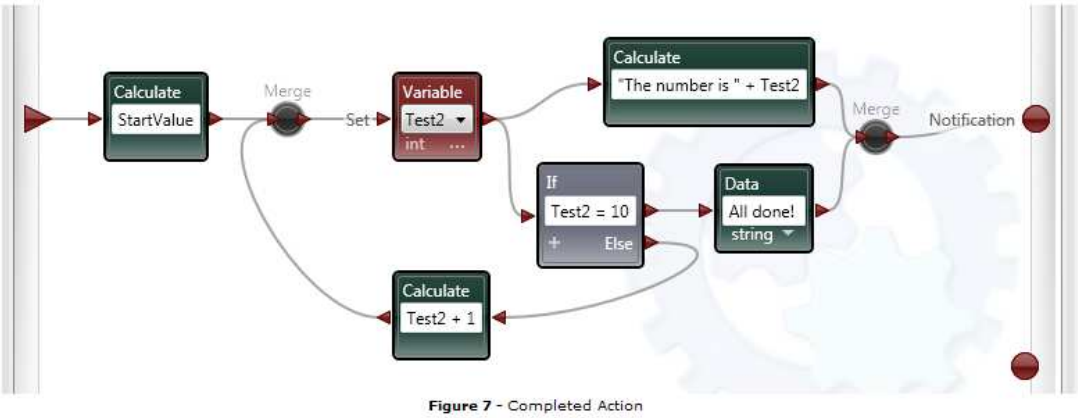
Εικόνα 9.1.5 :Δημιουργία νέας Calculate και σύνδεσή της με την Variable ,δημιουργία ενός Data και σύνδεση με το If,δημιουργία Merge και σύνδεση με Calculate και Data .

Για να συνδεθεί η έξοδος της ροής πληροφοριών, επιλέξτε την εντολή **Actions and Notifications** από το **Edit** μενού. Στο dialog box, κάνετε κλικ στο **Notifications (Εικόνα 9.1.6)** και έπειτα στο **Add** κουμπί για να δημιουργηθεί μια νέα ανακοίνωση (Notification). Κατόπιν με κλικ στο **Add** κάτω από το **Notification values** δίνεται νέο όνομα στη νέα τιμή **Text.** Κάνοντας κλικ στο **Add** ορίζεται και ο τύπος ως **string**, και με OK ολοκληρώνεται η διαδικασία.



Εικόνα 9.1.6 : Παράθυρο του Actions and Notifications .

Μπορεί τώρα να συνδεθεί η έξοδος του δεύτερου **Merge** block δραστηριότητας με τη στρογγυλή ακίδα σύνδεσης **Notification** στα δεξιά σύνορα της σελίδας action (**Εικόνα 9.1.7**). Επιλέξτε φυσικά το **Notification** που δημιουργήθηκε προηγουμένως. Η ροή πληροφοριών της δραστηριότητας πρέπει να μοιάζει με την ακόλουθη **Εικόνα 9.1.7**.



Εικόνα 9.1.7 : Σύνοψη διαγράμματος.

Η επιστροφή στην κύρια σελίδα διαγραμμάτων γίνεται είτε με κλείσιμο της σελίδας δραστηριότητας είτε με κλικ στο **Diagram** tab. Επόμενο βήμα είναι η σύνδεση της εξόδου του **Data** block της δραστηριότητας με το νέο **Count To10** block (**Εικόνα 9.1.8**).

Για να δημιουργηθεί αντίγραφο της δραστηριότητας που δημιουργήθηκε γίνεται αντιγραφή και επικόλληση του **Count To10.**

Τέλος, παρεμβάλλεται ένα **text-to-Speech** block της δραστηριότητας και γίνεται σύνδεσή του με την στρογγυλή ακίδα ανακοίνωσης του block δραστηριότητας **Count To10**. Στις συνδέσεις From: πληκτρολογήστε **Notification** και στις To: **Say Text**. Τέλος ορίζεται η σύνδεση data από **το Tex σε Speech Text**.

Εικόνα 9.1.8 : Σύνδεση Data με CountTo10και TexttoSpeechTTS με ένα άλλο CountTo10.

Κατά την εκτέλεση του προγράμματος φαίνεται ότι η μέτρηση δεν γίνεται κανονικά. Μερικές φορές μπορεί οι αριθμοί να είναι εκτός σειράς και να επαναλαμβάνονται. Το πρώτο πρόβλημα συμβαίνει επειδή τα μηνύματα μεταφέρονται με διαφορετικές ταχύτητες και μερικές φορές μπερδεύονται μεταξύ τους. Το δεύτερο πρόβλημα συμβαίνει επειδή νέα μηνύματα στέλνονται στο Text-To-Speech πριν αυτό τελειώσει να λέει το προηγούμενο μήνυμα. Μπορεί να χρησιμοποιηθεί το **SayTextSynchronous** αντί του **Say Text** επειδή η συγχρονισμένη λειτουργία δε στέλνει πίσω μία ανταπόκριση μέχρι να τελειώσει η ομιλία. Όμως αυτό απαιτεί αλλαγές στο πρόγραμμα που πρέπει να περιμένει το Text-To-Speech να τελειώσει μετά από κάθε μήνυμα.

9.2 Ρομπότ σε πορεία τετραγώνου

Σε αυτή την εφαρμογή γράφεται ο κώδικας για να τεθεί σε κίνηση ένα ρομπότ σε σχήμα τετραγώνου χρησιμοποιώντας μερικές από τις προηγμένες λειτουργίες του Generic Differential Drive.

Απαιτήσεις Υλικού

Αυτή η εφαρμογή χρησιμοποιείται για το iRobot Create (πραγματικό ή simulation) αλλά και για οποιοδήποτε ρομπότ με μια διαφορική κίνηση.

Για να χρησιμοποιηθεί ένα διαφορετικό ρομπότ πρέπει να αλλάξει το manifest που χρησιμοποιείται από το Generic Differential Drive.

Λογισμικό

Η εφαρμογή αυτή σχεδιάστηκε για να χρησιμοποιηθεί με την Microsoft Visual Programming Language (VPL), η οποία συμπεριλαμβάνεται σαν μέρος του Microsoft Robotics Developer Studio

Βήμα 1: Έναρξη του ρομπότ

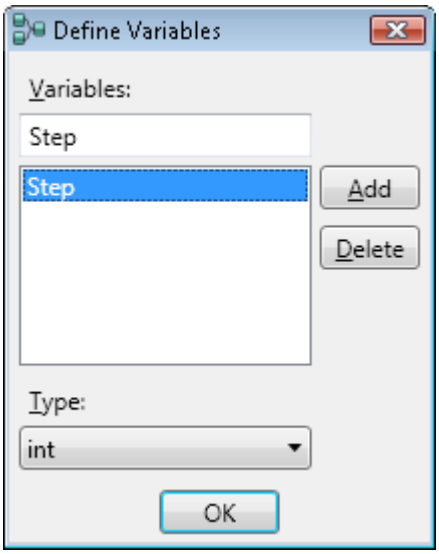
Αρχικά γίνεται άνοιγμα ενός νέου διαγράμματος VPL. Ο πρώτος στόχος που πρέπει να επιτευχθεί είναι η δημιουργία μιας μεταβλητής. Μια μεταβλητή είναι μια θέση στη μνήμη όπου αποθηκεύονται πληροφορίες και ενημερώνεται ενώ το πρόγραμμα εκτελείται.

Σε αυτό το πρόγραμμα, η μεταβλητή χρησιμοποιείται για να κρατήσει τη διαδρομή την οποία θα αποδώσει έπειτα το ρομπότ κινούμενο γύρω από ένα τετράγωνο.

Αυτό είναι ένα πολύ απλό παράδειγμα που καλείται *state machine* όπου η επικρατούσα κατάσταση κρατιέται σε μια μεταβλητή και το ρομπότ εκτελεί τις διάφορες διαδικασίες ανάλογα με την τιμή της.

Από το edit μενού, επιλέγονται οι μεταβλητές. Εμφανίζεται ένα dialog. Το Variable dialog που παρουσιάζεται στην **Εικόνα 9.2.1** δείχνει πώς πρέπει να είναι η εικόνα μετά από την ολοκλήρωση αυτών των βημάτων.

- Με κλικ στο κουμπί Add δημιουργείστε μια νέα μεταβλητή με default όνομα.
- Γίνεται εισαγωγή νέου ονόματος στη μεταβλητή, το Step.
- Επιλέγεται ο τύπος δεδομένων του ως INT (ένας ακέραιος αριθμός είναι ένας συνολικός αριθμός, δηλ. καμία δεκαδική θέση) επειδή είναι ένας μετρητής που θα αυξάνεται μετά την ολοκλήρωση του κάθε βήματος.



Εικόνα 9.2.1 : Παράθυρο του Define Variables.

Το όνομα της μεταβλητής Step απεικονίζει το γεγονός ότι το ρομπότ θα περάσει από μια σειρά βημάτων καθώς κινείται γύρω από το τετράγωνο. Είναι καλό να χρησιμοποιούνται ονόματα με νόημα για όλα τα αντικείμενα που δημιουργούνται σε ένα πρόγραμμα VPL.

Αφού υπάρχει η μεταβλητή, ακολουθεί σύρσιμο μιας δραστηριότητας Data στο διάγραμμα (**Εικόνα 9.2.2**). Πρέπει να έχει τιμή μηδέν (0) και τύπο INT. Ακολουθεί σύρσιμο μιας δραστηριότητας Variable (variable activity) στο διάγραμμα και επιλέγεται το Step από τη λίστα. Στη συνέχεια συνδέεται το block του Data με το Variable block όπως φαίνεται στο σχήμα. Αυτό αρχικοποιεί την τιμή της μεταβλητής Step σε μηδέν όταν αρχίζει το πρόγραμμα.

Το διάγραμμα περιμένει να πατηθεί OK όταν το ρομπότ είναι έτοιμο. Με σύρσιμο ενός SimpleDialog στο διάγραμμα και προσθήκη ενός data block εισάγεται σε αυτό ένα μήνυμα. Αυτό είναι σημαντικό για λόγους ασφάλειας με ένα πραγματικό ρομπότ, αλλά επίσης με αυτό τον τρόπο παρέχεται ο χρόνος που απαιτείται για να ελεγχθεί ότι το ρομπότ είναι συνδεδεμένο πριν το πρόγραμμα ξεκινήσει να του στέλνει εντολές. Ακόμη και στην προσομοίωση, παίρνει λίγο χρόνο στον προσομοιωτή να ξεκινήσει για αυτό οι εντολές δεν πρέπει να δοθούν αμέσως.

Το τελευταίο block σύνδεσης είναι ένα GenericDifferentialDrive το οποίο επιλέγεται με σύρσιμο και τοποθετείται στο διάγραμμα. Κατόπιν στις ιδιότητες (Properties) επιλέγεται η επιλογή διαμόρφωσης(configuration option) «Use a manifest», και κάνετε κλικ στο κουμπί import. Επιλέξτε κατάλληλο manifest για το ρομπότ. (Εάν χρησιμοποιηθεί το παρεχόμενο διάγραμμα, είναι ήδη τροποποιημένο για simulation).

Στη συνέχεια συνδέεται η έξοδος του SimpleDialog με την είσοδο του GenericDifferentialDrive και ορίζεται στο Data Connection το Distance (απόσταση) να είναι 0.5, το Power να είναι 0.5 και το DriveDistanceStage να είναι DriveStage.InitialRequest. Αυτό θα κάνει το ρομπότ να κινηθεί στο πρώτο μέρος της τετραγωνικής κίνησης του.

Εικόνα 9.2.2 : Διάγραμμα χρησιμοποιώντας τις δραστηριότητες Data,Variable,SimpleDialog και GenericDifferentialDrive.

Βήμα 2: Ειδοποιήσεις(ανακοινώσεις) ολοκλήρωσης

Εισάγεται άλλο ένα GenericDifferentialDrive στο διάγραμμα (**Εικόνα 9.2.3**). Ένας διάλογος (dialog) θα εμφανιστεί. Πρέπει να χρησιμοποιείται η υπάρχουσα υπηρεσία κι όχι να έχει δημιουργηθεί νέα.

Στη συνέχεια προστίθεται μια IF δραστηριότητα στο διάγραμμα. Γίνεται σύνδεση από την στρογγυλή ακίδα του GenericDifferentialDrive στο block του

IF. Επιλέγεται ο τύπος μηνυμάτων (message type) ως DriveDistance. Μέσα στο IF εισάγεται η έκφραση.

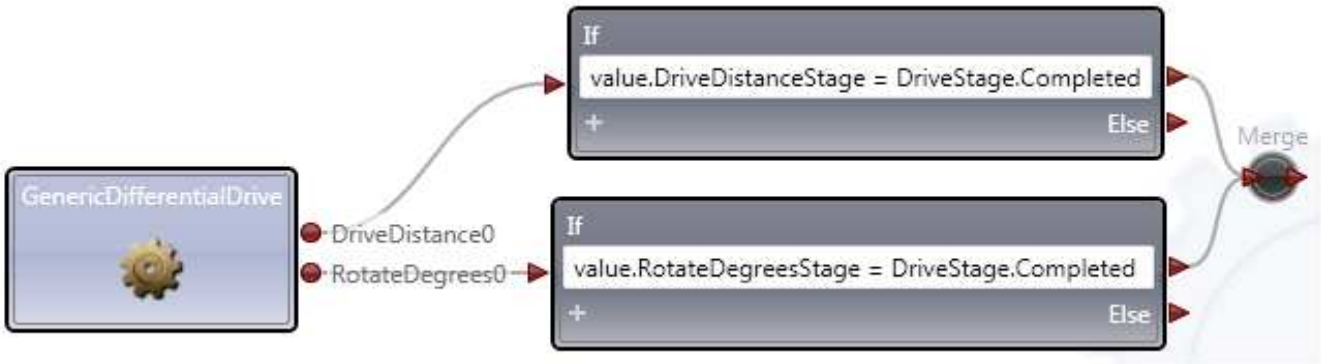
```
value.DriveDistanceStage = DriveStage.Completed
```

Αυτό ελέγχει εάν η κίνηση έχει ολοκληρώσει την DriveDistance λειτουργία. Εντούτοις, για να κινηθεί σε ένα τετράγωνο το ρομπότ πρέπει επίσης να περιστραφεί επί τόπου. Παρατηρείτε ότι η έξοδος του else δεν χρησιμοποιείται. Αυτό σημαίνει ότι οποιαδήποτε άλλη τιμή του DriveDistanceStage θα αγνοηθεί.

Προστίθεται άλλη μια IF στο διάγραμμα. Δημιουργείτε έτσι άλλη μια σύνδεση από την στρογγυλή ακίδα του IF με το GenericDifferentialDrive, μόνο που αυτή τη φορά ο τύπος μηνυμάτων (message type) επιλέγεται ως RotateDegrees. Μέσα στην IF εισάγετε την έκφραση:

```
value.RotateDegreesStage = DriveStage.Completed
```

Ακολουθεί συνδυασμός των εξόδων των δύο IF με την προσθήκη ενός merge block το οποίο συνδέεται με τα IF block. Ο κώδικας πρέπει μέχρι τώρα να έχει τη μορφή που παρουσιάζεται στην Εικόνα9.2.3 που ακολουθεί.

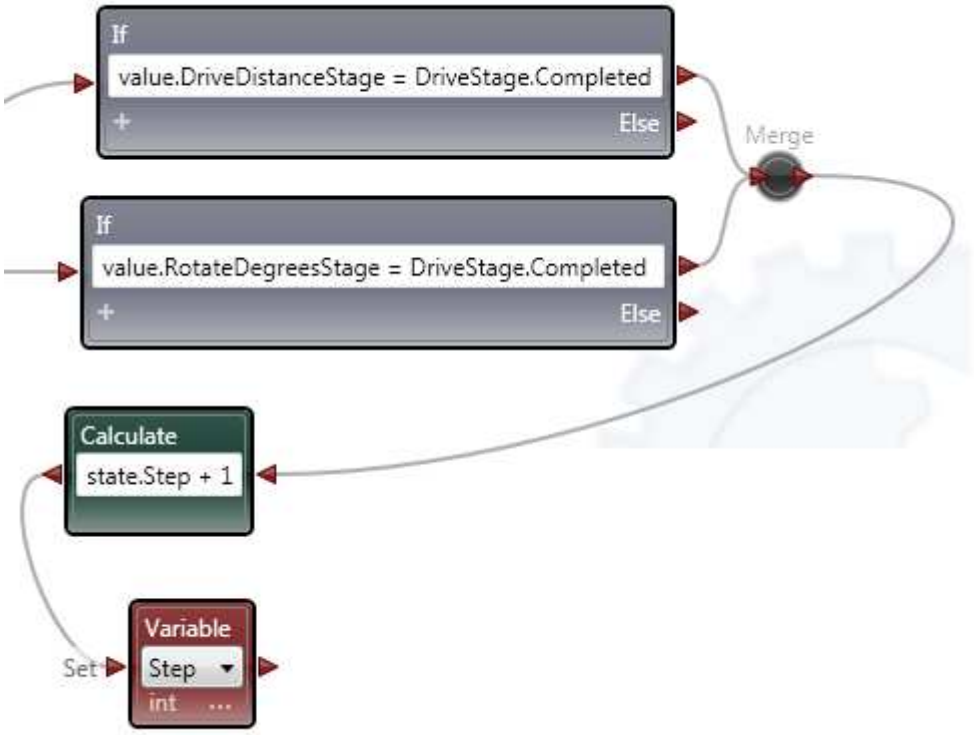


Εικόνα 9.2.3 :Προσθήκη στο διάγραμμα ενός GenericDifferentialDrive και σύνδεσή του με δύο νέες If ,σύνδεση ενός Merge με τις If.

Βήμα 3: Αύξηση του μετρητή βημάτων

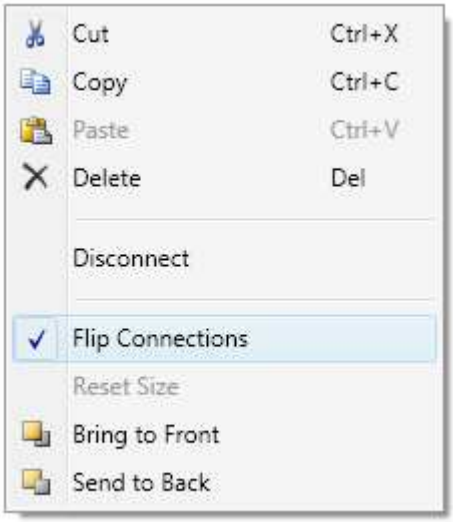
Κάθε φορά που φθάνει μια ανακοίνωση για να πει ότι μια λειτουργία DriveDistance ή RotateDegrees έχει ολοκληρωθεί, ένα μήνυμα στέλνεται από το merge. Αυτό χρησιμοποιείται για να αυξήσει την τιμή της μεταβλητής που

ονομάζεται Step όπως φαίνεται στην **Εικόνα 9.2.4** χρησιμοποιώντας μια δραστηριότητα Calculate και περνώντας ένα μήνυμα στη δραστηριότητα Variable.



Εικόνα 9.2.4 :Προσθήκη στο διάγραμμα ενός Calculate και μίας Variable που συνδέονται μεταξύ τους και η Calculate με την Merge .

Η δραστηριότητα Calculate στην παραπάνω εικόνα είναι έτσι ώστε το εισερχόμενο μήνυμα από τη merge να συνδέεται στη δεξιά πλευρά, όχι την αριστερή πλευρά. Ο προσανατολισμός του block αλλάζει χρησιμοποιώντας το μενού. Με δεξί κλικ στο Calculate block επιλέγεται το Flip Connections όπως φαίνεται στην **Εικόνα 9.2.5**. Υπάρχει δυνατότητα να φαίνεται συνεχώς ποια είναι η ακίδα εισόδου και ποια είναι η ακίδα εξόδου επειδή επιδεικνύονται ως βέλη που δείχνουν την κατεύθυνση της ροής μηνυμάτων. Αυτό δεν είναι ένα ουσιαστικό βήμα, αλλά είναι χρήσιμο για την απλούστευση των διαγραμμάτων.



Εικόνα 9.2.5 :Εμφάνιση μενού κάνοντας κλικ στο Calculate block και επιλογή του Flip Connections.

Το Step είναι μια στατική μεταβλητή για αυτό το πρόγραμμα. Ελέγχει ποια λειτουργία το ρομπότ εκτελεί έπειτα. Έτσι αυξάνοντας το Step αναγκάζει το ρομπότ να περάσει από μια ακολουθία διαδικασιών.

Βήμα 4: Εφαρμογή των βημάτων για το τετράγωνο

Αρχικά σύρετε μια δραστηριότητα Switch στο διάγραμμα (**Εικόνα 9.2.6**). Αυτό λειτουργεί σαν μια σειρά από IF δραστηριότητες. (Σε μερικές γλώσσες προγραμματισμού αυτό αναφέρεται ως δήλωση «περιπτώσεων» -- επιλέγει από μια από τις διαφορετικές περιπτώσεις). Στην εικόνα παρατηρείται ότι υπάρχουν επτά διαφορετικές τιμές, και ένα Else. Μπορεί να προστεθούν στις πρόσθετες εκφράσεις στο Switch με κλικ στο σύμβολο συν (+) στο κάτω-αριστερό μέρος του Switch.

Στη συνέχεια η έξοδος του calculate (η νέα αυξημένη τιμή του Step) συνδέεται με την είσοδο του Switch. Πρέπει τώρα να είναι προφανές πώς λειτουργεί. Η τιμή του Step έχει τεθεί σε μηδέν κατά τη διάρκεια της αρχικοποίησης. Κάθε φορά που φθάνει ένα μήνυμα ολοκλήρωσης, το Step αυξάνεται κατά ένα έτσι το Switch στέλνει ένα μήνυμα στην επόμενη έξοδο της ακολουθίας. Τελικά η τιμή φθάνει σε 8 και μπαίνουμε στο Else, το οποίο είναι το τέλος του προγράμματος.

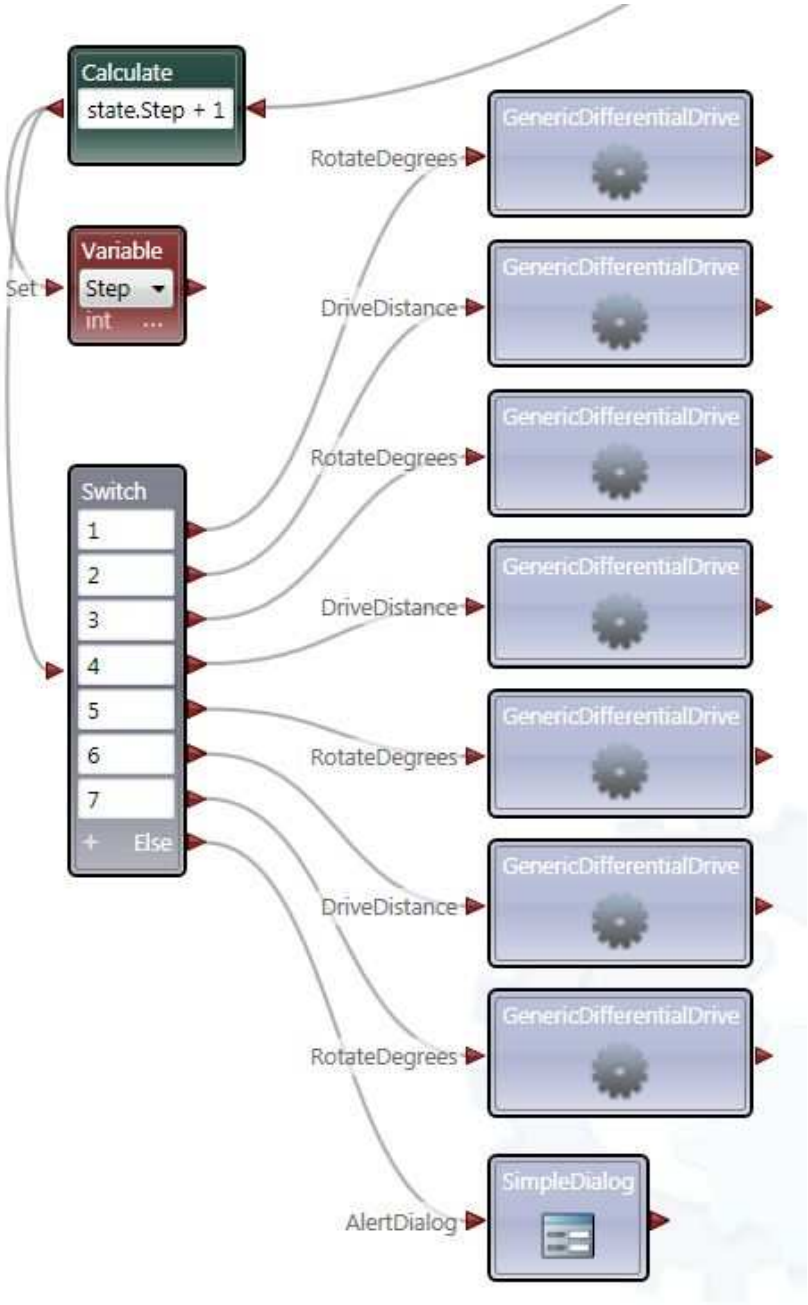
Τώρα θα χρειαστεί να συνδεθούν οι σωστές λειτουργίες drive (κίνησης) σε κάθε έξοδο του Switch, και ένα SimplyDialog στο Else για να εμφανισθεί ένα μήνυμα το οποίο θα λέει ότι το πρόγραμμα έχει τερματίσει.

Η διαδικασία είναι η ακόλουθη:

- Εισαγωγή ενός καινούριου GenericDifferentialDrive block
- Σύνδεση του καινούργιου GDD στην επόμενη έξοδο του Switch
- Επιλογή εναλλακτικά των RotateDegrees ή Drive Distance λειτουργιών όταν προστίθεται η λειτουργία κίνησης.

- Για τις RotateDegrees λειτουργίες ορίζεται η Degrees παράμετρος στο 90. Για κάθε μια από τις λειτουργίες **DriveDistance** ορίζεται η Distance παράμετρος στο 0,5 .

Τέλος, προστίθενται στο διάγραμμα ένα SimplyDialog και συνδέεται με την έξοδο του Else του Switch. Κάθε τιμή του Step μεγαλύτερη του 7 (η μικρότερη του 1) θα ενεργοποιήσει την έξοδο του Else. Όταν ολοκληρωθεί αυτό το μέρος του διαγράμματος θα πρέπει να φαίνεται όπως παρακάτω (**Εικόνα 9.2.6**).



Εικόνα 9.2.6 :Προσθήκη στο διάγραμμα μίας Switch και σύνδεση με την Calculate,σύνδεση εφτά GDD με την Switch και σύνδεση μίαςSipmleDialog με την Switch.

Βήμα 5 : Τρέξτε το πρόγραμμα

Το πρόγραμμα αποθηκεύεται και είναι έτοιμο να τρέξει. Θα πρέπει να πατηθεί το OK κουμπί στο διάλογο προειδοποίησης (Alert dialog) πριν το ρομπότ αρχίσει να κινείται.

Εάν χρησιμοποιείται ο προσομοιωτής, οι πιθανότητες το ρομπότ να σχηματίσει ένα τετράγωνο είναι πολύ καλές (αν και η ακρίβεια εξαρτάται ακόμα από τον τύπο του προσομοιωτή, του ρομπότ που έχει επιλεγεί και από το drive power). Ωστόσο, εάν χρησιμοποιείται αληθινό ρομπότ είναι πολύ πιθανόν το ρομπότ να μην κινηθεί σε τετράγωνο. Αυτό συμβαίνει επειδή οι Drivedistance και Rotatedegrees λειτουργίες δεν εγγυώνται ότι θα εκτελέσουν με ακρίβεια το έργο τους.

Λόγω της ανεκτικότητας στη βιομηχανία των drivers, τη διολίσθηση των τροχών και πολλών ακόμα παραγόντων είναι πολύ πιθανόν το ρομπότ να μην σχηματίσει ποτέ ένα τέλειο τετράγωνο. Είναι ένα πραγματικό πρόβλημα για το οποίο οι άνθρωποι που ασχολούνται με την ρομποτική ξοδεύουν πολύ χρόνο για να το διορθώσουν.

9.3 Κίνηση ρομπότ το οποίο ανιχνεύει τοίχο και αλλάζει πορεία .

Απαιτήσεις Υλικού

Η εφαρμογή αυτή δεν απαιτεί ειδικό υπολογιστικό υλικό.

Λογισμικό

Η εφαρμογή αυτή σχεδιάστηκε για να χρησιμοποιηθεί με την Microsoft Visual Programming Language (VPL), η οποία συμπεριλαμβάνεται σαν μέρος του Microsoft Robotics Developer Studio.

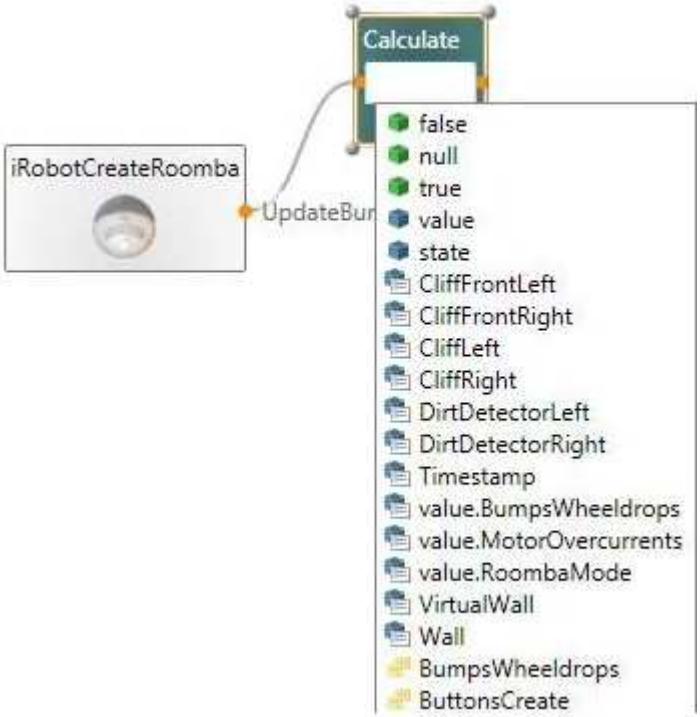
Βήμα 1: Προσθήκη ενός I Robot Create Lite

Το **I Robot** επισημαίνεται στις **υπηρεσίες** και προστίθεται ένα **IRobot Create Lite** στο διάγραμμα (**Εικόνα 9.3.1**). Προκειμένου να χρησιμοποιηθεί ο αισθητήρας του IRobot εντοπισμού ενός τοίχου απαιτείται είτε ένα **IRobot Create /Roomba** ή ένα **IRobot Create Lite** στο διάγραμμα.

Πώς λαμβάνεται το Wall Signal

Προστίθεται ένα **Calculate** box από το **Basic Activities Panel** και συνδέεται η στρογγυλή έξοδος του **IRobotCreateLite** με την είσοδο του **Calculate**. Υπενθυμίζεται στο σημείο αυτό ότι μια στρογγυλή έξοδος στη VPL χρησιμοποιείται για τις ανακοινώσεις. Με τη σύνδεση στη στρογγυλή έξοδο του **iRobotCreateLite**, μπορεί να επιλεγεί μια ανακοίνωση για να ληφθεί από την υπηρεσία. Όταν γίνει η σύνδεση, ένας κατάλογος θα ανοίξει. Στον κατάλογο φαίνονται όλες οι διαθέσιμες επιλογές. Για ένα update του *Wall*

Signal.. υπάρχει μια επιλογή του με όνομα **UpdateBumpsCliffsAndWalls**. Αφού επιλεχθεί, τοποθετείται ο δείκτης του ποντικιού στο **Calculate** box. Ένα κυλιόμενο μενού με τις διαθέσιμες επιλογές πρέπει να εμφανιστεί όπου το **Wall** update είναι διαθέσιμο.



Εικόνα 9.3.1 : Διάγραμμα με την χρήση μίας Calculate,ενός IRobotCreateLite.

Βάζοντας τον δείκτη του ποντικιού πάνω στη μεταβλητή **Wall** στο κυλιόμενο μενού, φαίνεται ότι η μεταβλητή είναι τύπου Boolean. Αυτό που αναζητείται είναι η μεταβλητή *Wall Signal* που είναι τύπου int. αυτό λαμβάνεται μέσω του **UpdateCliffDetails.**

Κάνοντας δεξί κλικ στη σύνδεση μεταξύ του IRobotCreateLite και του **Calculate** και επιλέγοντας **Connections** από το μενού που εμφανίζεται, μπορεί να επιλεγεί το **UpdateCliffDetails** αντί του **UpdateBumpsCliffsAndWalls**. Αν τώρα ο δείκτης του ποντικιού τοποθετηθεί στο **Calculate** box θα εμφανιστεί το **Wall Signal** τύπου **int** στον κυλιόμενο κατάλογο. Επιλέγοντας αυτή τη μεταβλητή, ή εισάγοντας το κείμενο **Wall Signal**. Το διάγραμμα θα πρέπει να είναι όπως φαίνεται στην **Εικόνα 9.3.2** που ακολουθεί.

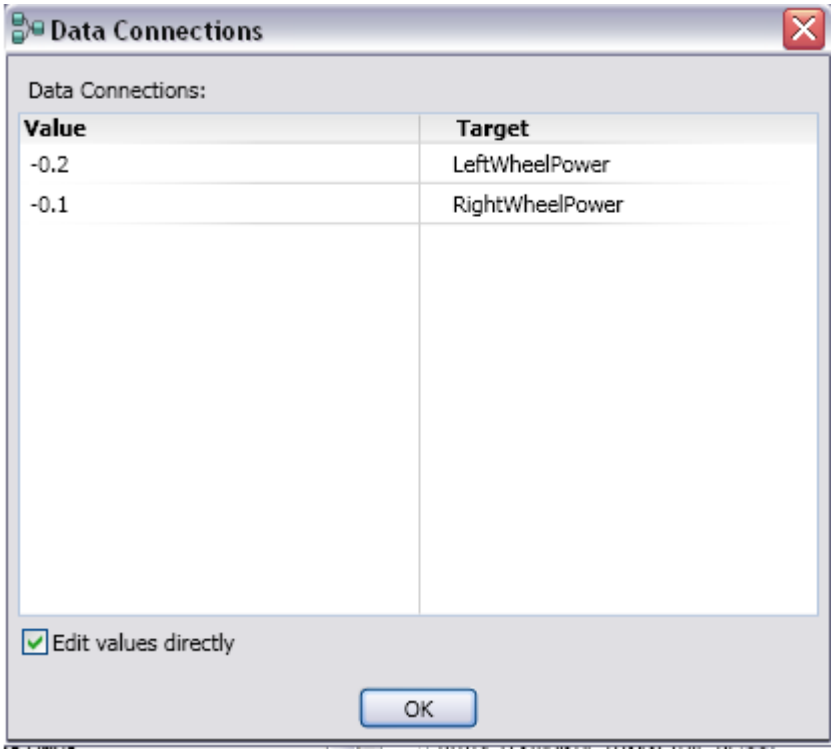
Εικόνα 9.3.2 : Διάγραμμα με την χρήση μίας Calculate και ενός IRobotCreateLite.

Στη συνέχεια επιλέγεται το Manifest και συνδέεται με το ρομπότ. Με διπλό κλικ στο IRobotCreateLife, εμφανίζεται νέο παράθυρο επιλογών όπου επιλέγεται από το Set Configuration το Use a manifest. Περισσότερες επιλογές εμφανίζονται με κλικ στο Import Manifest και μετά επιλέξτε το IRobot.CreateLiteSimulation στο TestWall SensorSim.Manifest.xml. Όταν το διάγραμμα τρέχει η IRobot υπηρεσία θα αρχίσει.

Βήμα 2: Η λογική τις αποφυγής του τοίχου

Από το Activities panel προστίθεται ένα If στο διάγραμμα του οποίου η είσοδος συνδέεται με την έξοδο του Calculate. Η If κατάσταση ορίζεται value > 5 ή κάποιον άλλο ακέραιο επιλογής προκαλώντας αξιολόγηση της κατάστασης ως true εάν το Wall Signal είναι μεγαλύτερο από το 5.

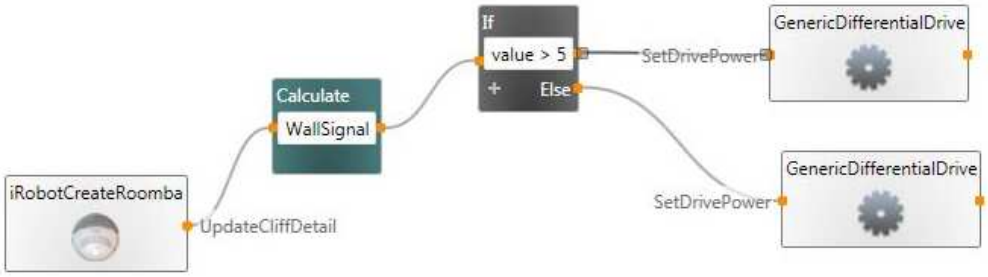
Από το Services panel προστίθεται ένα Generic Differential Drive το οποίο συνδέεται στο If στην κίνηση και επιλέγεται SetDrivePower. Στο Data Connections (**Εικόνα 9.3.3**), επιλέγεται το checkbox Edit values απευθείας στο κάτω αριστερό μέρος, και εισάγεται για το **the power for the left and right wheels** οι τιμές -0.2 και -0, 1 αντίστοιχα.



Εικόνα 9.3.3 : Το παράθυρο του Data Connections με τις ρυθμίσεις του.

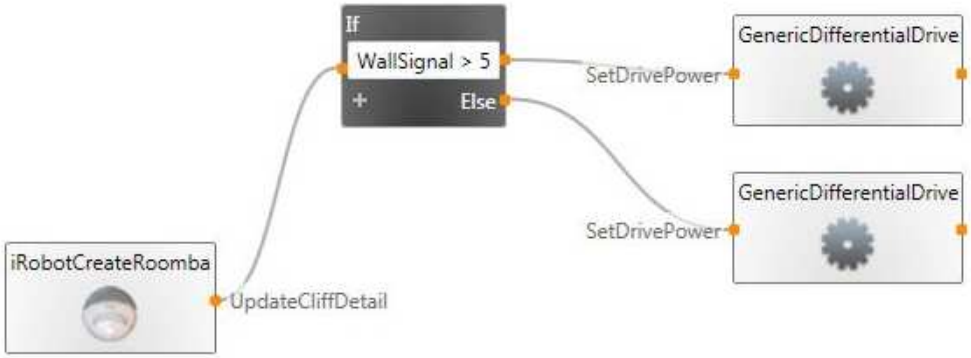
Σύμφωνα με το διάγραμμα το ρομπότ θα κινηθεί προς τα πίσω αν το **Wall Signal** είναι μεγαλύτερο από 5. Αν αυτό δεν ισχύει τότε το ρομπότ πρέπει να κινηθεί προς τα εμπρός. Για να χρησιμοποιηθεί το ίδιο differential drive όπως πριν (κι όχι μια νέα ανεξάρτητη υπηρεσία)επιλέγεται το **GenericDifferentialDrive** που είναι ήδη στο διάγραμμα (**Εικόνα 9.3.4**) και πατιέται κλικ στο **CTRL- c**,, και έπειτα στο **CTRL-V** για να κολλήσει ένα αντίγραφο. Επιλέγοντας τις ιδιότητες της αντιγραμμένης κίνησης στο κάτω δεξί μέρος υπάρχει ένα πεδίο ονόματος το οποίο παρατίθεται ως **GenericDifferentialDrive**. Εάν αλλάξει αυτό το όνομα σε κάτι άλλο, για παράδειγμα **My Drive**, και τα δύο boxes **GenericDifferentialDrive** θα ενημερωθούν.

Για να κινηθεί το ρομπότ προς τα εμπρός μένει να συνδεθεί το δεύτερο **GenericDifferentialDrive** με το else και να επιλεγεί το **SetDrivePower** και στο **Data Connections**, πάλι να επιλεγεί το **Edit values directly**, και να οριστεί το **power to the left και right wheels** στο **0,2** Το διάγραμμα θα πρέπει να έχει τη μορφή που παρουσιάζεται στην **Εικόνα 9.3.4** που ακολουθεί.



Εικόνα 9.3.4 : Το διάγραμμα με την προσθήκη μίας If και δυο GenericDifferentialDrive.

Το διάγραμμα αυτό απλοποιείται με αφαίρεση του **Calculate** box (**Εικόνα 9.3.5**). Για να γίνει αυτό με όσο το δυνατό λιγότερες αλλαγές στο τρέχον διάγραμμα, αρχικά κάνετε κλικ στη σύνδεση μεταξύ **Calculate** και **If**, και έπειτα στη διαγραφή. Στη συνέχεια επιλέγετε τη σύνδεση μεταξύ των δύο υπηρεσιών ώστε να αποσυνδεθεί το **Calculate** από το **IRobotCreateRoomba** και να κινηθεί εκείνο το τέλος της σύνδεσης προς το **If**. Σύροντας το τέλος που συνδέεται με το **Calculate** στο **If** συνδέεται το **IRobotCreateRoomba** με το **If** χωρίς να πρέπει να επιλεγεί ξανά το **UpdateCliffDetail**. Παρατηρείστε ότι έχει αλλάξει και το value της **If** σε **WallSignal**.



Εικόνα 9.3.5 : Το διάγραμμα με την αφαίρεση μίας Calculate και αλλαγή κάποιων ρυθμίσεων.

Για να οριστεί το αρχείο manifest για το **GenericDifferentialDrive** κάνετε διπλό κλικ σε ένα από τα boxes (δεν πειράζει αφού αναφέρονται στην ίδια κίνηση). Στη συνέχεια επιλέγεται το **Use a Manifest** και εισάγετε το **Simulated GenericDifferentialDriveTestWall SensorSim.Manifest .xml**.

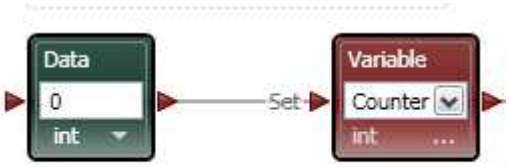
Χρήση του Manifest Editor για διαμόρφωση του Create για να λάβει τις ανακοινώσεις αισθητήρων.

Στην παράγραφο αυτή πρόκειται να αναλυθεί το πώς θα χρησιμοποιηθεί ο Manifest Editor ώστε να ληφθούν συγκεκριμένες ανακοινώσεις από το Create.

Στη VPL πριν από την εκτέλεση του προγράμματος επιλέγεται ένα **Manifest** για **το GenericDifferentialDrive**. Αυτό το **Manifest** είναι ένα αρχείο XML που παρέχει τις πληροφορίες για τις υπηρεσίες που είναι σχετικές για να λειτουργήσει η κίνηση. Ο **Manifest Editor** μπορεί να χρησιμοποιηθεί για να διαμορφώσει αυτές τις υπηρεσίες, να διευκρινίσει τις σχέσεις μεταξύ των υπηρεσιών, να προσθέσει τις νέες σχέσεις υπηρεσιών και ούτως καθεξής. Σε αυτό το βήμα παρουσιάζεται ο τρόπος που μπορεί να χρησιμοποιηθεί για να διαμορφώσει μια υπηρεσία.

Βήμα 3: Χρήση ενός timer για την αποφυγή μερικών ανακοινώσεων των αισθητήρων

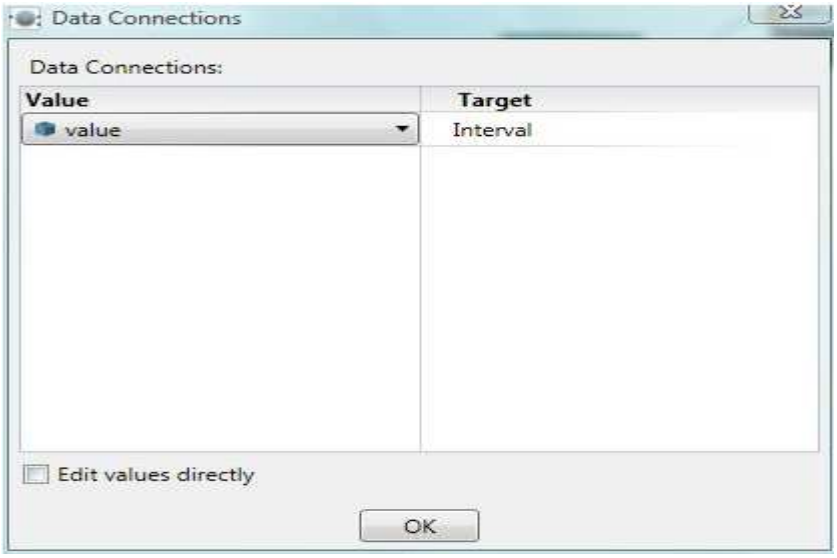
Στόχος είναι η δημιουργία ενός timer που θα λειτουργεί περιοδικά και θα αυξάνει έναν μετρητή. Όποτε υπάρχει ένας update αισθητήρας, ο μετρητής θα ελέγχεται. Εάν ο μετρητής είναι σε ένα κατώτατο όριο θα γίνεται reset στο μετρητή. Αλλιώς θα αγνοείτε το update του αισθητήρα. Αρχικά δημιουργείτε μια μεταβλητή του μετρητή και αρχικοποιείτε τη τιμή του στο μηδέν (**Εικόνα 9.3.6**).



Εικόνα 9.3.6 :Δημιουργία μετρητή με χρήση ενός Data και ενός Variable .

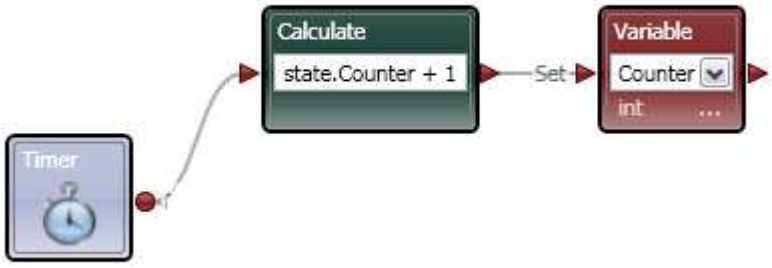
Στη συνέχεια πρέπει να αρχικοποιηθεί και ο timer. Από το **Services** σύρετε ένα **Timer,** ο οποίος πρέπει να λειτουργεί κάθε **50mms**. Για να γίνει αυτό προστίθεται ένα **Data** box με int **50** (**Εικόνα 9.3.7**). Ακολουθεί η σύνδεσή του χρησιμοποιώντας **Set Timer**, αφού πρώτα έχει επιλεγεί το **value** στο **Data Connections** (**Εικόνα 9.3.8**).

Εικόνα 9.3.7 : Δημιουργία ενός timer με τη χρήση ενός Data και ενός Timer.



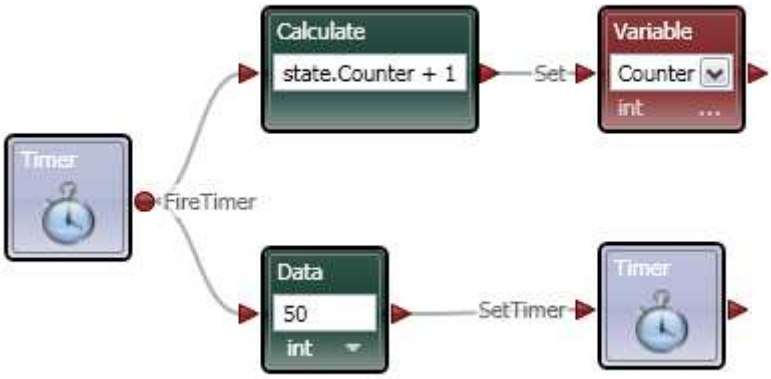
Εικόνα 9.3.8 : Παράθυρο Data Connections και επιλογή τιμής value.

Όταν ο timer ολοκληρωθεί, πρέπει να αυξηθεί η τιμή του Counter. Για να γίνει αυτό πρέπει να γίνει copy-and-paste του timer και να συνδεθεί σε αυτόν ένα **Calculate.** Αφού γίνει αυτό επιλέγεται το **Timer Complete**. Μέσα στο **Calculate** τοποθετείται η τιμή του Counter και προστίθεται **1** σε αυτό, πληκτρολογώντας **state.Counter + 1**. Το αποτέλεσμα του **Calculate** χρησιμοποιείται για να οριστεί ο **Counter(Εικόνα 9.3.9)**.



Εικόνα 9.3.9 : Σύνδεση ενός νέου Timer με μία Calculate και αυτή με ένα Variable .

Όταν ο timer ολοκληρωθεί, θα πρέπει επίσης να τεθεί ξανά σε λειτουργία. Αυτό επιτυγχάνεται με τη χρήση ενός **Data** block του οποίου η τιμή θα περαστεί στον **Timer** χρησιμοποιώντας το **Set Timer**. Ακολουθεί η σύνδεση του **Data** block στον **Timer Complete** όπως και προηγουμένως (**Εικόνα 9.3.9.α**).

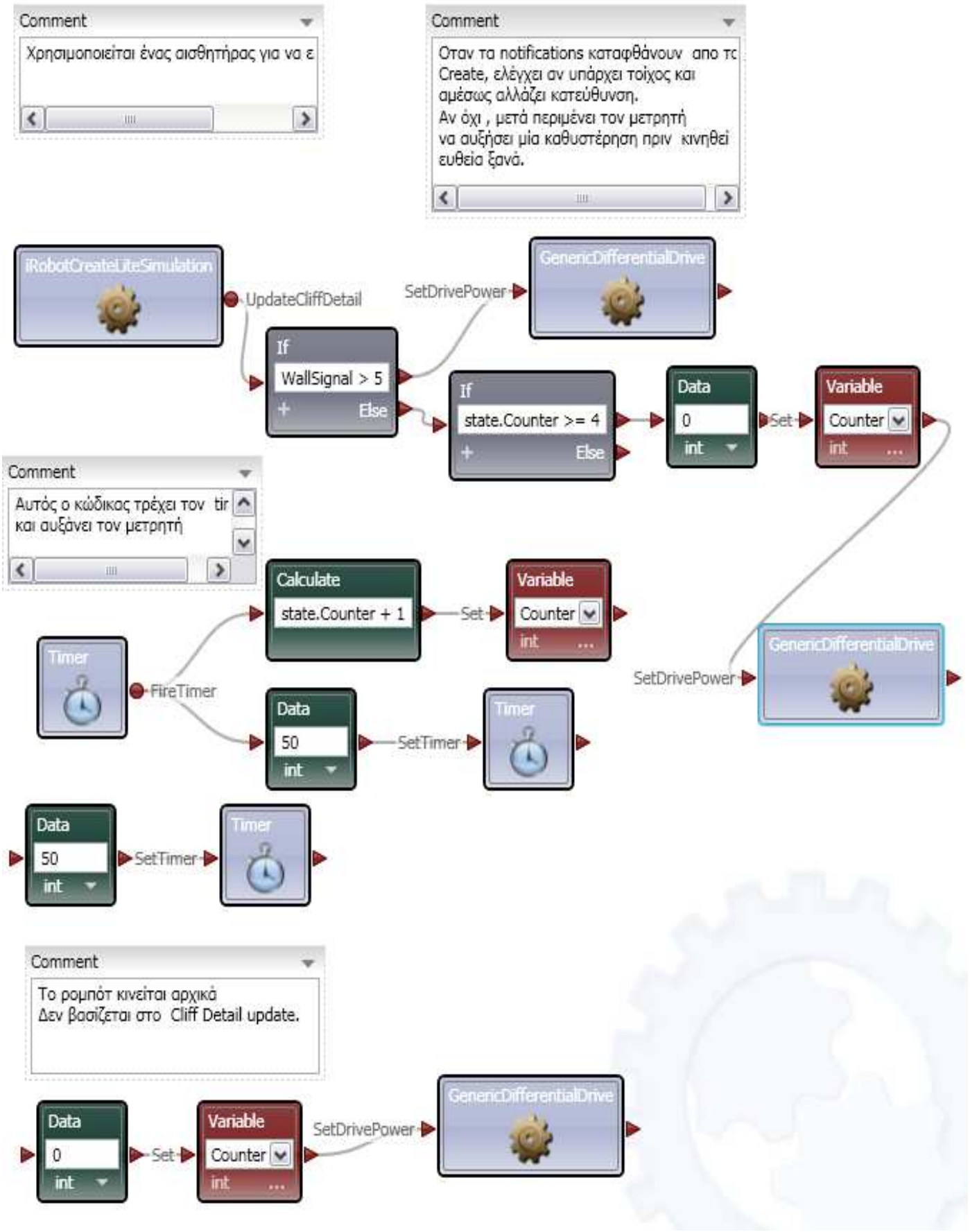


Εικόνα 9.3.9.α : Σύνδεση των δυο διαγραμμάτων της εικόνας 9.3.7 και 9.3.9 .

Στη συνέχεια πρέπει να χρησιμοποιηθεί ο **Counter** για να αποφασιστεί αν θα αγνοηθεί το update του αισθητήρα ή αν θα αξιολογηθεί το **WallSignal** και θα σταλεί η κατάλληλη εντολή στο **GenericDifferentialDrive**. Προστίθεται ένα **If** στο διάγραμμα (**Εικόνα 9.3.9.β**). Η κατάσταση πρέπει να οριστεί ως if **state.Counter >=4.** Αυτή η κατάσταση ορίζεται όταν η ανακοίνωση **UpdateCliffDetail** έχει ληφθεί. Χρησιμοποιείται η κατάσταση **>=** αντί **>** ώστε να καταναλώνεται λίγος χρόνος μέχρι το ρομπότ να λάβει την ανακοίνωση του αισθητήρα. Αν η κατάσταση βγει αληθινή, τότε πρέπει να αρχικοποιηθεί ο μετρητής.

Εικόνα 9.3.9.β : Πρόσθεση μίας If στο διάγραμμά μας και κάποιες ρυθμίσεις.

Αν η κατάσταση του μετρητή είναι αληθινή πρέπει επίσης να γίνει παρέμβαση και στην ανακοίνωση του αισθητήρα. Θα πρέπει η ροή να περάσει από αυτή την κατάσταση στο **If** για το **WallSignal.** Με σύνδεση του **If** για το **WallSignal** στην έξοδο του **If** για τον **Counter,** το **Counter If** θα υποδεχθεί το **UpdateCliffDetail** στο δεύτερο **If**. Όπως και να 'χει εξακολουθεί να παρέχεται πρόσβαση στο **WallSignal** όπως απαιτείται(**Εικόνα 9.3.9.γ**).



Εικόνα 9.3.9.γ : Το διάγραμμα ολοκληρωμένο.

9.4 Παιχνίδι Sumo

Οι υπηρεσίες Sumo Competition παρέχουν ένα περιβάλλον προσομοίωσης στο οποίο οι αγώνες sumo(**Εικόνα 9.4.1**) που φιλοξενούνται καθώς επίσης και μια υπηρεσία παίχτη(player) του sumo μπορούν να χρησιμοποιηθούν στο περιβάλλον προσομοίωσης και με φυσικά ρομπότ.

Εικόνα 9.4.1 : Ένας αγώνας Sumo με χρήση ρομπότ σε περιβάλλον προσομοίωσης .

Ο αγώνας sumo πραγματοποιείται σε ένα στρογγυλό ρινγκ. Το εσωτερικό μέρος του ρινγκ είναι μια μη ανακλαστική μαύρη επιφάνεια ενώ το εξωτερικό μέρος του ρινγκ είναι χρωματισμένο (κόκκινο) και ανακλαστικό. Δύο ρομπότ τοποθετούνται μέσα στο ρινγκ και πρέπει να παραμείνουν μέσα στο ρινγκ. Μια μάχη αποτελείται από τρεις γύρους που είναι κάθε ένας 60 δευτερόλεπτα διάρκεια. Ένας γύρος τελειώνει όταν πηγαίνει ένα ρομπότ έξω από το ρινγκ και ένας βαθμός απονέμεται στο άλλο ρομπότ. Ο βαθμός απονέμεται ακόμα κι αν το ρομπότ δεν έχει εκτοπιστεί από το ρινγκ. Το ρομπότ με τους περισσότερους βαθμούς στο τέλος της μάχης κερδίζει. Εάν ο γύρος τελειώνει με κανένα ρομπότ εκτοπισμένο από το ρινγκ, κανένας βαθμός δεν απονέμεται.

Τα ρομπότ έχουν μια διαφορική κίνηση(differential drive) που επιτρέπει σε αυτά να κινηθούν προς τα εμπρός ή προς τα πίσω σε οποιαδήποτε γωνία και επίσης να γυρίζουν σε μία θέση. Τα ρομπότ έχουν αριστερούς και δεξιούς αισθητήρες bumper για να έχουν την αίσθηση άλλων ρομπότ και αισθητήρες

IR στο κάτω μέρος ώστε να ανιχνεύουν μία αλλαγή στην IR ανακλαστικότητα κάτω από το ρομπότ. Η διαφορετική ανακλαστικότητα του εξωτερικού μέρους του ρινγκ επιτρέπει στα ρομπότ να αισθάνονται τη στιγμή όταν είναι έτοιμα να αφήσουν το ρινγκ.

Η υπηρεσία του διαιτητή Sumo

Η υπηρεσία διαιτητή επιτρέπει την επιλογή των υπηρεσιών των παικτών που συμμετέχουν στη μάχη (**Εικόνα 9.4.2**) και την εκτέλεση κάθε ενός από τους τρεις γύρους, που παρακολουθεί ταυτόχρονα το αποτέλεσμα και δηλώνει έναν νικητή στο τέλος.

Εικόνα 9.4.2 : Παράθυρο υπηρεσίας διαιτητή με τις ανάλογες επιλογές .

Όταν η υπηρεσία διαιτητή αρχίσει, παρουσιάζει μία κυλιόμενη λίστα διαθέσιμων παικτών. Στην αρχή θα υπάρχει ένας μόνο παίχτης στη λίστα(αποκαλούμενος SumoPlayer). Εάν έχει προστεθεί πρόσφατα ένας καινούριος παίχτης και δεν εμφανίζεται στην λίστα, επιλέγεται το refresh από την λίστα και ο διαιτητής θα κάνει refresh στο contact cache που απαριθμεί όλες τις διαθέσιμες υπηρεσίες. Μπορεί να επιλεγεί οποιοσδήποτε από τους αριθμημένους παίχτες ή μπορεί να επιλεγεί και ο none επιτρέποντας έτσι σε

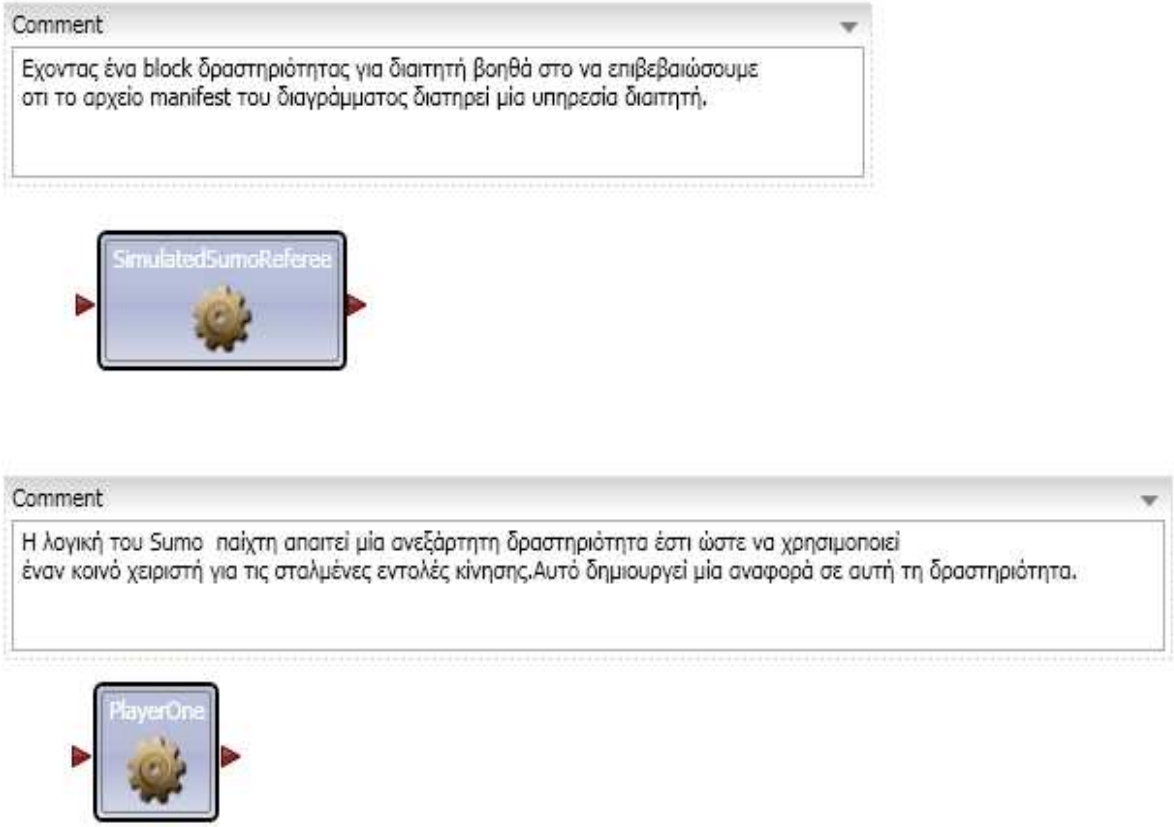
ένα ρομπότ να τρέξει μόνο του. Στην έναρξη κάθε γύρου, τα ρομπότ τοποθετούνται πίσω από τις καφετιές γραμμές στο ρινγκ με έναν τυχαίο προσανατολισμό. Όταν ο timer αρχίζει, η υπηρεσία διαιτητή ωθεί το play button σε κάθε simulated ρομπότ. Τα ρομπότ περιμένουν 5 δευτερόλεπτα πριν κινηθούν. Εάν ένα ρομπότ αποτύχει να περιμένει πέντε δευτερόλεπτα , θα χάσει τον γύρο.

Ένας γύρος τελειώνει όταν ένα από τα ρομπότ βγαίνει από το ρινγκ ή όταν λήγει ο χρόνος. Σε εκείνο το σημείο, η υπηρεσία διαιτητή ωθεί το κουμπί advance σε κάθε ρομπότ που επισημαίνει στην υπηρεσία παίχτη του sumo ότι ο γύρος έχει λήξει. Μετά από τρεις γύρους, ο διαιτητής θα αναγγείλει το νικητή και ένας νέος αγώνας μπορεί να αρχίσει με τους νέους αντιπάλους.

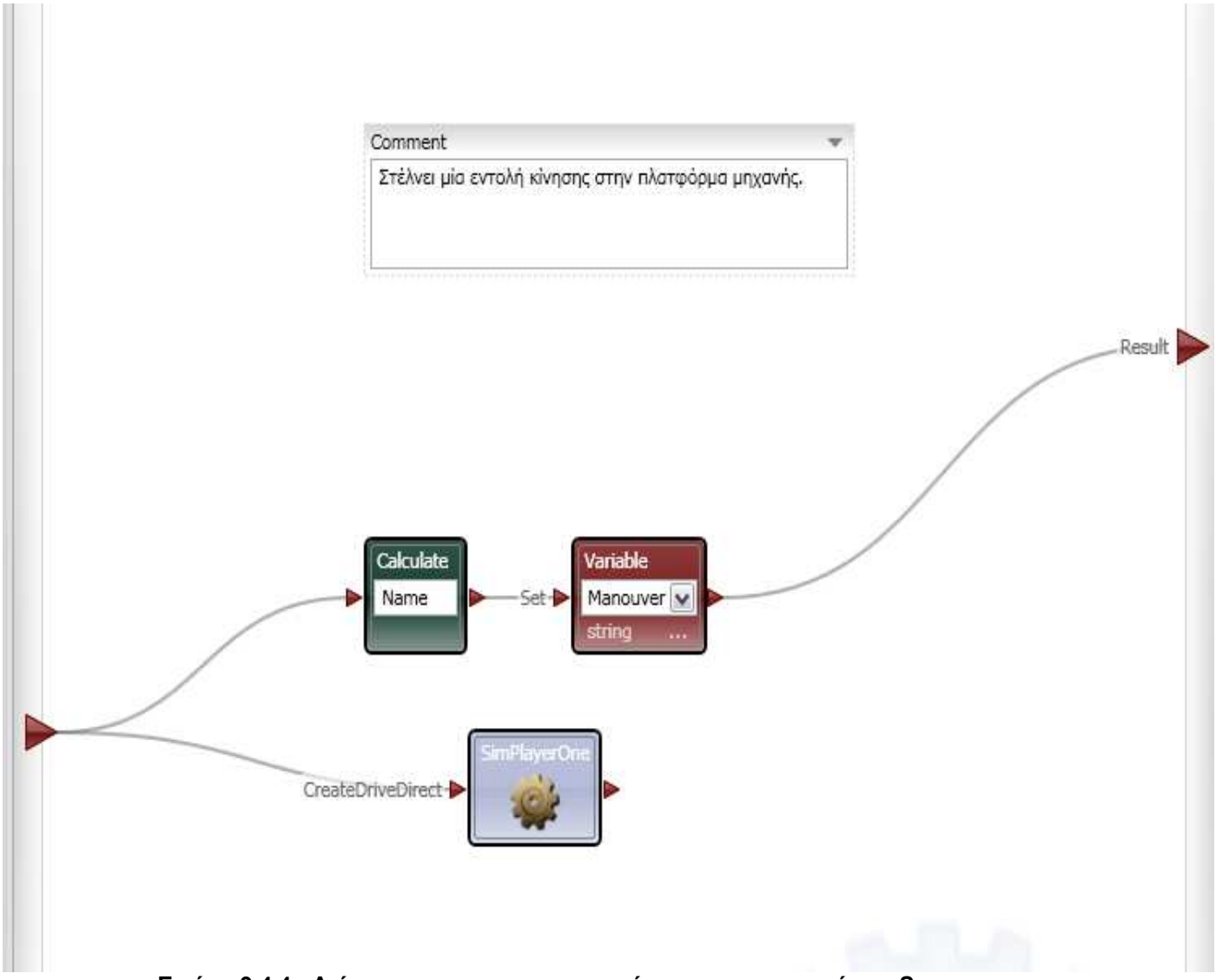
Η υπηρεσία του παίχτη του Sumo

Η υπηρεσία παίχτη του Sumo (SumoPlayer) παρέχει ένα παράδειγμα για το πώς μια υπηρεσία να ερμηνεύσει τα στοιχεία αισθητήρων που προέρχονται από το ρομπότ και να οδηγήσει το ρομπότ για να εφαρμόσει μια στρατηγική. Η υπηρεσία δίνει ένα παράδειγμα των απαραίτητων συστατικών που απαιτούνται για μια υπηρεσία παίχτη sumo αλλά δεν προορίζεται να είναι ένα παράδειγμα της καλύτερης δυνατής στρατηγικής sumo. Η πρώτη μέθοδος που καλείται στην υπηρεσία είναι έναρξη (). Καλεί ConnectWithRobot () που συνδέεται με την υπηρεσία υλικού iRobotLite σε μια φυσική συσκευή και με την υπηρεσία SimulatedIRobotLite στην περίπτωση προσομοίωσης. Επειδή η διεπαφή σε αυτές τις υπηρεσίες είναι η ίδια, ο ίδιος φορέας sumo μπορεί να χρησιμοποιηθεί στο περιβάλλον προσομοίωσης και στο φυσικό ρινγκ sumo. Η κύρια ρουτίνα εκτέλεσης στην υπηρεσία φορέων sumo είναι Timer Handler (). Αυτή η μέθοδος καλείται για 20 φορές ανά δευτερόλεπτο για να λάβει μια απόφαση για τη δράση που θα αποδώσει έπειτα. Εάν ένα πλαίσιο είναι διαθέσιμο από τη κάμερα, υποβάλλεται σε επεξεργασία και το αποτέλεσμα χρησιμοποιείται για να βοηθήσει να ληφθεί η απόφαση. Τα πλαίσια μπορούν να είναι διαθέσιμα μία φορά κάθε δευτερόλεπτο. Η υπηρεσία παίχτη του sumo προκαθορίζει την περιπλάνηση ώστε προσπαθεί να βρει τον αντίπαλό. Όταν ο αλγόριθμος επεξεργασίας όρασής του δείχνει ότι το άλλο ρομπότ έχει βρεθεί, μεταπηδά στην καταδίωξη του . Οποιαδήποτε στιγμή, ένας αισθητήρας στο ρομπότ μπορεί να αλλάξει τη κατάστασή του και ο αντίστοιχος χειριστής καλείται. Αυτό μπορεί να συμβεί όταν ένας bumper πιέζεται ή όταν ένας ή περισσότεροι από τους αισθητήρες IR στο κατώτατο σημείο του ρομπότ δείχνουν ότι η ανακλαστικότητα κάτω από το ρομπότ έχει αλλάξει. Το ρομπότ πρέπει συνήθως αμέσως να αλλάξει τον τρόπο του για να αποκριθεί σε αυτό το στοιχείο αισθητήρων. Η αποτυχία να αποκριθεί γρήγορα σε μια ανακοίνωση αισθητήρων IR, παραδείγματος χάριν, μπορεί να σημαίνει ότι το ρομπότ βγαίνει εκτός ρινγκ προτού να μπορέσει να αποκριθεί.

Παρακάτω ακολουθεί η παρουσίαση των δραστηριοτήτων που χρησιμοποιούνται στο Sumo και το διάγραμμα του.(**Εικόνα 9.4.3** και **Εικόνα 9.4.4**)



Εικόνα 9.4.3 : Δραστηριότητες που χρησιμοποιούνται για την εφαρμογή του Sumo .



Εικόνα 9.4.4 : Διάγραμμα που χρησιμοποιείται στην εφαρμογή του Sumo .

9.5 Κίνηση ρομπότ με joystick

Σε αυτή την εφαρμογή θα γραφτεί ο κώδικας οδήγησης ενός ρομπότ με τη χρήση ενός joystick. Στην αρχή παρουσιάζεται ο κώδικας σε γλώσσα VPL και στη συνέχεια ο κώδικας στη C#.

Απαιτήσεις Υλικού

Η εφαρμογή αυτή δεν απαιτεί ειδικό υπολογιστικό υλικό

Λογισμικό

Το πρώτο μέρος της εφαρμογής αυτής σχεδιάστηκε για να χρησιμοποιηθεί με την Microsoft Visual Programming Language (VPL), η οποία συμπεριλαμβάνεται σαν μέρος του Microsoft Robotics Developer Studio

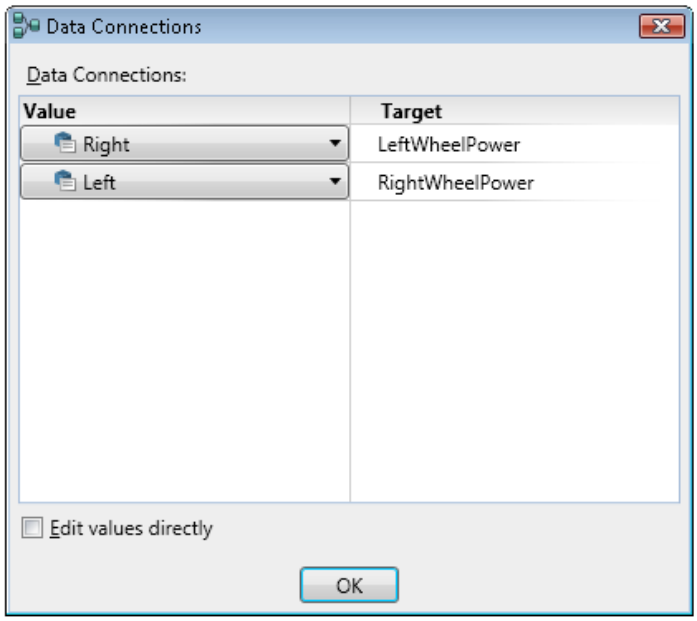
Το δεύτερο μέρος της εφαρμογής αυτής σχεδιάστηκε για να χρησιμοποιηθεί με την Microsoft Visual C# 2008 Express Edition (C sharp)

9.5.1 Ο Κώδικας σε γλώσσα VPL

Δημιουργία του διαγράμματος και εκτέλεση της προσομοίωσης

Αρχικά δημιουργείτε ένα νέο project με κλικ στο new από το file μενού. Στη συνέχεια τοποθετείται ένας **XInput controller** είτε με διπλό κλικ στο εικονίδιο είτε με σύρσιμο της υπηρεσίας από το services toolbox .

Αφού τοποθετηθεί μια υπηρεσία Generic Differential Drive, συνδέεται με την ακίδα εξόδου του **XInput controller**. Στο connection dialog box επιλέγεται στο from: **Triggers Change** και στο to: **SetDrivePower**. Στο **data connections (Εικόνα 9.5.1.α)** με κλικ στο right επιλέγεται το **LeftWheelPower** και με κλικ στο left επιλέγεται **το RightWheelPower**. Με κλικ στο ok ολοκληρώνεται η διαδικασία.

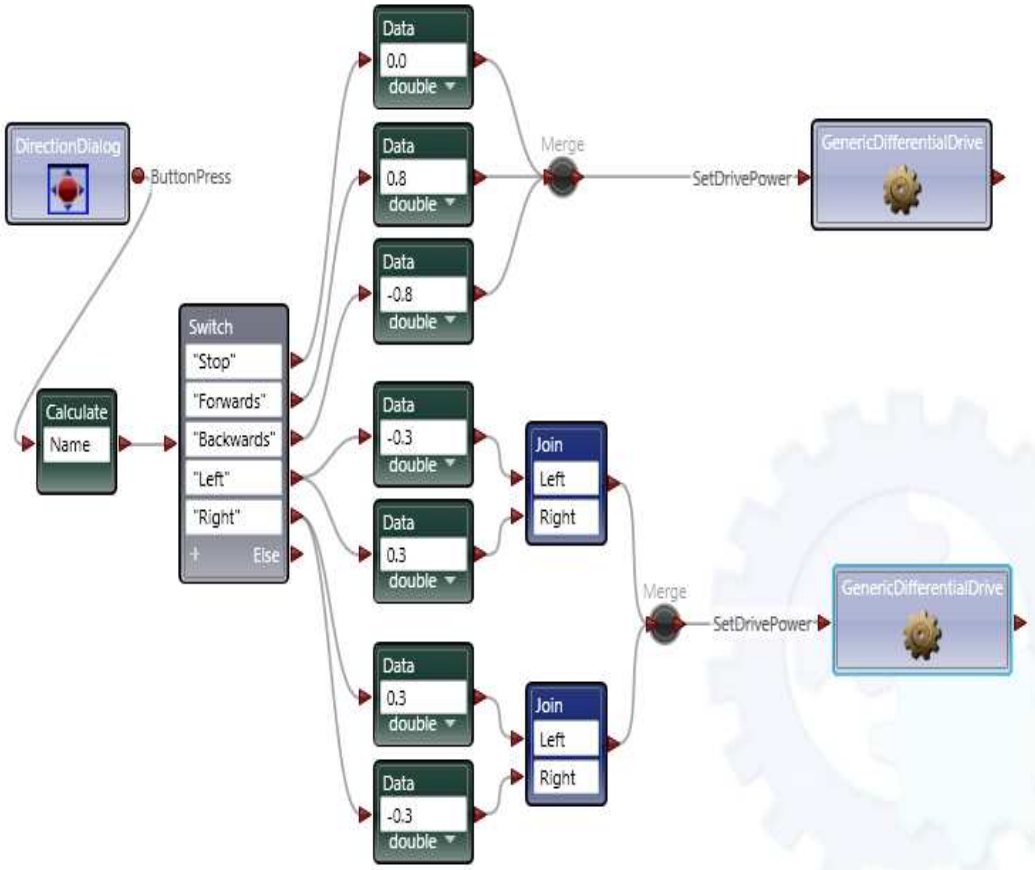


Εικόνα 9.5.1.α : Το παράθυρο του Data Connections .

Το διάγραμμα πρέπει να έχει την μορφή που παρουσιάζεται στην **Εικόνα 9.5.1.β**.

Εικόνα 9.5.1.β : Διάγραμμα με χρήση του GenericDifferentialDrive και ενός XInputController .

Εάν χρησιμοποιήθηκε το direction dialog,τότε το διάγραμμα θα πρέπει να έχει τη μορφή που παρουσιάζεται στην **Εικόνα 9.5.1.γ**.



Εικόνα 9.5.1.γ : Διάγραμμα με χρήση του direction dialog .

Επιλογή ενός manifest αρχείου

Για να γίνει η σύνδεση με το ρομπότ προσομοίωσης πρέπει να επιλεγεί το σωστό αρχείο manifest για το generic differential drive. Εάν υπάρχουν διάφορα **generic differential drive boxes** στο διάγραμμα, πρέπει να οριστεί το manifest αρχείο για ένα από αυτά. Αυτό γίνεται ως εξής: κάντε δεξί κλικ στο box και επιλέξτε το **set configuration**. Από την κυλιόμενη λίστα, κάντε κλικ στο **use a manifest** και στη συνέχεια στο import manifest από όπου επιλέγετε

LEGO.NXT.Tribot.Simulation.manifest.xml, ή
MobileRobots.P3DX.Simulation.manifest.xml, ή
SimulationTutorial3.manifest.xml.

9.5.2 Ο Κώδικας σε γλώσσα C Sharp (C#)

Βήμα 1: Καθορισμός λειτουργιών υπηρεσίας

Η κλάση RoboticsTutorial4Operations καθορίζει πέντε νέες λειτουργίες. Κάθε μία από αυτές τις λειτουργίες έχει ορισμό κλάσης για την λειτουργία και ορισμό κλάσης για τον τύπο δομής του μηνύματος.

Αυτοί είναι οι ορισμοί για την λειτουργία εκκίνησης(start).

```
[ServicePort]
public class
RoboticsTutorial4Operations :
PortSet<
DsspDefaultLookup,
DsspDefaultDrop,
Get,
Replace,
Stop,
Forward,
Backward,
TurnLeft,
TurnRight>
{
}
```

Τύπος μηνύματος και τύπος αίτησης για την λειτουργία Stop:

```
public class Stop : Submit<StopRequest,
PortSet<DefaultSubmitResponseType, Fault>>
                        {
    public Stop()
        : base(new StopRequest())
    {
                        }
}
[DataContract]
public class StopRequest { }
```

Αυτό ορίζει μία κλάση λειτουργίας (Operation class) Stop, η οποία είναι χωριστά από το γενικό τύπο Submit<TBody, TResponse>, έχοντας ένα τύπο δομής StopRequest. Με τον ίδιο τρόπο έχει οριστεί η λειτουργία Stop, το αρχείο επίσης περιέχει ορισμούς για τα Forward, Backward, TurnLeft και TurnRight. Η υπηρεσία ανταποκρίνεται σε αυτές τις λειτουργίες στέλνοντας εντολές σε έναν drive partner.

Κάθε μία από αυτές τις λειτουργίες έχει έναν «ταιριαστό» χειριστή handler στην κύρια υπηρεσία της κλάσης χειριστής για την λειτουργία Forward είναι ο ForwardHandler.

- **ForwardHandler**

Αν η μηχανή δεν είναι ενεργοποιημένη, ο ForwardHandler καλεί το EnableMotor. Το EnableMotor στέλνει ένα μήνυμα στο drive για να ενεργοποιήσει τη μηχανή.

```
 [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public virtual IEnumerator<ITask> ForwardHandler(Forward
forward)
{
    if (!_state.MotorEnabled)
                                {
        yield return EnableMotor();
    }

    // αυτό το παράδειγμα θέτει το  power σε 75%.
    // βασίζεται στο λογισμικό hardware,
    // μπορούν να γίνουν αλλαγές στις τιμές .
    drive.SetDrivePowerRequest request = new
drive.SetDrivePowerRequest();
    request.LeftWheelPower = 0.75;
    request.RightWheelPower = 0.75;

    yield return Arbiter.Choice(
        _drivePort.SetDrivePower(request),
        delegate(DefaultUpdateResponseType response) { },
        delegate(Fault fault)
        {
            LogError(null, "Unable to drive forwards",
fault);
        }
    );
```

Για να κινηθεί το ρομπότ μπροστά, ο ForwardHandler δημιουργεί μία αίτηση SetDrivePowerRequest και την τυπώνει στο _drivePort χρησιμοποιώντας την λειτουργία βοήθειας SetDrivePower.

Το πεδίο, _drivePort είναι η θύρα της λειτουργίας μίας συνεταιρικής υπηρεσίας που χρησιμοποιεί την υπηρεσία drive.

Οι χειριστές Handlers για τα υπόλοιπα τέσσερα μηνύματα ορίζονται με τον ίδιο τρόπο.

- **EnableMotor**

Οι χειριστές (handlers) για τα Forward, Backward, LeftTurn και RightTurn καλούν τη μέθοδο EnableMotor για να ενεργοποιηθεί η μηχανή εάν η υπηρεσία έχει θέσει τη ρύθμιση MotorEnabled σε κατάσταση false.

Αυτή η μέθοδος στέλνει ένα μήνυμα EnableDrive στην υπηρεσία drive service της. Καθώς επιστρέφεται ο τύπος Choice, η μέθοδος μπορεί να

επιστρέψει το αποτέλεσμα της λειτουργίας χρησιμοποιώντας το yield return, περιμένοντας μέχρι μία απάντηση να ληφθεί για αυτό το μήνυμα.

```
private Choice EnableMotor()
{
    drive.EnableDriveRequest request = new
drive.EnableDriveRequest();
    request.Enable = true;

    return Arbiter.Choice(
        _drivePort.EnableDrive(request),
        delegate(DefaultUpdateResponseType response) { },
        delegate(Fault fault)
        {
            LogError(null, "Unable to enable motor", fault);
        }
    );
}
```

- **NotifyDriveUpdate Handler**

Αυτή η υπηρεσία βοηθά στο να μπορεί η υπηρεσία drive service να αντιλαμβάνεται και να προσαρμόζεται με το αν οι μηχανές είναι ενεργοποιημένες. Σε μερικά ρομπότ, οι μηχανές μπορεί να είναι απενεργοποιημένες για συγκράτηση μπαταρίας και για να επιβεβαιωθεί ότι το ρομπότ δε θα κινηθεί σε ανύποπτη στιγμή. Στη μέθοδο Start, η λειτουργία αυτής της υπηρεσίας έχει ξεκινήσει και η μέθοδος NotifyDriveUpdate έχει ενεργοποιηθεί για να χειριστεί Update ανακοινώσεις (notifications) από την υπηρεσία drive

```
_drivePort.Subscribe(_driveNotify);
 Activate(Arbiter.Receive<drive.Update>(true,
      _driveNotify, NotifyDriveUpdate));
```

Το μήνυμα NotifyDriveUpdate αντιγράφει την ρύθμιση IsEnabled από τη δομή του update μηνύματος στη ρύθμιση MotorEnabled μίας καινούριας μεταβλητής StateType η οποία μετά τυπώνεται σε αυτή την υπηρεσία σε ένα μήνυμα Replace. Η κλάση StateType ορίζεται:

```
private void NotifyDriveUpdate(drive.Update
update)
{
      RoboticsTutorial4State state = new
           RoboticsTutorial4State();
    state.MotorEnabled =
update.Body.IsEnabled;

    _mainPort.Post(new Replace(state));
}
```

Βήμα 2: Ξεκίνημα της φόρμας

Το WinForms απαιτεί μία φόρμα να χρησιμοποιείται μόνο στο σημείο στο οποίο ξεκίνησε. Για να επιβεβαιωθεί ότι η υπηρεσία μπορεί να ακολουθήσει το σημείο στο οποίο μία φόρμα ξεκίνησε, η υπηρεσία στέλνει μηνύματα στο WinFormsServicePort. Αυτή είναι μία εσωτερική θύρα που είναι καθορισμένη στη κλάση DsspServiceBase.

Για να εμφανιστεί μία φόρμα πρέπει να τυπωθεί ένα μήνυμα RunForm. Για να κληθεί ο κώδικας που απευθύνεται στον έλεγχο της φόρμας τυπώνεται το μήνυμα FormInvoke. Η μέθοδος Start στην υπηρεσία της κλάσης αναπαριστά τον τρόπο που αυτά μπορούν να κληθούν.

```
          Protected override void Start()
{
    base.Start();

    WinFormsServicePort.Post(new
RunForm(StartForm));

    _drivePort.Subscribe(_driveNotify);
    Activate(Arbiter.Receive<drive.Update>(true,
_driveNotify, NotifyDriveUpdate));
}
```

Το μήνυμα RunForm παίρνει σαν μία ξεχωριστή παράμετρο στο κατασκευαστή του έναν εκπρόσωπο, ο οποίος καλείται να δημιουργήσει μία φόρμα. Αυτός ο εκπρόσωπος επιστρέφει ένα αντικείμενο System.Windows.Forms.Form. Πιο απλά, αυτό το αντικείμενο δημιουργεί τη φόρμα και επιστρέφει το πιο πρόσφατο αντικείμενο που δημιουργήθηκε. Στον κώδικα παρακάτω, ο τίτλος της φόρμας έχει τεθεί στο μονοπάτι της υπηρεσίας drive που χρησιμοποιείται. Αυτό αναπαριστά πώς χρησιμοποιείται το μήνυμα FormInvoke.

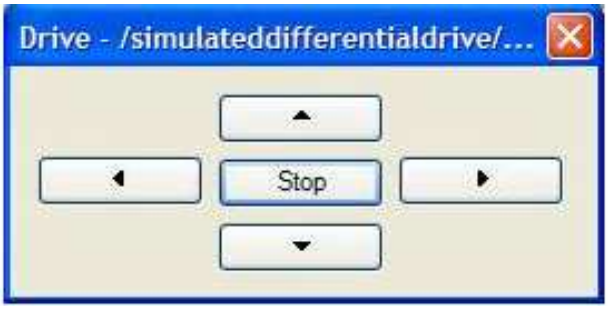
```
Private System.Windows.Forms.Form StartForm()
{
             RoboticsTutorial4Form form = new
            RoboticsTutorial4Form(_mainPort);

    Invoke(delegate()
        {
            PartnerType partner = FindPartner("Drive");
            Uri uri = new Uri(partner.Service);
            form.Text = string.Format(
                Resources.Culture,
                Resources.Title,
                uri.AbsolutePath
            );
        }
    );

    return form;
}
```

Αυτή η μέθοδος βοήθειας στέλνει ένα μήνυμα FormInvoke στο WinFormsServicePort. Αυτό επιβεβαιώνει ότι ο κώδικας που τροποποιεί τη φόρμα εκτελείται στο σωστό σημείο .

```
Private void
Invoke(System.Windows.Forms.MethodInvoker mi)
{
    WinFormsServicePort.Post(new FormInvoke(mi));
                            }
```

Βήμα 3: Εκτέλεση της υπηρεσίας από τη φόρμα

Στο βήμα 2, η φόρμα δημιουργήθηκε ως μέρος του ξεκινήματος μίας υπηρεσίας. Όταν δημιουργήθηκε η φόρμα, το πεδίο _mainPort περάστηκε σαν παράμετρος στο κατασκευαστή της φόρμας. Αυτό δίνει στη φόρμα την ικανότητα να στέλνει μηνύματα στην υπηρεσία. Η φόρμα έχει 5 κουμπιά (**Εικόνα 9.5.2.α**) που ανταποκρίνονται στις 5 λειτουργίες που έχουν οριστεί στο βήμα 1.



Εικόνα 9.5.2.α : Φόρμα με πέντε κουμπιά .

Όταν ένα κουμπί πατιέται, αρχίζει ο χειρισμός ενός γεγονότος τυπώνοντας την κατάλληλη λειτουργία στην κύρια θύρα της υπηρεσίας. Αυτή είναι η λειτουργία του χειριστή event handler για το κουμπί **Stop**.

```
private void btnStop_Click(object sender, EventArgs e)
                                {
    _mainPort.Post(new Stop());
}
```

Παρόμοιες μέθοδοι υπάρχουν και για τα υπόλοιπα κουμπιά.

Όταν η φόρμα κλείνει, η υπηρεσία δε μπορεί να κάνει την αρχική λειτουργία. Ο κώδικας της φόρμας στέλνει ένα Drop στην θύρα της υπηρεσίας όταν η φόρμα κλείνει από το χρήστη.

```
protected override void OnClosed(EventArgs e)
{
                _mainPort.Post(new
  DsspDefaultDrop(DropRequestType.Instance));

    base.OnClosed(e);
}
```

Βήμα 4: Εκτέλεση της υπηρεσίας

Αυτή η υπηρεσία ,χρησιμοποιεί τη σύμβαση drive. Αυτό επιτρέπει να χρησιμοποιείται η σύμβαση για να κινηθεί οποιαδήποτε υπηρεσία που καλύπτει τη σύμβαση.

Στον κατάλογο Samples\Config, υπάρχουν πολλά αρχεία με την κατάληξη .manifest.xml, και περιέχουν μία λίστα από υπηρεσίες που μπορούν να εκτελεστούν όταν το DSS node έχει ξεκινήσει.

9.6 Αναγνώριση περιοχών χρωμάτων

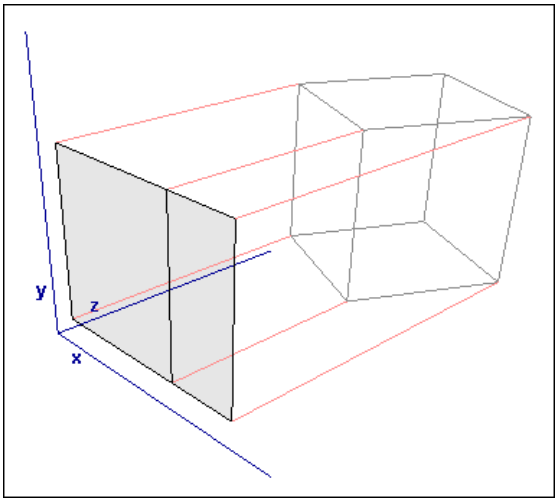
Σε αυτό το πρόγραμμα θα χρησιμοποιηθεί μια προσομοιωμένη κάμερα, για αυτό είναι σημαντική η επεξήγηση κάποιων βασικών εννοιών που σχετίζονται με την κάμερα.

9.6.1 Επεξήγηση βασικών εννοιών σχετικών με την κάμερα

Projection Matrix

Ο όρος projection ορίζεται ως ένα σύνολο εξισώσεων που αφορούν τις πραγματικές θέσεις (τρεις μεταβλητές x,y,z) σε σχέση με τις θέσεις στις καρτεσιανές συντεταγμένες.

Στους υπολογιστές μια camera matrix ή projection matrix χρησιμοποιείται για να περιγράψει τη χαρτογράφηση και τη μετατροπή από τα τρισδιάστατα σημεία στον κόσμο μας στα δυσδιάστατα σημεία μιας εικόνας. Ένα matrix στηρίζεται πάνω σε βασικές έννοιες τριγωνομετρίας. Μία οθόνη υπολογιστή είναι μια δυσδιάστατη επιφάνεια, έτσι εάν πρέπει να παρουσιαστούν τρισδιάστατες εικόνες, τότε χρειάζεται ένας τρόπος μετασχηματισμού της τρισδιάστατης γεωμετρίας σε μια μορφή που μπορεί να δοθεί ως 2D εικόνα. Και αυτό ακριβώς είναι που κάνει το projection matrix.για παράδειγμα, προκειμένου να προβληθεί ένα τρισδιάστατο αντικείμενο πάνω σε μια 2D επιφάνεια ένας τρόπος είναι η προσθήκη της διάστασης z σε ένα σχήμα με συντεταγμένες x και y όπως φαίνεται στην **Εικόνα 9.6.1.1**.



Εικόνα 9.6.1.1 : Προβολή πάνω στο x y της διάσταση z

Φυσικά, αυτό δεν είναι υπερβολικά απλό στις περισσότερες περιπτώσεις. Οι δυσδιάστατες συντεταγμένες x y θα μετασχηματίσουν τη γεωμετρία τους σε έναν νέο όγκο, αποκαλούμενο κανονικό όγκο άποψης (canonical view volume). Οι ακριβείς συντεταγμένες του κανονικού όγκου άποψης μπορούν να ποικίλουν από μια γραφική παράσταση σε άλλη (ανάλογα τη κωδικοποίηση της μετατροπής από 2D σε 3D), αλλά για λόγους

ευκολίας στο παραπάνω παράδειγμα, υποθέτουμε ότι το κιβώτιο εκτείνεται από (-1, -1, 0) (1, 1, 1), αυτές οι τιμές χρησιμοποιούνται άλλωστε και στο Direct3D.

Μόλις χαρτογραφηθούν όλα τα σημεία της δυσδιάστατης εικόνας στον κανονικό όγκο άποψης, μόνο οι X και Y συντεταγμένες τους χρησιμοποιούνται για να προβληθεί η εικόνα στην οθόνη. Η z συντεταγμένη δεν είναι άχρηστη, αφού χρησιμοποιείται για τον προσδιορισμό διαφάνειας προκειμένου να δοθεί η έννοια του βάθους. Αυτός είναι ο λόγος που μετασχηματίζεται σε έναν νέο όγκο .

Με αυτόν τον τρόπο, λαμβάνονται οι πραγματικές μετατροπές προβολής. Υπάρχουν πολλές διαφορετικές μέθοδοι προβολής όπως για παράδειγμα η ορθογραφική και η προοπτική.

Αν ληφθεί υπόψη η χαρτογράφηση που παράγεται από το matrix, οι συντεταγμένες εικόνας που προκύπτουν μπορούν να μετασχηματιστούν. Αυτό περιλαμβάνει τις δυσδιάστατες μετατροπές και τους γενικούς δυσδιάστατους μετασχηματισμούς προοπτικής.

The Projection Transformation

Ο καθορισμός του μετασχηματισμού που αφορά την προβολή είναι όπως η επιλογή ενός φακού για μια κάμερα. Ο μετασχηματισμός αυτός καθορίζει το οπτικό πεδίο και τα αντικείμενα που βρίσκονται μέσα σε αυτό, κι ως ένα βαθμό το πώς μοιάζουν. Αυτό είναι ισοδύναμο με την επιλογή μεταξύ των κανονικών φακών, των τηλεφακών, και των ευρυγώνιων φακών, για παράδειγμα. Με ένα ευρυγώνιο φακό, καλύπτεται μεγαλύτερο μέρος της περιοχής στην τελική φωτογραφία συγκριτικά με αυτή που καλύπτεται από ένα τηλεφακό. Όμως ένας τηλεφακός επιτρέπει τη φωτογράφηση αντικειμένων που να φαίνονται πολύ κοντά από ότι είναι στη πραγματικότητα.

Εκτός από τις εκτιμήσεις οπτικού πεδίου, ο μετασχηματισμός προβολής καθορίζει το πώς τα αντικείμενα προβάλλονται επάνω στην οθόνη. Δύο βασικοί τύποι προβολών παρέχονται από το OpenGL, μαζί με τις αντίστοιχες εντολές για την περιγραφή των σχετικών παραμέτρων με διαφορετικούς τρόπους. Ο ένας τύπος είναι η προβολή με προοπτική, η οποία ταιριάζει το πώς φαίνονται τα πράγματα στην καθημερινή ζωή. Η προοπτική κάνει τα αντικείμενα που είναι μακριά να εμφανίζονται μικρότερα. Η προβολή με προοπτική χρησιμοποιείται για ρεαλιστικές εικόνες και ορίζεται σε αυτό το παράδειγμα κώδικα με την εντολή glFrustum ().

Παράδειγμα μετασχηματισμένου κύβου

```
#include <GL/gl.h>
```

```
#include <GL/glu.h>
#include <GL/glut.h>

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glLoadIdentity ();             /* clear the matrix */
           /* viewing transformation  */
   gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
   glScalef (1.0, 2.0, 1.0);       /* modeling transformation
*/
   glutWireCube (1.0);
   glFlush ();
}

void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
   glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Ο άλλος τύπος προβολής είναι ο ορθογραφικός, ο οποίος χαρτογραφεί τα αντικείμενα απευθείας επάνω στην οθόνη χωρίς να επηρεάζεται από το σχετικό τους μέγεθος. Η ορθογραφική προβολή χρησιμοποιείται στις εφαρμογές αρχιτεκτονικού σχεδιασμού και με προγράμματα σχεδιασμού μέσω υπολογιστή όπου η τελική εικόνα πρέπει να απεικονίζει τις μετρήσεις των αντικειμένων κι όχι το πώς μοιάζουν. Οι αρχιτέκτονες δημιουργούν τα σχέδια με προοπτική για να δείξουν πώς εμφανίζονται τα κτήρια ή τα εσωτερικά τους μέρη από διάφορα οπτικά πεδία. Η ανάγκη για ορθογραφική προβολή προκύπτει κατά τη δημιουργία μηχανολογικών σχεδίων ή σχεδίων ανελκυστήρων τα οποία χρησιμοποιούνται στην ανέγερση κτιρίων.

Πριν κληθεί η glFrustum () για να θέσει το μετασχηματισμό προβολής, πρέπει να γίνει μια προετοιμασία. Η εντολή glMatrixMode () χρησιμοποιείται πρώτα (στο παραπάνω παράδειγμα κώδικα), με το όρισμα GL_PROJECTION. Αυτό δείχνει ότι η τρέχουσα μήτρα διευκρινίζει το μετασχηματισμό προβολής. Οι κλήσεις μετασχηματισμού που ακολουθούν έχουν επιπτώσεις στη μήτρα προβολής. Όπως φαίνεται, μερικές γραμμές πιο κάτω στον κώδικα ξανακαλείται η glMatrixMode (), αυτή τη φορά με το GL_MODELVIEW ως όρισμα. Αυτό δείχνει ότι οι επιτυχόντες μετασχηματισμοί έχουν επιπτώσεις τώρα στη μήτρα modelview αντί της μήτρας προβολής.

Στο σημείο αυτό αξίζει να αναφερθεί ότι η glLoadIdentity () χρησιμοποιείται για να τοποθετήσει τις αρχικές τιμές στην τρέχουσα μήτρα προβολής έτσι ώστε μόνο ο συγκεκριμένος μετασχηματισμός προβολής να έχει μια επίδραση. Τώρα η glFrustum () μπορεί να κληθεί, μαζί με τα ορίσματα που καθορίζουν τις παραμέτρους του μετασχηματισμού προβολής. Σε αυτό το παράδειγμα, και ο μετασχηματισμός προβολής και η άποψη του μετασχηματισμού περιέχονται στην ρουτίνα reshape(), η οποία καλείται όταν δημιουργείται αρχικά το παράθυρο και οποτεδήποτε αυτό κινείται ή αναδιαμορφώνεται. Αυτό έχει νόημα, δεδομένου ότι και η προβολή και η εφαρμογή της άποψης σχετίζονται άμεσα με την οθόνη και ειδικά με το μέγεθος ή τον λόγο των δύο διαστάσεων του παραθύρου στην οθόνη.

The Viewport Transformation

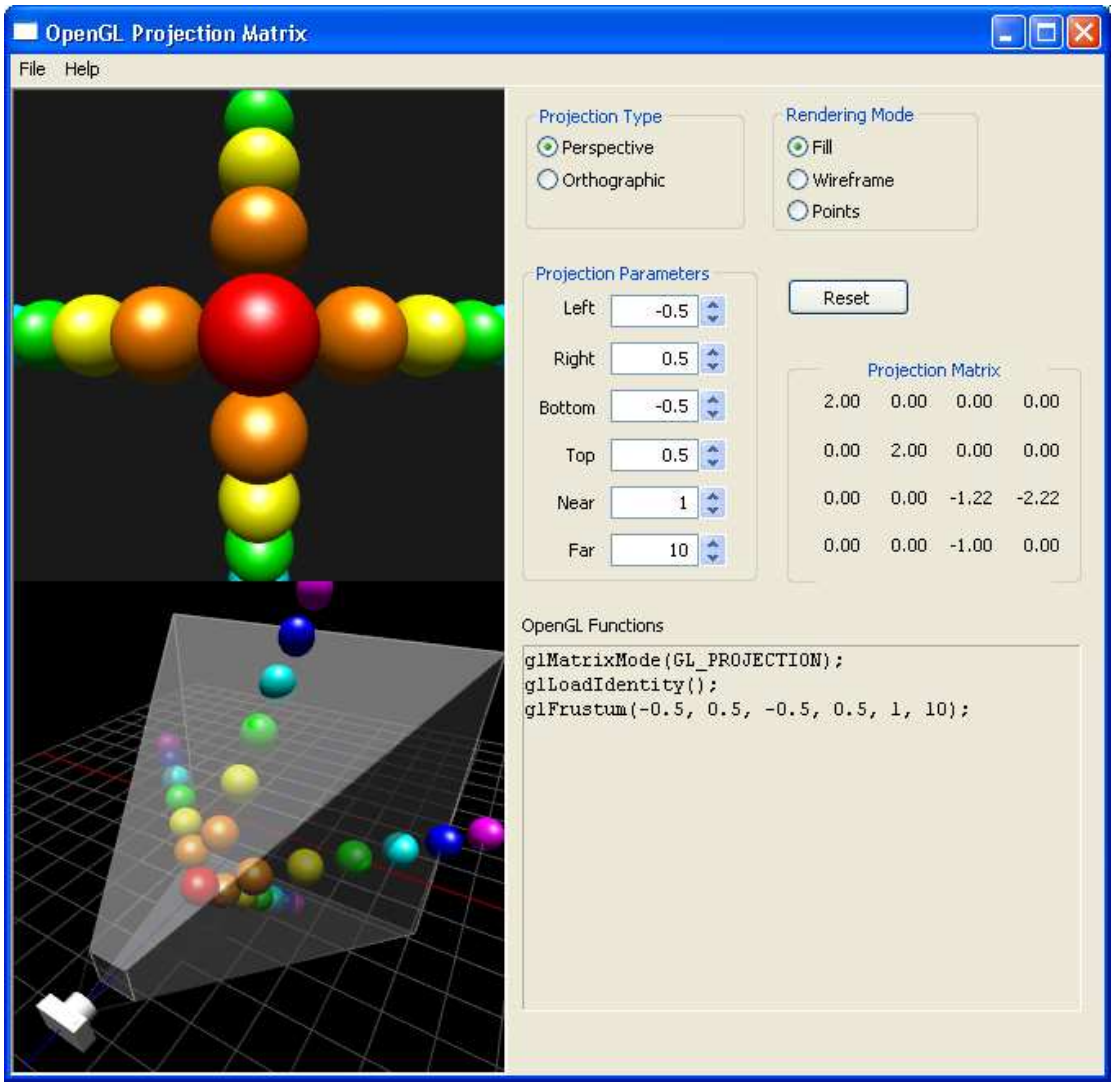
Τόσο ο μετασχηματισμός προβολής όσο και η άποψη του μετασχηματισμού καθορίζουν το πώς μια εικόνα χαρτογραφείται επάνω στην οθόνη του υπολογιστή. Ο μετασχηματισμός προβολής διευκρινίζει τη μηχανική για το πώς θα εμφανιστεί η χαρτογράφηση, και η άποψη δείχνει τη μορφή της διαθέσιμης περιοχής της οθόνης στην οποία χαρτογραφείται η εικόνα. Δεδομένου ότι η άποψη διευκρινίζει την περιοχή που καταλαμβάνει η εικόνα στην οθόνη ενός υπολογιστή, η άποψη του μετασχηματισμού, για παράδειγμα, καθορίζει το μέγεθος και τη θέση της τελικής επεξεργασμένης φωτογραφίας, είτε η φωτογραφία χρειάζεται μεγέθυνση είτε σμίκρυνση.

Τα ορίσματα της glViewport () περιγράφουν την προέλευση της διαθέσιμης περιοχής της οθόνης εντός του παραθύρου - (0, 0) σε αυτό το παράδειγμα – καθώς και το πλάτος και το ύψος της διαθέσιμης περιοχής της οθόνης. Όλα μετρούνται σε εικονοστοιχεία στην οθόνη. Για αυτό και η εντολή αυτή πρέπει να κληθεί μέσα στο reshape()- εάν το παράθυρο αλλάζει το μέγεθος, και η άποψη πρέπει να αλλάξει αναλόγως. Το πλάτος και το ύψος διευκρινίζονται χρησιμοποιώντας το πραγματικό πλάτος και το ύψος του παραθύρου.

Projection Matrix (GL_PROJECTION)

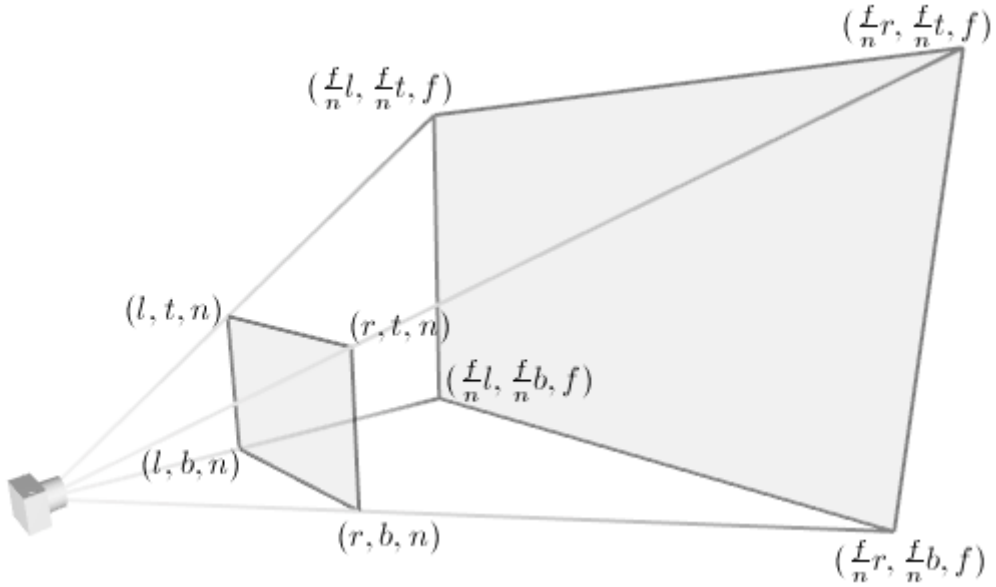
Η μήτρα GL_PROJECTION χρησιμοποιείται για να καθορίσει ένα κόλουρο κώνο. Αυτός ο κωνικός σχηματισμός καθορίζει ποια αντικείμενα ή

μέρη των αντικειμένων θα μειωθούν. Επίσης, καθορίζει πώς η τρισδιάστατη εικόνα προβάλλεται επάνω στην οθόνη (**Εικόνα 9.6.1.2**).



Εικόνα 9.6.1.2: Παράδειγμα Projection Matrix

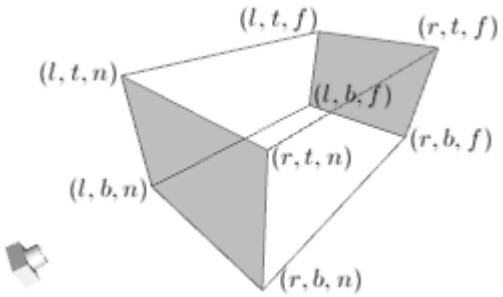
Το OpenGL παρέχει 2 λειτουργίες για το μετασχηματισμό GL_PROJECTION. Το **glFrustum ()** που χρησιμοποιείται για την παραγωγή μιας προβολής με προοπτική, και το **glOrtho ()** που χρησιμοποιείται για την παραγωγή μιας ορθογραφικής (παράλληλης) προβολής. Και οι δύο λειτουργίες απαιτούν 6 παραμέτρους για να διευκρινίσουν τα επίπεδα μείωσης: *αριστερό*, *δεξί*, *κατώτατο σημείο*, *κορυφή*, *κοντινά* και *μακρινά* επίπεδα. Οκτώ κορυφές του κόλουρου κώνου παρουσιάζονται στην ακόλουθη **Εικόνα 9.6.1.3**.



Εικόνα 9.6.1.3:OpenGL Perspective Viewing Frustum

Οι κορυφές στο πίσω επίπεδο μπορούν να υπολογιστούν απλά από την αναλογία των όμοιων τριγώνων, για παράδειγμα, το αριστερό του μακρινού επιπέδου θα είναι (**Εικόνα 9.6.1.4**):

$$\frac{far}{near} = \frac{left_{far}}{left}, \quad left_{far} = \frac{far}{near} \cdot left$$



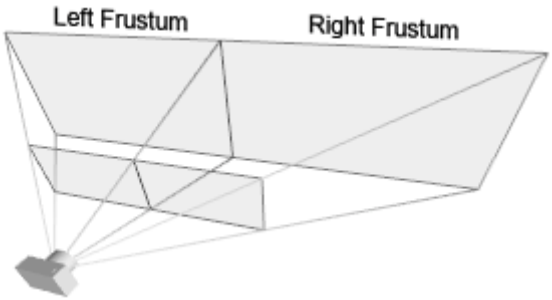
Εικόνα 9.6.1.4:OpenGL Orthographic Frustum

Για την ορθογραφική προβολή, αυτή η αναλογία θα είναι 1, έτσι οι τιμές του *αριστερού*, του *δεξιού*, του *κατώτατου σημείου* και της *κορυφής* του μακρινού επιπέδου θα είναι ίδιες με του κοντινού επιπέδου.

Υπάρχει επίσης η δυνατότητα να χρησιμοποιηθούν οι λειτουργίες gluPerspective () και gluOrtho2D () με λιγότερες παραμέτρους. Η **gluPerspective ()** απαιτεί μόνο 4 παραμέτρους: το κάθετο οπτικό πεδίο (fov), το λόγο δύο διαστάσεων του πλάτους προς το ύψος και οι αποστάσεις στα

κοντινά και μακρινά επίπεδα. Η ισοδύναμη μετατροπή από gluPerspective () σε glFrustum () περιγράφεται στον ακόλουθο κώδικα.

```
// Αυτό δημιουργεί ένα συμμετρικό frustum.
// Μετατρέπει σε 6 παραμέτρους(l, r, b, t, n, f) για την  glFrustum()
// 4 παράμετροι (fovy, aspect, near, far)
void makeFrustum(double fovY, double aspectRatio, double front, double back)
{
   const double DEG2RAD = 3.14159265 / 180;

   double tangent = tan(fovY/2 * DEG2RAD);   // εφαπτομένη του μισού fovY
double height = front * tangent;          // ήμισυ του ύψους του κοντινού πλάνου
double width = height * aspectRatio;       // ήμισυ του πλάτους του κοντινού πλάνου
   // παράμετροι : left, right, bottom, top, near, far
   glFrustum(-width, width, -height, height, front, back);
}
```



Εικόνα 9.6.1.5: Παράδειγμα ασύμμετρου frustum

Φαίνεται ξεκάθαρα ότι πρέπει να χρησιμοποιηθεί η glFrustum () αν πρέπει να δημιουργηθεί ένας μη συμμετρικός όγκος μιας εικόνας(**Εικόνα 9.6.1.5**). Παραδείγματος χάριν, για να δοθεί μια μεγαλύτερη εικόνα σε δύο γειτονικές οθόνες, θα πρέπει ο κόλουρος κώνος να χωριστεί σε δύο ασύμμετρους κόλουρους κώνους (τον αριστερό και τον δεξιό) και στη συνέχεια θα πρέπει να δοθεί μια εικόνα για κάθε κόλουρο κώνο.

OpenGL Transformation Matrix

$$\begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix}$$

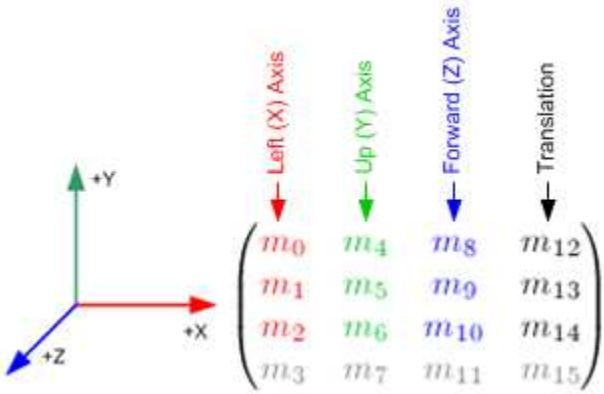
Εικόνα 9.6.1.6 :OpenGL Transform Matrix

Η OpenGL χρησιμοποιεί μια μήτρα 4 X 4 για τους μετασχηματισμούς. Τα δεκαέξι στοιχεία της μήτρας αποθηκεύονται σε μορφή ενός πίνακα στήλης. Ο πίνακας αυτός πρέπει να μετατεθεί στην μήτρα ώστε να την μετατρέψει στην τυποποιημένη μορφή του πίνακα γραμμής (**Εικόνα 9.6.1.6**).

Το OpenGL έχει 4 διαφορετικούς τύπους μήτρων: τον **GL_MODELVIEW**, τον **GL_PROJECTION**, τον **GL_TEXTURE**, και τον **GL_COLOR**. Για τη μετατροπή του τρέχοντος τύπου πρέπει να χρησιμοποιηθεί στον κώδικα η **glMatrixMode ()**. Παραδείγματος χάριν, προκειμένου να επιλεχτεί η μήτρα GL_MODELVIEW, πρέπει να χρησιμοποιηθεί η **glMatrixMode (GL_MODELVIEW)**.

Model-View Matrix (GL_MODELVIEW)

Η μήτρα GL_MODELVIEW συνδυάζει τη μήτρα θεώρησης και τη μήτρα διαμόρφωσης, σε μια μήτρα. Προκειμένου να μετασχηματιστεί η άποψη (δηλαδή η κάμερα), πρέπει να κινηθεί ολόκληρη την εικόνα με αντίστροφο μετασχηματισμό. Η **gluLookAt ()** χρησιμοποιείται κυρίως για να θέσει τον νέο μετασχηματισμό.



Εικόνα 9.6.1.7 :4 columns of GL_MODELVIEW matrix

Τα 3 στοιχεία της μήτρας(**Εικόνα 9.6.1.7**) στη δεξιά στήλη (*m12*, *m13*, *m14*) αφορούν το μετασχηματισμό των συντεταγμένων, **glTranslatef ()**. Το στοιχείο *m15* είναι η ομοιογενής (homogeneous) συντεταγμένη, η οποία χρησιμοποιείται για τον μετασχηματισμό στην προβολή.

Τρία σύνολα στοιχείων, τα (*m0*, *m1*, *m2*), (*m4*, *m5*, *m6*) και (*m8*, *m9*, *m10*) αφορούν τον Ευκλείδειο και τον συγγενικό μετασχηματισμό, όπως η περιστροφή **glRotatef ()** ή η κλίμακα **glScalef ()**. Τα τρία αυτά σύνολα αντιπροσωπεύουν στην πραγματικότητα τους τρεις ορθογώνιους άξονες:

- (*m0*, *m1*, *m2*): + άξονας X, *αριστερό* διάνυσμα, (1 ,0, 0) εξ ορισμού
- (*m4*, *m5*, *m6*): + άξονας Y, *επάνω* διάνυσμα, (0 ,1, 0) εξ ορισμού
- (*m8*, *m9*, *m10*): + άξονας *ζ*, *μπροστινό* διάνυσμα, (0 ,0, 1) εξ ορισμού

Υπάρχει δυνατότητα άμεσης κατασκευής της μήτρας GL_MODELVIEW από τις γωνίες ή το υπάρχον διάνυσμα χωρίς τη χρησιμοποίηση των λειτουργιών μετασχηματισμού της OpenGL.

Η OpenGL εκτελεί τους πολλαπλασιασμούς των μήτρων στην αντίστροφη διάταξη εάν οι πολλαπλάσιοι μετασχηματισμοί εφαρμόζονται στην κορυφή. Παραδείγματος χάριν, εάν η κορυφή μετασχηματίζεται πρώτα από *το MA*, και δεύτερον μετασχηματίζεται από *το MB*, η OpenGL αποδίδει πρώτα τον πολλαπλασιασμό *MB* x *MA* και έπειτα τον πολλαπλασιασμό των κορυφών. Έτσι, η τελευταία μετατροπή έρχεται πρώτη και η πρώτη μετατροπή εμφανίζεται στο τέλος του κώδικα.

$$v' = M_B \cdot (M_A \cdot v) = (M_B \cdot M_A) \cdot v$$

```
// Σημειώστε ότι το αντικείμενο θα μεταφραστεί πρώτα και μετά θα περιστραφεί
glRotatef(angle, 1, 0, 0);    // περιστρέψετε τη γωνία του αντικειμένου γύρω από τον άξονα X
glTranslatef(x, y, z);        // μετακινεί το αντικείμενο to (x, y, z)
drawObject();
```

View Frustum

Στα τρισδιάστατα γραφικά υπολογιστών, το **viewing frustum** ή **view frustum** είναι η περιοχή στο διαμορφωμένο κόσμο που μπορεί να εμφανιστεί στην οθόνη δηλαδή είναι το οπτικό πεδίο της εννοιολογικής κάμερας. Η ακριβής μορφή αυτής της περιοχής ποικίλλει ανάλογα με τι είδους φακό καμερών μιμείται. Ο όρος frustum χρησιμοποιείται συνήθως στην γραφιστική

για να περιγράψει την τρισδιάστατη περιοχή που είναι ορατή στην οθόνη

Στην τρισδιάστατη γραφιστική, το **Viewing frustum culling** είναι η διαδικασία που χρησιμοποιείται για να καθορίσει ποιες επιφάνειες και μέρη των επιφανειών δεν είναι ορατά από μια ορισμένη άποψη. Ένας κρυμμένος αλγόριθμος προσδιορισμού επιφάνειας είναι μια λύση στο πρόβλημα διαφάνειας, το οποίο ήταν ένα από τα πρώτα σοβαρά προβλήματα στον τομέα της τρισδιάστατης ηλεκτρονικής γραφιστικής. Η διαδικασία του κρυμμένου προσδιορισμού επιφάνειας καλείται μερικές φορές κρύψιμο, και ένας τέτοιος αλγόριθμος καλείται μερικές φορές hider. Υπάρχουν πολλές τεχνικές για τον κρυμμένο προσδιορισμό επιφάνειας.

9.6.2 Η αναγνώριση περιοχών χρωμάτων σε C#

Η υπηρεσία Color Segment αναλύει εικόνες από μία κάμερα ή προσομοιωμένη κάμερα, και βασίζεται σε χρωματικούς περιορισμούς που δηλώνονται μέσα στη υπηρεσία, αναγνωρίζει περιοχές χρωμάτων σε κάθε εικόνα.

Προαπαιτούμενα

Hardware

Τυπικά αυτό το παράδειγμα απαιτεί μία κάμερα. Μπορεί να εκτελεστεί χρησιμοποιώντας μία προσομοιωμένη κάμερα στο περιβάλλον προσομοίωσης.

Software

Θα χρειαστεί ο Microsoft Internet Explorer ή άλλος συμβατός web browser.

Εκτελώντας την ColorSegment υπηρεσία

Για να ξεκινήσει η ColorSegment υπηρεσία , ξεκινήστε ένα κόμβο (node) χρησιμοποιώντας την ακόλουθη εντολή από το DSS Command Prompt:

```
Console

        dsshost /p:50000 /m:"samples\config\Colorsegment.manifest.xml"
```

Αυτό θα ξεκινήσει την ColorSegment υπηρεσία και το default παράδειγμα της υπηρεσίας της κάμερας. Για να αλληλεπιδράσει η εκτελούμενη ColorSegment υπηρεσία χρειάζεται πλοήγηση στο http://localhost:50000/colorsegment, αυτό θα παρουσιάσει την διεπαφή χρήστη της υπηρεσίας.

User Interface (Διεπαφή χρήστη)

Η διεπαφή του ColorSegment έχει τους ακόλουθους τομείς.

- **Εικόνες** – η εικόνα της κάμερας στα αριστερά, η επεξεργασμένη εικόνα στα δεξιά.
- **Χειρισμός της κάμερας**
  - Το παράθυρο Refresh Interval σας επιτρέπει να θέσετε το όριο στο οποίο ο φυλλομετρητής διαβάζει εικόνες από την κάμερα και μπορεί να οριστεί σε χιλιοστά του δευτερολέπτου (milliseconds).
  - Το κουμπί Start ξεκινά την ενημέρωση των εικόνων από την κάμερα
  - Το κουμπί Stop σταματά την ενημέρωση των εικόνων από την κάμερα
  - Το κουμπί Refresh παίρνει την τρέχουσα εικόνα από την κάμερα και την υπηρεσία ColorSegment.
- **Ρυθμίσεις**
  - Η Threshold ρύθμιση είναι ένα μέτρο που μας δείχνει πόσο πολύ ταιριάζει ένα χρώμα, όσο μεγαλύτερος είναι ο αριθμός, τόσο λιγότερο ταιριάζει το χρώμα.
  - Η Minimum Blob Size ρύθμιση είναι το μικρότερο σέτ συνδεδεμένων pixels που θα αναφερθούν σε μία subscribing υπηρεσία. Αυτό μετριέται σε pixels.
  - Το κουμπί Change επικυρώνει τις αλλαγές στις ρυθμίσεις της υπηρεσίας.

    Σημείωση: οι αλλαγές στις ρυθμίσεις δεν έχουν επίδραση μέχρι να πατηθεί το κουμπί.
- **Ορισμός Χρωμάτων (Color Definitions) –** Κάθε ορισμός χρωμάτων έχει τα υπόλοιπα μέρη.
  - Name – Αυτό είναι ένα όνομα που δίνεται στο χρώμα όταν πρωτοπροστέθηκε. Οι ορισμοί με το ίδιο όνομα αναφέρονται μαζί και όταν εμφανίζονται σε μια εικόνα τα pixels είναι χρωματισμένα με το χρώμα του πρώτου ορισμού χρώματος με το ίδιο όνομα
  - Y Cb Cr – Αυτό είναι το χρώμα όπως ορίζεται στο διάστημα Y Cb Cr. Όλες οι χρωματικές προσαρμογές γίνονται εσωτερικά σε αυτό το διάστημα. επίσης δείχνει την κλίμακα των χρωμάτων που είναι σε κάθε διάσταση.
  - R G B – Αυτό είναι το χρώμα που καθορίστηκε στο διάστημα R G B.
  - Delete – Πιέζοντας αυτό το κουμπί διαγράφουμε το χρώμα που ορίσαμε.
  - Επέκταση του Y – Αυτό επεκτείνει την κλίμακα των χρωμάτων που υπάρχουν στη διάσταση Y από το διάστημα Y Cb Cr. Αυτό προκαλεί την υπηρεσία ColorSegment να αναγνωρίζει ανοιχτόχρωμα και σκουρόχρωμα χρώματα ως ίδια.

Καθορισμός χρώματος

Για τον καθορισμό ενός χρώματος ακολουθούνται οι επόμενες ενέργειες.

1. Αρχικά αναγνωρίζεται η περιοχή της εικόνας που έχει το χρώμα που πρόκειται να καθοριστεί.
2. Πατιέται κλικ σε μία γωνία αυτής της περιοχής στην εικόνα .
3. Πατιέται κλικ σε άλλη γωνία της περιοχής. Αυτό θα δημιουργήσει ένα κόκκινο πολύγωνο που θα εμφανιστεί στην εικόνα δείχνοντας την περιοχή που θα συμπεριληφθεί καθορίζοντας το χρώμα
4. Αν η περιοχή που έχει επιλεχθεί είναι η επιθυμητή τότε:
   - Πατιέται κλικ στο όνομα ενός υπάρχον χρωματικού περιορισμού , ή
   - Πληκτρολογείτε το όνομα ενός καινούργιου χρώματος στο ειδικό παράθυρο κειμένου που είναι στην επιλογή πρόσθεσης στο τέλος των επιλογών ορισμού
5. Πατιέται κλικ στο κουμπί Add button για να προστεθεί ο καινούριος καθορισμός.

Αυτή η διαδικασία στέλνει τις συντεταγμένες του ορθογωνίου και του ονόματος που εισάγονται, στην ColorSegment υπηρεσία. Η υπηρεσία μετά εκτελεί μία απλή στατιστική ανάλυση της επιλεγμένης περιοχής της τρέχουσας εικόνας για να υπολογίσει το επικρατέστερο χρώμα και την κλίμακα των χρωμάτων που βρέθηκε σε αυτήν την περιοχή.

Χρήση της ColorSegment υπηρεσίας από άλλη υπηρεσία

Για να χρησιμοποιηθεί η ColorSegment υπηρεσία από άλλη υπηρεσία χρειάζεται να προστεθεί η ColorSegment ως συνεταιρική στην άλλη υπηρεσία.

Αρχικά προστίθεται μία αναφορά στην ColorSegment υπηρεσία στο πρόγραμμα, και στη συνέχεια προστίθεται στο πάνω μέρος της υπηρεσίας μια δήλωση χρήσης της.

C#
```
using cs = Microsoft.Robotics.Services.Sample.ColorSegment.Proxy;
```

Στη συνέχεια, στην κλάση της υπηρεσίας πρέπει να δηλωθεί η συνεταιρική.

```
C#
[Partner("ColorSegment", Contract = cs.Contract.Identifier,
CreationPolicy = PartnerCreationPolicy.UsePartnerListEntry)]
cs.ColorSegmentOperations _colorSegmentPort = new
cs.ColorSegmentOperations();
cs.ColorSegmentOperations _colorSegmentNotify = new
cs.ColorSegmentOperations();
```

Στην μέθοδο Start της κλάσης, δηλώνεται η υπηρεσία ColorSegment, και στο συγκεκριμένο παράδειγμα δηλώνεται το μήνυμα cs.UpdateColorAreas.

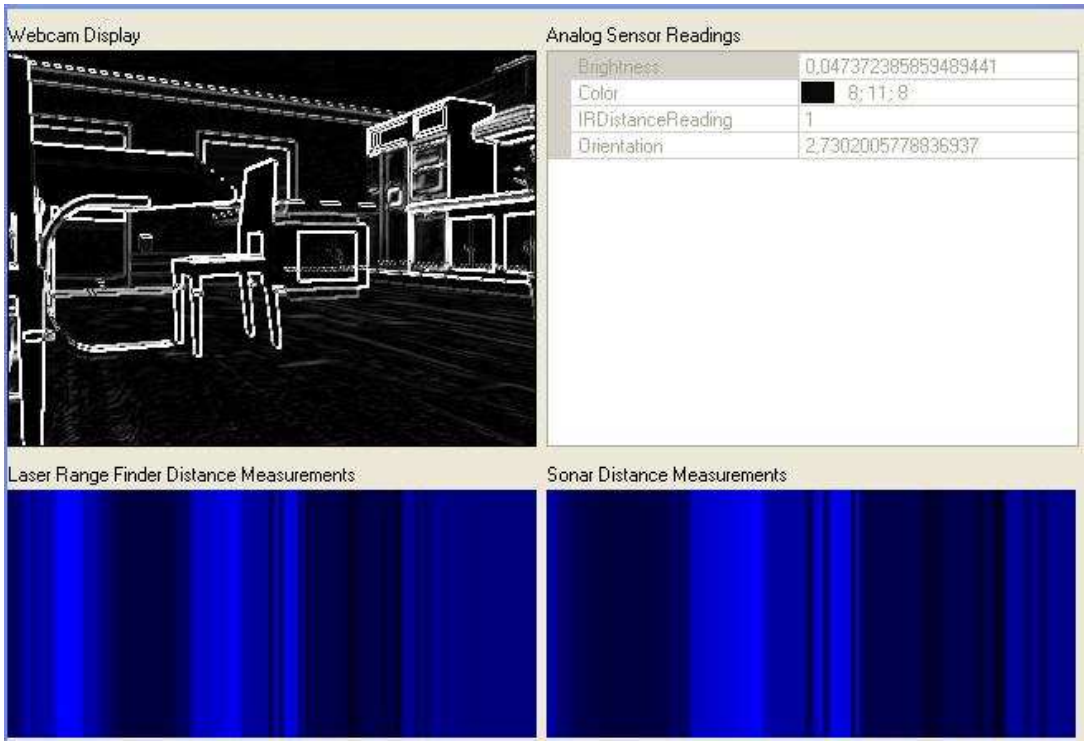
Επίσης ενεργοποιείται ένας έλεγχος για τις ειδοποιήσεις από αυτή την υπηρεσία

```
C#
 protected overide void Start()
{
    base.Start();

    _colorSegmentPort.Subscribe(_colorSegmentNotify,
typeof(cs.UpdateColorAreas));

    Activate(
        Arbiter.Receive<cd.UpdateColorAreas>(true, _colorSegmentNotify,
OnUpdateColorAreas)
    );
}
```

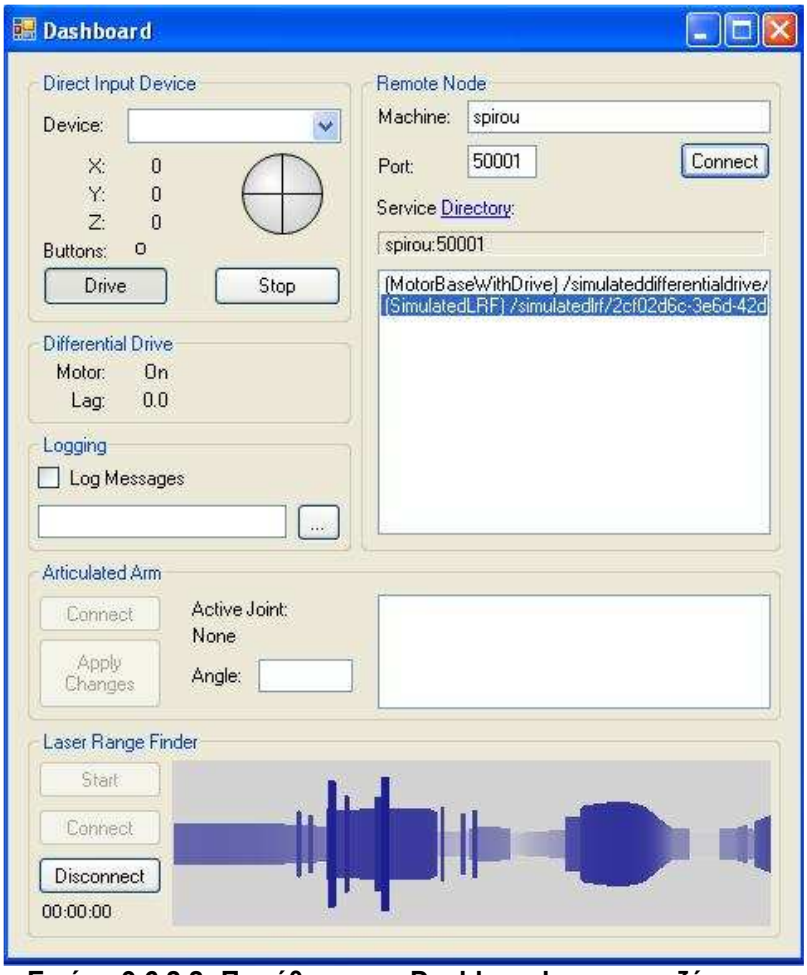
Η εφαρμογή της μεθόδου χειρισμού που θα καλεστεί από την υπηρεσία ColorSegment επεξεργάζεται κάθε εικόνα και αναγνωρίζει το χρώμα.

```
C#
 void OnUpdateColorAreas(cs.UpdateColorAreas update)
{
    // διαδικασία αναγνώρισης χρωμάτων περιοχής
```

Στη συνέχεια παρουσιάζονται μερικές εικόνες που σχετίζονται με τη χρήση της κάμερας του ρομπότ, **Εικόνα 9.6.2.1**, **Εικόνα 9.6.2.2**, **Εικόνα 9.6.2.3**.



Εικόνα 9.6.2.1 : Παράθυρο που δείχνει την εικόνα που τραβάει η κάμερα με διάφορες ρυθμίσεις που αφορούν για παράδειγμα το χρώμα, την φωτεινότητα.



Εικόνα 9.6.2.2: Παράθυρο του Dashboard που χειριζόμαστε για την προσομοίωση μίας εικόνας που τραβάει η κάμερα.

Εικόνα 9.6.2.3: Παράθυρο του Simulation όπου φαίνεται τι τραβάει η κάμερα του ρομπότ που είναι σε ένα χώρο .

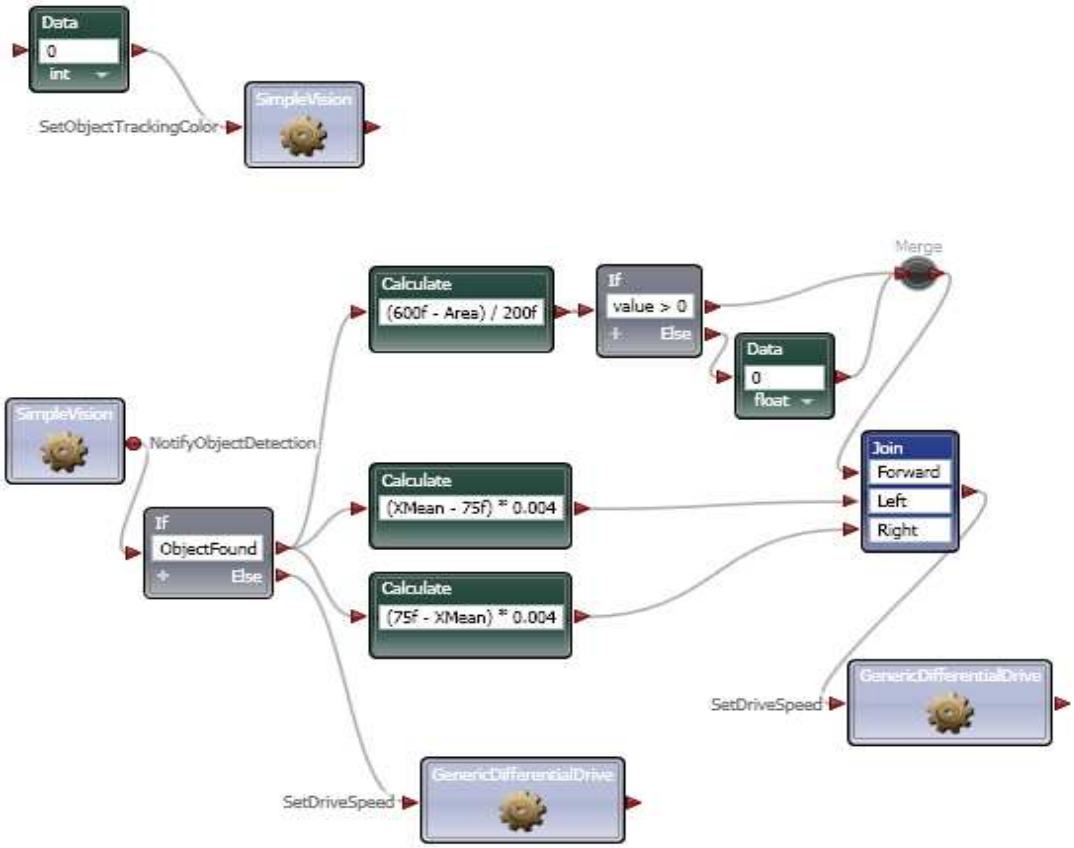
Κεφάλαιο 10

Παρουσίαση Advanced προγραμμάτων σε γλώσσα VPL του Microsoft Robotics Developer Studio

10.1 Κίνηση ρομπότ το οποίο ανιχνεύει και ακολουθεί μία κόκκινη μπάλα καθώς κινείται τυχαία.

10.1.1 Παρουσίαση προγράμματος στην VPL και σχολιασμός της λειτουργίας του.

Το πρόγραμμα που θα αναλύσουμε(δείτε παρακάτω **Εικόνα10.1.1.α**) αποτελεί την δημιουργία ενός γραφικού περιβάλλοντος προσομοίωσης ,στο οποίο θα κινείται με μία συγκεκριμένη ταχύτητα μία κόκκινη μπάλα τυχαία ,την οποία θα ανιχνεύει το ρομπότ και θα ακολουθεί την πορεία της.



Εικόνα 10.1.1.α : Διάγραμμα ρομπότ στη VPL που ανιχνεύει και ακολουθεί κόκκινη μπάλα καθώς κυλάει.

Η **simple Vision activity** χρησιμοποιείται για να αναλύσει το βίντεο από την ενσωματωμένη κάμερα του ρομπότ.

Η **Data activity** στο επάνω αριστερά μέρος της **Εικόνας 10.1.1.α** στέλνει μήνυμα αρχικοποίησης στην είσοδο **SetObjectTrackingColor** της **Simple Vision activity**. Αυτό που κάνει είναι να θέτει το χρώμα που θέλουμε να κοιτάζουμε στο κόκκινο. Μόνο τα Pixels που είναι κοντά σε αυτήν την τιμή θα υπολογίζονται από την υπηρεσία **Simple Vision**.

Η υπηρεσία στέλνει περιοδικά μηνύματα για να δείξει αν ακολουθεί ή όχι ένα αντικείμενο. Αν το **ObjectFound** είναι ψευδές (false) τότε ένα μήνυμα στέλνεται στην **simulated Differential Drive** υπηρεσία για να σταματήσει το ρομπότ την κίνησή του. Οπότε αν η κάμερα του ρομπότ δεν βλέπει την μπάλα, τότε σταματά εκεί που είναι.

Αν η υπηρεσία δείξει ότι η μπάλα εμφανίζεται στην κάμερα τότε τρεις **Calculate activities** αποφασίζουν τις τιμές που πρέπει να δοθούν στην μηχανή έτσι ώστε το ρομπότ να κινηθεί προς την μπάλα αλλά όχι πολύ κοντά!

Συγκεκριμένα , α)για την κίνηση εμπρός έχουμε:
Αν το αντικείμενο (η μπάλα) είναι πολύ μακριά (δηλαδή το area πλησιάζει την τιμή 0) τότε το ρομπότ κινείται γρήγορα μπροστά. Όσο το αντικείμενο έρχεται πιο κοντά τότε η τιμή του area πλησιάζει το 600 και έτσι το ρομπότ που θα πάρει την τιμή της εξίσωσης θα κινηθεί πιο αργά προς τα εμπρός!

Για να αποφύγουμε την περίπτωση που το area πάρει μεγαλύτερη τιμή από 600 (δηλαδή το αντικείμενο έρθει πολύ κοντά) τότε βάζουμε μια συνθήκη if η οποία θα στέλνει 0 στην τιμή forward!Αυτό το κάνουμε για να μην κινηθεί προς τα πίσω!

β)Οι εξισώσεις για αριστερά και δεξιά.
Όσο η τιμή του **XMean** έχει απόκλιση από το 75f (που είναι το κέντρο που εστιάζει η κάμερα) τόσο οι τιμές left και right θα «παίζουν» έτσι ώστε να τείνουν να διορθώσουν το X (να κεντράρουν το αντικείμενο στο κέντρο της κάμερας. Για παράδειγμα, αν το αντικείμενο είναι δεξιά στην οθόνη τότε η τιμή για τον αριστερό κινητήρα θα πάρει μεγαλύτερη τιμή από αυτή του δεξιού κινητήρα το οποίο θα έχει ως αποτέλεσμα το ρομπότ να στρίψει δεξιά.

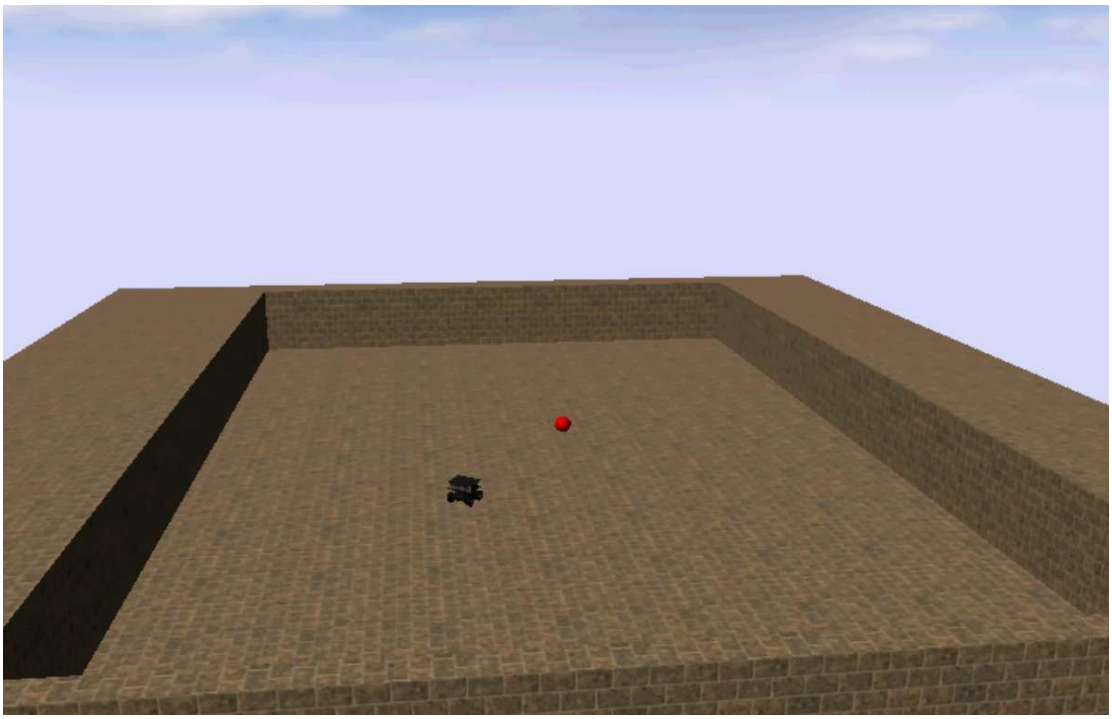
Τέλος όλες αυτές οι τιμές συγκεντρώνονται μέσω της **Join** (η οποία θυμίζουμε ότι περιμένει αποτελέσματα σε όλες στις θύρες της για να δώσει τιμή) και υπολογίζουμε τις τιμές που θα πάρει ο αριστερός και δεξιός κινητήρας με βάση αυτές τις εξισώσεις:
LeftWheelSpeed = value.Left + value.Forward
RightWheelSpeed = value.Right + value.Forward

Παρατηρούμε ότι αθροίζουμε τις τιμές και δεν παίρνουμε το γινόμενο τους. Αυτό το κάνουμε έτσι ώστε οι δύο διαφορετικές συμπεριφορές (στροφή και κίνηση προς τα εμπρός) να αντιμετωπίζονται χωριστά. Για παράδειγμα αν

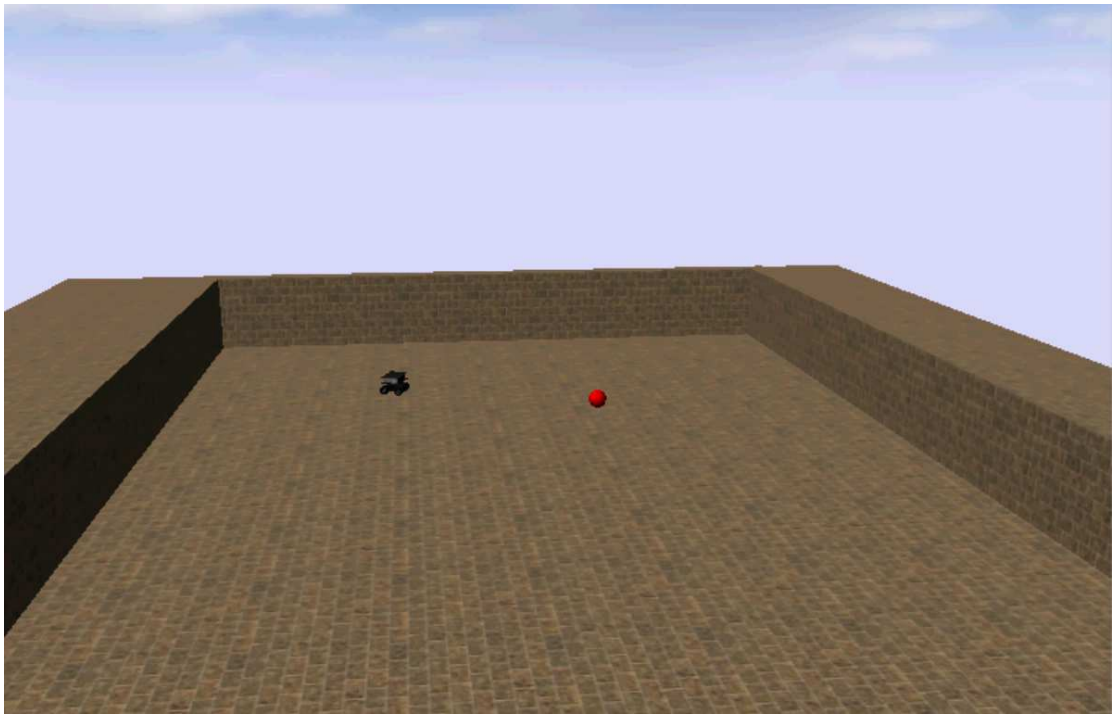
η μπάλα ήταν πολύ κοντά τότε η τιμή value.Forward θα ήταν 0 και αυτό θα είχε ως συνέπεια το ρομπότ να μη μπορούσε να στρίψει!

<u>Σημείωση</u> : Στο Παράρτημα μπορείτε να δείτε την υλοποίηση της παραπάνω εφαρμογής σε κώδικα C# με σχόλια.

10.1.2 Εικόνες της προσομοίωσης του προγράμματος



Εικόνα 10.1.2.α: Προσομοίωση ρομπότ που ανιχνεύει και ακολουθεί κόκκινη μπάλα.

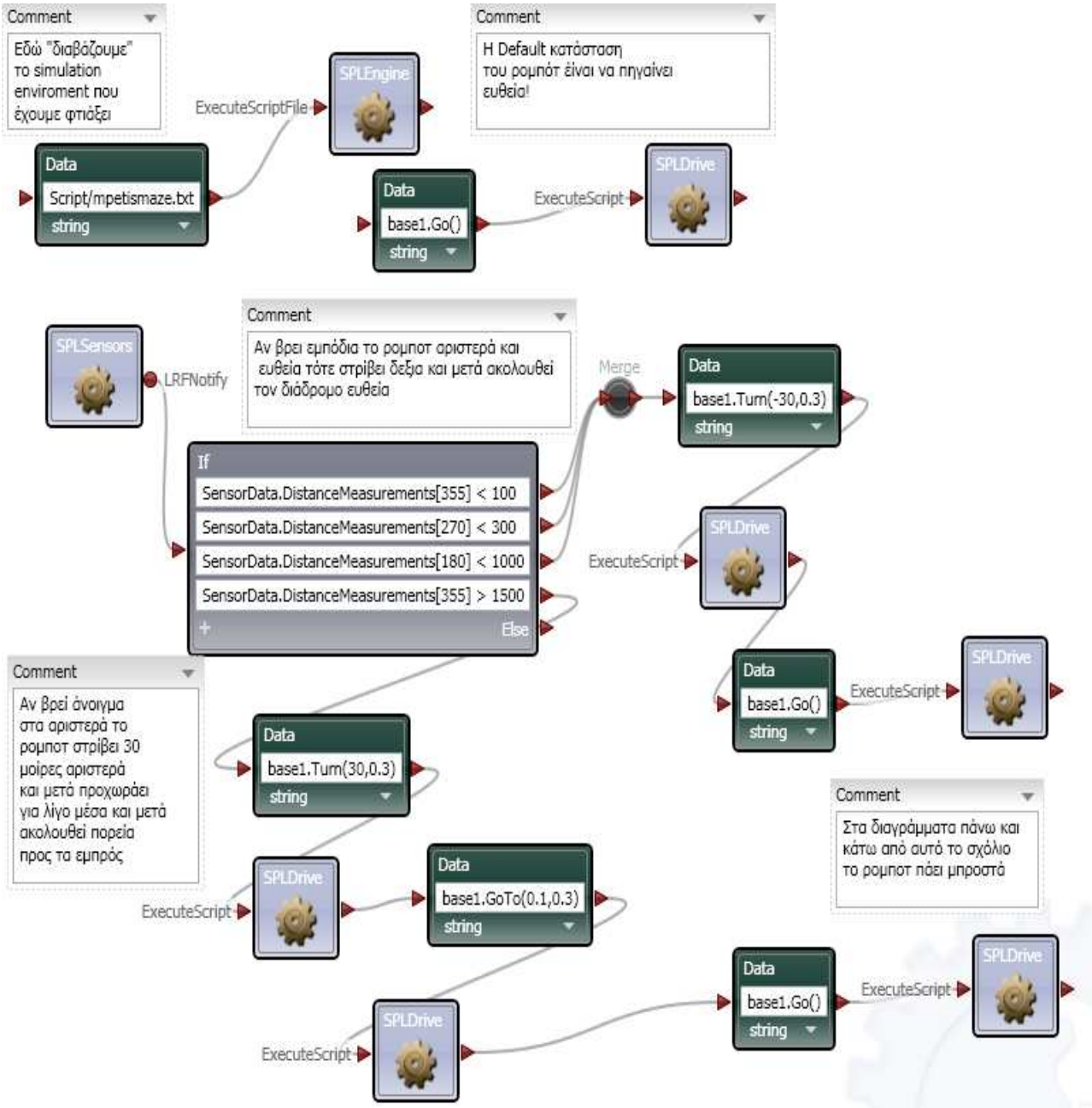


Εικόνα 10.1.2.β : Προσομοίωση ρομπότ που ανιχνεύει και ακολουθεί κόκκινη μπάλα.

10.2 Κίνηση ρομπότ σε λαβύρινθο όπου ανιχνεύει την έξοδο.

10.2.1 Παρουσίαση προγράμματος στην VPL και σχολιασμός της λειτουργίας του.

Το πρόγραμμα που θα αναλύσουμε(δείτε παρακάτω **Εικόνα10.2.1.α**) αποτελεί την δημιουργία ενός γραφικού περιβάλλοντος προσομοίωσης ,στο οποίο θα κινείται ένα ρομπότ σε ένα λαβύρινθο και θα βρίσκει τον έξοδο του λαβυρίνθου.



Εικόνα 10.2.1.α: Διάγραμμα ρομπότ στη VPL που κινείται σε λαβύρινθο και βρίσκει την έξοδό του.

Το πρώτο μπλοκ **Data** περιέχει ως συνάρτηση το **script / mpetismaze.txt.**Εδώ δίνουμε το path(διαδρομή) που έχουμε σώσει τον λαβύρινθό μας και αυτό που κάνουμε είναι να το συνδέσουμε με το **SPLEngine** το οποίο θα διαβάσει το αρχείο αυτό που του δείχνει το path και θα το εκτελέσει.

Μέσα στο simulation έχουμε κάποιες οντότητες. Μια οντότητα είναι το **base1** που είναι το όνομα του ρομπότ. Αυτό λειτουργεί με το **SPLDrive.** Εμείς θέλουμε το ρομπότ να προχωράει ευθεία. Αυτό θα το δηλώναμε βάζοντας στο δεξί και τον αριστερό κινητήρα μια τιμή. Το ίδιο κάνουμε και εδώ μόνο που το κάνουμε με script. Απλά του λέμε **base1**(το όνομα του ρομπότ) **GO** (πήγαινε) και (). Μέσα στην παρένθεση μπορούμε να βάλουμε ότι τιμή θέλουμε για

αριστερό ή δεξί κινητήρα αρκεί να το χωρίζουμε με ένα κόμμα. Αν δεν βάζουμε τιμές τότε παίρνει από μόνο του την τιμή 0.2 και στους 2 κινητήρες όποτε προχωράει ευθεία. Άρα θα μπορούσαμε και να γράψουμε **base1.Go(0.2,0.2)**.

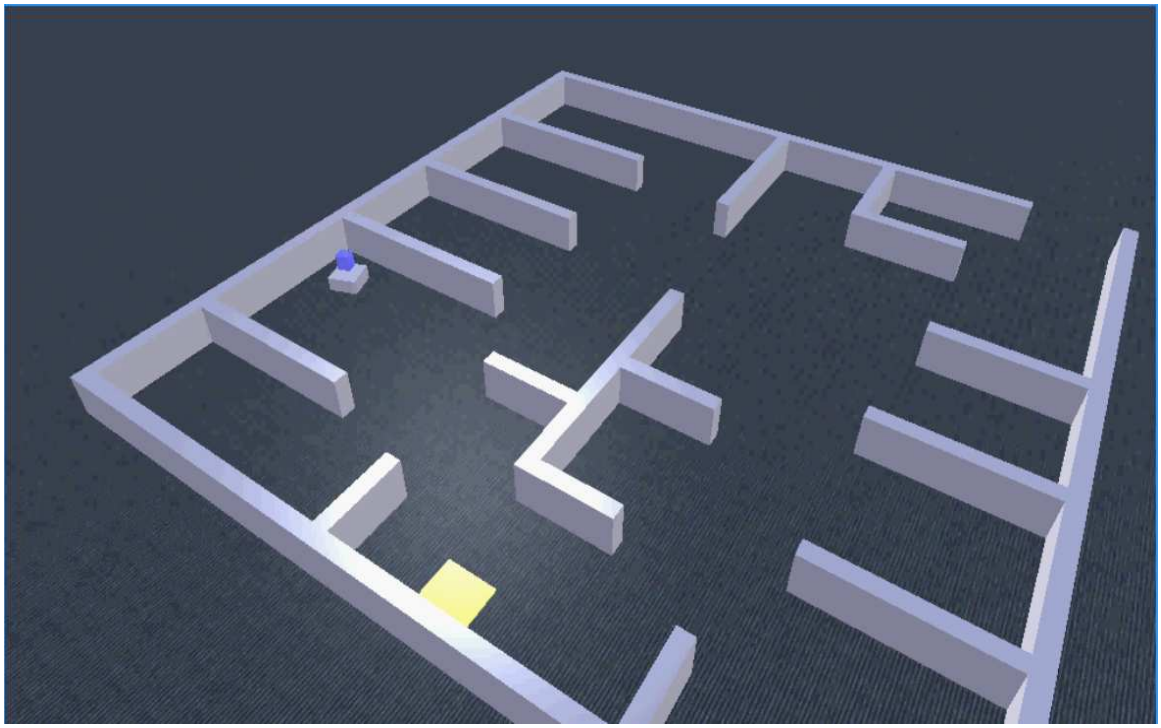
Η άλλη οντότητα που βάλαμε στο simulation η οποία είναι partner(συνεργάτης) του **base1** είναι η **LRF.** Αυτή προσομοιώνει μια ακριβή συσκευή μέτρησης απόστασης. Αυτή την χρησιμοποιούμε για να μετράμε την απόσταση από τον τοίχο. Χρησιμοποιώντας το **SPLSensors** παίρνουμε κάποιες τιμές από την συσκευή αυτή. Οπότε το μόνο που έχουμε να κάνουμε είναι να αποφασίσουμε πως θα κινηθεί το ρομπότ μέσα στον λαβύρινθο ανάλογα με τις τιμές που θα παίρνει.

Εμείς για να βγει από τον λαβύρινθο το ρομπότ θα ακολουθήσουμε τον κανόνα του αριστερού χεριού. Δηλαδή θα πάρουμε τον αριστερό τοίχο και θα τον ακολουθήσουμε μέχρι να βρούμε την έξοδο! Για να το κάνουμε αυτό πρέπει να μετρήσουμε τις αποστάσεις από τους τοίχους γύρω μας. Αν απομακρυνόμαστε από τον αριστερό τοίχο δηλαδή να στρίβουμε αριστερά για να τον ξαναπλησιάζουμε, αν δε ερχόμαστε πολύ κοντά να στρίβουμε δεξιά για να αποφύγουμε σύγκρουση με αυτόν. Το ίδιο θα κάνουμε και αν συναντήσουμε εμπόδιο μπροστά μας. Θα στρίψουμε δεξιά για να το αποφύγουμε.

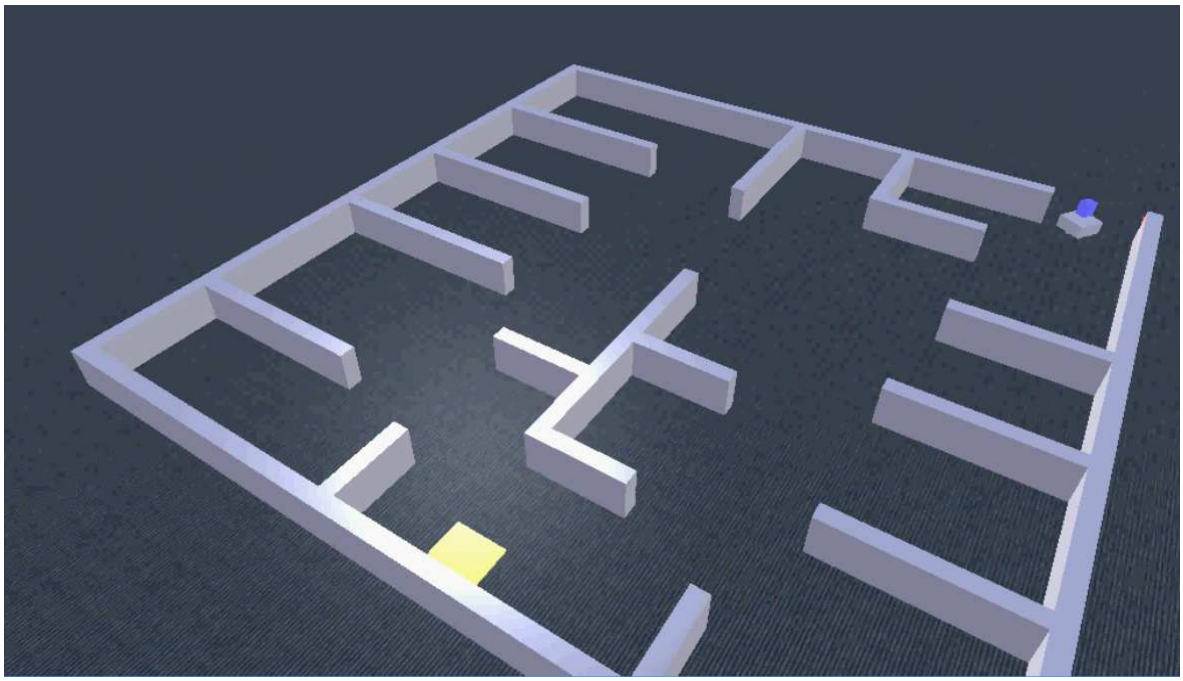
Οπότε για να το κάνουμε αυτό χρησιμοποιούμε την **if** και λέμε σε ποιες περιπτώσεις θέλουμε να στρίψουμε δεξιά; 1) όταν είναι πολύ κοντά στον αριστερό τοίχο 2) όταν βρει μπροστά εμπόδιο. Άρα **if SensorData.DistanceMeasurements[355] < 100** που σημαίνει αν δω τοίχο στις **177,5 μοίρες** (**355/2**) δηλαδή πολύ αριστερά μου και αυτός ο τοίχος είναι πιο κοντά από 10 εκατοστά τότε εκτέλεσε την εξής εντολή : **base1.Turn(-30,0.3)** δηλαδή στρίψε **30 μοίρες** δεξιά με ταχύτητα **0.3** και μετά προχώρησε ευθεία. Το ίδιο κάνε όταν έχεις και **SensorData.DistanceMeasurements[270] < 300** δηλαδή βρεις τοίχο στα 30 εκατοστά κοντά σου ο οποίος να είναι σε γωνία [**270/2**] **135 μοίρες** (δηλαδή περίπου ανάμεσα από την ευθεία σου και την αριστερή σου γωνία) αλλά και το ίδιο κάνε αν βρεις στο 1 μέτρο μπροστά σου (δηλαδή στις **90 μοίρες** [**180/2**]) τοίχο (**SensorData.DistanceMeasurements[180] < 1000**) και στην περίπτωση που βρεις άνοιγμα αριστερά μεγαλύτερο από 1,5 μέτρα (είτε επειδή απομακρύνθηκες από τον αριστερό τοίχο είτε γιατί ο τοίχος κάνει γωνία και πρέπει να πάρεις στροφή να τον ακολουθήσεις) τότε εκτέλεσε το **base1.Turn(30,0.3)** δηλαδή στρίψε αριστερά **30 μοίρες** με ταχύτητα **0.3.** Μετά **base1.GoTo(0.1,0.3)** (δηλαδή πήγαινε μπροστά για 10 εκατοστά) και μετά πάλι ευθεία κανονικά. Τα 10 εκατοστά ευθεία που το βάζουμε να κάνει ούτως ή άλλως είναι για να μην πάρει πολύ κλειστή την ευθεία και πέσει στον τοίχο.

<u>Σημείωση</u> : Στο Παράρτημα μπορείτε να δείτε την υλοποίηση της παραπάνω εφαρμογής σε κώδικα C# με σχόλια.

10.2.2 Εικόνες της προσομοίωσης του προγράμματος



Εικόνα 10.2.2.α: Προσομοίωση ρομπότ που προσπαθεί να βρει την έξοδο σε λαβύρινθο.



Εικόνα 10.2.2.β: Προσομοίωση ρομπότ που βρήκε την έξοδο σε λαβύρινθο.

Παράρτημα

1)Υλοποίηση εφαρμογής ενός ρομπότ που ανιχνεύει και ακολουθεί μία κόκκινη μπάλα καθώς κυλάει σε κώδικα C# με σχόλια.

```
//-------------------------------------------------------
-----------------------
// Simulated Quad Differential Drive Service
//
// Professional Microsoft Robotics Developer Studio
//
//
//          Generated          with:          dssnewservice
/s:SimulatedQuadDifferentialDrive /i:"\Microsoft Robotics
Studio                        (1.5)\bin\RoboticsCommon.dll"
/alt:"http://schemas.microsoft.com/robotics/2006/05/drive
.html"
//
//-------------------------------------------------------
-----------------------
// φόρτωμα βιβλιοθηκών
// κάθε ένα από τα using είναι μια βιβλιοθήκη που
φορτώνει
// για να χρησιμοποιήσει κάποιες εντολές η γλώσσα
// εδώ δεν χρειάζεται να πούμε κάτι παραπάνω μια και για
παράδειγμα η πρώτη //εντολή φροτρώνει τον πυρήνα της CCR
η δεύτερη το DSS και τα λοιπά

using Microsoft.Ccr.Core;
using Microsoft.Dss.Core;
using Microsoft.Dss.Core.Attributes;
using Microsoft.Dss.ServiceModel.Dssp;
using Microsoft.Dss.ServiceModel.DsspServiceBase;
using Microsoft.Dss.Services.SubscriptionManager;
using W3C.Soap;

using System;
using System.Collections.Generic;
using System.ComponentModel;
using dssphttp = Microsoft.Dss.Core.DsspHttp;
using pxdrive = Microsoft.Robotics.Services.Drive.Proxy;
using xml = System.Xml;
using xna = Microsoft.Xna.Framework;
using                        submgr                        =
Microsoft.Dss.Services.SubscriptionManager;
using simtypes = Microsoft.Robotics.Simulation;
using simengine = Microsoft.Robotics.Simulation.Engine;
using physics = Microsoft.Robotics.Simulation.Physics;
using corobot = ProMRDS.Simulation.Corobot;
using Microsoft.Robotics.PhysicalModel;
```

```
//όνομα υπηρεσίας
namespace ProMRDS.Simulation.QuadDifferentialDrive
{

    /// <summary>
    /// Simulated Quad Differential Drive Service
    /// </summary>
    /// κινιτήρας με 4 τροχους
    [DisplayName("Simulated QuadDifferentialDrive")] //το
ονομα που θα εμφανίζεται
    [Description("A four-wheel drive version of the
differential drive service.")] // η περιγραφή του
    [Contract(Contract.Identifier)]// κάθε υπηρεσία
πρέπει να ένα contract και ένα alertnate contract αυτό το
μπλοκ κώδικα τον παίρνουμε όπως και πολλούς άλλους
αυτουσιο από τα παραδείγματα των ρομποτ γιατί είναι
στάνταρ
    [AlternateContract(pxdrive.Contract.Identifier)]
    public class SimulatedQuadDifferentialDriveService :
DsspServiceBase //ονομα κλάσης
    {
        //ορισμός μεταβλητων προσμοίωσης
        #region Simulation Variables
        corobot.CorobotEntity _entity; // ορισμός
μεταβλητής για το ρομπότ όνομα corobot ( το οποίο υπάρχει
στην βιβλιοθήκη της VPL)
        simengine.SimulationEnginePort
_notificationTarget;//Μεταβλητή για τις πόρτες του
κινητήρα του ρομπότ
        #endregion

        /// <summary>
        ///
Microsoft.Robotics.Services.Drive.SimulatedQuadDifferenti
alDrive //Partner στην ουσία εδώ δείχνουμε την υπηρεσία
        /// που θα λειτουργεί μαζί με αυτή και θα
ξεκινήσουν ακόμη και αν το //περιβάλλον προσομοίωσης
είναι άδειο!
        /// </summary>
        [Partner("SubMgr", Contract =
submgr.Contract.Identifier, CreationPolicy =
PartnerCreationPolicy.CreateAlways, Optional = false)]
        private SubscriptionManagerPort _subMgrPort = new
Microsoft.Dss.Services.SubscriptionManager.SubscriptionMa
nagerPort();

        // πόρτα για να δέχεται εντολες του generic
differential drive
        [AlternateServicePort(
            AllowMultipleInstances = true,
```

```
            AlternateContract                                =
pxdrive.Contract.Identifier)]
        private pxdrive.DriveOperations _diffDrivePort =
new
Microsoft.Robotics.Services.Drive.Proxy.DriveOperations()
;

        /// <summary>
        /// κύρια πόρτα για να παίρνει τις εντολες του
quad drive
        /// </summary>
        [ServicePort("/simulatedquaddifferentialdrive",
AllowMultipleInstances = true)]
        private QuadDriveOperations _mainPort = new
QuadDriveOperations();//ονομάζουμε _mainPort και την
ορίζουμε σαν //QuadDriveOperations()

        /// <summary>
        /// αρχική κατάσταση της υπηρεσίας
Simulation.QuadDifferentialDrive.SimulatedQuadDifferentia
lDriveService
        /// </summary>
        [InitialStatePartner(Optional = true, ServiceUri
= "SimulatedQuadDifferentialDriveService.Config.xml")]
        private DriveDifferentialFourWheelState _state =
new DriveDifferentialFourWheelState();// το ίδιο και εδώ
ορίζουμε νέα DriveDifferentialFourWheelState
     //Ο λόγος που το κάναμε αυτό είναι αφου ορίσαμε τι
τυπος και τι θα είναι οι κλάσεις παραπάνω τις δίνουμε και
ένα όνομα το οποίο για να θυμόμαστε πιο είναι είναι το
ίδιο με την κλάση
        /// <summary>
        /// κλάση για να κατασκευάζει πόρτες η υπηρεσία
        /// </summary>
        public
SimulatedQuadDifferentialDriveService(DsspServiceCreation
Port creationPort)
            :
                base(creationPort)//Χρησιμοποιούμε αυτό
γιατί φτιάχνουμε πάνω από μια πόρτες
        {
        }

        #region Initialization and State
        /// <summary>
        /// εδώ ξεκινάει να τρέχει η υπηρεσία
        /// </summary>
        protected override void Start()
        {
            if (_state == null)//αν δεν αλλάξει η
κατάσταση
```

```
                CreateDefaultState();//ξεκίνα          την
προεπιλεγμένη κατάσταση

            _notificationTarget           =           new
simengine.SimulationEnginePort();//άλλαξε το notification

            // PartnerType.Service is the entity instance
name.

simengine.SimulationEngine.GlobalInstancePort.Subscribe(S
erviceInfo.PartnerList, _notificationTarget);
            //από την στιγμή που δεν αλλάζει η κατάσταση
"πετάει" όλα τα πακέτα που λαμβάνει στις πόρτες


//επεξήγηση του Arbiter που χρησιμοποιούμε παρακάτω.. Οι
Arbiter εντολες μας //επιτρέπουν να επιμάσει τον κώσικα
μέσα προς εκτέλεση.. αν δεν το κάναμε αυτό //και δεν
έκανε drop τα πακέτα τότε θα είχαμε πρόβλημα με την μνήμη
του PC μας
// Αυτό το πετυχαινουμε με το Activate και το κάθε πακέτο
μέσα στην Arbiter
            Activate(new Interleave(
                new TeardownReceiverGroup
                (

Arbiter.Receive<simengine.InsertSimulationEntity>(false,
_notificationTarget,
InsertEntityNotificationHandlerFirstTime),

Arbiter.Receive<DsspDefaultDrop>(false,         _mainPort,
DefaultDropHandler),

Arbiter.Receive<DsspDefaultDrop>(false,    _diffDrivePort,
DefaultDropHandler)
                ),
                new ExclusiveReceiverGroup(),// και αυφου
πετάει τις τιμές που δεν χρειαζόμαστε ξαναφτιάχνει νέο
ExclusiveReceiverGroup και ConcurrentReceiverGroup
                new ConcurrentReceiverGroup()
            ));
        }
        //εδώ ο κώδικας της προεπιλεγμένης κατάστασης
        void CreateDefaultState()
        {
            //για    κάθε    ρόδα    ξεχωριστά    ορίζουμε
κατάσταση(ρόδας και κινητήρα)
            //ο κώδικας αυτός καλείται από την start
μέθοδο αν δεν έχουμε αρχική διαμόρφωση
            _state                  =                  new
DriveDifferentialFourWheelState();
```

```
            _state.LeftWheel              =              new
Microsoft.Robotics.Services.Motor.Proxy.WheeledMotorState
();
            _state.RightWheel             =              new
Microsoft.Robotics.Services.Motor.Proxy.WheeledMotorState
();
            _state.LeftWheel.MotorState        =        new
Microsoft.Robotics.Services.Motor.Proxy.MotorState();
            _state.RightWheel.MotorState       =        new
Microsoft.Robotics.Services.Motor.Proxy.MotorState();
            _state.LeftWheel.EncoderState      =        new
Microsoft.Robotics.Services.Encoder.Proxy.EncoderState();
            _state.RightWheel.EncoderState     =        new
Microsoft.Robotics.Services.Encoder.Proxy.EncoderState();
            _state.RearLeftWheel           =           new
Microsoft.Robotics.Services.Motor.Proxy.WheeledMotorState
();
            _state.RearRightWheel          =           new
Microsoft.Robotics.Services.Motor.Proxy.WheeledMotorState
();
            _state.RearLeftWheel.MotorState     =      new
Microsoft.Robotics.Services.Motor.Proxy.MotorState();
            _state.RearRightWheel.MotorState    =      new
Microsoft.Robotics.Services.Motor.Proxy.MotorState();
        }

        void UpdateStateFromSimulation()//αντίθετα αυτή η
μέθοδος καλείται για να αναβαθμίσει
            // την κατάσταση πρίν γίνει η εκπομπή
        {
            if  (_entity  !=  null)//αν  δηλαδή  έχουμε
αλλαγές ( αν δεν είναι κενό //το _entity
            {
                _state.TimeStamp                        =
DateTime.Now;//αναβάθμισε-δες την τωρινή //κατάσταση της
κάθε ρόδας του ρομποτ
                _state.LeftWheel.MotorState.CurrentPower
= _entity.FrontLeftWheel.Wheel.MotorTorque;
                _state.RightWheel.MotorState.CurrentPower
= _entity.FrontRightWheel.Wheel.MotorTorque;

_state.RearLeftWheel.MotorState.CurrentPower            =
_entity.RearLeftWheel.Wheel.MotorTorque;

_state.RearRightWheel.MotorState.CurrentPower           =
_entity.RearRightWheel.Wheel.MotorTorque;

_state.LeftWheel.EncoderState.CurrentReading            =
_entity.FrontLeftEncoder;
```

```
_state.RightWheel.EncoderState.CurrentReading =
_entity.FrontRightEncoder;
                _state.Position =
_entity.State.Pose.Position;
            }
        }
        #endregion
        //εδώ φτιάχνουμε τις συνδέσεις μεταξύ των θυρών
        #region Entity Handlers
        void
InsertEntityNotificationHandlerFirstTime(simengine.Insert
SimulationEntity ins)
        {
            //η υπηρεσία γράφεται στην τοπική θύρα
            DirectoryInsert();

            InsertEntityNotificationHandler(ins);


//SpawnIterator(TestDriveDistanceAndRotateDegrees);
        }

        //εδώ αποθηκευουμε μια αναφορά για το entity έτσι
ώστε να μπορεί να χρησιμοποιηθεί και από άλλες
        //μεθόδους
        void
InsertEntityNotificationHandler(simengine.InsertSimulatio
nEntity ins)
        {
            _entity = (corobot.CorobotEntity)ins.Body;
            _entity.ServiceContract =
Contract.Identifier;//  δείχνουμε  ότι  ελένχεται  από
ελέγχεται από την υπηρεσία

            //ετοιμάζουμε την αρχική κατάσταση του ρομπότ
βάση της φυσικής οντότητάς του
            //αυτή η κατάσταση δεν θα αλλάξει με το πέρας
του χρόνου παρα μόνο τώρα.
            xna.Vector3          separation =
_entity.FrontLeftWheel.Position -
_entity.FrontRightWheel.Position;
            _state.DistanceBetweenWheels =
separation.Length();


_state.LeftWheel.MotorState.PowerScalingFactor =
_entity.MotorTorqueScaling;

_state.RightWheel.MotorState.PowerScalingFactor =
_entity.MotorTorqueScaling;
```

```
            // ενεργοποιούμε και τις άλλες μεθόδους τώρα
που έγινε η σύνδεση
            Activate(new Interleave(
                new    TeardownReceiverGroup//    η    ίδια
δουλειά με τους arbiterw από επάνω( εξηγήσαμε την
χρησιμότητά τους)
                (

Arbiter.Receive<DsspDefaultDrop>(false,         _mainPort,
DefaultDropHandler),

Arbiter.Receive<DsspDefaultDrop>(false,    _diffDrivePort,
DefaultDropHandler)
                ),
                new ExclusiveReceiverGroup
                (
                    Arbiter.Receive<SetPose>(true,
_mainPort, SetPoseHandler),

Arbiter.ReceiveWithIterator<pxdrive.DriveDistance>(true,
_diffDrivePort, DriveDistanceHandler),

Arbiter.ReceiveWithIterator<pxdrive.RotateDegrees>(true,
_diffDrivePort, RotateHandler),

Arbiter.ReceiveWithIterator<pxdrive.SetDrivePower>(true,
_diffDrivePort, SetPowerHandler),

Arbiter.ReceiveWithIterator<pxdrive.SetDriveSpeed>(true,
_diffDrivePort, SetSpeedHandler),

Arbiter.ReceiveWithIterator<pxdrive.AllStop>(true,
_diffDrivePort, AllStopHandler),

Arbiter.Receive<simengine.InsertSimulationEntity>(true,
_notificationTarget,
InsertEntityNotificationHandler),//αυτό το ορίζουμε έτσι
ώστε να δουλευει και να ανανεώνεται η κατάσταση ακόμη

Arbiter.Receive<simengine.DeleteSimulationEntity>(true,
_notificationTarget,  DeleteEntityNotificationHandler)  //
και αν σβήσουμε ή βάλουμε και άλλεες φυσικές οντότητες
στην προσομοίωση
                ),
                new   ConcurrentReceiverGroup//   η   ίδια
δουλειά με τους arbiterw από επάνω( εξηγήσαμε την
χρησιμότητά τους)

                (
```

```
Arbiter.ReceiveWithIterator<dssphttp.HttpGet>(true,
_mainPort, MainPortHttpGetHandler),

Arbiter.ReceiveWithIterator<Get>(true,          _mainPort,
MainPortGetHandler),

Arbiter.ReceiveWithIterator<dssphttp.HttpGet>(true,
_diffDrivePort, HttpGetHandler),

Arbiter.ReceiveWithIterator<pxdrive.Get>(true,
_diffDrivePort, GetHandler),

Arbiter.ReceiveWithIterator<pxdrive.Subscribe>(true,
_diffDrivePort, SubscribeHandler),

Arbiter.ReceiveWithIterator<pxdrive.ReliableSubscribe>(tr
ue, _diffDrivePort, ReliableSubscribeHandler),

Arbiter.ReceiveWithIterator<pxdrive.EnableDrive>(true,
_diffDrivePort, EnableHandler)
                )
            ));
        }

        void
DeleteEntityNotificationHandler(simengine.DeleteSimulatio
nEntity del)
        {
            _entity = null;

            // η μέθοδος αυτή σετάρει το global_entity σε
null (κενό) και απενεργοποιεί όλα τα άλλα σήματα εκτός
από το
            // drop και το InsertEntityNotification
            Activate(new Interleave(
                new TeardownReceiverGroup
                (

Arbiter.Receive<simengine.InsertSimulationEntity>(false,
_notificationTarget,
InsertEntityNotificationHandlerFirstTime),

Arbiter.Receive<DsspDefaultDrop>(false,         _mainPort,
DefaultDropHandler),

Arbiter.Receive<DsspDefaultDrop>(false,    _diffDrivePort,
DefaultDropHandler)
                ),
                new ExclusiveReceiverGroup(),
                new ConcurrentReceiverGroup()
```

```
            ));
        }
        #endregion

        #region Subscribe Handling
        /// <summary>
        ///  εδώ  το  μόνο  που  μας  μένει  είναι  να
εφαρμόσουμε το πως θα χειριστεί //το κάθε
        /// μήνυμα κάθε πόρτα!
        /// </summary>
        /// <param name="subscribe"></param>
        /// <returns></returns>
        ///    //στα    πρώτα    2    χρησιμοποιούμε    το
SubscriptionManager για να χειριστουν αιτήματα από αλλες
        /// //υπηρεσίες


//Εδώ θα αναφερθούμε μια φορά σε κάθε εντολή με την
επεξήγησή της και την χρησιμότητά της μια και
επαναλαμβάνονται μέχρι το τέλος του κώδικά..
//    Συγκεκριμένα    έχουμε    τις    εξής    εντολές    :
IEnumerator<ITask>,  (Activate  και  Arbiter  εξηγήθηκαν
παραπάνω), delegate, yield break και throw

IEnumerator<ITask> = ένα Task περιέχει την αναφορά σε
έναν handler (handler= κομμάτια κώδικα που τρέχουν σε
multi threaded περιβάλλον)η CCR χρησιμοποιεί το ITask
interface, αυτό το κάνει για να έχει ο κώδικας μια δομή
έτοιμη να αποθηκευθεί και να μεταφερθεί, οπότε βάζεις τον
κώδικά σαν ένα Itask πριν φύγει από την υπηρεσία σαν
μήνυμα. IEnumarator είναι ένα τμήμα κώδικα το οποιο
επιστρέφει μια ακολουθία τιμών σε σειρά του ίδιου τύπου (
εδώ ορίζουμε πάντα τον τύπο ITask) ο λόγος που
χρησιμοποιούμε το IEnumarator σαν ένα βρόχο επανάληψης
στην ουσία και όχι έναν κοινό βρόχο επανάληψης είναι
γιατί ο IEnumarator λειτουργεί ασύχρονα και μπορεί να
κρατάει τις τιμές οσότου του ζητήθούν στην ουσία έχει
συμπεριφορά FIFO που αυτή θέλουμε για τα μηνύματά μας

Arbiter = ορίζει πότε οι συνθήκες είναι κατάλληλες για να
φύγει ένα μήνυμα (εξηγήθηκε με το Activate)

Delegate είναι μια μέθοδος θυμίζει τον pointer της C αλλά
χωρίς τον ακριβή ορισμό της.. εδώ χρησιμοποιείται για να
χειρίζεται τα success μηνύματα (να στέλνει μήνυμα success
στην προεπιλεγμένη πόρτα) καθώς και τα fault μηνυματα σε
αυτή την περίπτωση κάνει log to error και στέλνει flag
ότι η κίνηση απέτυχε.

Yield Break= εδώ τερματίζει την λειτουργία του βρόχου
επανάληψης ( του IEnumarator)
```

Throw= η έκφραση throw χρησιμοποιείται για να δηλώσει μια μη επιθημητή κατάσταση την ώρα που εκτελείται ο κώδικάς για παράδειγμα :

```
public void SetPoseHandler(SetPose setPose)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation      entity      not
registered with service");
```

Αν η _entity για την υπηρεσία αυτή είναι κενή τότε πέτα μήνυμα ότι δεν έχει //γίνει η δήλωση της υπηρεσίας στην προσομίωση

```
        public                          IEnumerator<ITask>
SubscribeHandler(pxdrive.Subscribe subscribe)
        {
            Activate(Arbiter.Choice(
                SubscribeHelper(_subMgrPort,
subscribe.Body, subscribe.ResponsePort),
                delegate(SuccessResult success)
                {
                    _subMgrPort.Post(new submgr.Submit(
                        subscribe.Body.Subscriber,
DsspActions.UpdateRequest, _state, null));
                },
                delegate(Exception ex) { LogError(ex); }
            ));

            yield break;
        }

        public                          IEnumerator<ITask>
ReliableSubscribeHandler(pxdrive.ReliableSubscribe
subscribe)
        {
            Activate(Arbiter.Choice(
                SubscribeHelper(_subMgrPort,
subscribe.Body, subscribe.ResponsePort),
                delegate(SuccessResult success)
                {
                    _subMgrPort.Post(new submgr.Submit(
                        subscribe.Body.Subscriber,
DsspActions.UpdateRequest, _state, null));
                },
                delegate(Exception ex) { LogError(ex); }
            ));
            yield break;
        }
        #endregion
```

```
        #region _mainPort handlers

        //τα επόμενα 2 υποστηρίζουν την get και την
Httpget αίτηση στην main πόρτα
        public                          IEnumerator<ITask>
MainPortHttpGetHandler(dssphttp.HttpGet get)
        {
            UpdateStateFromSimulation();
            get.ResponsePort.Post(new
dssphttp.HttpResponseType(_state));
            yield break;
        }

        public  IEnumerator<ITask>  MainPortGetHandler(Get
get)
        {
            UpdateStateFromSimulation();
            get.ResponsePort.Post(_state);
            yield break;
        }
        //εδώ υποστηρίζεται η SetPost αίτηση
        public void SetPoseHandler(SetPose setPose)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation      entity      not
registered with service");

            Task<corobot.CorobotEntity, Pose> task = new
Task<corobot.CorobotEntity,                 Pose>(_entity,
setPose.Body.EntityPose, SetPoseDeferred);
            _entity.DeferredTaskQueue.Post(task);
        }

        void        SetPoseDeferred(corobot.CorobotEntity
entity, Pose pose)
        {
            entity.PhysicsEntity.SetPose(pose);
        }
        //αν είναι συνεδεμένο δημιουργεί μία εργασία η
οποία καθορίζει την στάση ανάλογα με
        //το αίτημα. Η εργασία αυτή θα προσθεθεί σε σειρά
στο DeferredTaskQueue με σκοπό
        //να εκτελεστεί αν και εφόσον η physics μηχανή
ΔΕΝ είναι σε λειτουργία.
        #endregion

        #region _diffDrivePort handlers
        //εδώ κάνουμε την ίδια δουλεια με επάνω αλλά αντί
για την main πόρτα
        //χρησιμοποιούμε την _diffDrive
```

```
        public                          IEnumerator<ITask>
HttpGetHandler(dssphttp.HttpGet get)
        {
            UpdateStateFromSimulation();
            pxdrive.DriveDifferentialTwoWheelState
_twoWheelState =

(pxdrive.DriveDifferentialTwoWheelState)((pxdrive.DriveDi
fferentialTwoWheelState)_state).Clone();
            get.ResponsePort.Post(new
dssphttp.HttpResponseType(_twoWheelState));
            yield break;
        }

        public  IEnumerator<ITask>  GetHandler(pxdrive.Get
get)
        {
            UpdateStateFromSimulation();
            pxdrive.DriveDifferentialTwoWheelState
_twoWheelState =

(pxdrive.DriveDifferentialTwoWheelState)((pxdrive.DriveDi
fferentialTwoWheelState)_state).Clone();
            get.ResponsePort.Post(_twoWheelState);
            yield break;
        }
        //εδώ προωθούμε τα αιτίματα καλώντας είτε την
DriveDistanse ή
        //RotateDegrees (περιμένουν πρώτα την ασύχρονη
απάντηση πριν στείλουν μηνυμα)
        public                          IEnumerator<ITask>
DriveDistanceHandler(pxdrive.DriveDistance driveDistance)
        {
            SuccessFailurePort   entityResponse   =   new
SuccessFailurePort();
            _entity.DriveDistance(
                (float)driveDistance.Body.Distance,
                (float)driveDistance.Body.Power,
                entityResponse);

            yield return Arbiter.Choice(entityResponse,
                delegate(SuccessResult        s)        {
driveDistance.ResponsePort.Post(DefaultUpdateResponseType
.Instance); },
                delegate(Exception e)
                {
                    driveDistance.ResponsePort.Post(new
W3C.Soap.Fault());
                });

            yield break;
```

```
        }

        public                          IEnumerator<ITask>
RotateHandler(pxdrive.RotateDegrees rotate)
        {
            SuccessFailurePort   entityResponse   =   new
SuccessFailurePort();
            _entity.RotateDegrees(
                (float)rotate.Body.Degrees,
                (float)rotate.Body.Power,
                entityResponse);

            yield return Arbiter.Choice(entityResponse,
                delegate(SuccessResult        s)        {
rotate.ResponsePort.Post(DefaultUpdateResponseType.Instan
ce); },
                delegate(Exception e)
                {
                    rotate.ResponsePort.Post(new
W3C.Soap.Fault());
                });

            yield break;
        }
        //εδώ αυτές οι μέθοδοι αλλάζουν την γωνιακή
ταχύτητα των τροχών και στέλνουν
        //μια ειδοποίηση σε όλες τις υπερεσίες που είναι
συνεδεμένες ότι η καταστασή τους
        // έχει αλλάξει
        public                          IEnumerator<ITask>
SetPowerHandler(pxdrive.SetDrivePower setPower)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation     entity     not
registered with service");

            // Call simulation entity method for setting
wheel torque
            _entity.SetMotorTorque(
                (float)(setPower.Body.LeftWheelPower),
                (float)(setPower.Body.RightWheelPower));

            UpdateStateFromSimulation();

setPower.ResponsePort.Post(DefaultUpdateResponseType.Inst
ance);

            // εδώ η ειδοποίση ότι άλλαξε κατάσταση
            _subMgrPort.Post(new     submgr.Submit(_state,
DsspActions.UpdateRequest));
```

```
            yield break;
        }

        public                          IEnumerator<ITask>
SetSpeedHandler(pxdrive.SetDriveSpeed setSpeed)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation      entity      not
registered with service");

            _entity.SetVelocity(
                (float)setSpeed.Body.LeftWheelSpeed,
                (float)setSpeed.Body.RightWheelSpeed);

            UpdateStateFromSimulation();

setSpeed.ResponsePort.Post(DefaultUpdateResponseType.Inst
ance);

            // εδώ η ειδοποίση ότι άλλαξε κατάσταση
            _subMgrPort.Post(new     submgr.Submit(_state,
DsspActions.UpdateRequest));
            yield break;
        }
        //εδώ στέλνει το μήνυμα ενεργοποίησης ανάλογα με
την αίτηση και
        //ειδοποιεί για αλλάγή κατάστασης.
        public                          IEnumerator<ITask>
EnableHandler(pxdrive.EnableDrive enable)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation      entity      not
registered with service");

            _state.IsEnabled = enable.Body.Enable;
            _entity.IsEnabled = _state.IsEnabled;

            UpdateStateFromSimulation();

enable.ResponsePort.Post(DefaultUpdateResponseType.Instan
ce);

            // εδώ η ειδοποίση ότι άλλαξε κατάσταση
            _subMgrPort.Post(new     submgr.Submit(_state,
DsspActions.UpdateRequest));
            yield break;
        }
        //τέλος η AllStopHandler η οποία θέτει την
γωνιακή ταχύτητα στο 0
```

```
        //και φυσικά ενημερώνει για την αλλαγή καταστασής
της..
        public                          IEnumerator<ITask>
AllStopHandler(pxdrive.AllStop estop)
        {
            if (_entity == null)
                throw                                   new
InvalidOperationException("Simulation      entity      not
registered with service");

            _entity.SetMotorTorque(0, 0);
            _entity.SetVelocity(0);

            UpdateStateFromSimulation();

estop.ResponsePort.Post(DefaultUpdateResponseType.Instanc
e);

            // εδώ η ειδοποίση ότι άλλαξε κατάσταση
            _subMgrPort.Post(new     submgr.Submit(_state,
DsspActions.UpdateRequest));
            yield break;
        }
        #endregion


    }
}
```

2)Υλοποίηση εφαρμογής ενός ρομπότ που βρίσκει την έξοδο ενός λαβυρίνθου σε κώδικα C# με σχόλια.

```
//-------------------------------------------------------
----------------
//   This file is part of Microsoft Robotics Developer
Studio Code Samples.
//
//   Copyright (C) Microsoft Corporation.   All rights
reserved.
//
//  $File: SimulatedLRF.cs $ $Revision: 1 $
//-------------------------------------------------------
----------------
// φόρτωμα βιβλιοθηκών
using System;
using System.Collections.Generic;
using Microsoft.Ccr.Core;
using Microsoft.Dss.Core;
using Microsoft.Dss.Core.Attributes;
using permissions = System.Security.Permissions;
using                      svcbase                      =
Microsoft.Dss.ServiceModel.DsspServiceBase;
using dssp = Microsoft.Dss.ServiceModel.Dssp;
using                      sicklrf                      =
Microsoft.Robotics.Services.Sensors.SickLRF.Proxy;
using                       submgr                      =
Microsoft.Dss.Services.SubscriptionManager;
using simtypes = Microsoft.Robotics.Simulation;
using simengine = Microsoft.Robotics.Simulation.Engine;
using physics = Microsoft.Robotics.Simulation.Physics;
using Microsoft.Robotics.PhysicalModel;
using System.ComponentModel;
using Microsoft.Dss.Core.DsspHttp;
using System.Net;
//Το όνομα του προγράμματος
namespace
Microsoft.Robotics.Services.Simulation.Sensors.LaserRange
Finder
{
    /// <summary>
    /// Το πρόγραμμα αυτό μας δίνει πρόσβαση σε ένα
    /// εικονικό λέιζερ μετρητή απόστασης
    /// χρησιμοποιώντας την μηχανή φυσικής του εξομοιωτή
αλλά και την φυσική οντότητα του λειζερ
    /// που έχουμε βάλει εμείς στον εξομοιωτή στην ουσία
εδώ δείχνουμε την υπηρεσία
    /// που θα λειτουργεί μαζί με αυτή και θα ξεκινήσουν
ακόμη και αν το περιβάλλον προσομοίωσης είναι άδειο!
```

```
    /// </summary>
    [DisplayName("(User) Simulated Laser Range Finder")]
    [Description("Provides access to a simulated laser
range finder.\n(Uses the Sick Laser Range Finder
contract.)")]
    [AlternateContract(sicklrf.Contract.Identifier)]
    [Contract(Contract.Identifier)]

[DssServiceDescription("http://msdn.microsoft.com/library
/cc998493.aspx")]
    public class SimulatedLRFService :
svcbase.DsspServiceBase
    {
        #region Simulation Variables
        simengine.LaserRangeFinderEntity _entity;
        simengine.SimulationEnginePort
_notificationTarget;
        physics.PhysicsEngine _physicsEngine;
        Port<physics.RaycastResult> _raycastResults = new
Port<physics.RaycastResult>();
        #endregion

        private sicklrf.State _state = new
sicklrf.State();

        [ServicePort("/SimulatedLRF",
AllowMultipleInstances=true)]
        private sicklrf.SickLRFOperations _mainPort = new
sicklrf.SickLRFOperations();

        [Partner("SubMgr", Contract =
submgr.Contract.Identifier, CreationPolicy =
PartnerCreationPolicy.CreateAlways)]
        submgr.SubscriptionManagerPort _subMgrPort = new
submgr.SubscriptionManagerPort();

        /// <summary>
        /// Φτιάχνει μια θύρα για να ειδοποιήσει πότε η
υπηρεσία ξεκινάει
        /// </summary>
        /// <param name="creationPort"></param>
        public
SimulatedLRFService(dssp.DsspServiceCreationPort
creationPort) :
                base(creationPort)
        {
        }

        /// <summary>
        /// Η start κλάση ξεκινά την SimulatedLRFService
υπηρεσία και περιμένει για μηνύματα
```

```
        /// </summary>
        protected override void Start()
        {
            _physicsEngine                                =
physics.PhysicsEngine.GlobalInstance;
            _notificationTarget          =          new
simengine.SimulationEnginePort();

            // PartnerType.Service is the entity instance
name.

simengine.SimulationEngine.GlobalInstancePort.Subscribe(S
erviceInfo.PartnerList, _notificationTarget);


            //με αυτόν τον κώδικά εδω αυτό που κάνουμε
είναι να ελένχουμε
            //αν έχει έρθει η ειδοποίηση από την υπηρεσία
ότι ξεκίνησε, αν είναι false λοιπόν,
            //τότε πολύ απλά τα μόνα μηνύματα που αφήνει
να
            // περάσουν είναι τα drop μηνύματα για της
θύρες έτσι ώστε να μην έχουμε
            // έλλειμα μνήμης
            Activate(new Interleave(
                new TeardownReceiverGroup
                (

Arbiter.Receive<simengine.InsertSimulationEntity>(false,
_notificationTarget,
InsertEntityNotificationHandlerFirstTime),

Arbiter.Receive<dssp.DsspDefaultDrop>(false,    _mainPort,
DefaultDropHandler)
                ),
                new ExclusiveReceiverGroup(),
                new ConcurrentReceiverGroup()
            ));

        }
        //εδώ έχουμε μια σταθερά! ότι η μέγιστη απόσταση
του laser θα είναι 8 μέτρα!
        const float LASER_RANGE = 8f;
        //εδώ φτιάχνουμε την προεπιλεγμένη κατάσταση του
LaserRangeFinder[LRF]!
        //1) δηλώνουμε λοιπόν ότι θα έχουμε σαν μονάδα
μέτρησης τα χιλιοστά
        //2) ότι το ευρος του LRF θα είναι 180 μοίρες,
        //3) τέλος ότι η ανάλυση θα είναι σε 0,5 μοίρες
        void CreateDefaultState()
        {
```

```
            _state.Units = sicklrf.Units.Millimeters;
            _state.AngularRange = 180;
            _state.AngularResolution = 0.5f;
        }
        //η κλάση για να σβήνουμε την ειδοποίηση
        void
DeleteEntityNotificationHandler(simengine.DeleteSimulatio
nEntity del)
        {
            _entity = null;// απλά βάζουμε κενό στην
οντότητα
        }
        //εδώ ορίζουμε την πρώτη φορά που θα τρέξει η
υπηρεσία
        void
InsertEntityNotificationHandlerFirstTime(simengine.Insert
SimulationEntity ins)
        {
            InsertEntityNotificationHandler(ins);
            base.Start();
            MainPortInterleave.CombineWith(
                new Interleave(
                    new TeardownReceiverGroup(),
                    new ExclusiveReceiverGroup(

Arbiter.Receive<simengine.InsertSimulationEntity>(true,
_notificationTarget, InsertEntityNotificationHandler),

Arbiter.Receive<simengine.DeleteSimulationEntity>(true,
_notificationTarget, DeleteEntityNotificationHandler)
                    ),
                    new ConcurrentReceiverGroup()
                )
            );
        }
        // και εδώ για κάθε φορά που θα τρέχει
        void
InsertEntityNotificationHandler(simengine.InsertSimulatio
nEntity ins)
        {
            _entity                                    =
(simengine.LaserRangeFinderEntity)ins.Body;
            _entity.ServiceContract                    =
Contract.Identifier;

            CreateDefaultState();

            physics.RaycastProperties raycastProperties =
new physics.RaycastProperties();
            raycastProperties.StartAngle        =        -
_state.AngularRange / 2.0f;
```

```
            raycastProperties.EndAngle                    =
_state.AngularRange / 2.0f;
            raycastProperties.AngleIncrement              =
(float)_state.AngularResolution;
            raycastProperties.Range = LASER_RANGE;
            raycastProperties.OriginPose = new Pose();

            _entity.RaycastProperties                     =
raycastProperties;
            _entity.Register(_raycastResults);

            // Ενεργοποιούμε την υπηρεσία επανάληψης για
να δέχεται τα αποτελέσματα από τις αχτίνες
            // ο κώδικας και η όλη λειτουργία εξηγήτε στο
παρακάτω σχόλιο
            Activate(Arbiter.Receive(true,
_raycastResults, RaycastResultsHandler));
        }
        //ΠΡΟΣΟΧΗ ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ του πως δουλευει το LRF
        //Η LaserRangeFinderEntity εξομοιώνει έναν
ανιχνευτή ακτίνας λέιζερ με χύτευση ακτίνων στο
προσωμοιομένο
        //περιβάλλον που αντιστοιχούν σε δείγματα λέιζερ.
Αυτές οι ακτίνες διασταυρώνεται με τα φυσικά σχήματα
        //στο περιβάλλον, και μια σειρά από αποστάσεις
επιστρέφεται. Μικροί, κόκκινοι κύκλοι που αναβοσβήνουν,
        //φαίνονται στο περιβάλλον σε κάθε σημείο τομής.
Μπορούμε να παρατηρήσουμε ότι αυτά τα σημεία ορισμένες
φορές είναι
        //στον αέρα! Αυτό γίνεται επειδή το σχήμα φυσικής
που αντιπροσωπεύει μία οντότητα είναι
        //διαφορετική από την οπτική είκόνα που έχουμε σε
αυτό που αποτελεί την οντότητα. Με λίγα λόγια μπορεί να
έχουμε ένα
        // αντικείμενο το οποίο να φαινεται σε εμάς
κυκλικό αλλά στην πραγματικότητα στην μηχανή φυσικής να
έχει άλλο σχήμα
        //έστω ορθογώνιο (συνήθως για χάρην απλότητας και
ευκολίας) τότε λοιπόν το λέιζερ θα το δει ώς ορθογώνιο
και
        //οι ακτίνες που θα πε΄σουν επάνω του θα
συμπεριφερθουν ανάλογα!
        // (Γενικά το λέιζερ λειτουργεί πάντα σε σχήματα
φυσικής και όχι της οπτικής εικόνας που έχουμε εμεις)

        private                                    void
RaycastResultsHandler(physics.RaycastResult result)
        {
            //εδώ μόλις παίρνουμε τα αποτελέσματα από την
μηχανη φυσικής απλά χρησιμοποιούμε
```

```
            // την μέτρηση απόστασης από κάθε σημείο που
έγινε η επαφή! (ΑΛΛΑ η μηχανή σου
            //διαθέτει και την επιλογή να ελένξεις και
τις ιδιότητες του κάθε υλικου!
            //Αρα μπορεις να επιλέξεις αν οι ακτίνες θα
κάνουν ανάκλαση, αν θα έρθουν πίσω με θόρυβο
            //αν θα διαθλαστουν κ.τ.λ

            sicklrf.State     latestResults     =     new
sicklrf.State();
            latestResults.DistanceMeasurements    =    new
int[result.SampleCount+1];
            int initValue = (int)(LASER_RANGE * 1000f);
            //φυσικά εδώ σκανάρει για κάθε μια ακτίνα
ξεχωριστά ( παίρνει περίπου αποτελέσματα κάθε
            //δευτερόλεπτο)
            for (int i = 0; i < (result.SampleCount + 1);
i++)
                latestResults.DistanceMeasurements[i]    =
initValue;

            foreach  (physics.RaycastImpactPoint  pt  in
result.ImpactPoints)
            {
                // Η απόσταση από το σημείο που έγινε η
επαφή
                // έχει προυπολογιστεί από την θέση που
είναι το LRF

latestResults.DistanceMeasurements[pt.ReadingIndex]      =
(int)(pt.Position.W * 1000f);
            }

            latestResults.AngularRange                   =
(int)Math.Abs(_entity.RaycastProperties.EndAngle         -
_entity.RaycastProperties.StartAngle);
            latestResults.AngularResolution              =
_entity.RaycastProperties.AngleIncrement;
            latestResults.Units                          =
sicklrf.Units.Millimeters;
            latestResults.LinkState     =     "Measurement
received";
            latestResults.TimeStamp = DateTime.Now;

            //    στέλνει    στον    εαυτό    του    μήνυμα
αντικατάστασης
            sicklrf.Replace       replace       =       new
sicklrf.Replace();
            // αφου στέλνει η υπηρεσία στον εαυτό της
μήνυμα αντικατάστασης δεν χρειάζεται να
```

```
            // δώσουμε την πόρτα λήψης του μηνυματος, την
αφήνουμε κενή (null)!
            replace.ResponsePort = null;
            replace.Body = latestResults;
            _mainPort.Post(replace);
        }
        // εδώ αρχίζει ο κώδικας επανάληψης για όλες τις
υπηρεσίες
        //Εδώ θα αναφερθούμε μια φορά σε κάθε εντολή με
την επεξήγησή της και την χρησιμότητά της μια και
επαναλαμβάνονται μέχρι το τέλος του κώδικά..
        // Συγκεκριμένα έχουμε τις εξής εντολές :
IEnumerator<ITask>, (Activate και Arbiter εξηγήθηκαν
παραπάνω), delegate, yield break και throw

        //IEnumerator<ITask> = ένα Task περιέχει την
αναφορά σε έναν handler (handler= κομμάτια
        // κώδικα που τρέχουν σε multi threaded
περιβάλλον)η CCR χρησιμοποιεί το ITask
        //interface, αυτό το κάνει για να έχει ο κώδικας
μια δομή έτοιμη να αποθηκευθεί και
        //να μεταφερθεί, οπότε βάζεις τον κώδικά σαν ένα
Itask πριν φύγει από την υπηρεσία σαν μήνυμα.
        //IEnumarator είναι ένα τμήμα κώδικα το οποιο
επιστρέφει μια ακολουθία τιμών σε σειρά του ίδιου τύπου
        //( εδώ ορίζουμε πάντα τον τύπο ITask) ο λόγος
που χρησιμοποιούμε το IEnumarator σαν ένα βρόχο
        // επανάληψης στην ουσία και όχι έναν κοινό βρόχο
επανάληψης είναι γιατί ο IEnumarator
        //λειτουργεί ασύχρονα και μπορεί να κρατάει τις
τιμές οσότου του ζητήθούν στην ουσία
        // έχει συμπεριφορά FIFO που αυτή θέλουμε για τα
μηνύματά μας

        //Arbiter = ορίζει πότε οι συνθήκες είναι
κατάλληλες για να φύγει ένα μήνυμα
        //(εξηγήθηκε με το Activate)

        //Delegate είναι μια μέθοδος θυμίζει τον pointer
της C αλλά χωρίς τον ακριβή ορισμό της..
        //εδώ χρησιμοποιείται για να χειρίζεται τα
success μηνύματα (να στέλνει μήνυμα success στην
προεπιλεγμένη πόρτα)
        // καθώς και τα fault μηνυματα σε αυτή την
περίπτωση κάνει log to error και στέλνει flag ότι η
κίνηση απέτυχε.

        //Yield Break= εδώ τερματίζει την λειτουργία του
βρόχου επανάληψης ( του IEnumarator)
        //Throw= η έκφραση throw χρησιμοποιείται για να
δηλώσει μια μη επιθημητή κατάσταση την ώρα
```

```
        // που εκτελείται ο κώδικάς

        /// <summary>
        /// Πέρνουμε την κατάσταση του SimulatedLRF
        /// </summary>
        /// <param name="get"></param>
        /// <returns></returns>

[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
        public  IEnumerator<ITask>  GetHandler(sicklrf.Get
get)
        {
            get.ResponsePort.Post(_state);
            yield break;
        }


        /// <summary>
        ///Φτιάχνουμε ένα μήνυμα αντικατάστασης
        /// </summary>
        /// <param name="replace"></param>
        /// <returns></returns>

[ServiceHandler(ServiceHandlerBehavior.Exclusive)]
        public                          IEnumerator<ITask>
ReplaceHandler(sicklrf.Replace replace)
        {
            _state = replace.Body;
            if (replace.ResponsePort != null)

replace.ResponsePort.Post(dssp.DefaultReplaceResponseType
.Instance);

            // κοινοποιούμε την ειδοποίηση
            _subMgrPort.Post(new     submgr.Submit(_state,
dssp.DsspActions.ReplaceRequest));
            yield break;
        }

        /// <summary>
        /// Εγγραφη στην υπηρεσία SimulatedLRF
        /// </summary>
        /// <param name="subscribe"></param>
        /// <returns></returns>

[ServiceHandler(ServiceHandlerBehavior.Exclusive)]
        public                          IEnumerator<ITask>
SubscribeHandler(sicklrf.Subscribe subscribe)
        {
            yield return Arbiter.Choice(
```

```
                SubscribeHelper(_subMgrPort,
subscribe.Body, subscribe.ResponsePort),
                delegate(SuccessResult success)
                {
                    _subMgrPort.Post(new submgr.Submit(
                        subscribe.Body.Subscriber,
dssp.DsspActions.ReplaceRequest, _state, null));
                },
                delegate(Exception ex) { LogError(ex); }
            );
        }
```

Βιβλιογραφία

Για τη δημιουργία αυτού του εγγράφου αντλήθηκαν πληροφορίες από συγγράμματα τα οποία παρατίθενται παρακάτω.

Συγγράμματα

1. Documendation of Microsoft Robotics Developer Studio Express 2008
2. Programming Microsoft Robotics Studio by Sara Morgan

Ιστοσελίδες

1. http://msdn.microsoft.com
2. http://www.microsoft.com/robotics/
3. http://www.studentguru.gr/blogs/robotics/
4. http://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio/
5. http://blogs.msdn.com/MSRoboticsStudio/
6. http://en.wikipedia.org/wiki/Robotics_simulator
7. http://www.societyofrobots.com/programming_robot_simulation.shtml
8. http://www.generation5.org/content/2005/sim2005.asp
9. http://en.wikipedia.org/wiki/Microsoft_XNA
10. http://en.wikipedia.org/wiki/DirectX
11. http://msdn.microsoft.com/en-us/magazine/cc546547.aspx
12. http://en.wikipedia.org/wiki/Sensor
13. http://roboticsprimer.sourceforge.net/workbook/Sensors
14. http://en.wikipedia.org/wiki/Image_sensor
15. http://www.sensorsweb.com
16. http://www.elderweb.com/home/node/3003

17. http://usl.sis.pitt.edu/ulab/icra-07.pdf

18. http://forums.trossenrobotics.com/tutorials/introduction-129/choosing-sensors-for-your-robot-3254

19. http://en.wikibooks.org/wiki/Robotics/Sensors/Distance_Sensors

20. http://www.electronicsforu.com/efylinux/efyhome/cover/September2007/Sensor-For-Robot.pdf

21. http://www.geocities.com/robodave2000/sensors.htm

22. http://www.seattlerobotics.org/encoder/jul97/basics.html

23. http://www.scribd.com/doc/18679809/Robot-Sensors

24. http://www.robotshop.ca/sensors.html

25. http://www.societyofrobots.com/sensors_interpret.shtml

26. http://el.wikipedia.org/wiki/Lego_Mindstorms

27. http://www.cs.brown.edu/courses/cs148/old/2004fall/lecture_slides/cs148_3.pdf

28. http://msdn.microsoft.com/en-us/magazine/cc546547.aspx

29. http://ims.mii.lt/olimp/tin/boi2002/kita/tasks/robots-en.pdf

30. http://www.teiser.gr/icd/staff/fasoulas/vadizonta_robot.pdf

31. http://www.cs.uoi.gr/~foka/robotics/slides/robotics_02.pdf

32. http://ncart.scs.ryerson.ca/NCART1/EDUCATOR/Diff_Drive_Veh.pdf

33. http://www.eng.ucy.ac.cy/elias/Courses/PRY020/Control%20and%20Robotics%20B.pdf

34. http://www.societyofrobots.com/programming_differentialdrive.shtml

35. http://www.songho.ca/opengl/gl_projectionmatrix.html

36. http://www.songho.ca/opengl/gl_transform.html

37. http://www710.univ-lyon1.fr/~jciehl/Public/OpenGL_PG/ch04.html