ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βαρτζής Τριαντάφυλλος ΑΜ 657
Γκανάς Γιάννης ΑΜ 700

Θέμα:

« Ανάπτυξη ρομποτικών συστημάτων με το πακέτο εφαρμογής Webots »

Webots™ 6
fast prototyping and simulation of mobile robots

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

1.1 Εισαγωγή

Το Webots είναι ένα επαγγελματικό πακέτο λογισμικού προσομοίωσης ρομπότ. Προσφέρει ένα γρήγορο περιβάλλον διαμόρφωσης πρωτοτύπων, το οποίο επιτρέπει στο χρήστη να δημιουργήσει τους τρισδιάστατους εικονικούς κόσμους με ιδιότητες φυσικής, όπως η μάζα, οι ενώσεις, οι συντελεστές τριβής, κ.λπ. Ο χρήστης μπορεί να προσθέσει ενεργά αντικείμενα τα αποκαλούμενα κινητά ρομπότ. Αυτά τα ρομπότ μπορούν να έχουν διαφορετικό τρόπο μετακίνησης (τροχοφόρα ρομπότ, ρομπότ με πόδια ή πετούμενα ρομπότ). Επιπλέον, μπορούν να εξοπλιστούν με διάφορες συσκευές αισθητήρων και ενεργοποιητών, όπως οι αισθητήρες απόστασης, οι ρόδες κίνησης, οι κάμερες, τα servos, οι αισθητήρες αφής, οι εκπομποί, οι δέκτες, κ.λπ. Τέλος, ο χρήστης μπορεί να προγραμματίσει κάθε ρομπότ για να εκθέσει χωριστά την επιθυμητή συμπεριφορά.

Το Webots περιέχει επίσης διάφορες διεπαφές στα πραγματικά κινητά ρομπότ, έτσι ώστε μόλις συμπεριφερθεί το μιμούμενο ρομπότ όπως αναμένεται, μπορούμε να μεταφέρουμε το πρόγραμμα ελέγχου σε ένα πραγματικό ρομπότ όπως το Khepera, Hemisson, LEGO Mindstorms, Aibo, κ.λπ.

1.1.2 Πλεονεκτήματα – Μειονεκτήματα simulator

Η εκπαίδευση φοιτητών για τον χειρισμό-προγραμματισμό ρομποτικών συστημάτων επιφέρει δυσκολίες, οι οποίες σε επίπεδο θεωρίας σχετίζονται με την κατανόηση κα την περιγραφή της κινηματικής των στερεών σωμάτων στον χώρο με την βοήθεια καρτεσιανών συστημάτων συντεταγμένων. Όπως είναι γνωστό, τα ρομπότ είναι μηχανισμοί που κινούνται και μεταφέρουν αντικείμενα στον χώρο και για το λόγο αυτό πρέπει να λαμβάνονται πάντα κατάλληλα μέτρα προστασίας. Σε επίπεδο πρακτικής άσκησης, είναι δυνατό να προκύψουν προβλήματα από λανθασμένο χειρισμό-προγραμματισμό του ρομπότ, με αποτέλεσμα να υπάρχει κίνδυνος όχι μόνο επιπλοκή του ρομπότ, αλλά και καταστροφής μέρους του περιβάλλοντος χώρου εργασίας. Το μεγαλύτερο όμως πρόβλημα αποτελεί η μη ύπαρξη των απαιτούμενων κονδυλίων για την αγορά του απαραίτητου βασικού και παρελκόμενου εξοπλισμού ενός ρομπότ σε συνδυασμό με τα διάφορα προβλήματα που προκύπτουν, λόγω του μεγάλου αριθμού φοιτητών που πρέπει να εκπαιδευτούν κάθε ακαδημαϊκό εξάμηνο. Επίσης, κρίνεται επιτακτική η ύπαρξη τεχνικού προσωπικού κατάλληλα εκπαιδευμένου, που θα συντηρεί το ρομπότ σε τακτά χρονικά διαστήματα. Όλα τα παραπάνω συντελούν στην ανάγκη ανάπτυξης ενός εικονικού εργαστηρίου με την χρήση αναδραστικών ηλεκτρονικών μέσων. Τα τελευταία χρόνια έχει αναπτυχθεί ένας μεγάλος αριθμός από προγράμματα προσομοίωσης που εστιάζουν το ενδιαφέρον τους σε διάφορα ρομποτικά συστήματα σταθερής ή κινούμενης βάσης τόσο για εκπαιδευτικούς όσο και για ερευνητικούς σκοπούς. Ως παράδειγμα αναφέρουμε τα Yobotics (YOBOTICS Inc., 2000), Robotics toolbox for MATLAB(Corke P.I., 1996) , RoboMosp (Jaramillo-Botero, A., 2006) που εστιάζουν στηνκινηματική και δυναμική προσομοίωση ρομποτικών βραχιόνων, ενώ

τα, Khepera Simulator (KHEPERA simulator, 1995) και Webots (WEBOTS software, 2004) αναφέρονται στην προσομοίωση τροχοφόρων ρομπότ. Σε αυτή την εργασία παρουσιάζεται ένα εκπαιδευτικό λογισμικό που σχεδιάστηκεκαι υλοποιήθηκε για την προσομοίωση τροχοφόρων ρομπότ το webots. Η παρούσα εφαρμογή, βοηθάει με απλό τρόπο να γίνει κατανοητή φιλοσοφία λειτουργίας και ο τρόπος προγραμματισμού των ρομπότ. Επιπρόσθετα, μέσα από τις δυνατότητες που παρέχει το πρόγραμμα μπορεί να βοηθήσει οπτικά στην κατανόηση βασικών εννοιών της ρομποτικής που σχετίζονται με την κινηματική των στερεών σωμάτων στο χώρο.

1.1.3 Λίγα λόγια για το πρόγραμμα

Το Webots είναι ένας επαγγελματικός προσομοιωτής ρομπότ που χρησιμοποιείται ευρέως για εκπαιδευτικούς λόγους. Το πρόγραμμα Webots άρχισε το 1996, αρχικά να αναπτύσσεται από το Δρ Olivier Michel στο ελβετικό ομοσπονδιακό Τεχνολογικό Ινστιτούτο (EPFL) στη Λοζάνη, Ελβετία. Το Webots χρησιμοποιεί την ODE (Open Dynamics Engine) για την ανίχνευση των συγκρούσεων και τη μίμηση της άκαμπτης δυναμικής σωμάτων. Η βιβλιοθήκη ODE επιτρέπει στο webots να μιμηθεί ακριβώς τις σωματικές ιδιότητες των αντικειμένων όπως η ταχύτητα, η αδράνεια και η τριβή. Μια μεγάλη συλλογή των ελεύθερα τροποποιήσιμων προτύπων ρομπότ παρέχεται μαζί με τη διανομή λογισμικού. Είναι επίσης δυνατό να δημιουργηθούν και νέα μοντέλα από την αρχή. Κατά το σχεδιασμό ενός ρομπότ, τα χαρακτηριστικά τους τα ορίζουν οι γραφικές και φυσικές ιδιότητες των αντικειμένων. Οι γραφικές ιδιότητες περιλαμβάνουν τη μορφή, τις διαστάσεις, τη θέση και τον προσανατολισμό, τα χρώματα, και τη σύσταση του αντικειμένου. Οι φυσικές ιδιότητες περιλαμβάνουν τη μάζα, τον παράγοντα τριβής, καθώς επίσης και τις σταθερές άνοιξη και απόσβεσης. Το Webots περιλαμβάνει ένα σύνολο αισθητήρων και ενεργοποιείτο που χρησιμοποιούνται συχνά στα ρομποτικά πειράματα, π.χ. αισθητήρες εγγύτητας, ελαφριοί αισθητήρες, αισθητήρες αφής, α, επιταχύνετε, κάμερες, εκπομπή και δέκτες, σέρβο μηχανές (περιστροφικοί & γραμμικοί), αισθητήρας θέσης και δύναμης, LEDs, πένσες, γυροσκόπια και πυξίδα. Τα προγράμματα ελεγκτών ρομπότ μπορούν να γραφτούν στις γλώσσες προγραμματισμού C, C++, την java, Python και MATLAB. Το AIBO, η Nao και τα πρότυπα ρομπότ e-puck μπορούν επίσης να προγραμματιστούν με τη γλώσσα URBI (απαιτείται άδεια URBI).Το Webots προσφέρει τη δυνατότητα να ληφθούν screen shots της οθόνης σε PNG format και να καταγραφούν οι προσομοιώσεις ως MPEG (Mac/Linux) και ώς AVI (Windows).

Οι κόσμοι του Webots αποθηκεύονται σε .wbt format το οποίο μοιάζει πολύ με το VRML. Είναι επίσης δυνατό να εισαχθούν και να εξαχθούν οι κόσμοι Webots ή αντικείμενα σε VRML format. Ένα άλλο χρήσιμο χαρακτηριστικό γνώρισμα είναι ότι ο χρήστης μπορεί να αλληλεπιδράσει οποιαδήποτε στιγμή σε μία προσομοίωση ενώ αυτή τρέχει , δηλ. πιθανόν ενώ κινείται το ρομπότ να κινείται και άλλο αντικείμενο με το ποντίκι. Το Webots χρησιμοποιείται σε διάφορους online διαγωνισμούς προγραμματισμού ρομπότ. Το Robotstadium competition είναι μια προσομοίωση της τυποποιημένης ένωσης πλατφορμών RoboCup. Σε αυτήν την προσομοίωση δύο ομάδες του παιχνιδιού ποδοσφαίρου της Nao παίζουν ποδόσφαιρο με τους κανόνες παρόμοιους με το πραγματικό. Τα ρομπότ χρησιμοποιούν τις μιμούμενες κάμερες και τους αισθητήρες υπερήχου και πίεσης. Στο Rat's Life competition δύο e-puck ρομπότ ανταγωνίζονται για τους πόρους ενέργειας σε έναν λαβύρινθο Lego. Οι αντιστοιχίες οργανώνονται σε καθημερινή βάση και τα αποτελέσματα μπορούν να προσέξουν στα σε απευθείας σύνδεση βίντεο.

1.2 Βασικές έννοιες του προγράμματος

Οι βασικές έννοιες του προγράμματος είναι η προσομοίωση ,ο κόσμος (world),ο ελεγκτής και ο supervisor.

1.2.1 Προσομοίωση Webots

Μια προσομοίωση Webots αποτελείται από τα παρακάτω :

1. Ένα Webots αρχείο που καθορίζει ένα ή περισσότερα τρισδιάστατα ρομπότ και το περιβάλλον τους.

2. Προγράμματα ελεγκτών για τα ανωτέρω ρομπότ.

3. Ένας προαιρετικός επόπτης (Supervisor).

1.2.2 Τι είναι ένας κόσμος (WORLD) ;

Ένας κόσμος, σε Webots, είναι μια τρισδιάστατη περιγραφή των ιδιοτήτων των ρομπότ και του περιβάλλοντός τους. Περιέχει μια περιγραφή κάθε αντικειμένου: τη θέση, τον προσανατολισμό, τη γεωμετρία, την εμφάνιση (όπως το χρώμα ή τη φωτεινότητα), τις σωματικές ιδιότητες και τον τύπο αντικειμένου τους. Οι κόσμοι οργανώνονται ως ιεραρχικές δομές όπου τα αντικείμενα μπορούν να περιέχουν άλλα αντικείμενα (όπως σε VRML97). Παραδείγματος χάριν, ένα ρομπότ μπορεί να περιέχει δύο ρόδες, έναν αισθητήρα απόστασης και έναν σέρβο μηχανισμό που ο ίδιος περιέχει μια κάμερα, κ.λπ. Ένα world αρχείο δεν περιέχει τον κώδικα ελεγκτών των ρομπότ, αλλά διευκρινίζει μόνο το όνομα του ελεγκτή που απαιτείται για κάθε ρομπότ. Τα world σώζονται ως .wbt αρχεία. Τα .wbt αρχεία αποθηκεύονται στο world subdirectory κάθε προγράμματος Webots.

1.2.3 Τι είναι ελεγκτής (controller);

Ένας ελεγκτής είναι ένα πρόγραμμα που ελέγχει ένα ρομπότ σε ένα αρχείο world. Οι ελεγκτές (controller) μπορούν να γραφτούν σε οποιαδήποτε από τις γλώσσες προγραμματισμού υποστηρίζονται από το Webots: C, C++, java, URBI ή Python. Όταν μια προσομοίωση αρχίζει, το Webots ξεκινάει τους συγκεκριμένους ελεγκτές, και τους συνδέει με τα εικονικά ρομπότ.

Τα πηγαία αρχεία (source files) και τα δυαδικά αρχεία (binary files) κάθε ελεγκτή αποθηκεύονται μαζί σε έναν κατάλογο ελεγκτών. Ένας κατάλογος ελεγκτών τοποθετείται στο controllers subdirectory κάθε προγράμματος Webots.

1.2.4 Τι είναι supervisor;

Ο supervisor είναι ένας προνομιούχος τύπος ρομπότ που μπορεί να εκτελέσει διαδικασίες που μπορούν κανονικά να διενεργηθούν μόνο από έναν ανθρώπινο χειριστή και όχι από ένα πραγματικό ρομπότ. Ο supervisor συνδέεται κανονικά με ένα πρόγραμμα ελεγκτών που μπορεί επίσης να γραφτεί σε οποιεσδήποτε από τις γλώσσες προγραμματισμού. Εντούτοις σε αντίθεση με έναν κανονικό ελεγκτή ρομπότ, ο supervisor έχει δυνατότητα να εκτελέσει κάποιες ιδιαίτερες διαδικασίες όπως ο έλεγχος προσομοίωσης για παράδειγμα, κινώντας τα ρομπότ προς μια τυχαία θέση να γίνεται εγγραφή σε βίντεο της προσομοίωσης, κ.λπ.

1.2.5 Έναρξη του webots

Από το Start menu, επιλέγουμε Program Files — Cyberbotics και επιλέγουμε το webots 6.1.0. Στη συνέχεια θα εμφανιστεί το μήνυμα ”Welcome to Webots !” και θα εμφανιστεί ένα menu με μια λίστα επιλογών όπου διαλέγουμε τι θέλουμε να χρησιμοποιήσουμε.

Κεφάλαιο 2

2.1 MainWindow: Επιλογές και κουμπιά

Το Webots GUI αποτελείται από τέσσερα κύρια παράθυρα.

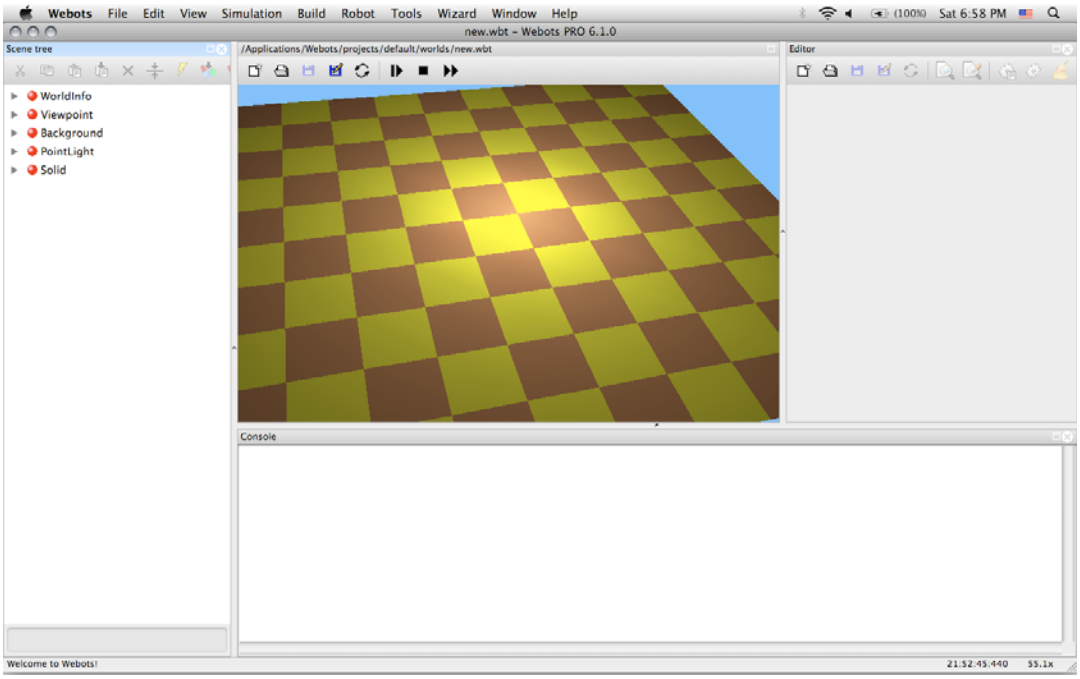
Αρχικά έχουμε το παράθυρο προσομοίωσης simulation window (or 3D window) Σε αυτό παρουσιάζεται ο εικονικός κόσμος που έχουμε φορτώσει και έχει μια μπάρα εργαλείων με shortcuts με τις πιο σημαντικές επιλογές.

Στη συνέχεια έχουμε το Scene Tree Window που είναι μια ιεραρχική αντιπροσώπευση του τρέχοντος κόσμου.

Τρίτον, έχουμε το text Editor Window που εκεί γράφουμε τον κώδικα ελεγκτών.

Τέλος έχουμε το Console Window όπου εκεί βλέπουμε διάφορες πληροφορίες για τη λειτουργία των ελεγκτών.

Αυτά τα παράθυρα μπορούμε να τα διαμορφώσουμε με όποιο τρόπο θέλουμε. Το κύριο παράθυρο έχει οκτώ επιλογές: File, Edit, View, Simulation, Build, Tools, Wizard and Help.



Εικόνα 1 : Βασικό παράθυρο του Webots

2.2 File Menu

Το **File** menu παρέχει τις συνηθισμένες επιλογές αρχείων όπως: loading, saving, κτλ.

 Το **New World...** menu (button) ανοίγει ένα νέο world στο simulation window που περιέχει μόνο ένα ElevationGrid, που επιδεικνύεται ως σκακιέρα 10 X 10 τετραγώνων σε μια επιφάνεια 1 μ X 1 μ.

 Το **Open World...** menu (button) ανοίγει έναν διάλογο επιλογής αρχείων που μας επιτρέπει να επιλέξουμε και να φορτώσουμε ένα .wbt αρχείο.

 Το **Save World...** menu (button) σώζει το world στο οποίο εργαζόμαστε χρησιμοποιώντας το τρέχον όνομα αρχείου (το όνομα αρχείου που εμφανίζεται στην κορυφή του κύριου παραθύρου).Σε περίπτωση επανάληψης του κουμπιού ξαναεγγράφεται το αρχείο.

 Το **Save World As...** menu (button) αποθηκεύει το αρχείο δημιουργώντας ένα καινούργιο .wbt με το όνομα που επιθυμούμε χωρίς να επικαλύπτει το προηγούμενο.

 Το **Revert World...** menu item (and button) ξαναφορτώνει τον τρέχοντα κόσμο από τη σωζόμενη έκδοση και ξανά αρχίζει την προσομοίωση από την αρχή.

 Το **New Text File...** menu item (and button) ανοίγει ένα κενό αρχείο κειμένων στο συντάκτη κειμένων.

 Το **Open Text File...** menu (button) ανοίγει ένα παράθυρο όπου μπορούμε να επιλέξουμε αρχεία και μας επιτρέπει να επιλέξουμε ένα αρχείο Text (παραδείγματος χάριν ένα αρχείο .java) και να το φορτώσουμε .

 Το **Save Text File...** menu (button) με αυτό το κουμπί σώζουμε το τρέχων αρχείο text file.

 Το **Save All Text Files...** menu item σώζει όλα τα ανοιγμένα αρχεία κειμένων καθώς και αρχεία που δεν έχουμε ήδη σώσει.

 Το **Save Text File As...** menu (button) σώζει το τρέχον αρχείο Text με ένα νέο όνομα αρχείου που εισάγεται από το χρήστη.

 Το **Revert Text File...** menu (button) ξαναφορτώνει το αρχείο Text από τη σωζόμενη έκδοση .

Το **Page setup...** menu item με αυτό το κουμπί ανοίγει ένα παράθυρο όπου μας επιτρέπει να διαχειριστούμε σχεδιαγράμματα σελίδων προκειμένου να τυπωθούν τα αρχεία από το συντάκτη κειμένων (text editor).

Το **Print...** menu item ανοίγει ένα παράθυρο όπου μας επιτρέπεται να τυπώσουμε το τρέχον αρχείο.

Το **Import VRML 2.0...** menu προσθέτει VRML97 αντικείμενα στο τέλος του scene tree. Αυτά τα αντικείμενα προέρχονται από το VRML97 αρχείο που πρέπει να φορτώσουμε .

Το **Export VRML 2.0...** μας επιτρέπει να σώσουμε τον ήδη φορτωμένο κόσμο ως .wrl αρχείο, που προσαρμόζεται στα πρότυπα VRML97. Ένα τέτοιο αρχείο μπορεί, στη συνέχεια, να ανοιχτεί με οποιοδήποτε VRML97**.**

Το **Make Animation...** μας επιτρέπει να δημιουργήσουμε ένα 3D animation ως .wva αρχείο. Αυτή η μορφή επιτρέπει στο αρχείου να μπορεί να αναπαραχθεί στο Webots Player .

Το **Make Movie** … μας επιτρέπει να δημιουργήσουμε βιντεο MPEG (Linux και Mac OS X) ή AVI (windows).

Όλα τα βίντεο που δημιουργούνται από το Webots έχουν 25 fps. Εν τούτοις είναι δυνατό να ελεγχθεί η ταχύτητα αναπαραγωγής ήχου αυτών των βίντεο επιλέγοντας what intervals (of simulated time) στις εικόνες που παράγονται από το Webots. Το χρονικό διάστημα μεταξύ δύο εικόνων είναι αποτέλεσμα συνδυασμού του basicTimeStep και displayRefresh fields (WorldInfo node). Παραδείγματος χάριν εάν χρειαστείτε ένα βίντεο να παίζει σε πραγματικό χρόνο, το time interval πρέπει να είναι 40 επειδή 1 / F = 1 / 25 = 0.04s

Έτσι μπορείτε να επιλέξετε παραδείγματος χάριν 10ms για το basicTimeStep και 4 για το displayRefresh, έτσι το διάστημα θα είναι 10 * 4 = 40, και το βιντεο θα παίξει σε πραγματικό χρόνο.

Ομοίως οποιοσδήποτε συνδυασμός basicTimeStep και displayRefresh που πολλαπλασιάζει σε 40, θα είναι πραγματικός χρόνος. Το βίντεο θα είναι ή γρηγορότερο ή πιο αργό σε πραγματικό χρόνο εάν είναι αντίστοιχα μεγαλύτερο ή μικρότερο από 40. Σημειώστε ότι η τροποποίηση του basicTimeStep μπορεί να έχει παρενέργειες στην προσομοίωσή σας γι αυτό είναι συνήθως ασφαλέστερο να αλλάξει μόνο displayRefresh εάν αυτό είναι δυνατόν.

Το **Take Screenshot...** μας επιτρέπει να πάρουμε ένα screenshot του παραθύρου του Webots. Έπειτα ανοίγει έναν παράθυρο όπου μπορούμε να σώσουμε το τρέχων screenshot ως εικόνα PNG.

Το **Quit Webots…** κλείνει την τρέχουσα προσομοίωση και κλείνει το πρόγραμμα Webots.

2.3 Edit Menu

Το **Undo** menu χρησιμοποιείται για να επαναφέρει την τελευταία τροποποίηση που έχει γίνει στο text editor.

Το **Redo** menu χρησιμοποιείται για να επαναφέρει μια Undo λειτουργία.

Το **Cut** menu χρησιμοποιείται για να αποκόψει μια επιλογή μας ,το οποίο επίσης μετέπειτα αφαιρεί την επιλογή αυτή.

Το **Copy** menu χρησιμοποιείται για να αντιγράψει μια επιλογή μας .

Το **Paste** menu χρησιμοποιείται για να κάνουμε επικόλληση μια ήδη υπάρχων επιλογή μας.

Το **Select All** menu χρησιμοποιείται για να επιλέξει όλο το κείμενο του αρχείου του τρέχοντος text editor.

 Το **Find...** menu ανοίγει ένα παράθυρο όπου μπορούμε να ψάξουμε μια λέξη στον text editor.

 Το **Replace...** menu ανοίγει ένα παράθυρο όπου μπορούμε να ψάξουμε μια λέξη και να την αντικαταστήσουμε στον text editor.

Το **Go to line...** menu ανοίγει ένα παράθυρο όπου μπορούμε να μεταβούμε σε μια συγκεκριμένη γραμμή του text editor.

2.4 View Menu

Το **View** menu μας επιτρέπει να ελέγχουμε το παράθυρο προσομοίωσης (simulation Window).

Το **Follow Object** menu μας επιτρέπει να μετατρέψουμε μια στατική άποψη (Static Viewpoint) σε μια άποψη που ακολουθεί ένα κινητό αντικείμενο (συνήθως ένα ρομπότ).

Το **Restore Viewpoint** αποκαθιστά τη θέση και τον προσανατολισμό της άποψης (Viewpoint) στις αρχικές τοποθετήσεις τους όταν το αρχείο φορτώθηκε.

Το **Projection** submenu μας επιτρέπει να επιλέξουμε μεταξύ **Perspective** και **Orthographic** προβολής. Ο τρόπος **Perspective** είναι η προεπιλογή του προγράμματος και αντιστοιχεί σε μια φυσική προβολή, δηλαδή στο όσο μακρύτερα είναι ένα αντικείμενο από το θεατή τόσο μικρότερο εμφανίζεται στην εικόνα. Με την **Orthographic** προβολή, η απόσταση από το θεατή δεν έχει επιπτώσεις στο πόσο μεγάλο εμφανίζεται ένα αντικείμενο.

Το **Rendering** submenu μας επιτρέπει να επιλέξουμε μεταξύ **Regular** (default) και **Bounding Objects.** Στο **regular**, τα αντικείμενα δίνονται με τα γεωμετρικά σχηματά τους, υλικά, χρώματα, με τον ίδιο τρόπο που φαίνονται συνήθως από ένα μάτι ή μια κάμερα.

Το **Show Contact Points** παρουσιάζει τα σημεία επαφής που παράγονται από τη μηχανή ανίχνευσης συγκρούσεων. Τα σημεία επαφής που δεν παράγουν κάποια δύναμη δεν παρουσιάζονται.

2.5 Simulation Menu

Το **Simulation** menu χρησιμοποιείτε για να ελέγχουμε και να εκτελέσουμε την προσομοίωση που δημιουργήσαμε

 Το **Stop** menu (button) σταματάει το simulation.

 Το **Step** menu (button) εκτελεί μία προσομοίωση. Η διάρκεια της προσομοίωσης καθορίζεται στο basicTimeStep του WorldInfo, και μπορεί να ρυθμιστεί στο scene tree window

 Το **Run** menu (button) τρέχει την προσομοίωση έως ότου να τη διακόψουμε πατώντας **Stop** ή **Step**.

 Το **Fast** menu (button) λειτουργεί όπως το run, εκτός από το ότι δεν εκτελείται το graphical rendering (στην έκδοση του Webots PRO υπάρχει αυτή η δυνατότητα). Δεδομένου ότι το graphical rendering είναι απενεργοποιημένο (μαύρη οθόνη) αυτό επιτρέπει μια γρηγορότερη προσομοίωση .

2.6 Build Menu

 Το **Compile** menu μας επιτρέπει να κάνουμε compile το αρχείο σύμφωνα με το Makefile που βρίσκεται στον ίδιο κατάλογο.

 Το **Build** menu μας επιτρέπει να δημιουργήσουμε ένα εκτελέσιμο αρχείο σύμφωνα με το Makefile που βρίσκεται στον ίδιο κατάλογο με το αρχείο.

 Το **Clean** menu μας επιτρέπει να εκτελέσουμε την καθαρή λειτουργία του makefile που βρίσκεται στον ίδιο κατάλογο με το αρχείο

 Το **Make JAR file** menu μας επιτρέπει να δημιουργήσουμε ένα .jar αρχείο από ένα .java.

 Το **Cross - compilation clean** menu μας επιτρέπει να καθαρίσουμε τα cross-compilation αρχεία.

2.7 Tools Menu

Το **Tools** menu μας ανοίγει τα διάφορα παράθυρα Webots.

The **Scene Tree** menu ανοίγει το SceneTree window με το οποίο μπορείς να επεξεργαστείς το virtual world

Το **Text Editor** menu ανοίγει τον Webots text editor.

Το **Console** menu item ανοίγει το Webots Console, το οποίο είναι μία read-only console που χρησιμοποιείται για να εμφανίζει τα Webots error messages και τα αποτελέσματα του controller.

Το **Robot Window** menu ανοίγει ένα Robot Window. Ο τύπος του παραθύρου εξαρτάται από τον τύπο του ρομπότ .Κάθε είδος robot Window επιτρέπει κάποια επίπεδο αλληλεπίδρασης με τους αισθητήρες και τους ενεργοποίησες ρομπότ. Αυτό το μενού ενεργοποιείται μόνο όταν επιλεγεί ένα ρομπότ.

Το **Motion Editor** menu ανοίγει το Motion Editor. Το Motion Editor είναι ένα εργαλείο που σχεδιάζει τις ακολουθίες κινήσεων για τα ρομπότ. Αυτές οι ακολουθίες μπορούν να σωθούν σε motionfiles και να παιχτούν έπειτα σε controller code.

Το **Upload to e-puck robot...** menu μας επιτρέπει να επιλέξουμε μία Bluetooth σύνδεση και να τη φορτώσουμε σε ένα e-puck robot.

Το **Preferences** ανοίγει ένα παράθυρο με δύο στοιχεία

Το **General** : Η γλωσσική αυτή επιλογή μας επιτρέπει να επιλέξει τη γλώσσα του Webots στη διεπαφή με τον χρήστη (απαιτείται επανεκκίνηση).

Το **Startup mode** επιτρέπει να επιλέξουμε το είδος της προσομοίωσης όταν το Webots ξεκινάει (stop, run, fast).

Το **Editor font** καθορίζει τα font που χρησιμοποιείται στο συντάκτη κειμένου του Webots.

Το **Java command** χρησιμοποιείται για να καθορίζει τη Java Virtual Machine (JVM) για την εκτέλεση των ελεγκτών java.

Το **Rendering** tab ελέγχει μερικές πτυχές της τρισδιάστατης απόδοσης στο παράθυρο προσομοίωσης:

Display global coordinate system παρουσιάζει το παγκόσμιο σύστημα συντεταγμένων (κατώτατη σωστή γωνία του παραθύρου) όπως κόκκινα, πράσινα και μπλε βέλη που αντιπροσωπεύουν τους άξονες X, Y και Z αντίστοιχα.

Display sensor rays παρουσιάζει ακτίνες αισθητήρων για τα μη-επιλεγμένα ρομπότ (οι ακτίνες των επιλεγμένων ρομπότ παρουσιάζονται πάντα). Οι ακτίνες αισθητήρων απόστασης σύρονται ως κόκκινες γραμμές (που γίνονται πράσινες πέρα από τα σημεία σύγκρουσης), και οι ελαφριές ακτίνες αισθητήρων επιδεικνύονται ως κίτρινες γραμμές.

Display device axes επιδεικνύει τους άξονες σέρβο και συνδέσεων. Οι άξονες σέρβο παρουσιάζονται ως μαύρες παχιές γραμμές ενώ οι άξονες συνδέσεων παρουσιάζονται ως πράσινες και μπλε γραμμές που αντιπροσωπεύουν τους άξονες Y και Z αντίστοιχα και μαύρες γραμμές που αντιπροσωπεύουν τις πιθανές ευθυγραμμίσεις.

Display camera frustums παρουσιάζει το OpenGL culling frustum για κάθε κάμερα χρησιμοποιώντας ένα magenta wire frame. Περισσότερες πληροφορίες για αυτήν την έννοια είναι διαθέσιμες στην τεκμηρίωση OpenGL.

Display lights παρουσιάζουν τα PointLight ως διάφορες τεμνόμενες κίτρινες γραμμές και ελέγχει το μέγεθος των σταυρών που αντιπροσωπεύουν τα κέντρα των Solid node (στερεό κόμβο) όταν επιλέγονται. Για κάθε (Solid node), δύο κέντρα μπορούν να εμφανίζονται: το γεωμετρικό κέντρο (geometrical center) (καθορίζεται από τους τομείς translation και rotation) και το κέντρο της μάζας (όπου καθορίζει τον physics node).

Axis size ελέγχει το μέγεθος των σέρβο αξόνων που παρουσιάζονται ως μαύρες παχιές γραμμές.

2.8 Wizard Menu

Με το **Wizard** menu δημιουργούμε νέα project και νέους ελεγκτές (new projects και new controllers).

Το **New Project Directory...** menu μας επιτρέπει να επιλέξουμε μια τοποθεσία στο filesystem και έπειτα αυτό δημιουργεί έναν κατάλογο προγράμματος. Ένας κατάλογος προγράμματος περιέχει διάφορα subdirectories που χρησιμοποιούνται για να αποθηκευθούν τα αρχεία σχετικά με ένα πρόγραμμα Webots, δηλ. τα αρχεία world, τα αρχεία ελεγκτών, τα αρχεία στοιχείων, plugins, κ.λπ. Το Webots θυμάται τον τρέχοντα κατάλογο και ανοίγουν αυτόματα και σώζουν οποιοδήποτε τύπο αρχείου από το αντίστοιχο subdirectory του τρέχοντος καταλόγου προγράμματος.

Το **New Robot Controller...** menu επιτρέπει να δημιουργήσουμε έναν νέο πρόγραμμα ελεκτή. Θα πρέπει πρώτα να επιλέξουμε έναν C, C++, Java ή έναν Python ελεκτή. Τότε το Webots θα ρωτήσει να εισάγουμε το όνομα του ελεγκτή και θα δημιουργήσει όλα τα απαραίτητα αρχεία (συμπεριλαμβανομένου και ενός template source code file).

2.9 Help menu

Στο **Help** menu, το **About** ανοίγει το About window το οποίο παρουσιάζει τις πληροφορίες αδειών.

Το **Webots Guided Tour** ξεκινάει μία παρουσίαση με μια σειρά από παραδείγματα

Το **Register** σας λέει το ComputerID και ανοίγει τη σελίδα εγγραφής Webots.

Το **OpenGL Information** δίνει τις πληροφορίες για το τρέχον υλικό OpenGL και τον οδηγό. Μπορεί να χρησιμοποιηθεί για να εντοπίσει προβλήματα.
Τα υπόλοιπα στοιχεία επιλογών φέρνουν τις διάφορες πληροφορίες όπως υποδεικνύονται, υπό μορφή σελίδων HTML, PDF έγγραφα, κ.λπ.

2.10 Speedometer and Virtual Time

Ένα speedometer (εικόνα 2) μας επιτρέπει να παρατηρούμε την ταχύτητα της εξομοίωσης στον υπολογιστή μας. Παρουσιάζεται στο κάτω δεξιό σημείο του κεντρικού παραθύρου και δείχνει πόσο γρήγορα τρέχει η προσομοίωση σε σχέση με τον πραγματικό χρόνο. Με άλλα λόγια, αντιπροσωπεύει την ταχύτητα του εικονικού χρόνου. Εάν δείχνει το speedometer 2, σημαίνει ότι η προσομοίωση στον υπολογιστή σας τρέχει δύο φορές πιο γρήγορα από τα αντίστοιχα πραγματικά ρομπότ. Αυτές οι πληροφορίες είναι διαθέσιμες τόσο για **Run** mode όσο και για **Fast** mode.

Εικόνα 2: Ένας Speedometer

Αριστερά του **speedometer**, ο εικονικός χρόνος παρουσιάζεται με την εξής μορφή:
H : MM : SS : MMM
Όπου **H** είναι οι ώρες, **MM** είναι τα λεπτά, **SS** είναι τα δευτερόλεπτα και **MMM** είναι τα χιλιοστά του δευτερολέπτου. Εάν η τιμή του speedometer είναι μεγαλύτερη από 1 τότε η ταχύτητα της προσομοίωσης είναι γρηγορότερη από την πραγματική.

Το βασικό χρονικό βήμα για την προσομοίωση μπορεί να οριστεί στο **basicTimeStep** του **WorldInfo** στο scene tree window. Αναφέρεται στα εικονικά χρονικά χιλιοστά του δευτερολέπτου. Η τιμή αυτή καθορίζει το μήκος του χρονικού βήματος που εκτελείται κατά τη διάρκεια του **Step** mode. Αυτό το βήμα πολλαπλασιάζεται με τον τομέα **displayRefresh** του **WorldInfo** για να καθορίσει πόσο συχνά η παρουσίαση ανανεώνεται.

2.11 Selecting an object

Ένα mouse click επιτρέπει να επιλέξουμε ένα αντικείμενο. Επιλέγοντας ένα ρομπότ ενεργοποιείται το **Robot View** και το **Motion Editor** στο **Tools** menu. Με διπλό κλικ στο αντικείμενο ανοίγει το Scene Tree ή RobotWindow.

2.12 Navigation in the scene

Με το σύρσιμο του ποντικιού και πιέζοντας ένα κουμπί του ποντικιού μπορούμε να κινήσουμε την κάμερα του τρισδιάστατοι παραθύρου.

Camera rotation: Στο 3D window, μπορούμε πιέζοντας το αριστερό κουμπί και σέρνοντας το ποντίκι για να επιλέξουμε ένα αντικείμενο να περιστρέφουμε την γωνία θέασής του. Εάν κανένα αντικείμενο δεν επιλέγεται, η κάμερα περιστρέφεται για την προέλευση του world coordinate system..

Camera translation: Στο τρισδιάστατο παράθυρο, πιέζοντας το δεξί κουμπί και σέρνοντας το ποντίκι μπορούμε να αλλάξουμε τη κάμερα με την κίνηση του ποντικιού.

Zooming / Camera rotation: Στο τρισδιάστατο παράθυρο, πιέζοντας και αριστερό και δεξί κουμπί του ποντικιού ταυτόχρονα (ή το μέσο κουμπί) και σέρνοντας το ποντίκι κάθετα, μπορούμε να μεγενθύνουμε προς τα μέσα ή προς τα έξω. Το σύρσιμο του ποντικιού οριζόντια θα περιστρέψει τη κάμερα για τον άξονα εξέτασης. Εναλλακτικά, η ρόδα ποντικιών μπορεί επίσης να χρησιμοποιηθεί για τη μεγέθυνση.

2.13 Moving a solid object

Για να κινηθεί ένα αντικείμενο: κρατάμε πατημένο το shift, κατόπιν επιλέγουμε το αντικείμενο και σέρνουμε το ποντίκι

Translation: Για να κινηθεί ένα αντικείμενο παράλληλα στο έδαφος: κρατάμε πατημένο το shift , πιέζουμε το αριστερό κουμπί του ποντικιού και σέρνουμε.

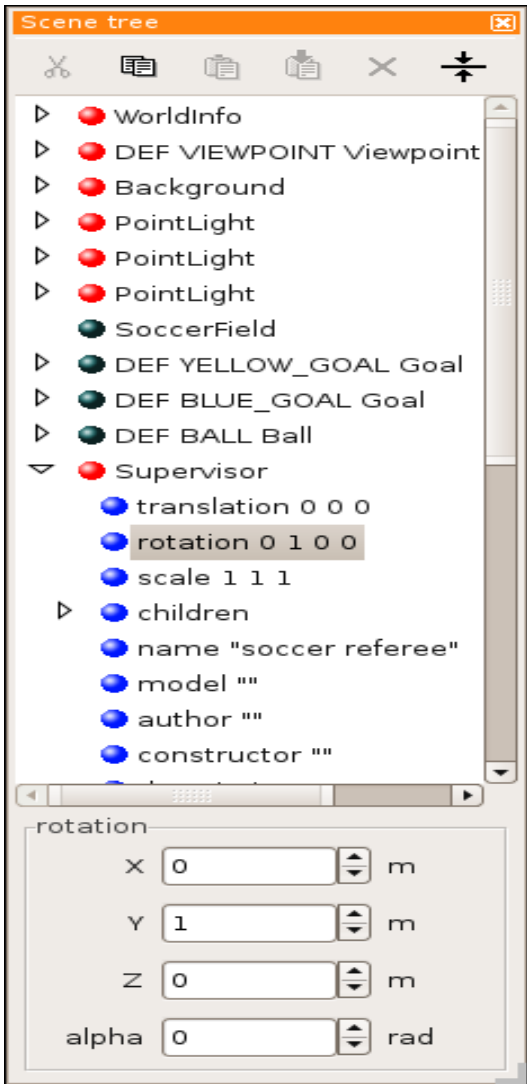
Rotation: Για να περιστραφεί ένα αντικείμενο: κρατάμε πατημένο το shift, πιέζουμε το δεξί κουμπί του ποντικιού και το σέρνουμε. Τον άξονα περιστροφής του αντικειμένου (X, Y ή Z) μπορούμε να τον αλλάξουμε γρήγορα ξαναπατώντας το κουμπί shift.

Lift: Για να αυξήσει ή να χαμηλώσει ένα αντικείμενο, κρατάμε πατημένο το κουμπί shift , πιέζουμε το αριστερό και δεξί κουμπί του ποντικιού (ή το μέσο κουμπί) και το σέρνουμε. Εναλλακτικά, η ρόδα από το ποντίκι που συνδυάζεται με το κουμπί shift μπορεί επίσης να χρησιμοποιηθεί.

2.14 Scene TreeWindow

Για να μπορούμε να έχουμε πρόσβαση στο **Scene Tree Window** μπορούμε είτε να επιλέξουμε **Scene Tree από το Tools** menu είτε με double-click του ποντικιού πάνω σε ένα αντικείμενο. Το Scene Tree Window περιέχει πληροφορίες οι οποίες περιγράφουν ένα Simulation world συμπεριλαμβανομένων και τις πληροφορίες για το ρομπότ ,το περιβάλλον καθώς και για το γραφικό παραβάλλον.

Παρακάτω βλέπουμε την διεπαφή του Scene Tree η οποία μας δίνει μια επισκόπηση των VRML97 κόμβων καθώς και των κόμβων Webots.



Εικόνα 3 : Scene Tree Window

2.14.1 Κουμπιά του Scene Tree Window

Οι κόμβοι (Nodes) του Scene Tree Window μπορούν να επεκταθούν με double-click του ποντικιού. Όταν επιλέξουμε ένα πεδίο εμφανίζονται τα χαρακτηριστικά του στο κάτω μέρος του Scene Tree Windows. Εάν πραγματοποιηθούν αλλαγές, τότε αυτές θα εμφανιστούν αμέσως στο 3D παράθυρο και τα ακόλουθα κουμπιά θα είναι διαθέσιμα για να μπορέσουμε να κάνουμε Edit τον κόσμο.

 **Cut:** Αποκοπή του επιλεγμένου κόμβου.

 **Copy:** Αντιγραφή των τιμών του επιλεγμένου πεδίου ή του κόμβου.

 **Paste:** Επικόλληση των προεπιλεγμένων τιμών του πεδίου ή του κόμβου.

 **Paste after:** Επικόλληση ενός κόμβου κάτω από τον επιλεγμένο κόμβο.

 **Delete:** Διαγραφή του επιλεγμένου κόμβου ή πεδίου.

 **Reset to default:** Επαναφορά πεδίου ή κόμβου στις αρχικές του τιμές.

 **Transform:** Μας επιτρέπει να αλλάξουμε τον τύπο από μερικά πεδία.

 **Insert after:** Εισάγει έναν κόμβο αμέσως μετά τον ήδη επιλεγόμενο κόμβο.

 **New node:** Εισάγει ένα νέο κόμβο.

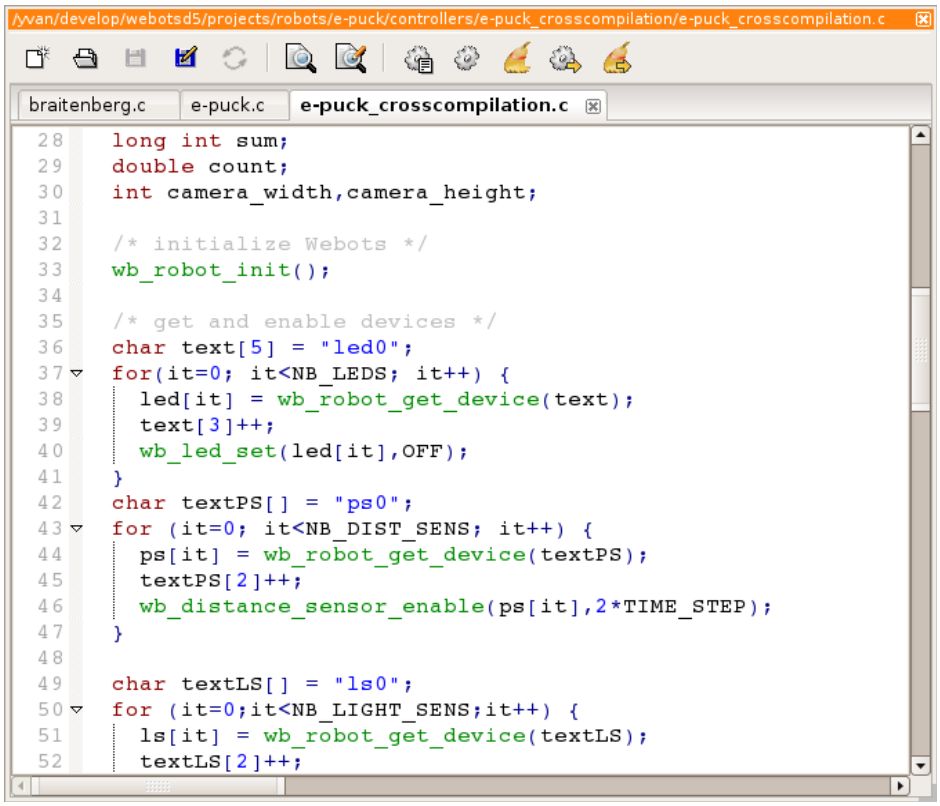
 **Export:** Εξαγάγει έναν κόμβο σε ASCII αρχείο. Οι εξαγόμενοι κόμβοι μπορούν έπειτα να εισαχθούν σε άλλους κόσμους.

 **Import:** Εισαγάγει έναν κόμβο στο scene tree.

 **Help:** Βοήθεια!!

2.15 Source Code Editor

Ο editor του Webots είναι ένας συντάκτης κειμένων που προσαρμόζεται ειδικά για την ανάπτυξη των ελεγκτών Webots. Ο editor αυτός μπορεί να υποστηρίζει πολλές γλώσσες όπως C, C++, Java, URBI και Python.



Εικόνα 4 :Ο Text Editor toy Webots

2.15.1 Σύνταξη στον Code Editor του Webots

Με τον code editor του Webots μπορούμε να γράψουμε σε γλώσσες προγραμματισμού όπως σε C, C++ ή Java και να τις μεταγλωττίσουμε σε αρχεία binary τα οποία στη συνέχεια μπορούν να εκτελεστούν σε μια προσομείωση. Τα αποτελέσματα του μεταγλωττισμένου αρχείου όπως για παράδειγμα errors ή Warnings φαίνονται στον Webots console. Για να διορθώσουμε τα λάθη αυτά, πάμε στον Webots console κάνοντας double-click με το ποντικιού μας στο σφάλμα που μας εμφανίζει το λάθος του κώδικα στον Code Editor του Webots.

Το **Compile** button. Με το πάτημα αυτού του κουμπιού αρχίζει να γίνεται η μεταγλώττιση του επιλεγμένου κώδικα. Το Webots εμφανίζει το μεταγλωττισμένο αρχείο με κατάληξη , gcc, g++ ή javac.

 Το **Build** button. Με το πάτημα αυτού του κουμπιού γίνεται η μεταγλώττιση του κώδικα και το μετατρέπει σε ένα εκτελέσιμο αρχείο (C / C++) ή σε bytecode (Java). Τα C/C++ αρχεία θα παραχθούν αυτόματα σε περίπτωση που χρειαστεί.

 Το **Clean** button. Το κουμπί αυτό διαγράφει τα ενδιάμεσα αρχεία σύνταξης στον κατάλογο του τρέχοντος αρχείου. Τα πηγαία αρχεία παραμένουν άθικτα.

 Το **Cross-compile** button. Το κουμπί αυτό μας επιτρέπει να γίνει η σύνταξη του τρέχων αρχείου κειμένου. Να σημειώσουμε επίσης ότι ένας συγκεκριμένος MakeFile απαιτείται στον κατάλογο controller's για την εκτέλεση αυτής της λειτουργίας. Για παράδειγμα σε ένα ρομπότ e-puck, αυτό το Makefile πρέπει να ονομαστεί makefile.e-puck.

 Το **Cross-compilation clean** menu. Το κουμπί αυτό μας επιτρέπει να γίνει η διαγραφή της σύνταξης του τρέχων αρχείου κειμένου.

2.16 Projects with Multiple Source Files

Το Webots μπορεί να υποστηρίξει projects με αρχεία από πολλές πηγές. Στην C και στην C++ το όνομα των πηγαίων αρχείων θα πρέπει να απαριθμούνται στο Makefile του προγράμματος. Για παράδειγμα εάν στο πρόγραμμα το όνομα του αρχείου είναι διαφορετικό του .cpp ή .cc, πρέπει να προσθέσουμε μία από αυτές τις γραμμές στο Makefile του προγράμματός μας :

```
CPP_SOURCES = my_first_file.cpp my_second_file.cpp my_last_file.cpp
CC_SOURCES = my_first_file.cc my_second_file.cc my_last_file.cc
```

Στη γλώσσα προγραμματισμού Java αυτό δεν είναι απαραίτητο μιας και ο Javac αυτόματα βρίσκει τα αρχεία που ανήκουν στο project.

2.17 Μη χρησιμοποίηση του Source Code Editor του Webots

Το Webots μας δίνει τη δυνατότητα να χρησιμοποιήσουμε και δικό μας Editor. Έτσι στην προκειμένη περίπτωση θα χρειαστούμε ένα terminal για να κάνουμε compile τον πηγαίο κώδικα μας. Στη συνέχεια θα πρέπει να καθορίσουμε τη μεταβλητή του περιβάλλοντος WEBOTS_HOME και να την κάνουμε να δείξει στο κατάλογο εγκαταστάσεων του Webots.
Ο καθορισμός μιας μεταβλητής εξαρτάται και από την πλατφόρμα μας και από το shell, Μερικά παραδείγματα: (Αυτά τα παραδείγματα υποθέτουν ότι Webots εγκαθίσταται στον προεπιλεγμένο κατάλογο) σε περίπτωση που χρησιμοποιείτε Linux, πληκρολογείστε αυτό :

```
$ export WEBOTS_HOME=/usr/local/webots
```
Ή προσθέστε αυτή τη γραμμή στο ~/.bash_profile file.

Σε λειτουργικό Mac OS X, πληκρολογείστε αυτό :

$ export WEBOTS_HOME=/Applications/Webots
Ή προσθέστε αυτή τη γραμμή στο ~/.profile file.

Μόλις καθοριστεί το WEBOTS_HOME, μπορούμε να το συντάξουμε σε ένα terminal, με την εντολή make.
Με τα editor buttons, θα είμαστε θέση να δημιουργήσουμε ολόκληρο το project, ή μόνο ένα ενιαίο binary file, κ.λπ.

$ make
$ make clean
$ make my_robot.class
$ make my_robot.

2.18 Motion Editor

Εισαγωγή :

Ο Motion Editor μπορεί να χρησιμοποιηθεί για να σχεδιάσει τις κινήσεις για τα ρομπότ με αρθρώσεις, παραδείγματος χάριν humanoids, κ.λπ.
Οι κινήσεις μπορούν έπειτα να σωθούν ως αρχεία .motion που μπορούν έπειτα να αναχαραχτούν από τον ελεγκτή κατά τη διάρκεια της προσομοίωσης. Ο Motion Editor δεν μπορεί να χρησιμοποιηθεί για να ελέγξει τις ρόδες ενός ρομπότ DifferentialWheels.

Ο Motion Editor συνδέεται άμεσα με τη μηχανή προσομοίωσης, αυτό έχει διάφορα πλεονεκτήματα: Κατ' αρχάς, το καθιστά σχετικά εύκολο να σχεδιάσει τις φυσικά εύλογες κινήσεις. Δεύτερον, είναι δυνατό να σχεδιαστούν οι κινήσεις που αλληλεπιδρούν με το περιβάλλον.

2.19 Using the Dialog

Προκειμένου να ανοιχτεί ο Motion Editor, επιλέγουμε ένα ρομπότ με το ποντίκι και επιλέγουμε έπειτα το Tool - Motion Editor menu. Το παράθυρο Menu Editor αποτελείται από δύο pane. Το αριστερό pane χρησιμοποιείται για να λειτουργήσει η ακολουθία των κινήσεων του ρομπότ. Το αριστερό pane χρησιμοποιείται για να επιλέξει και να τροποποιήσει τις θέσεις του επιλεγόμενου ρομπότ.

Η λίστα που εμφανίζεται στο αριστερό pane, διευκρινίζει τον επιθυμητό χρόνο καθώς και ένα όνομα για κάθε θέση του ρομπότ. Ο χρόνος επιδεικνύεται χρησιμοποιώντας το : min : sec : millis.

Εικόνα 5: Webots Motion Editor

Το Webots Motion Editor έχει κάποια κουμπιά. Αυτά είναι το Add, Remove, Up και Down τα οποία χρησιμοποιούνται για να προσθέσουμε, αφαιρέσουμε η να κινήσουμε το ρομπότ πάνω ή κάτω.

Το Fixed step παράθυρο ελέγχου διευκρινίζει το χρονικό διάστημα μεταξύ των poses που θα πρέπει να είναι σταθερό. Εάν αυτό ελέγχεται, τότε ο Motion Editor διαχειρίζεται τα χρονικά βήματα αυτόματα, διαφορετικά πρέπει να εισαχθούν με το χέρι.

Το πεδίο κάτω από το fixed step διευκρινίζει το κατώτερο μέγεθος των σταθερών χρονικών βημάτων στα χιλιοστά του δευτερολέπτου. Εάν αλλαχτεί αυτή η τιμή, αυτό θα απεικονιστεί αυτόματα σε κάθε posa του ρομπότ. Αυτό είναι ένας εύκολος τρόπος να τροποποιηθεί η γενική ταχύτητα αναπαραγωγής ήχου μιας κίνησης.

Το κουμπί Play μας δείχνει την κίνηση του ρομπότ. Πατώντας το κουμπί δίνει εντολή στα servos του ρομπότ παρόμοια με την κλήση της λειτουργίας wb_servo_set_postion (). Θα πρέπει να σημειώσουμε ότι το πρόγραμμα Webots θα πρέπει να είναι σε ”Run” mode ειδάλλως οι μηχανές δεν θα κινηθούν.

Το **Loop** check box μας δίνει τη δυνατότητα να επαναληφθεί η κίνηση του ρομπότ όταν αυτή ολοκληρωθεί.

Το **Reverse** check box μας δίνει την δυνατότητα να έχουμε κίνηση του ρομπότ να παίξει προς τα εμπρός ή προς τα πίσω.

Το Physics check box μας δίνει την δυνατότητα του ελέγχου μεταξύ μιας

φυσικής ή μιας κινηματικής προσομοίωσης του ρομπότ.
Το δεξί pane του Motion Editor θέτει την pose που επιλέγεται εκείνη την περίοδο στο αριστερό pane. Επίσης μας επιτρέπει να προσθέσουμε ή να επιλέξουμε μια ένωση και να επιλέξει μια θέση για εκείνη την ένωση.

Κεφάλαιο 3

3.1 Δημιουργεία 3D αντικειμένων

Μπορούμε να δημιουργήσουμε 3D αντικείμενα χρησιμοποιώντας προγράμματα όπως 3D Studio Max, Maya, AutoCAD, Pro Engineer, AC3D, or Art Of Illusion, MilkShape 3D,blender και το VRML97 (or VRML 2.0).

Αφού δημιουργήσουμε το 3D αντικείμενο σε ένα από τα προγράμματα που προαναφέραμε το κάνουμε export σε vrml97 αρχείο και στη συνέχεια μπορούμε να το εισάγουμε στο webots.(Προσοχή το webots δεν υποστηρίζει vrml 1.0)

Σκοπός: Διαμόρφωση και εξομοίωση του ρομπότ.

Ο στόχος αυτού του κεφαλαίου είναι να παρασχεθούν διάφορα παραδείγματα των ρομπότ, των κόσμων και των ελεγκτών. Ο πρώτος κόσμος είναι πολύ απλός εντούτοις εισάγει την κατασκευή οποιουδήποτε βασικού ρομπότ, και εξηγεί πώς να προγραμματίσει έναν ελεγκτή.

3.2 Ο πρώτος κόσμος μου: mybot.wbt

Σαν πρώτη εισαγωγή, πρόκειται να εξομοιώσουμε ένα πολύ απλό ρομπότ φτιαγμένο από κύλινδρο, δύο ρόδες και δύο υπέρυθρους αισθητήρες (δείτε το σχήμα 4.1). Το ρομπότ ελέγχεται από ένα πρόγραμμα εκτελώντας την αποφυγή εμποδίων που ακολουθεί τον αλγόριθμο Braitenberg. Κινείται σε ένα απλό περιβάλλον που περιέχει έναν τοίχο, καθώς και μερικά εμπόδια για να αποφύγει.

3.2.1 Εγκατάσταση

Πριν αρχίσουμε, πρέπει να ελέξουμε ότι το Webots εγκαταστάθηκε σωστά στον υπολογιστή μας Έπειτα, πρέπει να φτιάξουμε ένα φάκελο που θα περιέχει τα αρχεία που θα δημιουργήσουμε.

Στη συνέχεια πηγαίνουμε στην επιλογή **Wizard —New Project Directory** και επιλέγουμε το φάκελο που δημιουργήσαμε . Αυτή η λειτουργία δημιουργεί έναν κατάλογο αποκαλούμενο my webots και δύο υποκαταλόγους τους worlds και controllers .Ο πρώτος περιέχει τους κόσμους προσομοίωσης που δημιουργούμε, ενώ ο δεύτερος περιέχει τους ελεγκτές για τα μιμούμενα ρομπότ. Για να αρχίσουμε αυτό το project, αντιγράψτε απλά τον κόσμο mybot.wbt από τα Webots projects / sample / mybot / worlds στον δικό σας κατάλογο controllers .

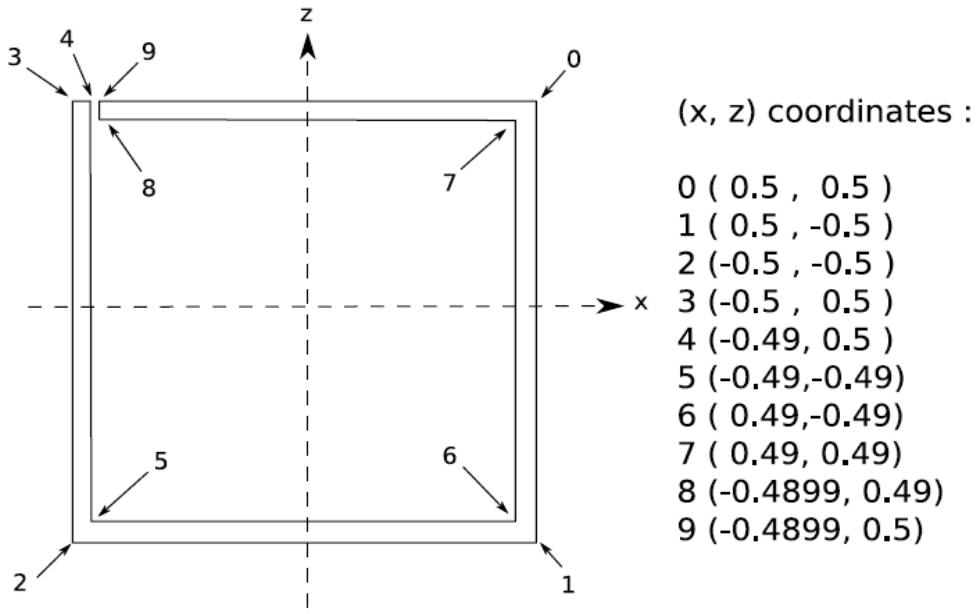
3.2.2 Περιβάλλον

Αυτός ο πρώτος μιμούμενος κόσμος είναι όσο το δυνατόν απλούστερος. Περιλαμβάνει πάτωμα, 4 εμπόδια και έναν περιβάλλοντα τοίχο για να αποτρέψει το ρομπότ από τη διαφυγή. Αυτός ο τοίχος διαμορφώνεται χρησιμοποιώντας έναν κόμβο εξώθησης(Extrusion node).
Οι συντεταγμένες του τοίχου παρουσιάζονται στο σχήμα 4.2.

Κατ' αρχάς, ξεκινάμε το Webots και σταματάμε την τρέχουσα προσομοίωση πατώντας το κουμπί stop. Πηγαίνουμε στο **File** menu, **New World** να δημιουργήσουμε έναν νέο κόσμο.
Κατόπιν ανοίγουμε το scene tree window από το **Scene Tree** στις επιλογές εργαλείων. Αυτό μπορεί επίσης να επιτευχθεί με διπλό κλικ στον τρισδιάστατο κόσμο.

Αρχίζουμε με την αλλαγή του φωτισμού της σκηνής:

1. Επιλέγουμε το PointLight, και πατάμε + μπροστά του. Μπορούμε τώρα να δούμε τους διαφορετικούς τομείς του κόμβου PointLight. Επιλέγουμε το ambientIntensity και εισάγουμε τιμή 0.6, έπειτα επιλέγουμε intensity και δίνουμε την τιμή 0. Τέλος, εισάγουμε στο location τις εξής τιμές [0.75 0.5 0.5].
2. Επιλέγουμε το PointLight αντίγραφο και κάνουμε copy paste με τα αντίστοιχα κουμπιά που βρίσκονται στην κορυφή του scene tree.. Ανοίγουμε αυτόν τον νέο κόμβο PointLight και εισάγουμε [-0.5 0.5 0.35] στο location.
3. Επαναλαμβάνουμε αυτήν την λειτουργία δύο φορές με [0.45 0.5 -0.5] στο location του τρίτου κόμβου PointLight, και [-0.5 0.5 -0.35] στο location του τέταρτου και τελευταίου κόμβου PointLight.
4. Η σκηνή πρέπει να είναι τώρα καλύτερη. Ανοίγουμε το Preferences από τις επιλογές εργαλείων tools (Linux και windows) ή από τις επιλογές Webots (Mac), επιλέγουμε το Rendering tab και τσεκάρουμε την επιλογή Display lights. Κάνουμε click στο πλήκτρο OK και ελέγουμε τις πηγές φωτός να είναι τώρα ορατές στη σκηνή. Δοκιμάζουμε τα διαφορετικά κουμπιά του ποντικιού, συμπεριλαμβανομένης της ρόδας εάν διαθέτουμε, και σέρνουμε το ποντίκι στη σκηνή για να πλοηγηθούμε και να παρατηρήσουμε τις θέσεις των πηγών φωτός.



Αφετέρου, δημιουργούμε τον τοίχο:

1. Επιλέγουμε το solid στο scene tree window (που είναι το πάτωμα) και πατάμε το κουμπί insert after.

2. Επιλέγουμε solid στην επιλογή new node.

3. Ανοίγουμε το πρόσφατα δημιουργημένο solid και πατώντας + ή κλικ με το ποντίκι στο βελάκι δίπλα από το solid γράφουμε “wall” στον τομέα του ονόματός.

4. Επιλέγουμε τον τομέα children και πατάμε **Insert after** και στη συνέχεια Shape node

5. Ανοίγουμε το Shape και επιλέγουμε το apperance ώστε να δημιουργήσουμε έναν Appearance κόμβο από το κουμπί **New node** . Χρησιμοποιούμε την ίδια τεχνική για να δημιουργηθεί material node στο material field. Επιλέγουμε το diffuseColor του material node και επιλέγουμε ένα χρώμα για να καθορίσουμε το χρώμα του τοίχου. Ας βάλουμε το ανοικτό καφέ. Προκειμένου το αντικείμενό μας να αλλάξει το χρώμα του ανάλογα με το φωτισμό, επιλέγουμε το specularColor του material node και επιλέγουμε ένα χρώμα για να καθορίσουμε το χρώμα του φωτισμένου τοίχου. Χρησιμοποιούμε ένα φωτεινότερο καφέ για να απεικονίσουμε την επίδραση του φωτός.

6. Ομοίως, είναι επίσης δυνατό να τροποποιηθούν τα χρώματα του εδάφους. Θα πρέπει να τροποποιήσουμε τους δύο τομείς χρώματος του τελευταίου κόμβου

μετατροπής Transform (αυτός που αντιστοιχεί στο έδαφος), οι οποίοι βρίσκονται στο / Shape / geometry / Color. Στα παραδείγματά μας το έχουμε αλλάξει σε ένα ασπρόμαυρο πλέγμα.

7. Τώρα δημιουργούμε ένα Extrusion στο τομέα geometry του Shape.

8. Θέτουμε το πεδίο convex ως FALSE. Κατόπιν, θέτουμε τις συντεταγμένες στο crossSection όπως φαίνεται στο σχήμα 4.2. Θα πρέπει να καταγράψουμε εκ νέου το πρώτο σημείο (0) στην τελευταία θέση (10) .

9. Στον τομέα spine, γράφουμε ότι ο τοίχος κυμαίνεται μεταξύ 0 και 0.1 κατά μήκος του άξονα Y (αντί των προκαθορισμένων τιμών 0 και 1).

10. Δεδομένου ότι θέλουμε να αποτρέψουμε το ρομπότ μας από τη διάβαση μέσω των τοίχων, πρέπει να καθορίσουμε τον τομέα boundingObject του τοίχου. Η οριοθέτηση των αντικειμένων δεν μπορεί να χρησιμοποιήσει τα σύνθετα αντικείμενα γεωμετρίας. Αυτά που μπορούν να χρησιμοποιηθούν περιορίζονται στο κιβώτιο, τον κύλινδρο, και τη σφαίρα. Ως εκ τούτου, θα πρέπει να δημιουργήσουμε τέσσερα κιβώτια (που αντιπροσωπεύουν τους τέσσερις τοίχους) για να καθορίσουμε τα όρια του περιβάλλοντος τοίχου. Επιλέγουμε τον τομέα boundingObject του τοίχου και δημιουργούμε ένα group που περιέχει τους τέσσερις τοίχους.

Σε αυτό το group , προσθέτουμε ένα Transform στον τομέα children. Δημιουργούμε ένα υλικό στο Appearance και προσαρμόζουμε τα diffuseColor και specularColor στο λευκό. Αυτό θα είναι χρήσιμο αργότερα όταν το ρομπότ
θα πρέπει να ανιχνεύσει τα εμπόδια, επειδή η ανίχνευση αισθητήρων είναι βασισμένη στο χρώμα. Τώρα δημιουργούμε ένα κιβώτιο ως geometry για το Shape. Θέτουμε το μέγεθος του κιβωτίου [0.01 0.1 1], έτσι ώστε να ταιριάζει με το μέγεθος ενός τοίχου. Θέτουμε το translation στο Transform [0.495 0.05 0], έτσι ώστε να ταιριάζει με τη θέση του πρώτου τοίχου.

11. Τώρα, κλείνουμε το Transform και κάνουμε copy και paste στη λίστα children. Αντί της δημιουργίας μιας νέας μορφής για αυτό το αντικείμενο, επαναχρησιμοποιούμε τη μορφή που δημιουργήσαμε για το πρώτο αντικείμενο. Πηγαίνουμε πίσω στο Transform του προηγούμενου αντικειμένου, ανοίγουμε το children, κάνουμε κλικ στο Shape, και θα δούμε στη δεξιά πλευρά του παραθύρου ότι μπορούμε να εισάγουμε ένα όνομα DEF. Γράφουμε WALL SHAPE ως όνομα DEF και επιστρέφουμε στο children του δεύτερου αντικειμένου. Αρχικά διαγράφουμε το Shape που υπάρχει και δημιουργούμε έναν νέο node μέσα σε αυτό. Στο Add a node μπορούμε τώρα να χρησιμοποιήσουμε το WALL SHAPE που καθορίσαμε.
Επιλέγουμε αυτό το στοιχείο και κάνουμε κλικ στο A. Ρυθμίζουμε στο translation του νέου κόμβου [-0.495 0.05 0], έτσι ώστε να ταιριάζει με τον απέναντι τοίχο. Επαναλάμβάνουμε αυτήν την λειτουργία με τους δύο υπόλοιπος τοίχους και ρυθμίζουμε στο τομέα rotation [0 1 0 1.57] έτσι ώστε να ταιριάζουν με τον προσανατολισμό των αντίστοιχων τοίχων.
Πρέπει επίσης να προσαρμόζουμε τους κόμβους translation έτσι ώστε να ταιριάζουν με τη θέση των αντίστοιχων τοίχων.

12. Κλείνουμε το συντάκτη tree, αποθηκεύουμε το αρχείο ως “my mybot.wbt” και βλέπουμε το αποτέλεσμα.

Ο τοίχος στο συντάκτη tree και το αποτέλεσμά του στο συντάκτη world είναι ορατά στο σχήμα 4.3

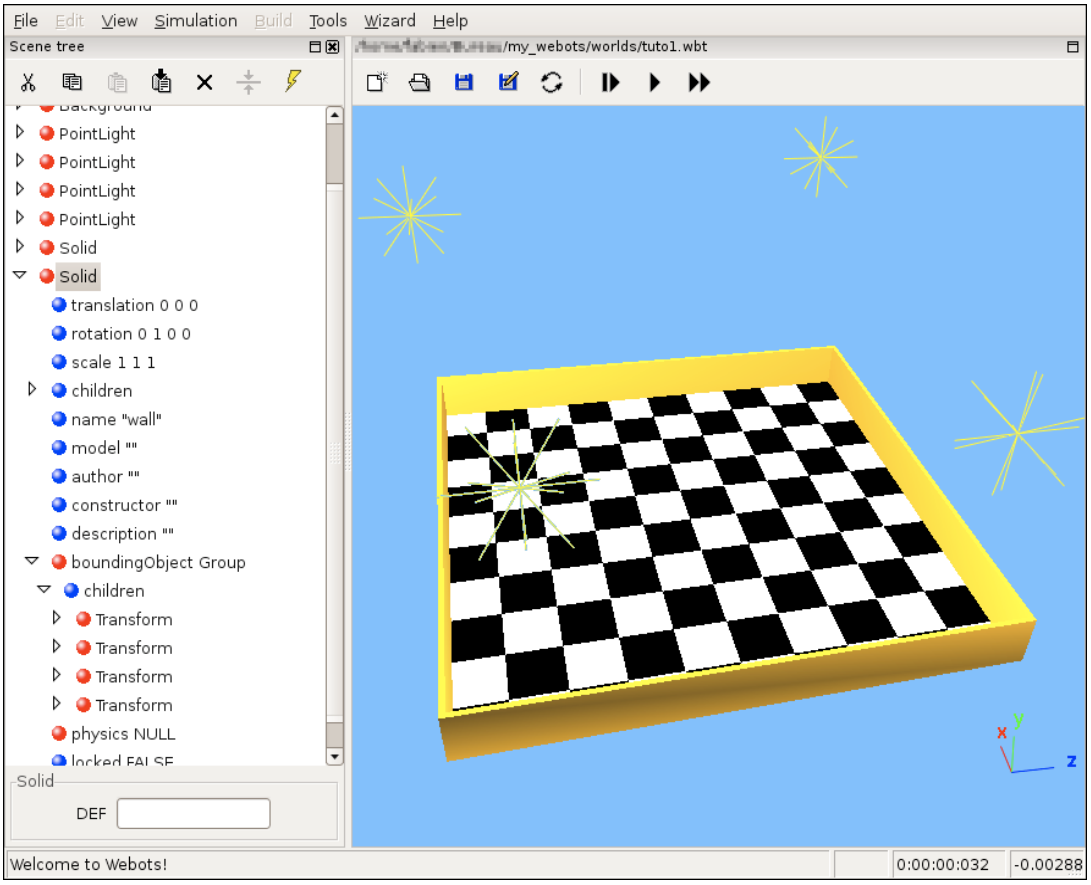
Τώρα ας δημιουργήσουμε τα εμπόδια:

1. Επιλέγουμε τον τελευταίο κόμβο Solid στο παράθυρο scene tree (που είναι ο τοίχος) και κάνουμε κλικ στο κουμπί **insert after**

2. Επιλέγουμε έναν κόμβο Solid.

3. Ανοίγουμε αυτόν τον νέο κόμβο Solid. Πατώντας το + πληκτρολογούμε ”green box” στον τομέα name.

4. Χρησιμοποιώντας την ίδια τεχνική όπως για τον τοίχο, προσθέτουμε πρώτα ένα Shape, έπειτα ένα Appearance και ένα Material.
Για το χρώμα, αλλάζουμε το πράσινο με ένα πιο ανοιχτό πράσινο για τα φωτισμένα μέρη.



Εικόνα 4.3 Ο τοίχος στο tree editor και στο world editor

5. Τώρα δημιουργούμε έναν κόμβο Box στον τομέα geometry του Shape και κάνουμε το μέγεθός του [0.23 0.1 0.1].
Θέτουμε το όνομα DEF ως BOX0.

6. Για να δημιουργήσουμε το boundingObject αυτού του αντικειμένου, δημιουργούμε έναν κόμβο Shape και επαναχρησιμοποιούμε το προηγούμενο DEF για τη geometry. Επίσης, δημιουργούμε ένα Appearance και ένα Material κόμβο και βάζουμε τα δύο χρώματα στο λευκό, όπως κάναμε για τον τοίχο.

7. Τέλος θέτουμε στον τομέα translation τις τιμές [-0.05 0.05 -0.25], αλλά αφήνουμε τον τομέα rotation στην προκαθορισμένη τιμή.

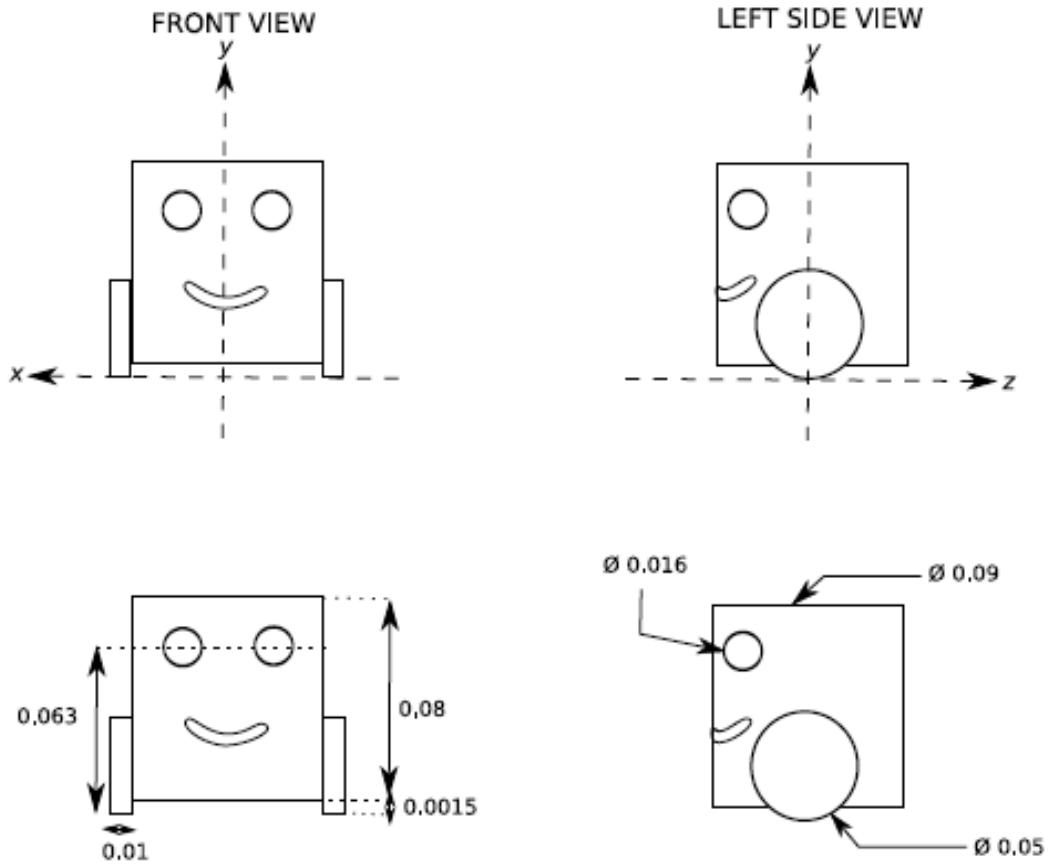
8. Τώρα επαναλαμβάνουμε αυτά τα βήματα για να δημιουργήσουμε τα τρία υπόλοιπα εμπόδια. Αρχικά δημιουργούμε το ένα αποκαλούμενο ”blue box” το οποίο έχει geometry (BOX1) [0.1 0.1 0.1], translation [0.2 0.05 0.27] και rotation [0 1 0 0.31].Κατόπιν δημιουργούμε ένα αποκαλούμενο ”yellow box” με geometry (BOX2) [0.05 0.1 0.3], translation [-0.2 0.05 0.15] rotation [0 1 0 0.4]. Τέλος δημιουργούμε ένα αποκαλούμενο ”red box” με geometry (BOX3) [0.15 0.1 0.08], translation [0.42 0.05 -0.1] και rotation το προεπιλεγμένο. Για όλα αυτά τα αντικείμενα, θέτουμε τα χρώματά τους σύμφωνα με τα ονόματά τους.

Κεφάλαιο 4

4.1 ΤΟ ΡΟΜΠΟΤ

Αυτό το κεφάλαιο περιγράφει πώς μπορούμε να διαμορφώσουμε το ρομπότ MyBot ως ένα DifferentialWheels node που περιέχει διάφορα όπως node μετατροπής για το σώμα, δύο στερεοί nodes για τις ρόδες, δύο nodes DistanceSensor για τους υπέρυθρους αισθητήρες και ένας node μορφής για το πρόσωπο.

Παρακάτω παρατηρούμε το Mybot



Για να διαμορφώσουμε το σώμα του ρομπότ:

1. Ανοίγουμε το scene tree window.

2. Επιλέγουμε τον τελευταίο solid node.

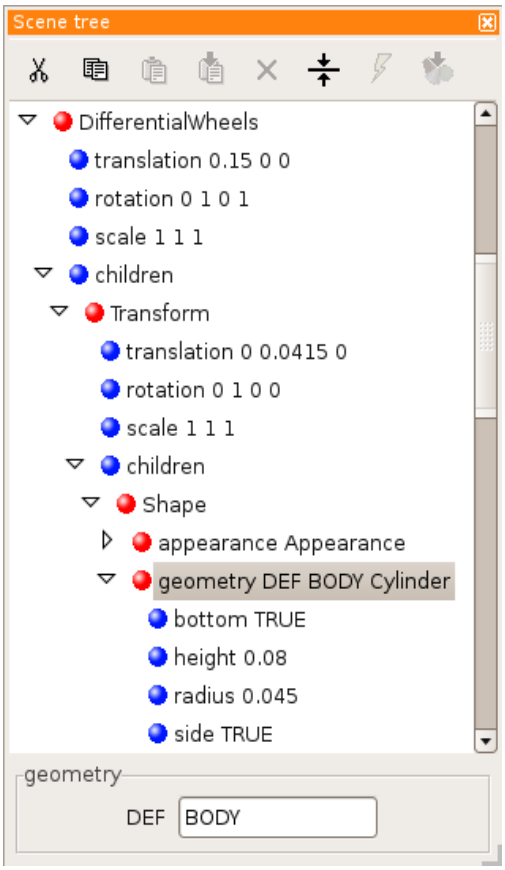
3. Στη συνέχεια προσθέτουμε έναν κόμβο DifferentialWheels , και το ονομάζουμε mybot .Επίσης στο Translation του βάζουμε τις τιμές [0.15 0.04 0]

4. Στο νέο τομέα children, προσθέτουμε έναν Shape node. Εκεί επιλέγοντας από το

appearance το material δίνουμε ένα χρώμα της επιλογής μας.
Στον τομέα γεωμετρίας, προσθέτουμε έναν node Cylinder (σχήμα κυλίνδρου). Θέτουμε τον τομέα ύψους του κυλίνδρου σε 0.08 και της ακτίνας 1 έως 0.045. Θέτουμε το όνομα DEF της γεωμετρίας στο BODY, έτσι ώστε να είμαστε σε θέση να το επαναχρησιμοποιήσουμε αργότερα. Τώρα θέτουμε τις τιμές [0 0.0415 0] στον Transform node.

Για να διαμορφώσουμε την αριστερή ρόδα του ρομπότ:

1. Επιλέγουμε τον Transform node που αντιστοιχεί στο σώμα του ρομπότ και insert after έναν στερεό κόμβο (Solid Node) που θα διαμορφώσει την αριστερή ρόδα. Δακτυλογραφήστε το ”left wheel” στο name field, έτσι ώστε αυτός ο στερεός κόμβος να αναγνωρίζεται ως αριστερή ρόδα του ρομπότ και θα περιστραφεί σύμφωνα με την εντολή μηχανών.



Σώμα του ρομπότ : cylinder

2. Ο άξονας της περιστροφής της ρόδας είναι ο X. Πληροφοριακά η ρόδα αποτελείται από έναν κύλινδρο που περιστρέφεται κατά pi/2 ακτίνια γύρω από τον άξονα Z. Για να λάβετε την κατάλληλη μετακίνηση της ρόδας, θα πρέπει να είστε προσεκτικοί και να μην μπερδέψετε αυτές τις δύο περιστροφές. Συνεπώς, πρέπει να προσθέσετε έναν Transform node στο children του Solid node.

3. Αφού προσθέσουμε αυτόν τον Transform Node, εισάγουμε ένα Shape στον geometry field σχήματος Cylinder. Μην ξεχνάμε να δώσουμε τα απαραίτητα χρώματα

στη ρόδα όπως εξηγείται προηγουμένως. Οι διαστάσεις του κυλίνδρου πρέπει να είναι 0.01 για το Height και 0.025 για το Radius. Θέτουμε το rotation [0 0 1 1.57]. Προσέχουμε στο πρόσημο του rotation γιατί εάν είναι λανθασμένο, η ρόδα θα γυρίσει στη λάθος κατεύθυνση.

4. Στο Solid node, θέτουμε [- 0.045 0.025 0] για να τοποθετήσουμε την αριστερή ρόδα, και να θέσουμε την περιστροφή της ρόδας γύρω από τον άξονα X: [1 0 0 0].

5. Δίνουμε για όνομα DEF στο Transform το WHEEL. Παρατηρούμε ότι καθορίσαμε το wheel translation στο επίπεδο Solid node, έτσι ώστε να μπορούμε να επαναχρησιμοποιήσουμε το WHEELTransform για τη Δεξιά ρόδα.

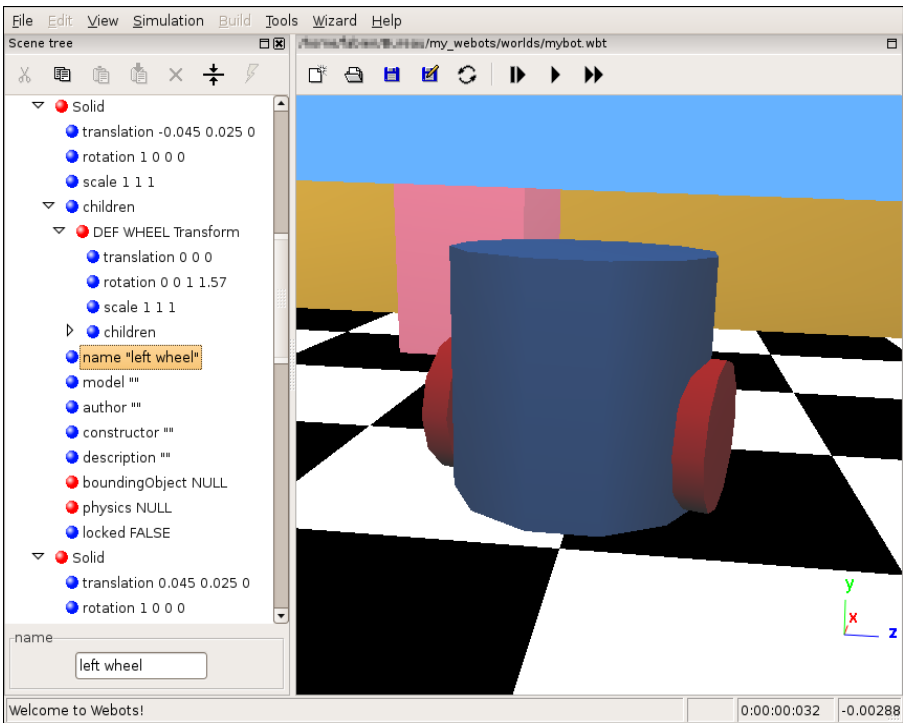
Για να διαμορφώσουμε τη δεξιά ρόδα του ρομπότ:

1. Επιλέγουμε το left wheel Solid node και πατάμε το insert after ενός άλλου Solid node. Το ονομάζουμε ”right wheel” στο Name Field. Θέτουμε το translation ως [0.045 0.025 0] και για το rotation [1 0 0 0].

2. Στον κόμβο children, πατάμε Insert after USEWHEEL.

3. Κλείνουμε το tree window και σώζουμε το αρχείο. Μπορούμε να εξετάσουμε το ρομπότ μας στον world editor, δηλαδή να δώσουμε κίνηση, και να κάνουμε ζουμ.

Το ρομπότ με τις δυο ρόδες μπορούμε να το δούμε και στην παρακάτω φωτογραφία



Κεφάλαιο 5

5.1 Εισαγωγή υπέρυθρων Αισθητηρίων Στο Ρομπότ

Στο ρομπότ μας θα εισάγουμε 2 υπέρυθρους αισθητηρες. Οι δύο υπέρυθροι αισθητήρες ορίζονται ως δύο κύλινδροι στο μέτωπο του σώματος ρομπότ. Η διάμετρός τους είναι 0.016 μ και το ύψος τους είναι 0.004 μ. Πρέπει να τοποθετήσουμε αυτούς τους αισθητήρες κατάλληλα έτσι ώστε οι ακτίνες αισθητήρων να δείχνουν προς στη σωστή κατεύθυνση.

Δημιουργία Αισθητηρίων

1. Στα Children του κόμβου DifferentialWheels, προσθέτουμε (**insert after)** έναν κόμβο Distance Sensor.

2. Πληκτρολογούμε το όνομα ”ir0”. Αυτό θα χρησιμοποιηθεί από το πρόγραμμα ελεγκτών.

3. Συνδέουμε ένα cylinder shape με αυτόν τον αισθητήρα:
Στην λίστα children του κόμβου Distance Sensor, προσθέτουμε (**Insert after)** ένα Transform node. Δώστε ένα όνομα DEF σε αυτό όπως: INFRARED, τις οποίες θα επαναχρησιμοποιήσουμε για το δεύτερο αισθητήρα IR.

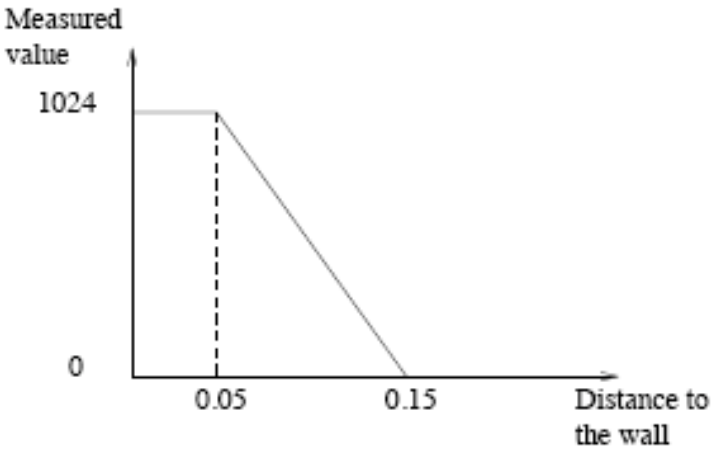
4. Στη συνέχεια στο Children του Transform node πατάμε Insert after - Shape node. Καθορίζουμε το shape και εισάγουμε με το insert έναν Cylinder στο πεδίο γεωμετρίας. Ρυθμίζουμε στο 0.004 για το ύψος και 0.008 για την ακτίνα.

5. Θέτουμε το rotation για το Transform node [0 0 1 1.57] για να ρυθμίσουμε τον προσανατολισμό του κυλίνδρου.

6. Στον κόμβο DistanceSensor, θέτουμε το translation για να τοποθετήσουμε τον αισθητήρα και την ακτίνα του στο : [- 0.02 0.063 -0.042]. Στο **Preferences** dialog, και στο **Rendering** tab, ελέγχουμε το **Display sensor rays** box προκειμένου να κατευθύνουμε την ακτίνα προς το μέτωπο του ρομπότ, θα πρέπει να θέσουμε το rotation [0 1 0 2.07].

7. Στον κόμβο DistanceSensor, θα πρέπει να εισάγουμε μερικές τιμές μέτρησης απόστασης για τους αισθητήρες στο lookupTable πεδίο, σύμφωνα με το παρακάτω σχήμα αυτές οι τιμές είναι:

```
lookupTable [ 0     1024  0,
              0.05  1024  0,
              0.15     0  0 ]
```



8. Για να διαμορφώσουμε το δεύτερο αισθητήρα IR, επιλέγουμε τον κόμβο DistanceSensor και με το **Insert after** έναν νέο κόμβο DistanceSensor. Πληκτρολογούμε ”ir1” ως όνομα. Θέτουμε το translation ως [0.02 0.063 -0.042] και το rotation ως [0 1 0 1.07].
Στο Children, **insert after** USEINFRARED. Στο lookupTable πεδίο, πληκτρολογούμε τις ίδιες τιμές όπως δίνονται παραπάνω.

9. Προκειμένου να ανιχνευθούν καλύτερα τα εμπόδια, θα χρησιμοποιήσουμε δύο ακτίνες ανά DistanceSensor.
Για να γίνει αυτό ανοίγουμε και τους δυο κόμβους DistanceSensor και θέτουμε την τιμή του numberOfRay πεδίου σε 2 και του aperture field 1.

Ρομπότ και οι δύο αισθητήρες του παρουσιάζονται στο σχήμα 4.8.

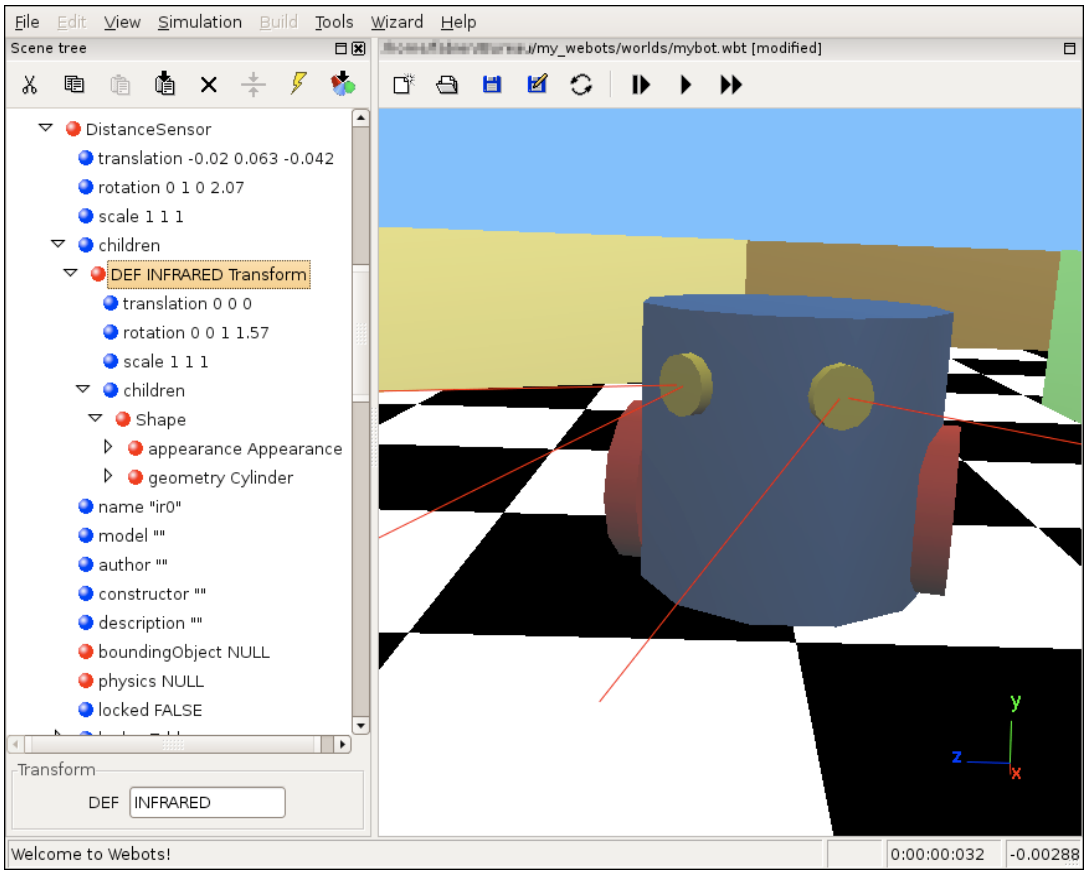
Σημείωση: Ένα texture μπορεί να χαρτογραφηθεί μόνο σε μια μορφή IndexedFaceSet. Το texCoord και texCoordIndex entries πρέπει να συμπληρωθούν.
Η εικόνα που χρησιμοποιείται ως texture πρέπει να είναι αρχείο .png ή ένα αρχείο .jpg, και το μέγεθός του πρέπει να είναι (2n) * (2n) pixels (παραδείγματος χάριν 8x8, pixels 16x16, 32x32, 64x64, 128x128 ή 256x256). Οι διαφανείς εικόνες επιτρέπονται σε Webots. Επιπλέον, οι εικόνες PNG πρέπει να χρησιμοποιούν 24 ή 32 bit ανά pixel (χαμηλότερο bpp ή επίπεδα του γκρι δεν υποστηρίζονται).
Beware των ορίων στις εικόνες texture που επιβάλλονται από τον τρισδιάστατο πίνακα γραφικής παράστασής: μερικοί παλαιοί τρισδιάστατοι πίνακες γραφικής παράστασης είχαν όριο τα 256x256 textures, ενώ ισχυρότεροι δέχονται 2048x2048 textures.

Για να δημιουργήσοyme ένα texture στο πρόσωπο του ρομπότ:

1. Επιλέγουμε τον τελευταίο κόμβο DistanceSensor και στο **Insert after** προσθέτουμε μετά έναν κόμβο Shape.

2. Δημιουργούμε έναν κόμβο Appearance στον τομέα appearance. Δημιουργούμε έναν ImageTexture κόμβο στον τομέα texture αυτού του κόμβου, με το ακόλουθο url: "textures/mybot.png".

3. Στον τομέα geometry δημιουργούμε έναν κόμβο IndexedFaceSet, με έναν Coordinate κόμβο στον τομέα coord. Πληκτρολογούμε τις συντεταγμένες των σημείων στον τομέα point:



Εικόνα 4.8: Οι κόμβοι Distance Sensor του MyBot robot

```
[ 0.015   0.038   -0.041,
  0.015   0.023   -0.041,
  0       0.023   -0.0455,
 -0.015   0.023   -0.041,
 -0.015   0.038   -0.041,
  0       0.038   -0.0455 ]
```

και εισάγουμε μετά στον coordIndex τομέα τις ακόλουθες τιμές: 0, 1, 2, 5, -1, 5, 2, 3, 4, -1. Η τιμή -1 υπάρχει για να δηλώσει το τέλος ενός προσώπου. Αυτό είναι χρήσιμο κατά το καθορισμό διάφορων προσώπων για τον ίδιο κόμβο IndexedFaceSet.

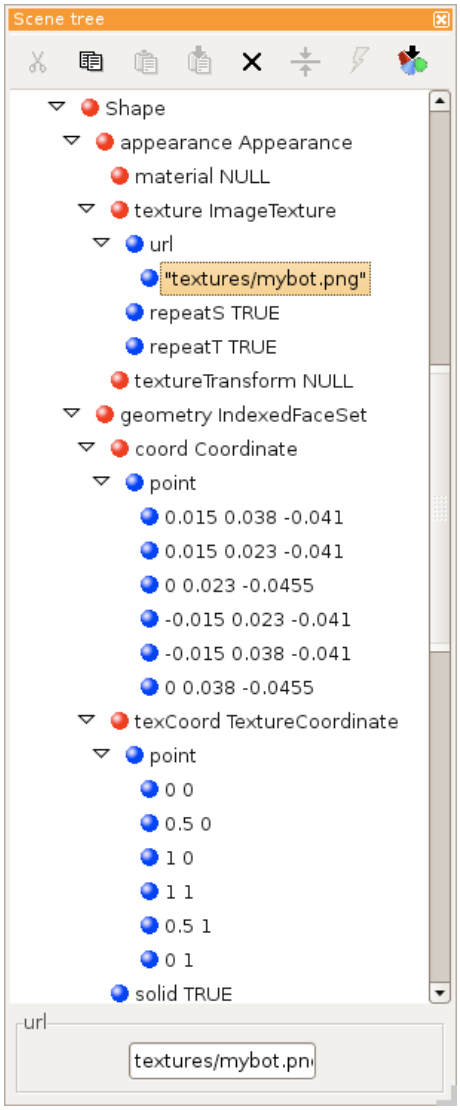
4. Στον τομέα texCoord, δημιουργούμε έναν κόμβο TexureCoordinate. Στον τομέα point, εισάγουμε τις συντεταγμένες του texture:

```
[ 0   0
  0.5 0
  1   0
  1   1
  0.5 1
  0   1 ]
```

και στον τομέα texCoordIndex, πληκτρολογούμε: 5, 0, 1, 4, -1, 4, 1, 2, 3, -1. Αυτό είναι ο τυποποιημένος τρόπος σε VRML97 και εξηγεί πώς το texture πρέπει να χαρτογραφηθεί στο αντικείμενο.

5. Στο παράδειγμά μας, έχουμε τροποποιήσει επίσης την τιμή του creaseAngle του IndexedFaceSet.
Αυτός ο τομέας τροποποιεί τον τρόπο που γίνεται η μετάβαση του φωτισμού μεταξύ των διαφορετικών προσώπων του IndexedFaceSet. Στο παράδειγμά μας, έχουμε θέσει την τιμή της σε 0.9 έτσι ώστε η μετάβαση φωτισμού να είναι ομαλή μεταξύ των δύο προσώπων.

6. Οι τιμές texture παρουσιάζονται στο σχήμα 4.9.



Σχήμα 4.9: Καθορισμός του texture του ρομπότ MyBot

Για να τελειώσουμε με τον κόμβο DifferentialWheels, πρέπει να συμπληρώσουμε μερικούς ακόμα τομείς:

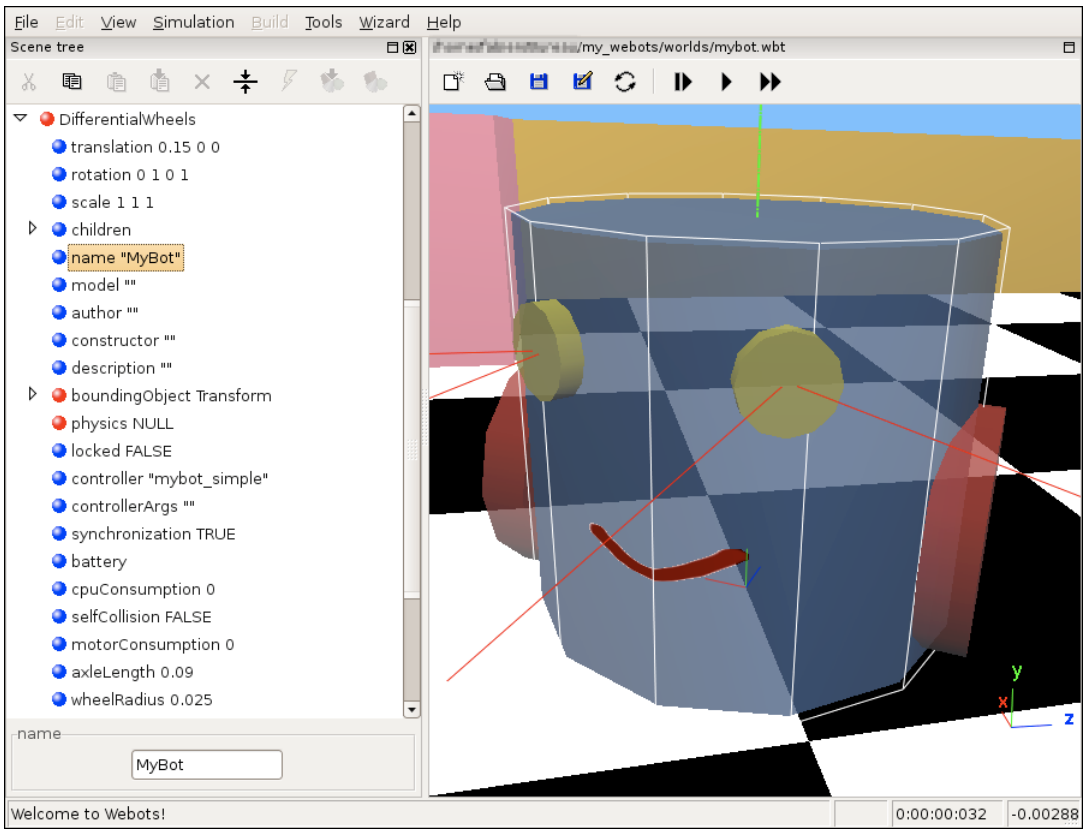
1. Στον τομέα controller, επιλέγουμε ”mybot simple,” που πρέπει να εμφανιστεί στη λίστα ελεγκτών που εμφανίζεται όταν πιέζετε το κουμπί επιλογής αρχείων. Χρησιμοποιείται για να καθορίσει ποιοι ελεκτές προγράμματος ελέγχουν το ρομπότ.

2. Ο τομέας boundingObject μπορεί να περιέχει έναν Transform κόμβο με έναν κύλινδρο. Δημιουργούμε έναν Transform κόμβο στον τομέα boundingObject, με translation [0 0.0415 0]. Επαναχρησιμοποιούμε τον κόμβο Body που καθορίσαμε προηγουμένως, και τον προσθέτουμε στο children του transform.

3. Στον τομέα axleLength, εισάγουμε το μήκος του άξονα μεταξύ των δύο ροδών: 0.09 (σύμφωνα με το σχήμα 4.4).

4. Στον τομέα wheelRadius, εισάγουμε την ακτίνα των ροδών: 0.025.

5. Οι τιμές για άλλους τομείς και το ολοκληρωμένο πλέον ρομπότ παρουσιάζονται στο σχήμα 4.10.



Σχήμα 4.10: Οι άλλοι τομείς του κόμβου DifferentialWheels

Το αρχείο mybot.wbt συμπεριλαμβάνεται στη Webots διανομή, στο φάκελο worlds.

5.2 Αρχές της ανίχνευσης σύγκρουσης

Η μηχανή ανίχνευσης σύγκρουσης είναι σε θέση να ανιχνεύσει κάθε σύγκρουση μεταξύ οποιονδήποτε δύο στερεών κόμβων. Με τον όρο στερεό κόμβο, εννοούμε το στερεό κόμβο Solid node και όλους τους παραγόμενους κόμβους, και αυτό περιλαμβάνει το DifferentialWheels, το ρομπότ, τους σέρβο κόμβους, κ.λπ. Ο σκοπός της ανίχνευσης σύγκρουσης είναι να αποτραπούν δύο οποιαδήποτε αντικείμενα από την αλληλοδιαπέραση. Αυτό επιτυγχάνεται με την παραγωγή των δυνάμεων επαφών που θα ωθήσουν τα στερεά χώρια όποτε αυτό είναι απαραίτητο. Η μηχανή ανίχνευσης σύγκρουσης υπολογίζει τη διατομή μεταξύ των αντικειμένων αυτών των στερεών κόμβων. Το αντικείμενο περιγράφεται στον τομέα boundingObject.

Ένα αντικείμενο αποτελείται από μια γεωμετρική μορφή ή μια ομάδα γεωμετρικών μορφών που καθορίζουν τα όρια του στερεού. Εάν ο τομέας boundingObject είναι NULL, τότε καμία ανίχνευση σύγκρουσης δεν θα εκτελεσθεί σε αυτό το συγκεκριμένο στερεό και επομένως τίποτα δεν θα την αποτρέψει από τη διάβαση μέσω άλλων στερεών.

Ένας στερεός κόμβος μπορεί να περιέχει άλλους στερεούς κόμβους στον κατάλογο children, ο κάθε ένας από αυτούς έχει το δικό του αντικείμενο. Επομένως είναι δυνατό να καθοριστούν οι σύνθετες στερεές ιεραρχίες, οι οποίες μπορούν να χρησιμοποιηθούν παραδείγματος χάριν για να διαμορφώσουν ένα αρθρωμένο ρομπότ.

5.3 Ένας απλός ελεγκτής (controller)

Αυτός ο πρώτος ελεγκτής είναι πολύ απλός, γι αυτό και τον ονομάσαμε mybot_simple. Το πρόγραμμα ελεγκτών διαβάζει απλά τις τιμές αισθητήρων και θέτει τις ταχύτητες των δύο μηχανών κατά τέτοιο τρόπο ώστε το MyBot να αποφεύγει τα εμπόδια.

Παρακάτω παρουσιάζεται ο πηγαίος κωδικός για τον ελεγκτή mybot_simple.c:

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/distance_sensor.h>
#define SPEED 60
#define TIME_STEP 64
int main()
{
wb_robot_init(); /* απαραίτητο για να ενεργοποιηθεί στο webots το ρομπότ */
/* Παίρνει και επιστρέφει τους αισθητήρες απόστασης */

WbDeviceTag ir0 = wb_robot_get_device("ir0");
WbDeviceTag ir1 = wb_robot_get_device("ir1");
wb_distance_sensor_enable(ir0, TIME_STEP);
wb_distance_sensor_enable(ir1, TIME_STEP);
while(wb_robot_step(TIME_STEP)!=-1) {

/* Παίρνει τις τιμές από τους αισθητήρες απόστασηςGet */
double ir0_value = wb_distance_sensor_get_value(ir0);
double ir1_value = wb_distance_sensor_get_value(ir1);
/* Υπολογίζει τις ταχύτητες μηχανών */
double left_speed, right_speed;
if (ir1_value > 500) {
/*Εάν και οι δύο αισθητήρες απόστασης ανιχνεύουν κάτι, αυτό σημαίνει ότι αντιμετωπίζουμε έναν τοίχο. Σε αυτήν την περίπτωση πρέπει να κινηθούμε προς τα πίσω.
*/
if (ir0_value > 500) {
left_speed = -SPEED;
right_speed = -SPEED / 2;
} else {
/*
Υπολογίζουμε τις τιμές των αισθητήρων επειδή όσο πιό κοντά είναι το ρομπότ στον τοίχο τόσο περισσότερο πρέπει να απομακρυνθεί.
*/
left_speed = -ir1_value / 10;
```

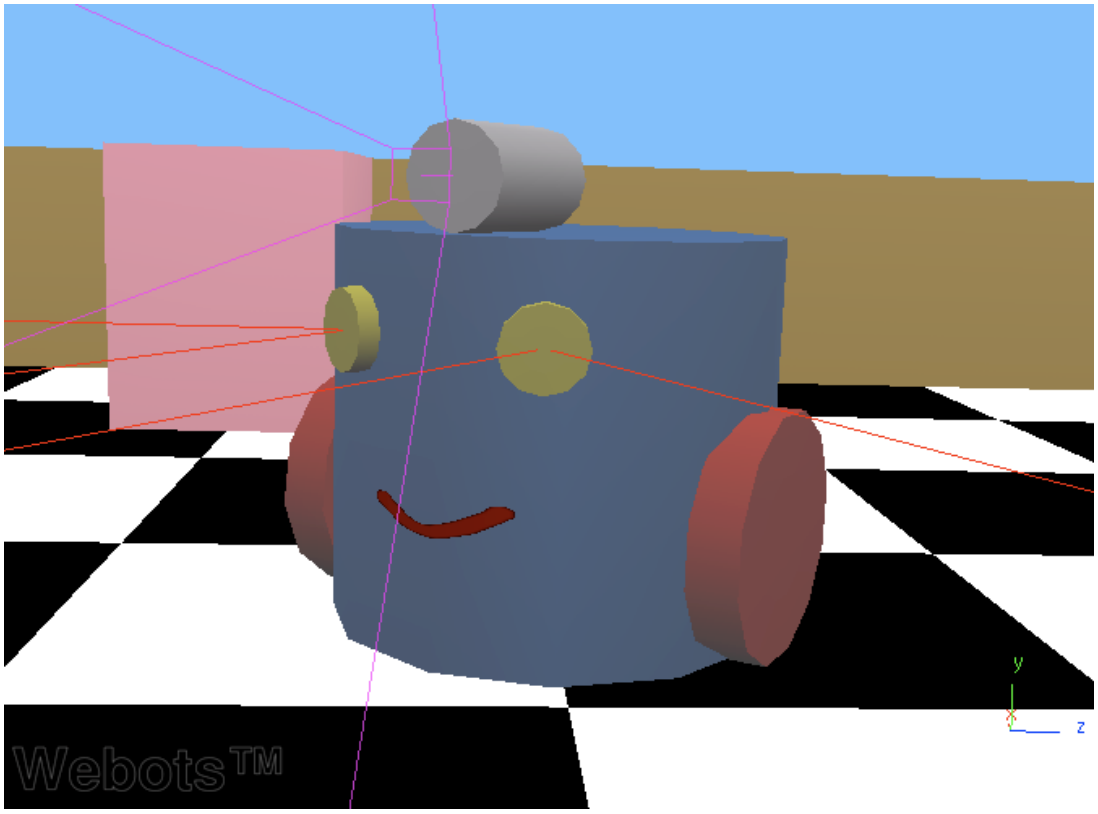
```
right_speed = (ir0_value / 10) + 5;
}
} else if (ir0_value > 500) {
left_speed = (ir1_value / 10) + 5;
right_speed = -ir0_value / 10;
} else {
/*
Εάν τίποτα δεν έχει ανιχνευθεί μπορούμε να κινηθούμε με τη μέγιστη ταχύτητα.
*/
left_speed = SPEED;
right_speed = SPEED;
}
/* Θέτουμε τις τιμές για την ταχύτητα των κινητήρων*/
}
return 0;
}
```

Αυτός ο ελεγκτής βρίσκεται στο subdirectory mybot_simple του καταλόγου ελεγκτών (στο projects/samples/mybot directory του Webots).

5.4 Προσθήκη μιας κάμερας στο ρομπότ MyBot

Αυτό το μάθημα μπορεί να θεωρηθεί ως άσκηση για να ελέγξει εάν κατανοούμε τις αρχές για τις συσκευές σε ένα ρομπότ. Η κάμερα που προσαρμόζεται είναι μια 2D έγχρωμη κάμερα, με μια εικόνα 80x60 pixels, και ένα οπτικό πεδίο 60 μοιρών.

Μπορούμε να διαμορφώσουμε τη μορφή της κάμερας ως έναν κύλινδρο, στην κορυφή του ρομπότ MyBot στο μέτωπό του. Οι διαστάσεις του κυλίνδρου είναι 0.01 για την ακτίνα και 0.03 για το ύψος. Δείτε το σχήμα 4.11



Σχήμα 4.11: Το ρομπότ MyBot με μια κάμερα

Ας προσπαθήσουμε να φτιάξουμε αυτήν την κάμερα. Το αρχείο mybot_camera.wbt συμπεριλαμβάνεται στη διανομή Webots, στον κατάλογο worlds, σε περίπτωση που το χρειαζόμαστε για βοήθεια.

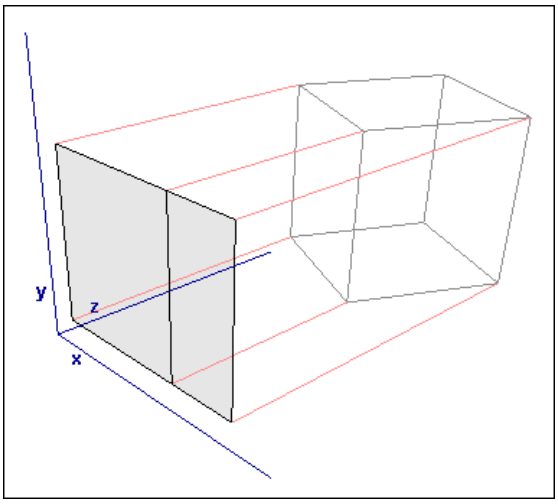
Ένα πρόγραμμα ελεγκτών για αυτό το ρομπότ, που ονομάζεται mybot_camera, συμπεριλαμβάνεται επίσης στο project mybot, στον κατάλογο controllers. Αυτό το πρόγραμμα καμερών δεν εκτελεί την επεξεργασία εικόνας, δεδομένου ότι είναι ένα πρόγραμμα επίδειξης, αλλά θα μπορούσαμε εύκολα να το επεκτείνουμε για να εκτελέσουμε την πραγματική επεξεργασία εικόνας. Το ρομπότ θα μπορούσε, παραδείγματος χάριν, να μάθει να αναγνωρίζει τα διαφορετικά αντικείμενα της σκηνής και να κινηθεί προς ή μακριά από αυτά ανάλογα με εάν το αντικείμενο είναι ταξινομημένο ως ”καλό” ή “κακό”. Ακολουθούν άλλωστε αντίστοιχα παραδείγματα!

5.5 Επεξήγηση βασικών εννοιών σχετικών με την camera:

5.5.1 Projection Matrix

Με τον όρο projection εννοούμε 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 δισδιάστατη επιφάνεια, έτσι εάν θέλουμε να παρουσιάσουμε τις τρισδιάστατες εικόνες, χρειαζόμαστε έναν τρόπο να μετασχηματίσουμε την τρισδιάστατη γεωμετρία σε μια μορφή που μπορεί να δοθεί ως 2D εικόνα. Και αυτό ακριβώς είναι που κάνει το projection matrix. Για να χρησιμοποιήσουμε ένα πολύ απλό παράδειγμα, ένας τρόπος να προβληθεί ένα τρισδιάστατο αντικείμενο επάνω σε μια 2D επιφάνεια θα ήταν να προσθέσουμε τη διάσταση z σε ένα σχήμα με συντεταγμένες x και y όπως φαίνεται στο σχήμα 1.



Σχήμα 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 παράδειγμά μας θα θεωρήσουμε 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 μετασχηματίζετε σε έναν νέο όγκο .

Με αυτόν τον τρόπο, μπορούμε να πάρουμε τις πραγματικές μετατροπές προβολής. Υπάρχουν πολλές διαφορετικές μέθοδοι προβολής όπως για παράδειγμα η ορθογραφική και η προοπτική.

Λαμβάνοντας υπόψη τη 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.

5.5.2 Transformations

Camera Transformations (Μετασχηματισμοί καμερών)

Λέγοντας camera transformation εννοούμε διαφόρους μετασχηματισμούς που γίνονται στην κάμερα βάση την κίνησης της κάμερας στο χώρο. Για παράδειγμα εάν η κάμερα δεν κινείται σωστά σε μια προσημείωση πρώτου προσώπου τότε οποιαδήποτε ελπίδα ρεαλισμού της λήψης χάνεται. Ευτυχώς για αυτό γιατί ο κώδικας καμερών που υπάρχει είναι σχετικά εύκολος να εφαρμοστεί χρησιμοποιώντας κάποια matrix (όπως έχουμε ήδη αναλύσει παραπάνω).

Κάθε κάμερα πρέπει να περιέχει ένα matrix που να καθορίζει τον προσανατολισμό της Έτσι στο camera matrix για παράδειγμα ενσωματώνονται κάποια matrix object, όπως το CT, το CX, την CY και το CZ που όπως καταλαβαίνουμε αφορούν διανύσματα στο χώρο. Ανάλογα με την εφαρμογή σας, τα διανύσματα μετατόπισης μπορεί να είναι και αρνητικά. Μερικές φορές το [dx, dy, DZ] και [X angle, Y angle, Z angle] πρέπει να είναι αρνητικά. Εάν η κάμερά μας κινείται προς τα εμπρός αντί προς τα πίσω, δεξιά αντί του αριστερού, επάνω αντί κάτω, κ.λπ. το διάνυσμα πρέπει να είναι κάπου αρνητικό. Το matrix τελικά δίνεται από τον παρακάτω τύπο :

$$[C] = [C] * [CT] * [CX] * [CY] * [CZ]$$

5.5.3 World Coordinates

Για να καθορίσουμε πώς βαθμονομείται μια κάμερα για να καθορίσει τη σχέση μεταξύ αυτού που εμφανίζεται στην camera και αυτού που βρίσκεται στον τρισδιάστατο κόσμο αρκεί να μελετήσουμε το σχηματισμό εικόνας και τη γεωμετρία καμερών με λίγο περισσότερες λεπτομέρειες

Στο σύστημα συντεταγμένων όπως παρουσιάζεται από ένα πρόγραμμα γραφικής παράστασης οι παγκόσμιες συντεταγμένες μπορούν να καθοριστούν ώστε να είναι κατάλληλες για την εφαρμογή. Ο σκοπός τους είναι να παραγάγουν ανεξάρτητα από το μέγεθος του πλαισίου. την πιο ελκυστική, κατάλληλα τοποθετημένη παρουσίαση.

5.5.4 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 ποιές 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.

5.6 Προσθήκη της φυσικής στην προσομοίωση MyBot

Το μοντέλο που έχουμε καθορίσει για το ρομπότ MyBot δεν περιλαμβάνει κάποια διαμόρφωση φυσικής, δεδομένου ότι δεν διευκρινίσαμε κάποια μάζα, κ.λπ. Αντ΄ αυτού είναι ένα απλό κινηματικό πρότυπο που μπορεί να χρησιμοποιηθεί εν τούτοις για πολλά κινητά πειράματα προσομοίωσης ρομποτικής όπου η αδράνεια και η τριβή μπορούν να παραμεληθούν. Παραδείγματος χάριν, είναι ρυθμισμένο για να μιμηθεί τα μικρά ρομπότ υπολογιστών γραφείου όπως Khepera ή Hemisson. Επιπλέον, οι προσομοιώσεις τρέχουν γρηγορότερα χωρίς φυσική.
Όταν όμως τα πράγματα είναι πιο σύνθετα, θα πρέπει να εισάγουμε κάποια φυσική στο πρότυπό μας.
Παραδείγματος χάριν, εάν το ρομπότ μας είναι βαρύ, δεν μπορούμε να παραμελήσουμε τα αποτελέσματα αδράνειας στην τροχιά του. Εάν θέλουμε να προσθέσουμε τα κινητά αντικείμενα, όπως τα κιβώτια ή μια σφαίρα, η προσομοίωση φυσικής γίνεται απαραίτητη. Τέλος, εάν θέλουμε να διαμορφώσουμε μια αρχιτεκτονική ρομπότ σημαντικά διαφορετική από τις συνηθισμένες με τις ρόδες μπορούμε να διαμορφώσουμε ένα ρομπότ με πόδια, ένα ρομπότ που κολυμπάει ή ένα ρομπότ που πετάει, όμως σε αυτή την περίπτωση χρειαζόμαστε στην οργάνωση πολλές παραμέτρους φυσικής.
Εμείς θα μελετήσουμε μια απλή προσομοίωση φυσικής στον κόσμο MyBot που επιτρέπει στο ρομπότ να παίξει με μια σφαίρα. Οι πιο σύνθετες προσομοιώσεις φυσικής μπορούν να εφαρμοστούν στο Webots, περιλαμβάνοντας παραδείγματος χάριν τα διαφορετικά σχέδια μετακίνησης βασισμένα στο ρομπότ και τους σέρβο κόμβους, που επιτρέπουν να χτιστούν τα σύνθετα τροχοφόρα και τα ρομπότ με πόδια. Άλλες δυνατότητες περιλαμβάνουν τα ρομπότ που κολυμπάνε και που πετάνε όπου απαιτούνται τα πρότυπα υδροδυναμικής. Αυτά τα χαρακτηριστικά γνωρίσματα δεν θα εξεταστούν εδώ.

5.6.1 Προετοιμασία του πατώματος για μια προσομοίωση φυσικής.

Επιλέγουμε τον κόμβο του πατώματος, που πρέπει να είναι ο πρώτος κόμβος μετατροπής στο scene tree, αμέσως μετά από τους κόμβους Point Light.
Αλλάζουμε το Transform στο Solid node χρησιμοποιώντας το κουμπί Transform
Τώρα είναι δυνατό να καθοριστεί ένα bounding Object για το πάτωμα. Δημιουργούμε έναν κόμβο μετατροπής Transform node που περιέχει ένα κιβώτιο ως αντικείμενο με το οποίο θα γίνει η οριοθέτηση. Θέτουμε το μέγεθος για το κιβώτιο (size field)[1 0.02 1] και τον translation field [0.05 -0.01 0.05]. Το αντικείμενο με το οποίο θα γίνει η οριοθέτηση που καθορίσαμε θα αποτρέψει το ρομπότ από να πέσει κάτω μέσω του πατώματος λόγω της βαρύτητας.

5.6.2 Προσθήκη της φυσικής στο ρομπότ MyBot

Το ρομπότ MyBot ήδη έχει καθορίσει ένα οριοθετημένο αντικείμενο. Εντούτοις, δεδομένου ότι θα κινείται, χρειάζεται επίσης τις παραμέτρους φυσικής που θα καθοριστούν στον τομέα φυσικής ως κόμβος φυσικής(Physics node). Δημιουργούμε έναν τέτοιο κόμβο (συστήνεται να χρησιμοποιηθεί η μάζα αντί της πυκνότητας)και θέτουμε την πυκνότητά του σε -1 και τη μάζα του σε 0.5. Η πυκνότητα εκφράζεται σε χιλιόγραμμα ανά κυβικό μετρητή, και η μάζα σε χιλιόγραμμα. Η μάζα δεν λαμβάνεται υπόψη όταν διευκρινίζεται η πυκνότητα.
Οι ρόδες του ρομπότ χρειάζονται επίσης μερικές ιδιότητες φυσικής για να καθορίσουν την τριβή με το πάτωμα. Αλλά πρώτα χρειάζονται ένα οριοθετημένο αντικείμενο. Θέτουμε τον καθορισμένο κόμβο ροδών (WHEEL node) ως bounding Object για κάθε στερεό ροδών.

Κατόπιν, προσθέτουμε έναν κόμβο φυσικής (Physics node)στην πρώτη ρόδα, και εισάγουμε τη φυσική ροδών(WHEEL PHYSICS) ως όνομα DEF. Τέλος, θέτουμε την πυκνότητα σε -1, τη μάζα σε 0.05, το coulombFriction σε 1 και το forceDependantSlip σε 0. Χρησιμοποιούμε αυτόν τον καθορισμό φυσικής ροδών για να καθορίσετε τη φυσική της δεύτερης ρόδας. Τώρα είμαστε έτοιμοι!

Σώζουμε τον κόσμο ως my_mybot_physics.wbt, τον ξαναφορτώνουμε και τρέχουμε την προσομοίωση. Θα παρατηρήσουμε ότι το ρομπότ δεν κινείται πολύ σταθερά (ειδικά εάν εξετάσουμε αυτό που η κάμερα του ρομπότ βλέπει). Αυτή είναι φυσική! Φυσικά μπορούμε να βελτιώσουμε τη σταθερότητα της μετακίνησης με τη ρύθμιση του οριοθετημένου αντικειμένου του ρομπότ, της ταχύτητας των ροδών, των παραμέτρων τριβής, κ.λπ.

Κεφάλαιο 6

6.1 Physics

6.1.1 Προσθήκη μιας σφαίρας στον κόσμο MyBot

Τώρα θα φτιάξουμε ένα παιχνίδι για το ρομπότ μας. Αντί να το δημιουργήσουμε από την αρχή θα το δανειστούμε αφού υπάρχει ήδη στον supervisor.wbt world. Με διπλό κλικ στη μπάλα ποδοσφαίρου θα ανοίξει το παράθυρο scene tree και θα επιλέξουμε το Ball solid. Απλά το αντιγράφουμε χρησιμοποιώντας το κουμπί Copy και μετά ανοίγουμε τον κόσμο mybot_physics.wbt. Ανοίγουμε το παράθυρο Scene Tree, επιλέγουμε το τελευταίο αντικείμενο του Scene Tree και κάνουμε κλικ στο κουμπί Paste After .Τώρα μπορούμε να δούμε τη μπάλα ποδοσφαίρου. Στη συνέχεια τοποθετούμε τη μπάλα μπροστά από το ρομπότ. Στη συνέχεια σώζουμε το World και έπειτα τρέχουμε την προσομοίωση. Το ρομπότ MyBot πρέπει να είναι σε θέση να κλωτσήσει τη μπάλα, κάνοντας κύκλους και αποφεύγοντας τους τοίχους.

6.1.2 Περιβάλλον

Το περιβάλλον αποτελείται από:
- μια σκακιέρα: ένας Solid node με έναν κόμβο ElevationGrid.
- ένα τοίχο γύρω από τη σκακιέρα: Solid node με έναν κόμβο Extrusion.
- ένα τοίχο μέσα στον κόσμο: ένας Solid node με έναν Cylinder node.

Αυτό το περιβάλλον παρουσιάζεται στο σχήμα 4.12

6.1.3 Ρομπότ με 16 sonars

Το ρομπότ (ένας κόμβος DifferentialWheels) αποτελείται από πέντε κύρια μέρη:

1. το σώμα: ένα Extrusion node.

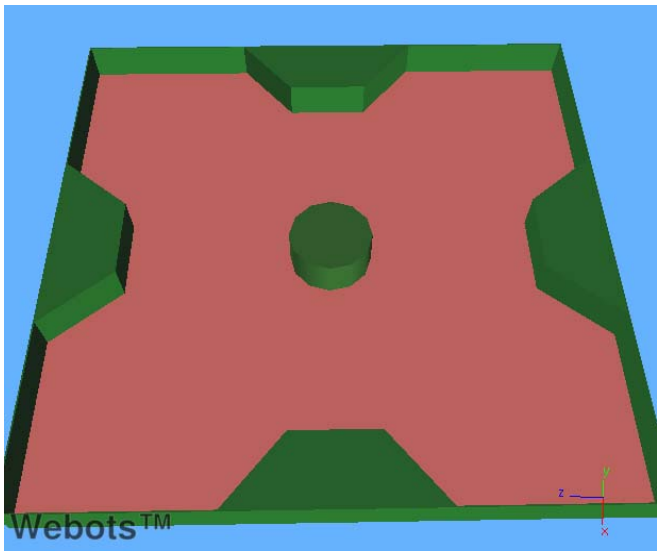
2. Ένα top plate: ένα Extrusion node.

3. δύο κύριες ρόδες: δυο Cylinder nodes.

4. μια οπίσθια ρόδα: ένα Cylinder node.

5. Μπροστινές και πίσω υποστηρίξεις αισθητήρων: δύο Extrusion nodes.

δέκα έξι sonars: δέκα έξι κόμβοι DistanceSensor.



Εικόνα 4.12: Ο τοίχος από τον Pioneer 2TM robot world

Το Pioneer 2 DXTM απεικονίζεται στο σχήμα 4.13.
Ανοίγουμε το Tree editor και προσθέτουμε έναν κόμβο DifferentialWheels. Εισάγουμε στον Chilren Field :

1. για το Body: ένα Shape node με ένα geometryExtrusion. Δείτε το σχήμα 4.14 για τις συντεταγμένες για το Extrusion.

2. για το Top Plate: ένας Shape node με ένα geometryExtrusion. Δείτε το σχήμα 4.15 για τις συντεταγμένες της Extrusion.

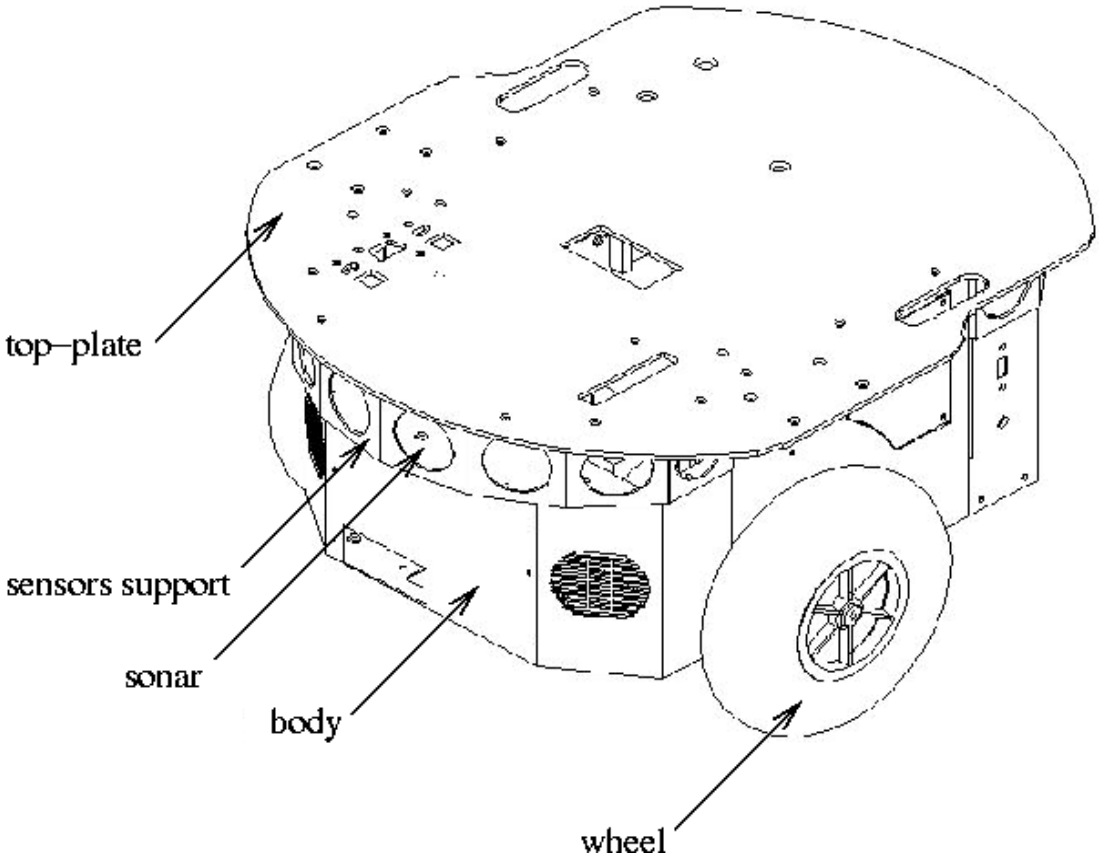
3. για τις δύο κύριες ρόδες: δύο Solid nodes. Κάθε Children του Solid node περιέχει ένα Transfer Node το οποίο περιέχει και αυτό ένα Shape node με ένα geometryCylinder. Κάθε solid node έχει ένα όνομα : Left Wheel και Right Wheel. Δείτε το σχήμα 4.16 για τις διαστάσεις που έχουν οι ροδές.

4. για την πίσω ρόδα: ένα Transform Node που περιλαμβάνει ένα shape node με έναν Geometry Field θέστε τις τιμές του Cylinder, όπως φαίνεται στο σχήμα 4.17

5. για την υποστηρίξη του sonar: δύο shape node με ένα geometryExtrusion. Δείτε το σχήμα 4.18 για τις Συντεταγμένες Extrusion.

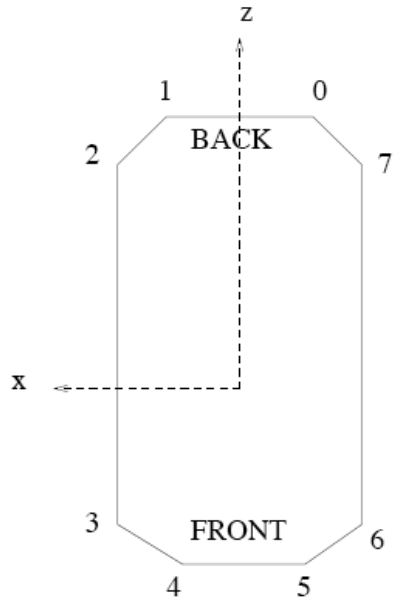
6. για 16 sonars: 16 DistanceSensor nodes. Κάθε node distanceSensor περιέχει ένα Transform Node. Ο Transform Node έχει έναν κόμβο shape που περιέχει ένα geometryCylinder.Δείτε το σχήμα 4.19 και το κείμενο από κάτω για περισσότερη εξήγηση.

Διαμόρφωση sonars:
Η αρχή είναι η ίδια όπως για το ρομπότ MyBot. Sonars είναι κύλινδροι με μια ακτίνα 0.0175 και ένα ύψος 0.002. Υπάρχουν 16 sonars, 8 στο μπροστινό μέρος του ρομπότ και 8 στο πίσω τμήμα του ρομπότ (δείτε το σχήμα 4.19). Οι γωνίες μεταξύ sonars και της αρχικής θέσης του DEFSONARTransform παρουσιάζονται



Εικόνα 413 The Pioneer 2 DXTM robot

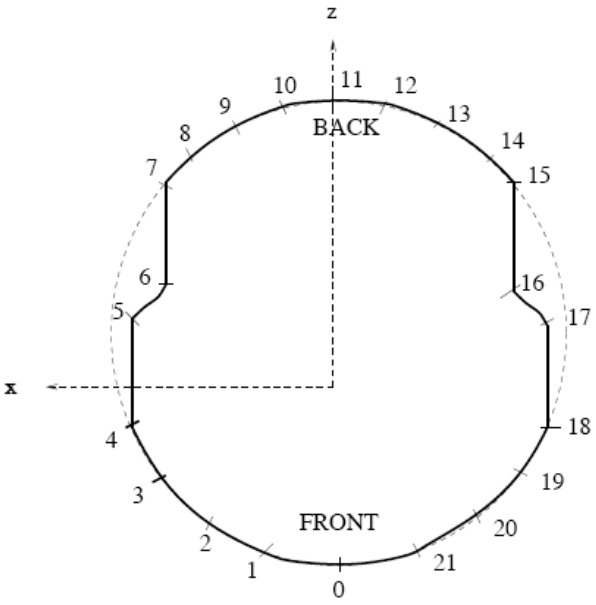
στο σχήμα 4.20. Ένα DEFSONARTransform περιέχει έναν Cylinder Node σε έναν Shap node με μια περιστροφή γύρω από τον άξονα Z. Αυτό το DEFSONAR Transform πρέπει να το προσαρμόσουμε σε rotate αλλά και σε translate με του αισθητήρες ds0, ds1, κ.λπ.
Κάθε sonar διαμορφώνεται ως ένα DistanceSensor node, στον οποίο μπορεί να κάνουμε μια περιστροφή γύρω από τον άξονας Y, ένα Translation, και ένα USESONAR Transform, με ένα όνομα (ds0, ds1,…) για να χρησιμοποιηθεί από τον ελεγκτή.
Για να ολοκληρώσουμε το ρομπότ pioneer 2TM, θα πρέπει να συμπληρώσουμε τους υπόλοιπους τομείς του κόμβου DifferentialWheels όπως φαίνεται στο σχήμα 4.21.



Coordinates of the crossSection field of the extrusion node:
0: x=–0.1, z=0.215
1: x=0.1, z=0.215
2: x=0.135, z=0.185
3: x=0.135, z=–0.095
4: x=0.08, z=–0.11
5: x=–0.08, z=–0.11
6: x=–0.135, z=–0.095
7: x=–0.135, z=0.185

0.059 < y <0.234

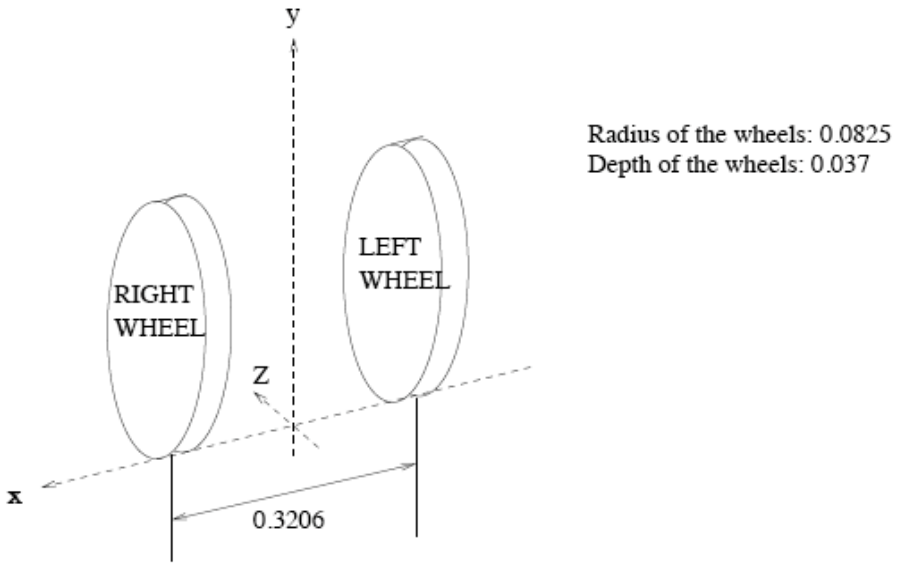
Εικόνα 4.14 Σώμα του Pioneer 2TM robot



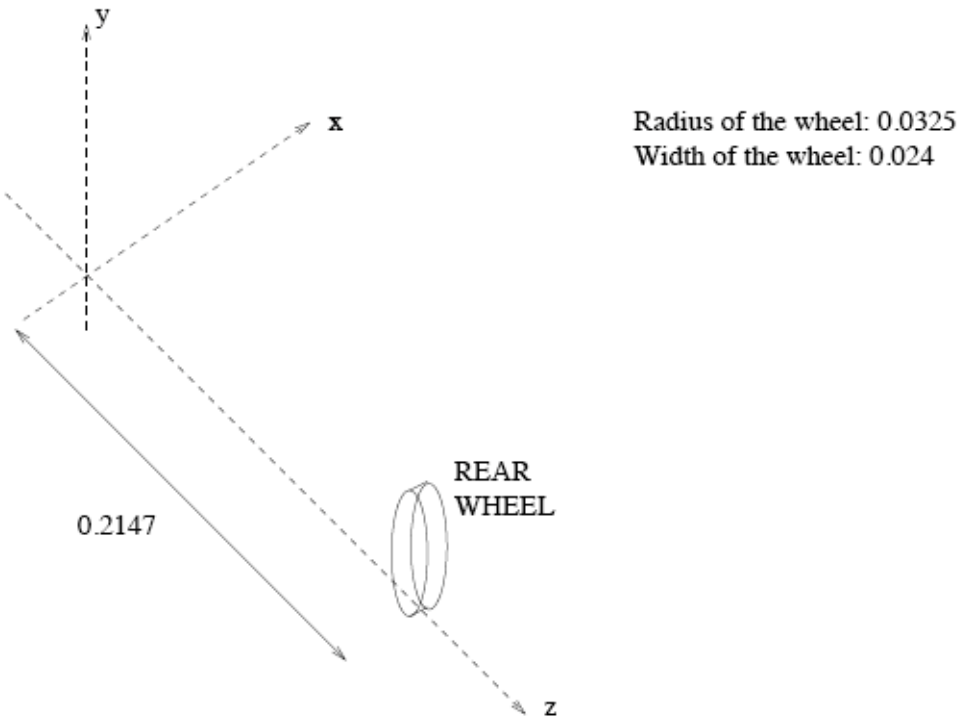
Coordinates of the crossSection field of the Extrusion node:
0: x=0 z=–0.174
1: x=–0.056 z=–0.166
2: x=–0.107 z=–0.145
3: x=–0.155 z=–0.112
4: x=–0.190 z=–0.064
5: x=–0.190 z=0.074
6: x=–0.160 z=0.096
7: x=–0.160 z=0.151
8: x=–0.155 z=0.2
9: x=–0.107 z=0.236
10: x=–0.056 z=0.256
11: x=0 z=0.264
12: x=0.056 z=0.256
13: x=0.107 z=0.236
14: x=0.155 z=0.2
15: x=0.160 z=0.151
16: x=0.160 z=0.096
17: x=0.190 z=0.074
18: x=0.190 z=–0.064
19: x=0.155 z=–0.112
20: x=0.107 z=–0.145
21: x=0.056 z=–0.166

0.234 < y < 0.24

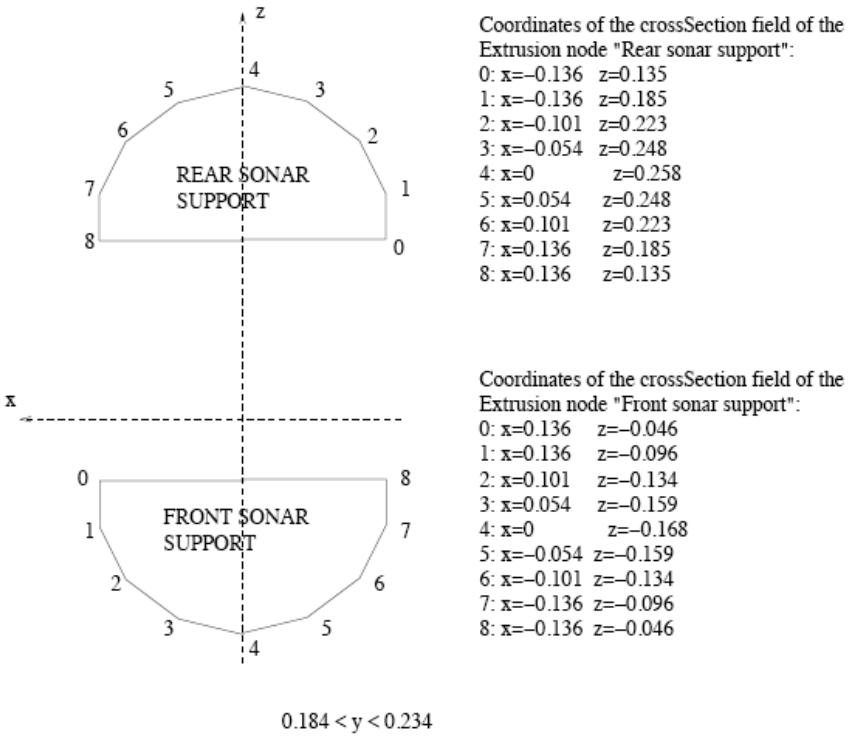
Εικόνα 4.15 Κορυφή του Pioneer 2TM robot



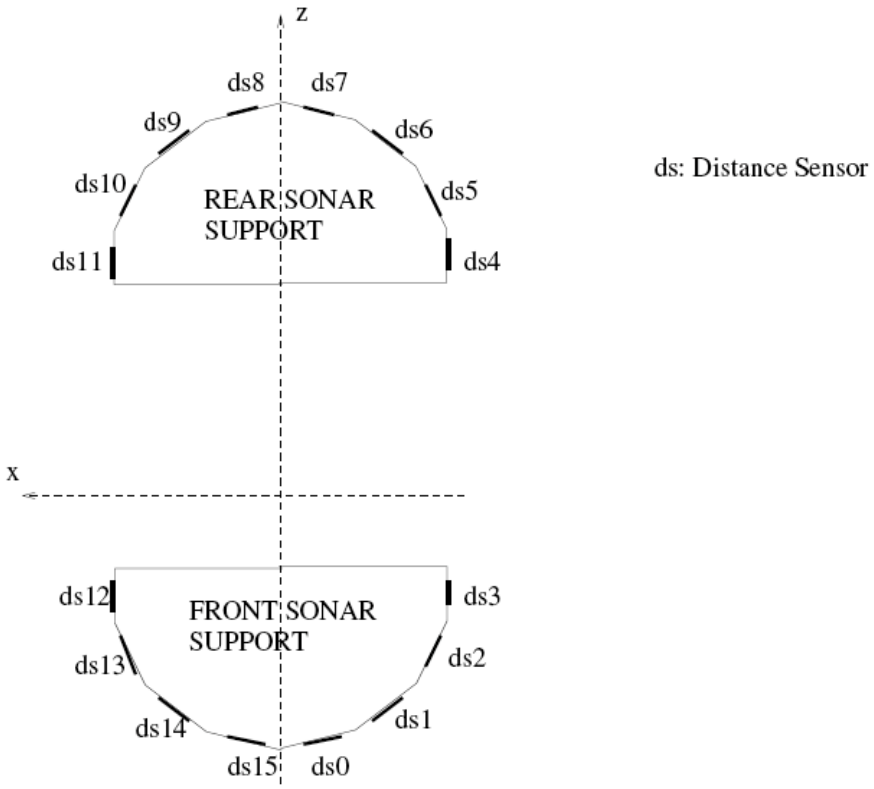
Εικόνα 4.16 Ρόδες του Pioneer 2TM robot



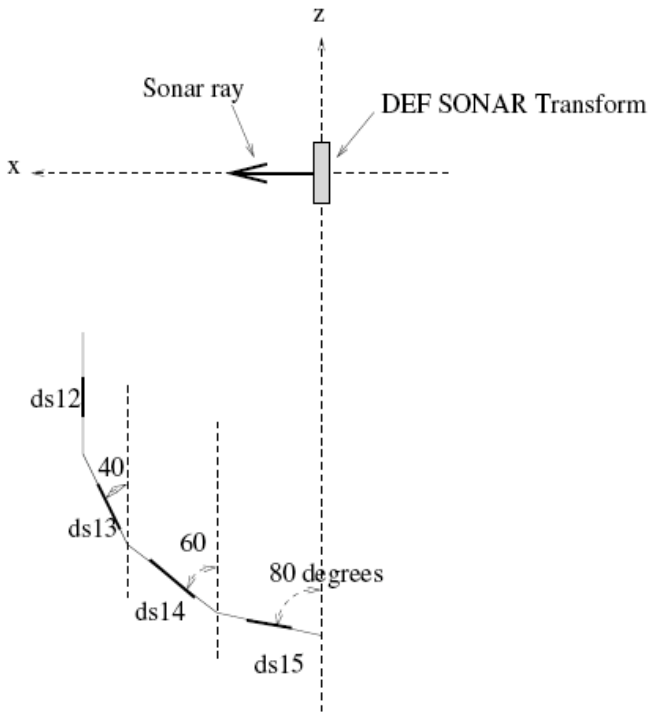
Εικόνα 4.17 Πίσω ρόδες του Pioneer 2TM robot



Εικόνα 4.18 Το υποστηριζόμενο sonar του Pioneer 2TM robot



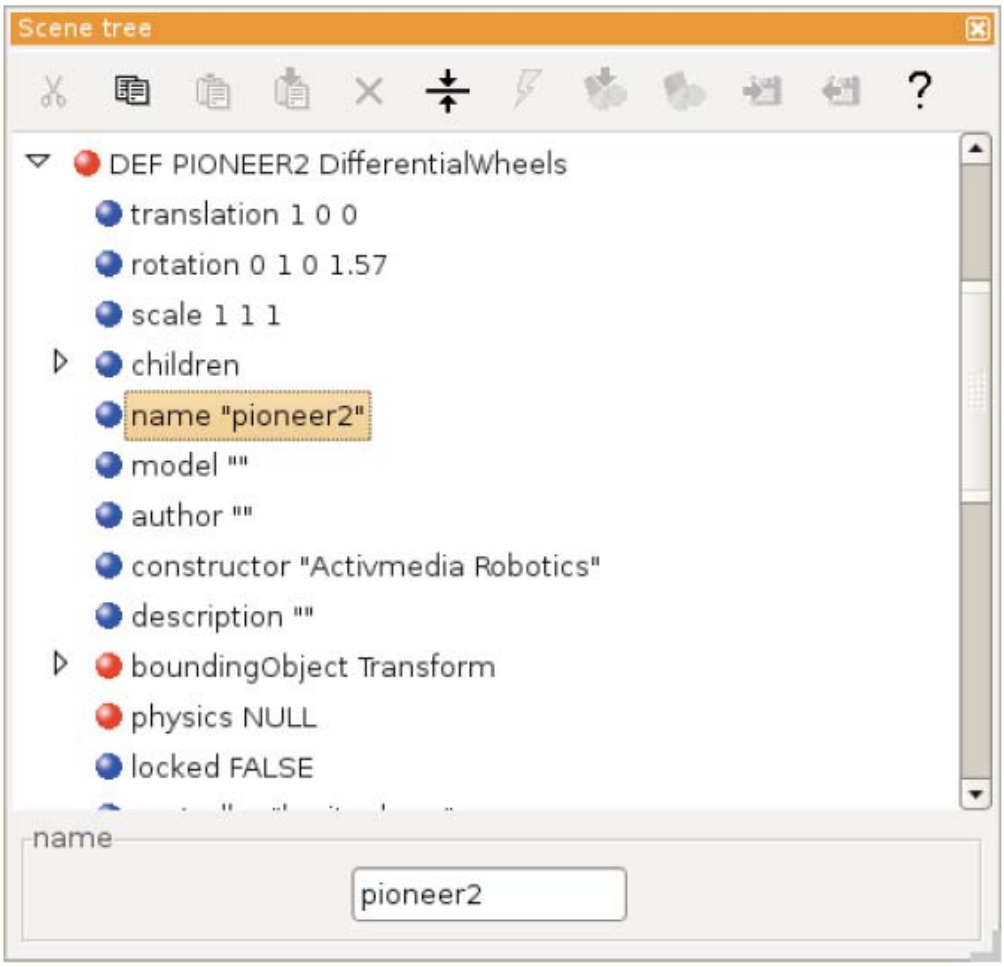
Εικόνα 4.29 Τοποθεσίες του sonar του Pioneer 2TM robot



Εικόνα 4.20 Μοίρες ανάμεσα στους αισθητήρες sonar του Pioneer 2TM robot

Sonar name	translation	rotation
ds0	-0.027 0.209 -0.164	0 1 0 1.745
ds1	-0.077 0.209 -0.147	0 1 0 2.094
ds2	-0.118 0.209 -0.11	0 1 0 2.443
ds3	-0.136 0.209 -0.071	0 1 0 3.14
ds4	-0.136 0.209 0.160	0 1 0 -3.14
ds5	-0.118 0.209 0.205	0 1 0 -2.443
ds6	-0.077 0.209 0.236	0 1 0 -2.094
ds7	-0.027 0.209 0.253	0 1 0 -1.745
ds8	0.027 0.209 0.253	0 1 0 -1.396
ds9	0.077 0.209 0.236	0 1 0 -1.047
ds10	0.118 0.209 0.205	0 1 0 -0.698
ds11	0.136 0.209 0.160	0 1 0 0
ds12	0.136 0.209 -0.071	0 1 0 0
ds13	0.118 0.209 -0.116	0 1 0 0.698
ds14	0.077 0.209 -0.147	0 1 0 1.047
ds15	0.027 0.209 -0.164	0 1 0 1.396

Εικόνα 4.1: Translation και rotation του Pioneer 2TM DEF SONARTransforms



Εικόνα 4.21: Μερικοί κόμβοι από το Pioneer 2TMDifferentialWheels

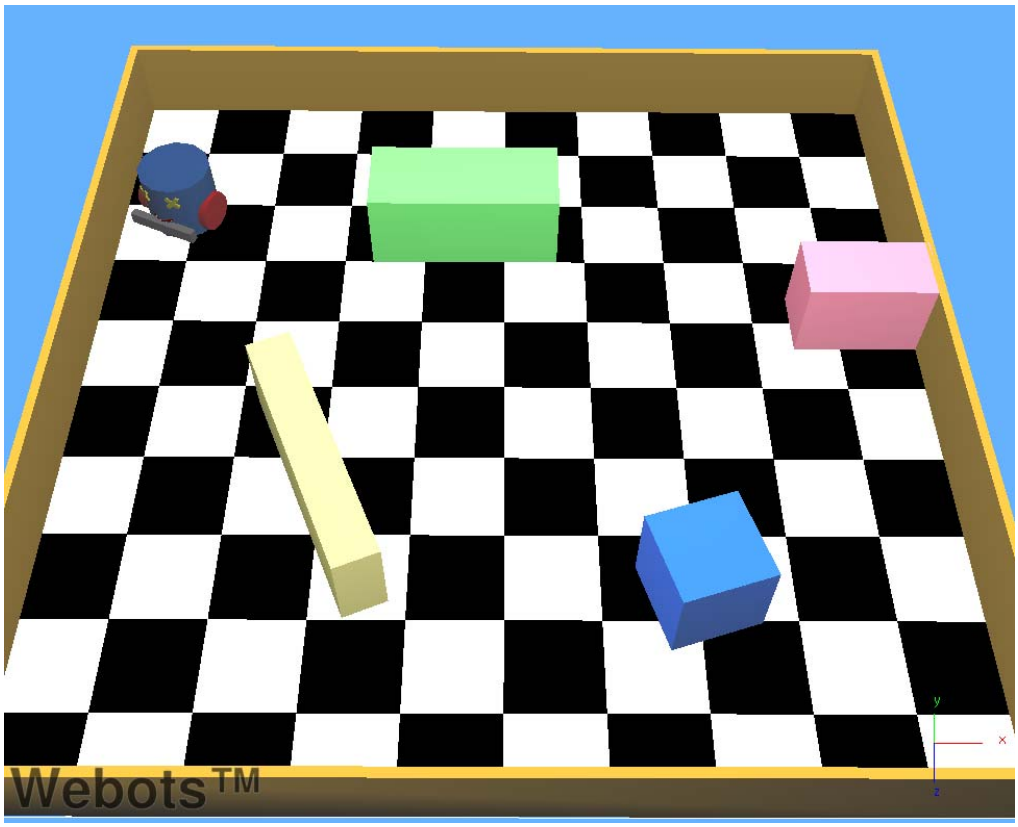
6.2 ΔΙΑΜΟΡΦΩΣΗ ΕΝΟΣ ΥΠΑΡΧΟΝΤΟΣ ΡΟΜΠΟΤ: PIONEER2.WBT 135 ελεγκτής 4.4.3

Ο ελεγκτής του ρομπότ PIONEER 2TM είναι αρκετά σύνθετος. Χρησιμοποιεί έναν ελεγκτή Braitenberg για να αποφύγει τα εμπόδια χρησιμοποιώντας τους αισθητήρες του. Ένα ενεργό Matrix έχει καθοριστεί για να υπολογίζει τις εντολές και τα λάθη των μηχανών από τις μετρήσεις των αισθητήρων. Εντούτοις, δεδομένου ότι η δομή του pionner 2TM δεν είναι κυκλική, χρησιμοποιούνται μερικά τεχνάσματα, όπως η δυνατότητα του ρομπότ να πάει προς τα πίσω προκειμένου να περιστραφεί ακίνδυνα για την αποφυγή των εμποδίων. Ο source code αυτού του ελεγκτή είναι ένα καλό παράδειγμα προγραμματισμού. Το όνομα αυτού του ελεγκτή είναι pionner2.

Κεφάλαιο 7

7.1 Bumper.wbt

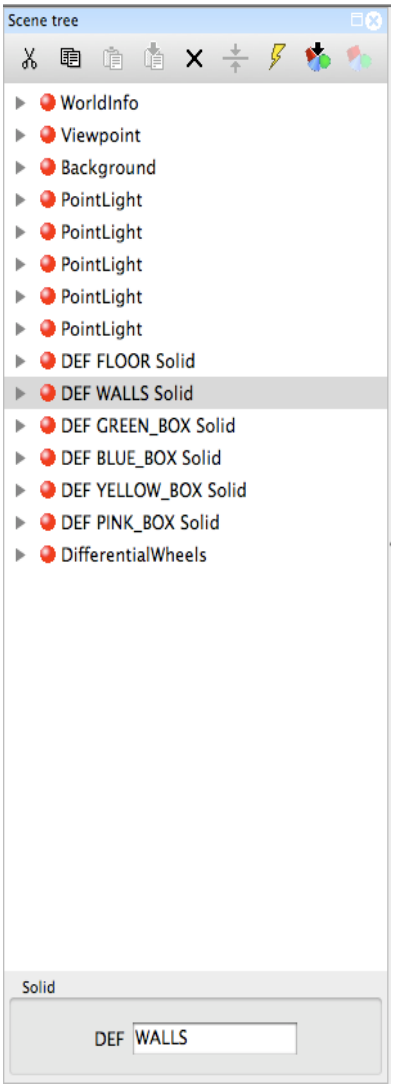
Έννοιες που θα χρησιμοποιηθούν σε αυτό το κεφάλαιο είναι TouchSensor, bumper, DifferentialWheels.

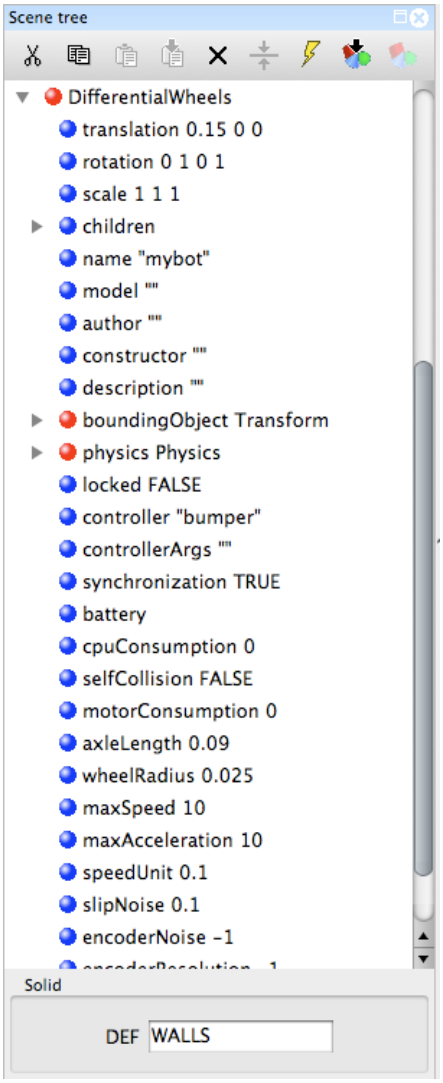


Εικόνα του bumper.wbt

Σε αυτό το μάθημα έχουμε το ρομπότ μας στο οποίο έχουμε προσθέσει ένα Bumper (Προφυλακτήρας) στο οποίο έχουμε εγκαταστήσει και ένα αισθητήρα TouchSensor. Έτσι το ρομπότ μας κινείται στο χώρο και όταν αυτό προσκρούσει σε ένα εμπόδιο (είτε αυτό είναι αντικείμενο, είτε είναι τείχος) τότε το ρομπότ κινείται με την όπισθεν, στρίβει και συνεχίζει να κινείται.

Από τα προηγούμενα μαθήματα ανοίγουμε το γνωστό πια κόσμο και αρχίζουμε να δουλεύουμε στο scene tree όπου κάνουμε διάφορες ρυθμίσεις.
Στις παρακάτω εικόνες βλέπουμε το scene tree windows με τις διάφορες ρυθμίσεις του και όχι μόνο.





Στη συνέχεια θα ασχοληθούμε με τον κώδικα του bumper.wbt.
Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες που είναι :

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/touch_sensor.h>
```

Και τις εξωτερικές μεταβλητές

```
#define SPEED 40
#define TIME_STEP 64
```

Όπου από τα ονόματα τους καταλαβαίνουμε και που αναφέρονται.
Με το παρακάτω κομμάτι κώδικα εισάγουμε τον Bumper στο ρομπότ μας

```
bumper = wb_robot_get_device("bumper");
  wb_touch_sensor_enable(bumper, TIME_STEP);
```

καθώς του βάζουμε και έναν αισθητήρα .
Όταν ο αισθητήρας μας εντοπίσει ένα αντικείμενο ή τείχος τότε ενεργοποιείται η

κίνηση αποφυγής του αντικειμένου. Αυτό το κατανοούμε καλύτερα βλέποντας το κομμάτι κώδικα :

```
if (wb_touch_sensor_get_value(bumper) > 0) {
   movement_counter = 15;
```

Εδώ χρησιμοποιούμε τον movement_counter για να κοντρολάρουμε καλύτερα τις κινήσεις του ρομπότ. Οι τιμές που δίνουμε είναι 0 που σημαίνει ότι το ρομπότ μας κινείτε ευθεία. Όταν η τιμή αυτή αλλάξει σημαίνει ότι ο αισθητήρας του ρομπότ έχει εντοπίσει ένα εμπόδιο και αρχίζει την αποφυγή του. Για την αποφυγή αυτή το ρομπότ μας κινείτε αρχικά προς τα πίσω κάνοντας ένα μικρό κύκλο και μετά συνεχίζει να κινείτε ευθεία.

Το κομμάτι του κώδικα όπου αναφέρονται τα παραπάνω :

```
if (movement_counter == 0) {
   left_speed = SPEED;
   right_speed = SPEED;

} else if (movement_counter >= 7) {
   left_speed = -SPEED;
   right_speed = -SPEED;
   movement_counter--;
} else {
   left_speed = -SPEED / 2;
   right_speed = SPEED;
   movement_counter--;
}
```

Τέλος ο τρόπος που δίνουμε κίνηση στα differential_wheels (στις ρόδες δλδ) φαίνεται εδώ.

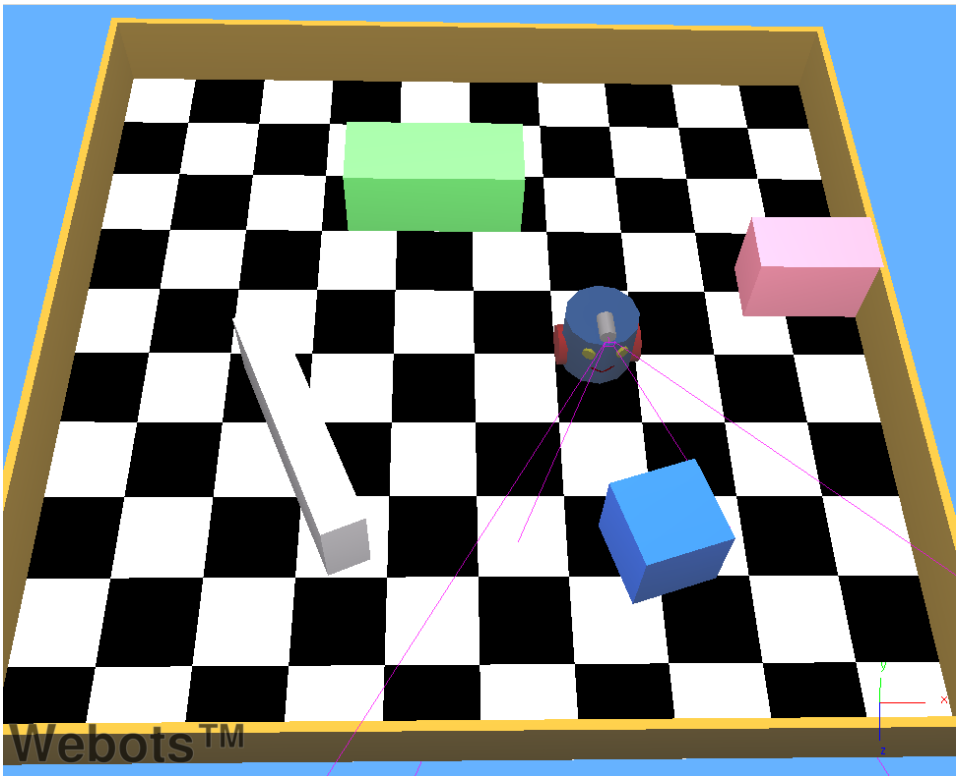
```
wb_differential_wheels_set_speed(left_speed, right_speed);


wb_robot_cleanup();

return 0;
```

Ο κώδικας του κεφαλαίου bumper.wbt φαίνετε ολοκληρωμένος στο παράρτημα.

Κεφάλαιο 8

8.1 Camera.wbt

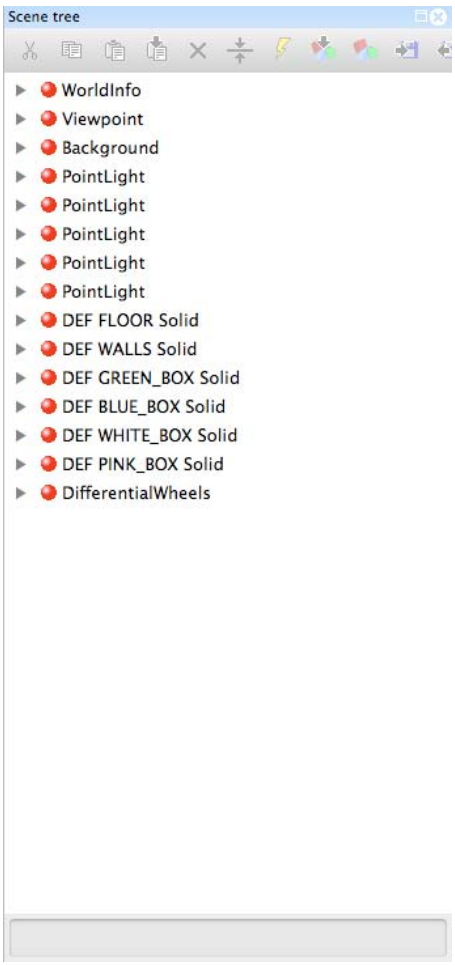
Έννοιες μου θα χρησιμοποιήσουμε σε αυτό το κεφάλαιο είναι οι εξής : Camera, image processing.

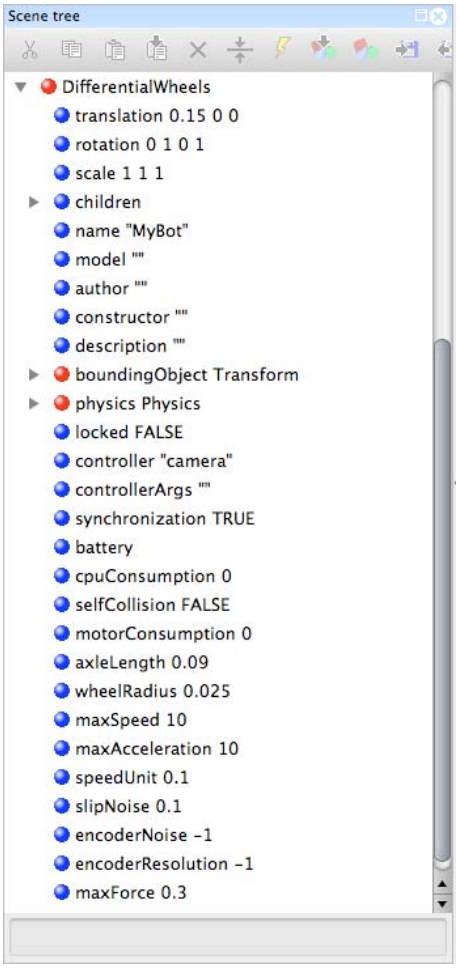


Εικόνα του ρομπότ με Camera

Σε αυτό το παράδειγμα το ρομπότ μας χρησιμοποιεί μια κάμερα για να μπορεί να ελέγχει τα χρωματιστά αντικείμενα του χώρου μας. Το ρομπότ μας αναλύει τα επίπεδα του RGB color σε κάθε pixel της εικόνας της κάμερας. Όταν εντοπίσει κάτι γυρίζει και σταματά για μερικά δευτερόλεπτα. Επίσης μας εμφανίζει ένα μήνυμα στην κονσόλα επεξηγώντας το χρώμα του αντικειμένου που εντόπισε. Μπορείτε να μετακινήσετε το ρομπότ σε μια διαφορετική θέση στο χώρο χρησιμοποιώντας το mouse προκειμένου να δείτε τι εντοπίζει το ρομπότ μας.

Από τα προηγούμενα μαθήματα ανοίγουμε το γνωστό πια κόσμο και αρχίζουμε να δουλεύουμε στο scene tree όπου κάνουμε διάφορες ρυθμίσεις.
Στις παρακάτω εικόνες βλέπουμε το scene tree windows με τις διάφορες ρυθμίσεις του και όχι μόνο.





Στη συνέχεια θα ασχοληθούμε με τον κώδικα του camera.wbt.
Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες που είναι :

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/camera.h>
```

Καθώς θέτουμε και τις εξωτερικές μεταβλητές για να μας διευκολύνουν στο πρόγραμμα μας.

```
#define SPEED 40
#define TIME_STEP 64
#define COLOR_BLOB_VALUE 4500000
#define GREY_BLOB_VALUE 4350000
```

Όπου από τα ονόματα τους καταλαβαίνουμε και που αναφέρονται αυτές οι μεταβλητές.

Πρώτα φτιάχνουμε την συσκευή της κάμερας καθώς και τις απαραίτητες ρυθμίσεις της, έπειτα την ανοίγουμε και την τοποθετούμε. Αποθηκεύουμε επίσης το ύψος και το πλάτος της για την περαιτέρω χρήση.
Αυτό μπορούμε καλύτερα να το δούμε στο παρακάτω κομμάτι κώδικα :

```
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, TIME_STEP);
wb_camera_move_window(camera, 0, 0);
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
```

Αφού τοποθετήσουμε την κάμερα χρησιμοποιούμε μια While όπου της δίνουμε μια τιμή για να γίνετε η απαραίτητη ανανέωση (refresh) της εικόνας της κάμερας. Στη συνέχεια το ρομπότ μας παίρνει μια εικόνα και αφού την λάβει σταματά.

```
while(wb_robot_step(TIME_STEP)!=-1) {
  image = wb_camera_get_image(camera);

  left_speed = 0;
  right_speed = 0;
```

Στην συνέχεια το ρομπότ μας σταματά για μερικά δευτερόλεπτα και αρχικοποιούμε τις τιμές των χρωμάτων.

```
  if (pause_counter == 0) {
    red = 0;
    green = 0;
    blue = 0;
```

Στο παρακάτω κομμάτι κώδικα αναλύουμε την εικόνα από την κάμερα. Ο στόχος μας είναι να ανιχνεύσει ένα σημείο χρώματος ενός καθορισμένου χρώματος. Για να το επιτύχουμε αυτό αναλύουμε την εικόνα pixel ανά pixel και ανάλογα με το χρώμα που ανιχνεύουμε αθροίζουμε την τιμή του στο αντίστοιχο χρώμα που ανιχνεύτηκε. Στη συνέχεια σταθμίζουμε την τιμή αυτή από την απόσταση του τρέχοντος pixel με το κέντρο της εικόνας έτσι ώστε να μπορούμε να πάρουμε το καλύτερο αποτέλεσμα.

```
    for (i = 0; i < width; i++)
{
     centering_weight = (i < (width / 2)) ? i - 10 : (width - i - 10);
```

Να σημειωθεί ότι διαβάζουμε το χαμηλό κομμάτι της εικόνας μιας και θέλουμε να αποφύγουμε επιρροές του χρώματος από τον ουρανό

```
     for (j = (height / 2); j < height; j++)
{
       red += wb_camera_image_get_red(image, width, i, j) * centering_weight;

       blue += wb_camera_image_get_blue(image, width, i, j) * centering_weight;

       green += wb_camera_image_get_green(image, width, i, j) * centering_weight;
     }
    }
```

Αφού τελειώσουμε με την ανίχνευση του χρώματος μπορούμε να υπολογίσουμε τα τελικά αποτελέσματα συγκρίνοντας το τελικό αποτέλεσμα με το κατώτατο όριο μιας σταγόνας

```
    if (blue >= COLOR_BLOB_VALUE) {
      printf("Looks like I found a blue blob !\n");
      pause_counter = 20;
    }

 else if (green >= COLOR_BLOB_VALUE)
 {
       printf("Looks like I found a green blob !\n");
       pause_counter = 20;
       wb_camera_save_image(camera,"green_blob.png",100);
// με το παραπάνω παίρνει ένα στιγμιότυπο της εικόνας
     }

else if (red >= COLOR_BLOB_VALUE)
{
       printf("Looks like I found a red blob !\n");
       pause_counter = 20;
       wb_camera_save_image(camera,"red_blob.jpg",100);
//  με το παραπάνω παίρνει ένα στιγμιότυπο της εικόνας

     }

else if (blue >= GREY_BLOB_VALUE && green >= GREY_BLOB_VALUE
              && red >= GREY_BLOB_VALUE) {
       printf("Looks like I found a grey blob !\n");
       pause_counter = 20;
     }

 Else
 {
       left_speed = -SPEED;
       right_speed = SPEED;
     }
   }
else
{
     pause_counter--;
```

Να σημειωθεί πως το ρομπότ θα πρέπει να κινηθεί προτού αρχίσουμε να αναλύσουμε την εικόνα, αυτό συμβαίνει γιατί αν δεν το κάνουμε θα έχουμε ακριβώς την ιδία εικόνα με αυτή που θα έχει ήδη ανιχνευτεί

```
    if (pause_counter <= 3) {
      left_speed = -SPEED;
      right_speed = SPEED;
    }
  }
```

Τέλος στο παρακάτω κομμάτι κώδικα φαίνεται η κίνηση που δίνετε στις ροδές του ρομπότ.

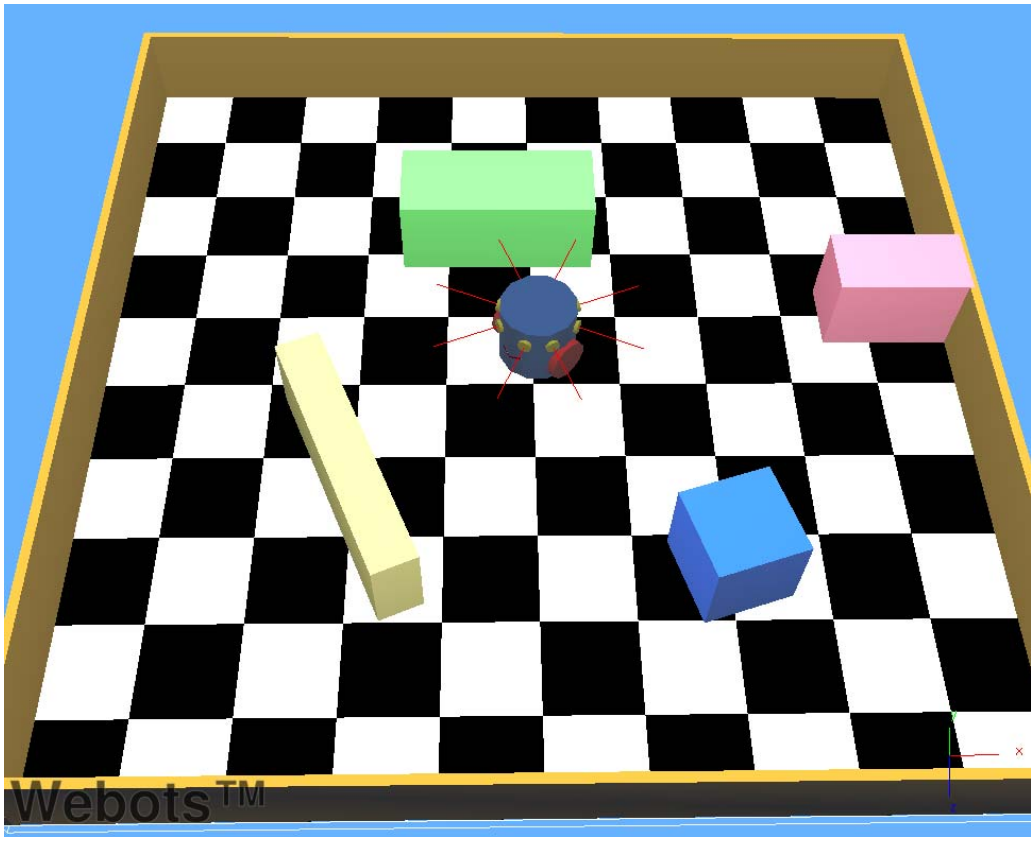
```
    wb_differential_wheels_set_speed(left_speed, right_speed);
  }

  wb_robot_cleanup();

  return 0;
}
```

Ο κώδικας του κεφαλαίου camera.wbt φαίνετε ολοκληρωμένος στο παράρτημα.

Κεφάλαιο 9

9.1 Distance sensor.wbt

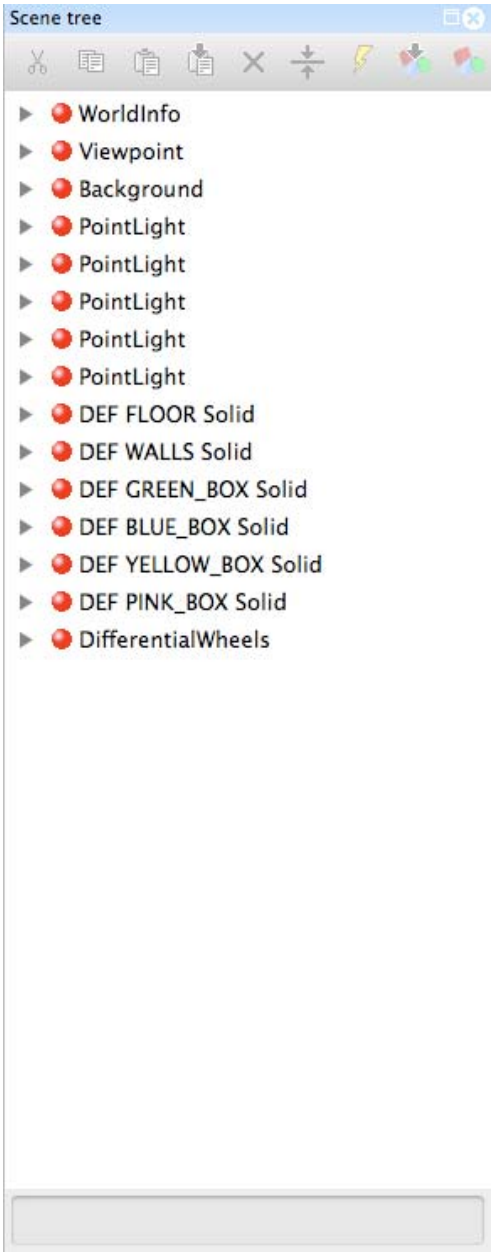
Έννοιες που θα χρησιμοποιήσουμε σε αυτό το κεφάλαιο είναι οι DistanceSensor, Braitenberg, DifferentialWheels.

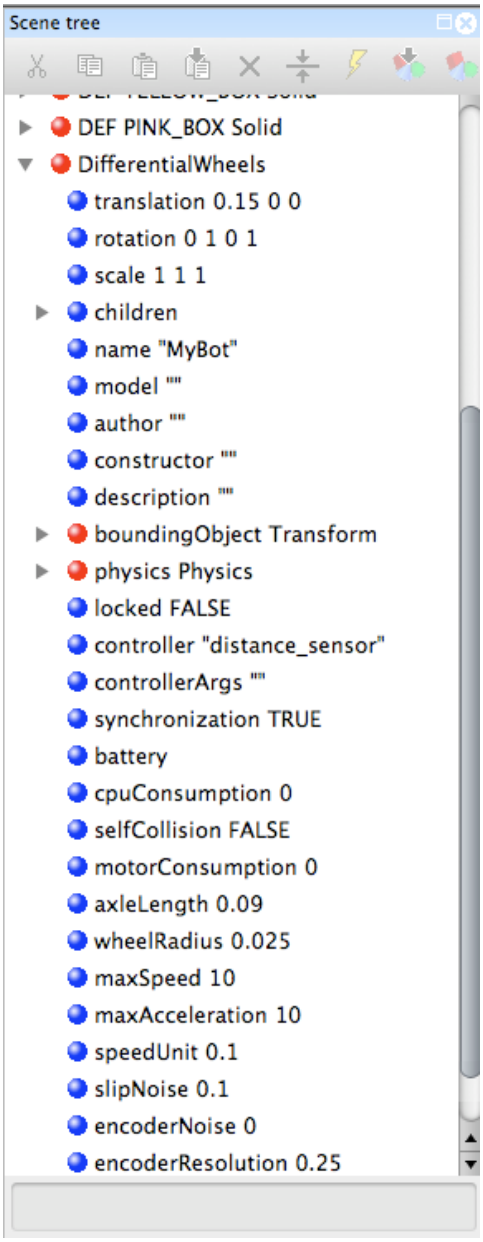


Εικόνα του ρομπότ μας distance sensor.wbt

Στο κεφάλαιο αυτό θα τοποθετήσουμε στο ρομπότ μας 8 αισθητήρες (Distance Sensors) γύρω από το σώμα του. Έτσι το ρομπότ μας κινείτε μέσα στον ήδη γνωστό μας κόσμο και όταν αυτοί οι αισθητήρες ανιχνεύσουν κάποιο από τα εμπόδια ή τοίχο το ρομπότ αποφεύγει τα εμπόδια χρησιμοποιώντας την τεχνική Braitenberg technique. Για την καλύτερη κατανόηση, αυτή η τεχνική θα φανεί καλύτερα μετά την υλοποίηση του μαθήματος.

Από τα προηγούμενα μαθήματα ανοίγουμε το γνωστό πια κόσμο και αρχίζουμε να δουλεύουμε στο scene tree όπου κάνουμε διάφορες ρυθμίσεις.
Στις παρακάτω εικόνες βλέπουμε το scene tree windows με τις διάφορες ρυθμίσεις του και όχι μόνο.





Στη συνέχεια θα ασχοληθούμε με τον κώδικα του distance sensor.wbt.
Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες που είναι :

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

Καθώς θέτουμε και τις εξωτερικές μεταβλητές για να μας διευκολύνουν στο πρόγραμμα μας.

```
#define NB_SENSORS 8
#define TIME_STEP  64
#define RANGE (1024 / 2)
```

```
 wb_robot_init();
```

Εδώ θα δημιουργήσουμε τους χειριστές (handlers) και θα ενεργοποιήσουμε τους αισθητήρες απόστασης

```
 for(i = 0; i < NB_SENSORS; i++)
{
   ps[i] = wb_robot_get_device(name);
```

Εδώ δημιουργούμε ένα handler στον αισθητήρα

Και παρακάτω κάνουμε μετρήσεις απόστασης και χιλιοστό του δευτερολέπτου ή TIME_STEP όπως βλέπουμε στον κώδικα μας.

```
   wb_distance_sensor_enable(ps[i], TIME_STEP);
```

Στη συνέχεια με τη γραμμή κώδικα name[2]++; Αυξάνουμε το όνομα των συσκευών σε "ir1", "ir2", κτλπ.

```
 }
```

Παρακάτω δημιουργούμε ένα βρόγχο έλεγχου

```
 while (wb_robot_step(TIME_STEP)!=-1)

   {

for (i = 0; i < NB_SENSORS; i++) {

     sensor_value[i] = wb_distance_sensor_get_value(ps[i]);

    }

   for (i = 0; i < 2; i++) {

     speed[i] = 0;

     for (j = 0; j < NB_SENSORS; j++) {
```

Έπειτα θα πρέπει να επαναπροσδιορίσουμε την τιμή του αισθητήρα για να είμαστε σε θέση να πάρουμε και τις αρνητικές του τιμές επίσης. Με αυτό θα μπορεί να γίνει κίνηση του ρομπότ και προς τα πίσω.

```
       speed[i] += matrix[i][j] * (1 - (sensor_value[j] / RANGE));
     }
   }
```

Τέλος η κίνηση του στις ροδές του ρομπότ δίνετε παρακάτω

```
   wb_differential_wheels_set_speed(2 * speed[0], 2 * speed[1]);
 }

 wb_robot_cleanup();
```

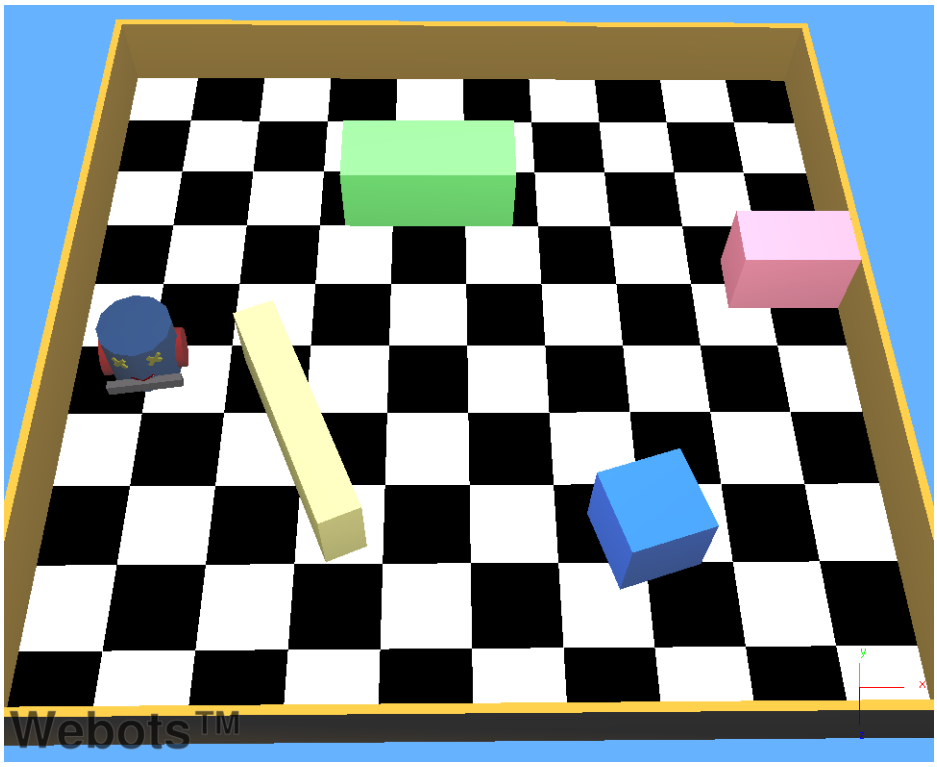
```
  return 0;
}
```

Ο κώδικας του κεφαλαίου distance sensor.wbt φαίνετε ολοκληρωμένος στο παράρτημα.

Κεφάλαιο 10

10.1 Force sensor.wbt

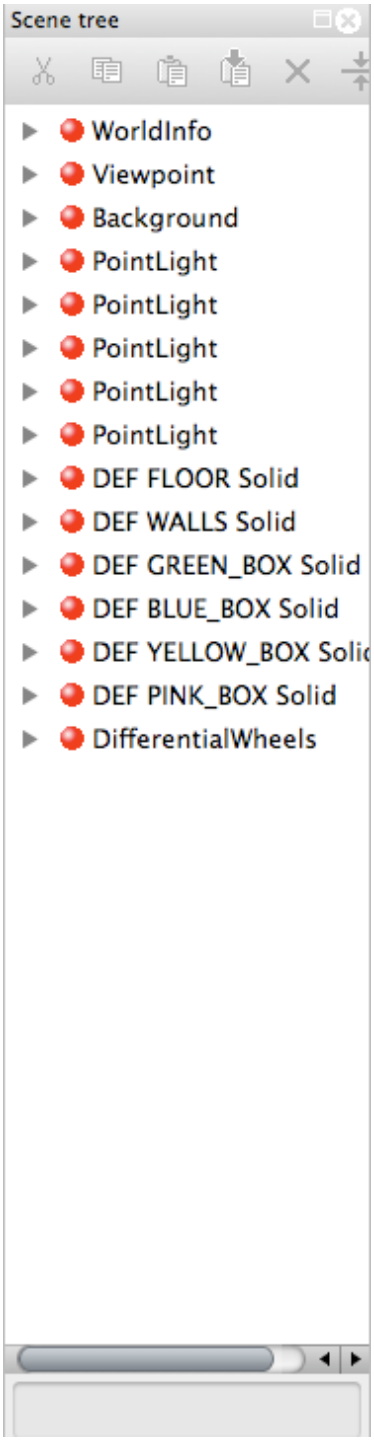
Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε τις έννοιες Force, TouchSensor, DifferentialWheels.

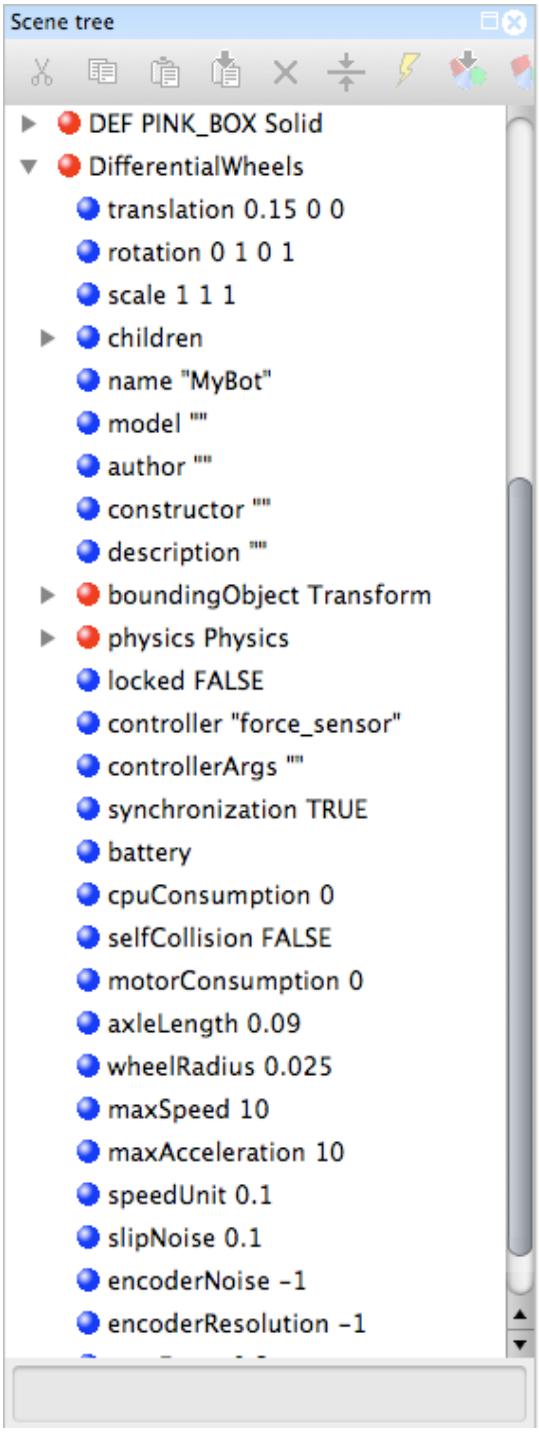


Εικόνα του force sensor.wbt

Αυτό το μάθημα μοιάζει πολύ με το προηγούμενο (δείτε το παραπάνω σχήμα) με την διαφορά ότι σε αυτή την περίπτωση έχουμε εγκαταστήσει στο ρομπότ μας ένα ”force” TouchSensor αντί για ένα ”bumper” που χρησιμοποιήσαμε στο προηγούμενο μάθημα. Έτσι το ρομπότ μας κινείτε στο γνωστό μας χώρο και όταν συγκρουστεί με αντικείμενα του χώρου ή τον τοίχο, αυτό κινείτε με όπισθεν και συνεχίζει την κίνηση του. Το σημαντικό εδώ είναι ότι κατά την σύγκρουση του το ρομπότ λόγο του ”force” TouchSensor μπορεί να υπολογίσει την δύναμη της κάθε σύγκρουσης και να μας το εμφανίσει στο Console window.

Από τα προηγούμενα μαθήματα ανοίγουμε το γνωστό πια κόσμο και αρχίζουμε να δουλεύουμε στο scene tree όπου κάνουμε διάφορες ρυθμίσεις.
Στις παρακάτω εικόνες βλέπουμε το scene tree windows με τις διάφορες ρυθμίσεις του και όχι μόνο.





Στη συνέχεια θα ασχοληθούμε με τον κώδικα του force sensor.wbt.
Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες που είναι :

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/touch_sensor.h>
```

Καθώς θέτουμε και τις εξωτερικές μεταβλητές για να μας διευκολύνουν στο πρόγραμμα μας.

```
#define SPEED 40
#define TIME_STEP 64
```

Στο κύριος μέρος του προγράμματος μας αφού δηλώσουμε τις απαραίτητες μεταβλητές φτιάχνουμε έναν ‘χειριστή’ (handler) στον αισθητήρα και στη συνέχεια τον ενεργοποιούμε.

```
 force = wb_robot_get_device("force");
 wb_touch_sensor_enable(force, TIME_STEP);

 while(wb_robot_step(TIME_STEP)!=-1) {
```

Αφού ενεργοποιήσουμε τον αισθητήρα φτιάχνουμε την κίνηση του ρομπότ. Έτσι μόλις ο αισθητήρας ανιχνεύσει κάτι αρχίζει η κίνηση αποφυγής των αντικειμένων το οποίο σε κώδικα φαίνετε παρακάτω.

```
force_value = wb_touch_sensor_get_value(force);

   if (force_value > 0.01) {
     printf("Detecting a collision of %g N\n", force_value);
     movement_counter = 15;
   }
```

Στη συνέχεια ενεργοποιούμε το movement_counter για να διαχειριστούμε τις κινήσεις του ρομπότ. Έτσι όταν η τιμή είναι 0 τότε το ρομπότ μας κινείτε ευθεία. Όταν όμως το ρομπότ μας έχει άλλη τιμή εκτός από το 0 τότε σημαίνει ότι αποφεύγει ένα εμπόδιο. Για την αποφυγεί κινείτε το ρομπότ αρχικά πίσω κάνοντας ένα κύκλο και στην συνέχεια αρχίζει πάλι την κίνηση του.
Παρακάτω βλέπουμε την περιγραφή μας σε κώδικα :

```
     if (movement_counter == 0)
{
        left_speed = SPEED;
        right_speed = SPEED;

   }
     else if (movement_counter >= 7)
  {
        left_speed = -SPEED;
        right_speed = -SPEED;
        movement_counter--;
   }
      Else
  {
        left_speed = -SPEED / 2;
        right_speed = SPEED;
        movement_counter--;
   }
```

Τέλος ενεργοποιούμε τα differential_wheels

```
   wb_differential_wheels_set_speed(left_speed, right_speed);
  }

 wb_robot_cleanup();

  return 0;
}
```

Ο κώδικας του κεφαλαίου Force sensor.wbt φαίνετε ολοκληρωμένος στο παράρτημα.

Κεφάλαιο 11

11.1 Εξάποδα Ρομπότ

Πριν όμως προχωρήσουμε στην υλοποίηση του hexapod.wbt ας κάνουμε μια μικρή αναφορά για τα ‘εξάποδα’ ρομπότ.

Ένα εξάποδο ρομπότ είναι ένα μηχανικό όχημα το οποίο μπορεί να περπατήσει με έξι πόδια. Δεδομένου ότι ένα ρομπότ μπορεί να είναι στατικά σταθερό με τρία ή με περισσότερα πόδια, ένα εξάποδο ρομπότ έχει καλύτερη ευελιξία στο πώς μπορεί να κινηθεί. Επίσης εάν μερικά από τα πόδια του δεν βρίσκονται σε λειτουργία τότε το ρομπότ μπορεί ακόμα να κινηθεί. Παράλληλα δεν είναι απαραίτητο να βρίσκονται σε λειτουργία όλα τα πόδια του για να μπορεί να σταθεί. Παρακάτω μπορούμε να δούμε ένα εξάποδο ρομπότ

Σχήμα 11.1 Εξάποδο ρομπότ

11.2 Μετακίνηση

Η κίνηση των εξάποδων ρομπότ γίνεται από τους ελεγχόμενους βηματισμούς των ποδιών που του επιτρέπουν να προχωρήση μπροστά, να γυρίσει η ακόμα να προχώρηση και στα πλάγια.

Αυτός ο βηματισμός μπορεί να επιτευχτεί ως εξής :

- Εναλλασσόμενη κίνηση των τριών ποδιών.
- Τετράποδος.
- Ένα πόδι τη φορά.

Οι βηματισμοί για τα hexapods είναι χαρακτηριστικά σταθεροί, ακόμη και στην ελαφρώς δύσκολη και ανώμαλη έκταση.

11.3 Βιολογικά Εμπνευσμένο

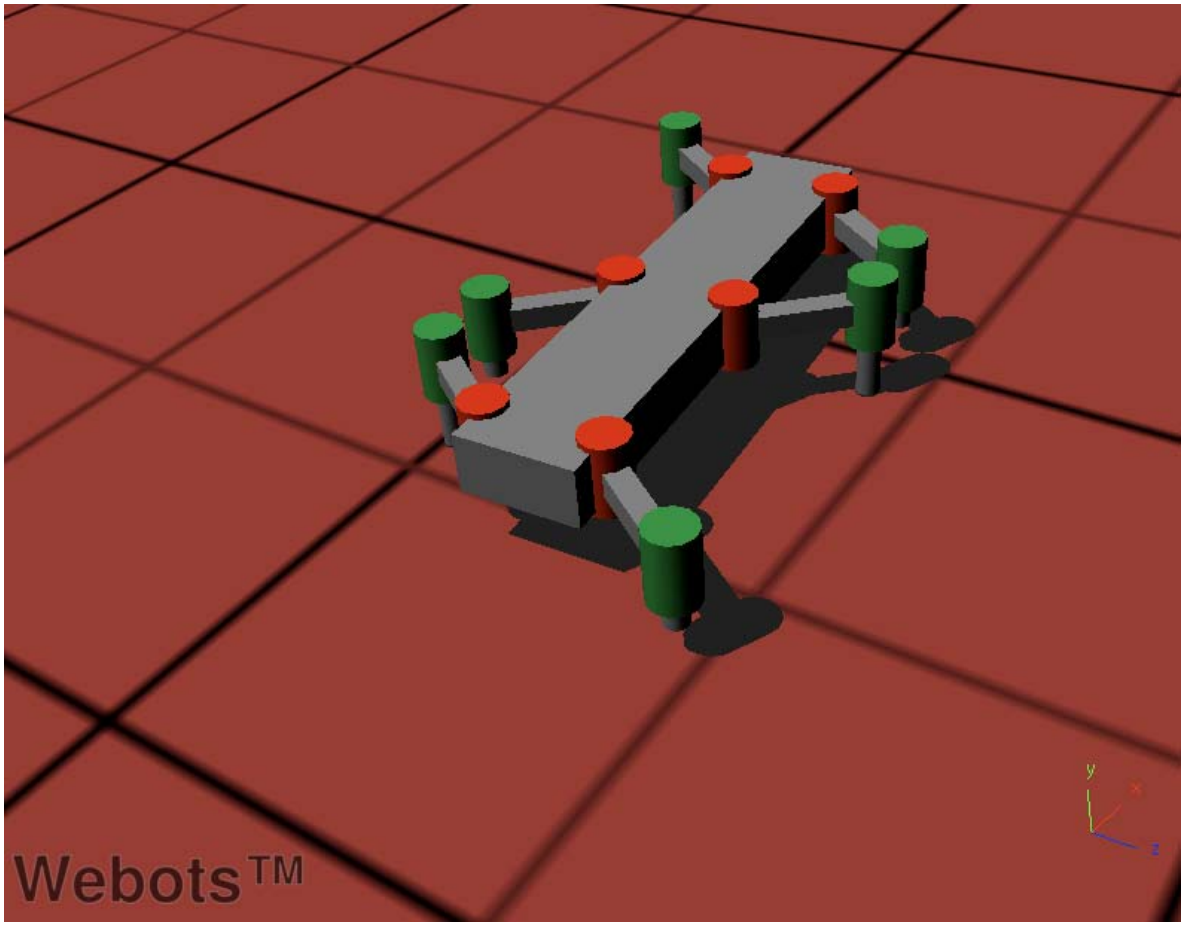
Τα έντομα επιλέγονται ως πρότυπα επειδή το νευρικό σύστημά τους είναι απλούστερο από άλλα ζωικά είδη. Επίσης, οι σύνθετες συμπεριφορές μπορούν να αποδοθούν ακριβώς σε μερικούς νευρώνες και η διάβιβαση μεταξύ των αισθητήριων εισαγωγής και της παραγωγής μηχανών είναι σχετικά πιό σύντομη. Η συμπεριφορά περπατήματος του εντόμου και η νευρική αρχιτεκτονική χρησιμοποιούνται για να βελτιώσουν τη μετακίνηση ρομπότ. Εναλλακτικά, οι βιολόγοι μπορούν να χρησιμοποιήσουν τα εξάποδα ρομπότ για τη δοκιμή των διαφορετικών υποθέσεων.

Τα βιολογικά εμπνευσμένα εξάποδα ρομπότ εξαρτώνται κατά ένα μεγάλο μέρος από τα είδη εντόμων που χρησιμοποιούνται ως πρότυπο. Η κατσαρίδα και το έντομο stick insect είναι τα δύο πιο συνηθέστερα χρησιμοποιημένα είδη εντόμων.

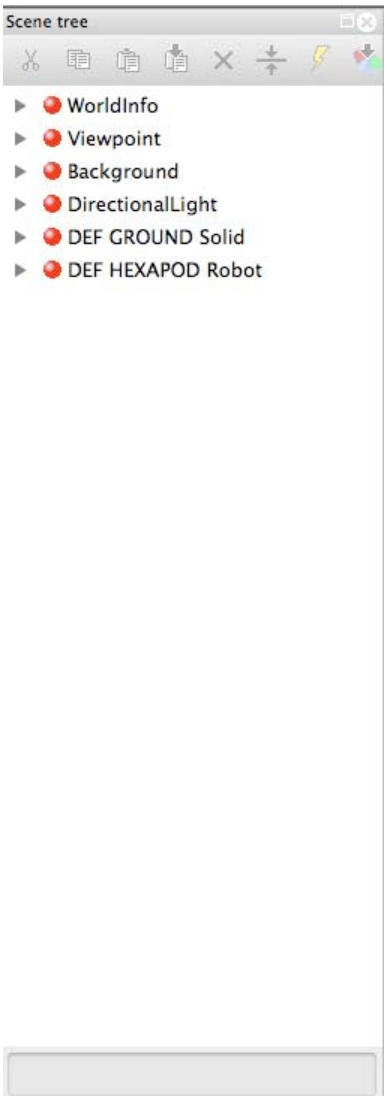
Οι βηματισμοί εντόμων λαμβάνονται συνήθως από δύο προσεγγίσεις: οι συγκεντρωμένες και αποκεντρωμένες αρχιτεκτονικές ελέγχου. Οι συγκεντρωμένοι ελεγκτές διευκρινίζουν άμεσα τις μεταβάσεις όλων των ποδιών, ενώ στις αποκεντρωμένες αρχιτεκτονικές, έξι κόμβοι (πόδια) συνδέονται σε ένα παράλληλο δίκτυο και οι βηματισμοί προκύπτουν από την αλληλεπίδραση μεταξύ των γειτονικών ποδιών.

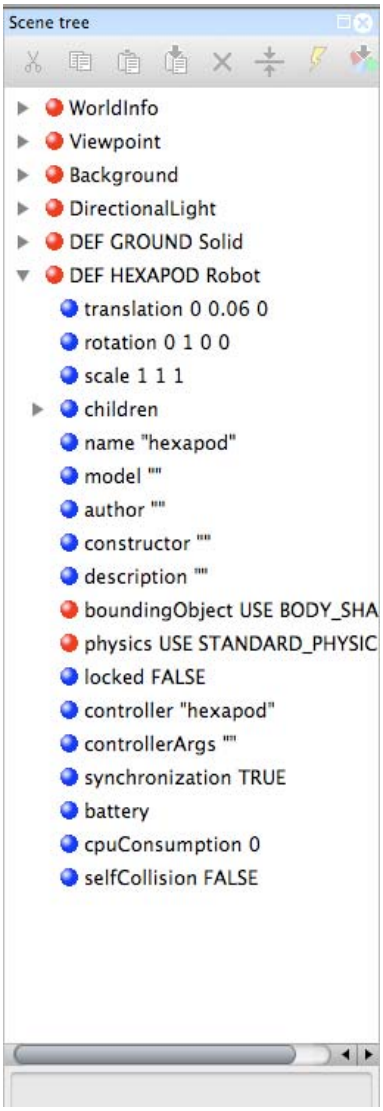
11.4 Hexapod.wbt

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε τις έννοιες : Legged robot, alternating tripod gait, linear Servo.



Εικόνα 1. hexapod.wbt


Scene tree
WorldInfo
Viewpoint
Background
DirectionalLight
DEF GROUND Solid
DEF HEXAPOD Robot


Scene tree
WorldInfo
Viewpoint
Background
DirectionalLight
DEF GROUND Solid
DEF HEXAPOD Robot
translation 0 0.06 0
rotation 0 1 0 0
scale 1 1 1
children
name "hexapod"
model ""
author ""
constructor ""
description ""
boundingObject USE BODY_SHA
physics USE STANDARD_PHYSIC
locked FALSE
controller "hexapod"
controllerArgs ""
synchronization TRUE
battery
cpuConsumption 0
selfCollision FALSE

Σε αυτό το παράδειγμα έχουμε ένα διαμορφωμένο ρομπότ που αποτελείται από ένα συνδυασμό γραμμικών και περιστροφικών σέρβο-συσκευών.
Η κίνηση του ρομπότ γίνεται χρησιμοποιώντας έναν εναλλασσόμενο βηματισμό τρίποδων

Στη συνέχεια θα ασχοληθούμε με τον κώδικα του Hexapod.wbt
Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες που είναι :

```
#include <webots/robot.h>
#include <webots/servo.h>
```

Καθώς θέτουμε και τις εξωτερικές μεταβλητές για να μας διευκολύνουν στο πρόγραμμα μας.

```
#define TIME_STEP 16
#define NUM_SERVOS 12
#define NUM_STATES 6

#define FRONT +0.7
#define BACK  -0.7
#define HI    +0.02
#define LO    -0.02
```

Παρακάτω έχουμε τη λειτουργία το ποδιών του hexapod το πότε θα είναι δηλαδή τα 3 από τα 6 στο έδαφος και ποτέ δεν θα είναι τα άλλα. Αυτό συνεχίζει να γίνεται εναλλάξ.

```
int main() {
const char *SERVO_NAMES[NUM_SERVOS] = {
"hip_servo_r0",
"hip_servo_r1",
"hip_servo_r2",
"hip_servo_l0",
"hip_servo_l1",
"hip_servo_l2",
"knee_servo_r0",
"knee_servo_r1",
"knee_servo_r2",
"knee_servo_l0",
"knee_servo_l1",
"knee_servo_l2"};
WbDeviceTag servos[NUM_SERVOS];
const double pos[NUM_STATES][NUM_SERVOS] = {
{BACK, FRONT, BACK, -FRONT, -BACK, -FRONT, LO, HI, LO, HI, LO, HI},
{BACK, FRONT, BACK, -FRONT, -BACK, -FRONT, HI, HI, HI, HI, HI, HI},
{BACK, FRONT, BACK, -FRONT, -BACK, -FRONT, HI, LO, HI, LO, HI, LO},
{FRONT, BACK, FRONT, -BACK, -FRONT, -BACK, HI, LO, HI, LO, HI, LO},
{FRONT, BACK, FRONT, -BACK, -FRONT, -BACK, HI, HI, HI, HI, HI, HI},
{FRONT, BACK, FRONT, -BACK, -FRONT, -BACK, LO, HI, LO, HI, LO, HI}
};


int elapsed = 0;
int state, i;

wb_robot_init();

for (i = 0; i < NUM_SERVOS; i++) {
servos[i] = wb_robot_get_device(SERVO_NAMES[i]);
if (!servos[i]) {
printf("could not find servo: %s\n",SERVO_NAMES[i]);
}
}

while(wb_robot_step(TIME_STEP)!=-1) {
elapsed++;
state = (elapsed / 25 + 1) % NUM_STATES;
for (i = 0; i < NUM_SERVOS; i++) {
```

```
wb_servo_set_position(servos[i], pos[state][i]);
}
}

wb_robot_cleanup();

return 0;
}
```

Ο κώδικας του κεφαλαίου Hexapod.wbt φαίνετε ολοκληρωμένος στο παράρτημα.

Κεφάλαιο 12

12.1 ΣΥΝΕΡΓΑΣΙΑ MATLAB ΜΕ WEBOTS

Το webots μας δίνει την δυνατότητα να συνεργαστούμε και με το πρόγραμμα matlab.

12.1.2 Εισαγωγή σε MATLAB

Το MATLAB είναι ένα αριθμητικό υπολογιστικό περιβάλλον και μας παρέχει μια ερμηνευμένη γλώσσα προγραμματισμού. Μας επιτρέπει τον εύκολο χειρισμό matrix, το χειρισμό των λειτουργιών και των στοιχείων, την εφαρμογή των αλγορίθμων και τη αλληλεπιδραση με τον χρήστη. Μπορείτε να πάρετε περισσότερες πληροφορίες για τον επίσημο ιστοχώρο του MathWorks. Χρησιμοποιείται ευρέως στη ρομποτική για την επεξεργασία εικόνας, τα νευρωνικά δίκτυα και τις εργαλειοθήκες αλγορίθμων γενετικής (Genetics Algorithms*)* του. Το Webots μας επιτρέπει να χρησιμοποιήσουμε άμεσα τα scripts του MATLAB ως προγράμματα ελεγκτών ρομπότ για τις προσομοιώσεις σας. Χρησιμοποιώντας τη διεπαφή MATLAB, γίνεται εύκολο να απεικονιστούν τα στοιχεία ελεγκτών ή εποπτών, παραδείγματος χάριν, επεξεργασμένες εικόνες, αναγνώσεις αισθητήρων, η απόδοση ενός αλγορίθμου βελτιστοποίησης, κ.λπ. ενώ η προσομοίωση τρέχει. Επιπλέον γίνεται και η επαναχρησιμοποίηση του υπάρχων κώδικα MATLAB άμεσα σε Webots.

Η τυποποιημένη έκδοση MatLabTM δεν παρέχει μια σαφή διεπαφή TCP/IP. Εντούτοις, μια ελεύθερη εργαλειοθήκη αποκαλούμενη TCP/UDP/IP εργαλειοθήκη 2.0.5, που αναπτύσσεται από τον κ. Mr. Peter Rydesliter, είναι διαθέσιμη. Αυτή η εργαλειοθήκη μπορεί να βρεθεί στο CD του webots, καθώς επίσης και στον ιστοχώρο MatLab. Μπορεί να χρησιμοποιηθεί για να επιτρέψει στο πρόγραμμα MatLab να συνδεθεί με τους ελεγκτές Webots tcpip στα ρομπότ κίνησης. Ένα MatLabTM TCP/IP Java interface για webots αναπτύχθηκε από τον κ.Harm Aarts από το Τεχνολογικό Πανεπιστήμιο του Ελσίνκι (HUT), Αυτό συστήνεται εάν αισθάνεστε πιο άνετοι με την Java από τη C. Μια άλλη επιλογή με MatLabTM είναι να χρησιμοποιηθεί η διεπαφή MatLabTM C για να συνδέσει έναν ελεγκτή Webots με τη μηχανή MatLabTM. Οποιαδήποτε στοιχεία μπορούν να περάσουν στο MatLabTM, παραδείγματος χάριν την εικόνα από μια κάμερα Webots, και οι εντολές MatLabTM μπορούν να επικαλεσθούν από μέσα από τον ελεγκτή Webots για να χρησιμοποιήσουν εκείνο το στοιχείο. Η μόνη πιθανή δυσκολία είναι η μετατροπή στοιχείων μεταξύ C και των διάφορων τύπων εκλεκτών MatLab .

12.2 Πως μπορούμε να τρέξουμε τα προγράμματα

Εάν έχουμε ήδη εγκαταστήσει το MATLAB μπορούμε άμεσα να τρέξουμε ένα από τα παραδείγματα του προγράμματος αυτού. Για να γίνει αυτό ανοίγουμε το Webots και ανοίγουμε το projects/packages/matlab/worlds/matlab.wbt ή το projects/contests/nao_robocup/worlds/nao2_matlab.wbt στο installation directory του Webots. Έτσι το webots αυτόματα ξεκινά το Matlab μόλις ανιχνεύσει ένα m-file σε έναν κατάλογο ελεγκτών (controller directory).

12.3 Πώς χρησιμοποιούμε το MATLAB API;

Το MATLAB API για Webots είναι πολύ παρόμοιο με το C API. Οι λειτουργίες ονομάζονται identically, μόνο ο τύπος και ο αριθμός παραμέτρων διαφέρουν ελαφρώς σε μερικές περιπτώσεις. Οι λειτουργίες και τα πρωτότυπα MATLAB περιγράφονται στο εγχειρίδιο αναφοράς Webots. Εδώ είναι ένα απλό παράδειγμα ελεγκτών:

```
%desktop
%keyboard

TIME_STEP = 32

camera = wb_robot_get_device («camera»)
wb_camera_enable (camera, TIME_STEP)

while 1
  wb_robot_step (TIME_STEP)
  rgb = wb_camera_get_image (camera)
  image (rgb)
  drawnow
  (500, -500)
end
```

Αυτό το παράδειγμα κάνει το ρομπότ DifferentialWheels να περιστραφεί επιδεικνύοντας τις εικόνες καμερών σε έναν αριθμό MATLAB .

Σημειώστε ότι το m-file πρέπει να καθοριστεί μετά από τον κατάλογό του, προκειμένου να προσδιοριστεί ως αρχείο ελεγκτών από το Webots. Παραδείγματος χάριν εάν ο κατάλογος ονομάζεται my_controller, τότε το m-file των ελεγκτών θα πρέπει να ονομαστεί my_controller/my_controller.m.

Κανένας ειδικός κώδικας έναρξης δεν είναι απαραίτητος στο m-file ελεγκτών. Στην πραγματικότητα το Webots καλεί ένα ενδιάμεσο αρχείο launcher.m που θέτει το περιβάλλον ελεγκτών Webots και καλεί έπειτα το m-file ελεγκτών. Το αρχείο launcher.m φορτώνει τη βιβλιοθήκη για την επικοινωνία με το Webots και προσθέτει στην πορεία τα m- file API. Τα m- file MATLAB API βρίσκονται στο lib/matlab directory της διανομής Webots.

12.4 Χρησιμοποιώντας το MATLAB

Προκειμένου να αποφύγει τα πάρα πολλά παράθυρα στον υπολογιστή το Webots αρχίζει να τρέχει το MATLAB ™ με - *nodesktop* επιλογή. Η Nodesktop επιλογή ξεκινάει το MATLAB ™ χωρίς ο χρήστης να μπορεί να παρέμβει στον ελεγκτή και επομένως κρατιέται η χρήση μνήμης χαμηλά που είναι χρήσιμη για τα πειράματα με τα ρομπότ. Εάν θα επιθυμούμε την αλληλεπιδράση με τον ελεγκτή θα πρέπει να προσθέσουμε αυτές τις δύο εντολές MATLAB ™ κάπου στην αρχή του m-αρχείου :

```
%desktop
%keyboard
```

Η εντολή desktop ξεκινάει την αλληλεπίδραση με το MATLAB. Το keyboard σταματά την εκτέλεση του ελεγκτή και δίνει τον έλεγχο στο πληκτρολόγιο .Κατόπιν το MATLAB ανοίγει το m-file ελεγκτών στον editor και η εκτέλεση σταματάει στην εντολή keyboard. Κατόπιν το m-file ελεγκτών μπορεί να διορθωθεί αμφίδρομα. Είναι δυνατό να συνεχιστεί η εκτέλεση του ελεγκτή με τη την εντολή return. Τελικά η εκτέλεση του ελεγκτή μπορεί να ολοκληρωθεί με το βασικό συνδυασμό CTRL-C.

Αφότου ολοκληρώθηκε ο ελεγκτής, η σύνδεση με Webots παραμένει ενεργή. Επομένως είναι δυνατό να δοθούν κι άλλες εντολές Webots άμεσα στο MATLAB, παραδείγματος χάριν μπορείτε αμφίδρομα να δώσετε τις εντολές για τους αισθητήρες, κ.λπ.:

```
>> (600, 600)
>> wb_robot_step (1000)
>> wb_gps_get_values (pst)

ANS =

  0.0001 0.0030 -0.6425
>> |
```

Σε αυτό το σημείο είναι επίσης δυνατό να επανακινησουμε τον ελεγκτής με την κλήση του m-file από την κονσόλα του MATLAB ™. Σημειώστε ότι αυτό θα επανακινηση τον ελεγκτή μόνο, όχι ολόκληρη τη προσομοίωση, έτσι το τρέχον ρομπότ και οι σέρβο θέσεις θα συντηρηθούν. Εάν θέλετε να επενακινησετε ολόκληρη την προσομοίωση πρέπει να χρησιμοποιήσετε το **Revert** κουμπί όπως συνήθως.

12.4.1 ΕΡΓΑΛΕΙΟΘΗΚΗ MATLAB TCP/IP

Η τυποποιημένη έκδοση MATLAB δεν παρέχει μια σαφή διεπαφή TCP/IP. Εντούτοις, μια ελεύθερη εργαλειοθήκη αποκαλούμενη TCP/UDP/IP εργαλειοθήκη 2.0.5, που αναπτύσσεται από τον κ. Peter Rydesäter, είναι διαθέσιμη. Αυτή η εργαλειοθήκη μπορεί να βρεθεί στο CD-rom του Webots (στον κοινό/κατάλογο UTIL), καθώς επίσης και στον ιστοχώρο MATLAB. Είναι γνωστό για να τρέχει στα παράθυρα, Linux και άλλα συστήματα της UNIX. Μπορεί να χρησιμοποιηθεί για να επιτρέψει τα προγράμματα MATLAB για να συνδέσει με τους ελεγκτές Webots tcpip στα ρομπότ κίνησης.

12.4.2 Κύρια Πλεονεκτήματα

Υπάρχουν διάφορα πλεονεκτήματα μιας τέτοιας διεπαφής. Κατ' αρχάς, μπορείτε να έχετε διάφορα μιμούμενα ρομπότ στον ίδιο κόσμο χρησιμοποιώντας διάφορες περιπτώσεις του ίδιου ελεγκτή tcpip, κάθε μια χρησιμοποιώντας έναν διαφορετικό port του TCP/IP, επιτρέποντας κατά συνέπεια στο λογισμικό σας για να ελέγξετε διάφορα ρομπότ μέσω διάφορων συνδέσεων TCP/IP. Για να επιτρέψετε στη διαδικασία tcpip και να ανοίξετε έναν διαφορετικό port ανάλογα με το ελεγχόμενο ρομπότ, πρέπει να δώσετε ένα διαφορετικό όνομα σε κάθε ρομπότ και να χρησιμοποιήσετε το robot_get_name () στον ελεγκτή tcpip για να ανακτήσετε αυτό το όνομα και να αποφασίσετε ποιο port να ανοίξει για κάθε ρομπότ.

Το δεύτερο πλεονέκτημα είναι ότι μπορείτε επίσης από μακριά να ελέγξετε ένα πραγματικό ρομπότ π.χ. Khepera από το λογισμικό σας χωρίς γράψιμο μιας γραμμής κώδικα. Απλά με τον τηλεχειρισμο στο παράθυρο Khepera θα επαναπροσανατολίσει την εισόδου-εξόδου στο πραγματικό ρομπότ μέσω της τμηματικής γραμμής.

Το τρίτο πλεονέκτημα είναι ότι μπορείτε να διαδώσετε τα προγράμματα ελεγκτών σας πέρα από ένα δίκτυο των υπολογιστών. Αυτό είναι ιδιαίτερα χρήσιμο εάν τα προγράμματα ελεγκτών εκτελούν υπολογιστικά τους ακριβούς αλγορίθμους όπως οι γενετικοί αλγόριθμοι ή άλλες τεχνικές εκμάθησης.

Τέλος, πρέπει να θέσετε το ελεγχόμενο ρομπότ στο σύγχρονο ή ασύγχρονο τρόπο ανάλογα με τον εάν θέλετε ή όχι τον προσομοιωτή Webots kai να περιμένετε τις εντολές από τους ελεγκτές σας. Στο σύγχρονο τρόπο (με τον τομέα synchronization του ρομπότ ίσου με TRUE), ο προσομοιωτής θα περιμένει τις εντολές από τους ελεγκτές σας. Το βήμα ελεγκτών που καθορίζεται από την παράμετρο robot_step τον ελεγκτή tcpip θα γίνει σεβόμενο. Στον ασύγχρονο τρόπο (με τον τομέα synchronization του ρομπότ καθορισμένου FALSE), ο προσομοιωτής θα τρέξει όσο το δυνατόν γρηγορότερα, χωρίς να αναμενη τις εντολές από τους ελεγκτές σας. Στην τελευταία περίπτωση, μπορείτε να θελήσετε να ελέγξετε τον τομέα runRealTime του κόμβου WorldInfo στο sceene treeWindow προκειμένου να υπάρξει μια real time προσομοίωση στην οποία τα ρομπότ πρέπει να συμπεριφερθούν όπως τα πραγματικά ρομπότ που ελέγχονται μέσω μιας ασύγχρονης σύνδεσης.

12.4.3 Περιορισμοί

Το κύριο μειονέκτημα TCP/IP που διασυνδέει είναι ότι εάν το ρομπότ σας έχει μια συσκευή καμερών, το πρωτόκολλο πρέπει να στείλει τις εικόνες στον ελεγκτή μέσω TCP/IP, το οποίο να είναι δίκτυο εντατικό. Ως εκ τούτου συστήνεται να υπάρξει ένα δίκτυο υψηλής ταχύτητας, ή να χρησησιμοποιεί τις μικρές ψηφιακες εικόνες καμερών, ή να συμπιέζει τα στοιχεία εικόνας πρίν τα στέλνει στον ελεγκτή. Αυτά τα γενικά έξοδα είναι αμελητέα εάν χρησιμοποιείτε μια χαμηλή κάμερα ψηφίσματος όπως το Khepera K213.

12.5 ΕΦΑΡΜΟΓΗ ΤΗΣ ΛΟΓΙΚΗΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΕΙ ΤΟ MATLAB(MATLAB FUZZY-LOGIC):

Για την αποφυγή των εμποδίων, θα καθορίσουμε 3 εισαγωγές και θα περιμένουμε 2 επιθυμητά αποτελέσματα.

Οι 3 επιλεγμένες εισαγωγές είναι οι εξής: (τιμές από την μέτρηση της απόστασης των αισθητήρων)

1. Μπροστινή απόσταση εμποδίων (FOD)

2. Αριστερή απόσταση εμποδίων (LOD)

3. Σωστή απόσταση εμποδίων (ROD)

Επιθυμητά Ouputs:

1. Αριστερή ταχύτητα ροδών (LWV)

2. Σωστή ταχύτητα ροδών (RWV)

12.5.1 ΕΦΑΡΜΟΓΗ ΤΟΥ FUZZY-SYSTEM:

Αυτό το παραδειγμα πραγματοποιείται αρχικά από το TOOLBOX MATLAB FUZZY-LOGIC και στη συνέχεια από το κώδικα του MATLAB.

Τα βήματα που ακολουθούνται είναι τα εξής:

1. Πληκτρολογούμε “anfisedit” στο command παράθυρο του MATLAB.

2. Πηγαίνουμε στο “File” , “New FIS”, Mamdani

3. Χρησιμοποιείται ο συντάκτης FIS για να καθορίσει τις εισαγωγές και τα αποτελέσματά μας.

4.Πηγαίνουμε στο “Edit”, “Add Variables”, “Input / Output”.

5. Στη συνεχεια επιλεγουμε συμφονα με τις αναγκες μας: δηλαδή τραπεζοειδής, τριγωνικός, γκαουσσιανός, κ.λπ.

6. Το Defuzzification γίνεται με τη μέθοδο Centroid.

12.5.2 ΚΑΘΟΡΙΣΜΟΣ ΤΩΝ ΕΙΣΑΓΩΓΩΝ & ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ:

1. ΕΙΣΑΓΩΓΕΣ: FOD, LOD, ROD μπορεί να είναι μακρινό Far(f) ή να πλησιάζει Near(N)

2. ΑΠΟΤΕΛΕΣΜΑΤΑ: LWV, RWV μπορεί να είναι γρήγορο fast(f) ή αργό Slow(S).

ΚΑΝΟΝΕΣ ΓΙΑ ΤΗ ΛΕΙΤΟΥΡΓΙΑ:

Sl.No	FOD	LOD	ROD	LWV	RWV
1	F	F	F	F	f
2	F	F	N	S	f
3	F	N	F	F	s
4	F	N	N	F	f
5	N	F	F	S	f
6	N	F	N	S	f
7	N	N	F	F	s
8	N	N	N	S	s

ΕΝΣΩΜΑΤΩΣΗ ΤΗΣ MATLAB FUZZY-LOGIC ΣΤΟΝ ΠΡΟΣΟΜΟΙΩΤΉ WEBOTS:

Ο ακόλουθος κώδικας MATLAB ενσωματώνει την MATLAB FUZZY-LOGIC στο Main-Controller Program.

Υπάρχουν 2 τρόποι να ενσωματωθεί η Fuzzy-Logic στο Webots και να λειτουργήσει:

1. Να προσθέσουμε τον Fuzzy-Logic κώδικα στον κώδικα ελεγκτών Webots.

2. Να πάρουμε τις τιμές (απόσταση-αισθητήρας) από το Webots και να τις προσθέσουμε στον κώδικα Fuzzy-Logic του Mamdani Matlab controller FIS Editor.

Στο παράδειγμά μας υοθετείται ο δεύτερος τρόπος.

```
%[System]

Name='fuzzy-logic';

Type='mamdani';

Version=2.0;

NumInputs=3;

NumOutputs=2;

NumRules=8;

AndMethod='min';

OrMethod='max';

ImpMethod='min';

AggMethod='max';

DefuzzMethod='centroid';

x = 0:0.1:6;

%Rules

if (FOD == 'F' & LOD == 'F'& ROD == 'F')
```

```
LWV = 'f';
RWV = 'f';
end
if (FOD == 'F' & LOD == 'F'& ROD == 'N')
LWV = 's';
RWV = 'f';
end
if (FOD == 'F' & LOD == 'N'& ROD == 'F')
LWV = 'f';
RWV = 's';
end
if (FOD == 'F' & LOD == 'N'& ROD == 'N')
LWV = 'f';
RWV = 'f';
end
if (FOD == 'N' & LOD == 'F'& ROD == 'F')
LWV = 's';
RWV = 'f';
end


if (FOD == 'N' & LOD == 'N'& ROD == 'N')
LWV = 's';
```

```
RWV = 's';
end

if (FOD == 'N' & LOD == 'F'& ROD == 'N')
LWV = 's';
RWV = 'f';
end

if (FOD == 'N' & LOD == 'N'& ROD == 'F')
LWV = 'f';
RWV = 's';
end

%[Input1]
Name='FOD'
Range=[0 6];
NumMFs=2;
if ( FOD == 'F' )
MF1= trimf ( x, [2 4 6] );
end
if ( FOD == 'N' )
MF2= trimf ( x, [0 2 4] );
end
plot(MF1,MF2), title('FOD'),
xlabel('FOD values'),
```

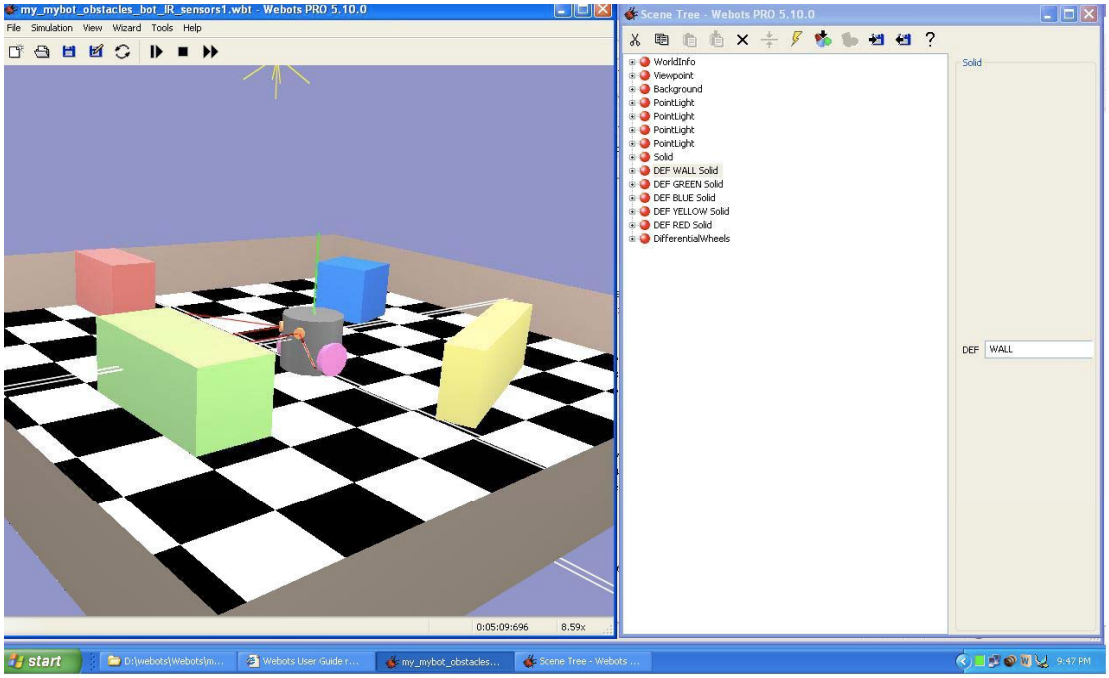
```
ylabel('Membership function value'),grid
out1 = defuzz(x,MF1,'centroid')
out11 = defuzz(x,MF2,'centroid')
out_1 = min(out1, out11)

%[Input2]
Name='LOD'
Range=[0 6];
NumMFs=2;
if ( LOD == 'F' )
MF1= trimf ( x, [3.5 4.5 5.5] );
end
if ( LOD == 'N' )
MF2= trimf ( x, [0.5 2.5 4.5] );
end
plot(MF1,MF2), title('LOD'),
xlabel('LOD values'),
ylabel('Membership function value'),grid
out2 = defuzz(x,MF1,'centroid')
out22 = defuzz(x,MF2,'centroid')
out_2 = min(out2, out22)

%[Input3]
Name='ROD'
Range=[0 6];
NumMFs=2;
```

```
if ( ROD == 'F' )
MF1= trimf ( x, [0.6 3.2 4.4] );
end
if ( ROD == 'N' )
MF2= trimf ( x, [3.6 5 6] );
end
plot(MF1,MF2), title('ROD'),
xlabel('ROD values'),
ylabel('Membership function value'),grid
out3 = defuzz(x,MF1,'centroid')
out33 = defuzz(x,MF2,'centroid')
out_3 = min(out3, out33)

out_final = min ( out_1, out_2, out_3 )



%[Output1]
Name='LWV'
Range=[0 4];
NumMFs=2;
if ( LWV == 'f' )
MF1= trimf ( x, [2 3.5 4] );
end
if ( LWV == 's' )
MF2= trimf ( x, [0 1 2] );
```

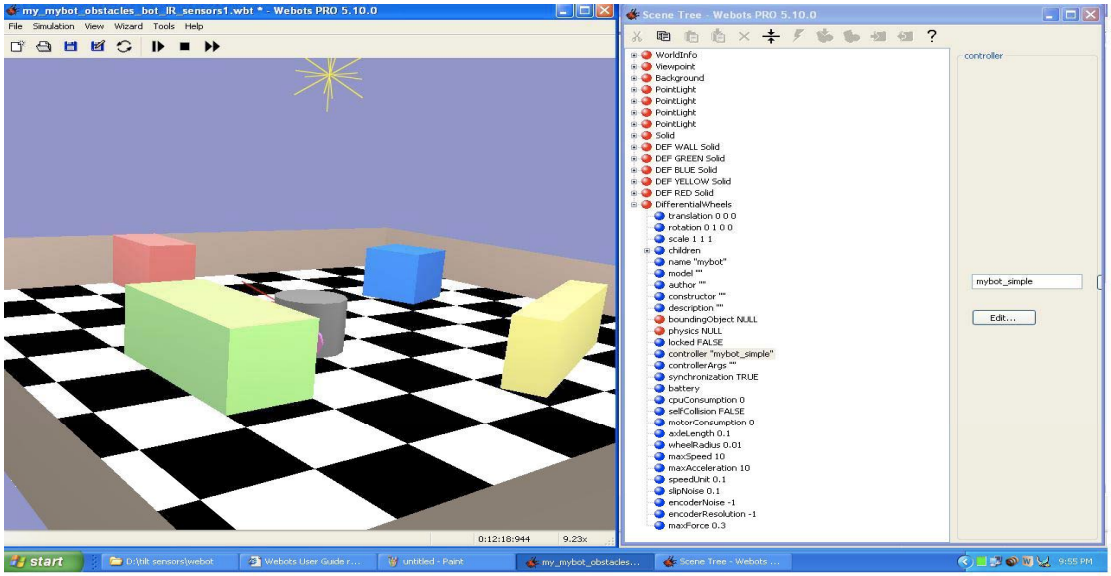
```
end
plot(MF1,MF2), title('LWV'),

xlabel('LWV values'),
ylabel('Membership function value'),grid
outf1 = defuzz(x,MF1,'centroid')
outf11 = defuzz(x,MF2,'centroid')
outf_1 = min(outf1, outf11)
outf_1net = outf_1./out_final

%[Output2]
Name='RWV'
Range=[0 4];
NumMFs=2;
if ( RWV == 'f' )
MF1= trimf ( x, [1.25 2.5 3.6] );
end
if ( RWV == 's' )
MF2= trimf ( x, [0 1.75 3] );
end
plot(MF1,MF2), title('RWV'),
xlabel('RWV values'),
ylabel('Membership function value'),grid
outf2 = defuzz(x,MF1,'centroid')
outf22 = defuzz(x,MF2,'centroid')
outf_2 = min(outf2, outf22)
```

```
outf_2net = outf_2./out_final

% [Rules]

% 1 1 2, 1 1 (1) : 1

% 1 1 1, 2 1 (1) : 1

% 1 2 2, 1 2 (1) : 1

% 1 2 1, 1 1 (1) : 1

% 2 1 2, 2 1 (1) : 1

% 2 1 1, 2 1 (1) : 1

% 2 2 2, 1 2 (1) : 1

% 2 2 1, 2 2 (1) : 1


% Defuzzification - Centroid

%out = defuzz(x,MF,'centroid')
```

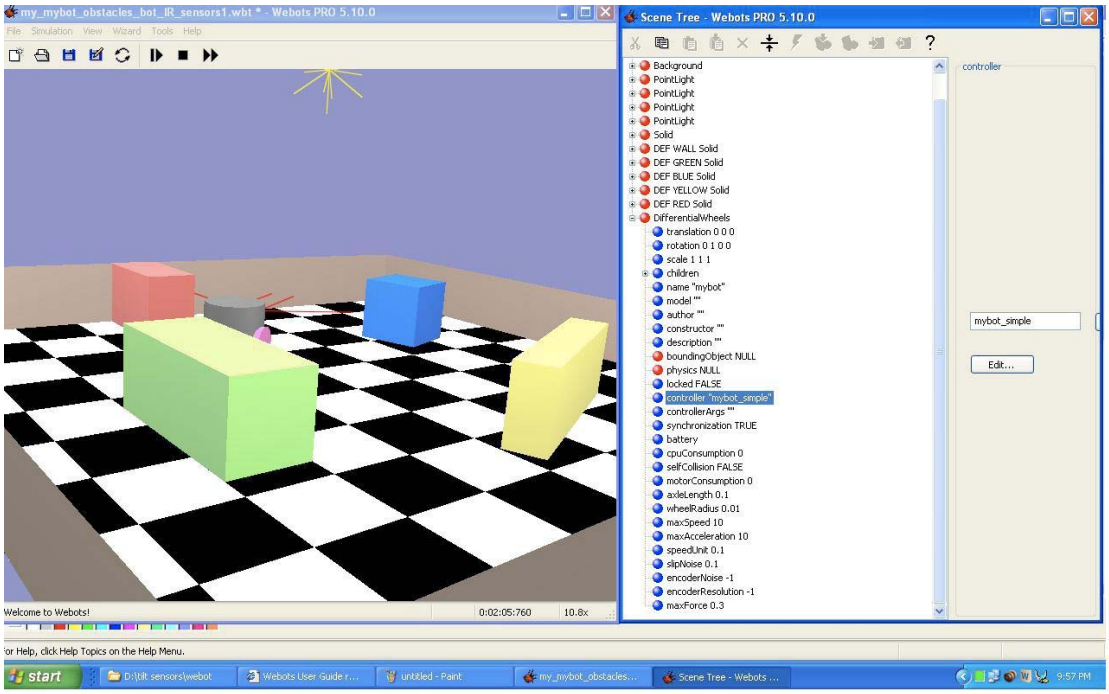
12.5.3 Αποτελέσματα από την προσομοίωση



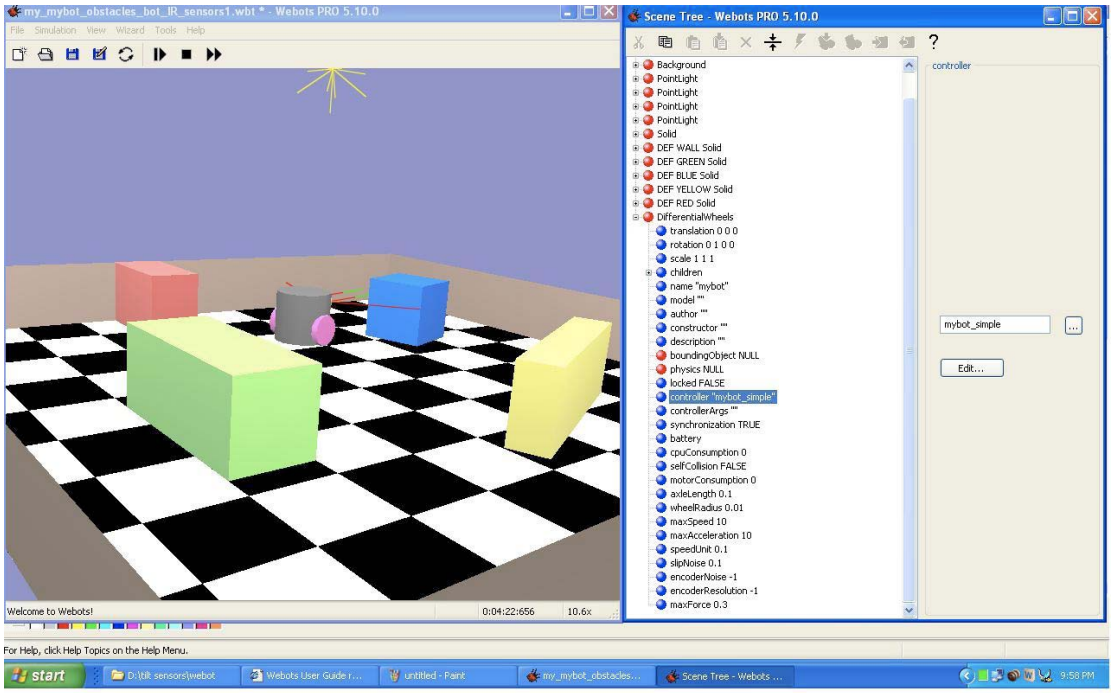
Σχήμα 1: Το ρομπότ πριν την αποφυγή εμποδίου(στην αρχική του θέση)



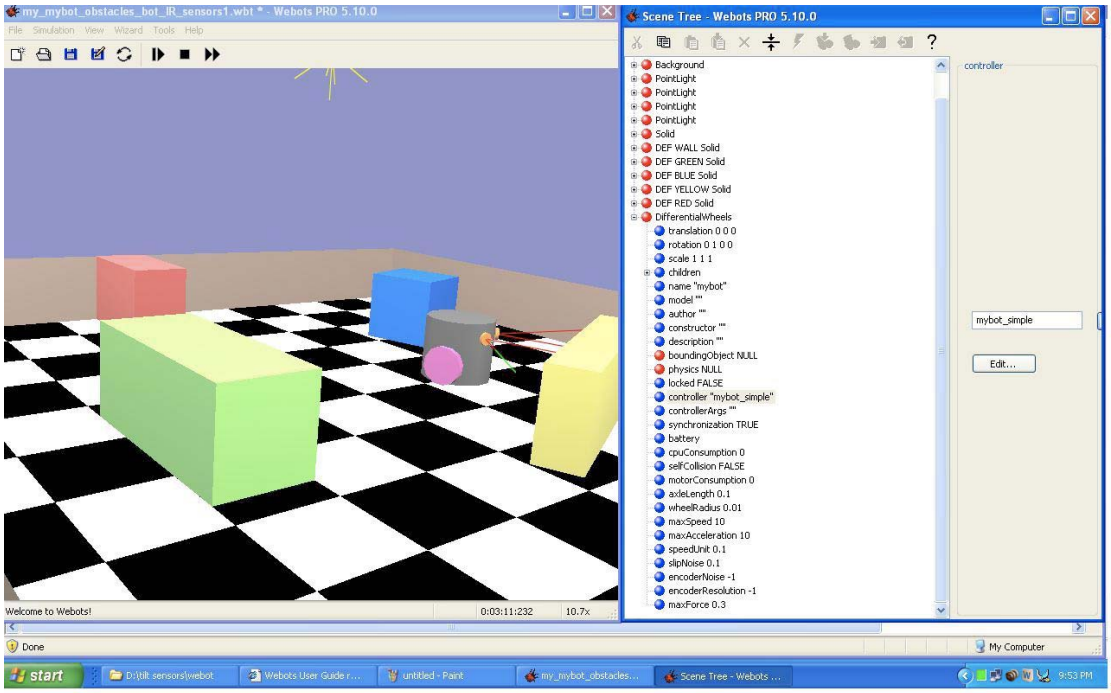
Σχήμα 2: Το ρομπότ αποφεύγει το πρώτο εμπόδιο (πράσινο)



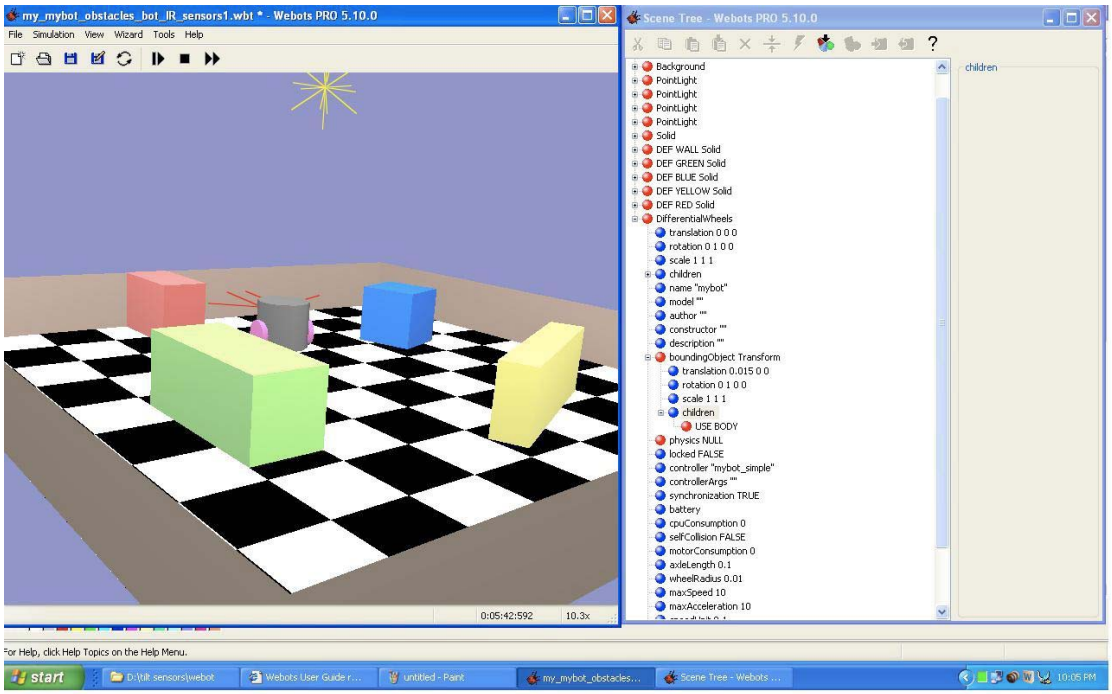
Σχήμα 3: Το ρομπότ αποφεύγει το δεύτερο εμπόδιο (κόκκινο)



Σχήμα 4: Το ρομπότ αποφεύγει το τρίτο εμπόδιο (μπλέ)

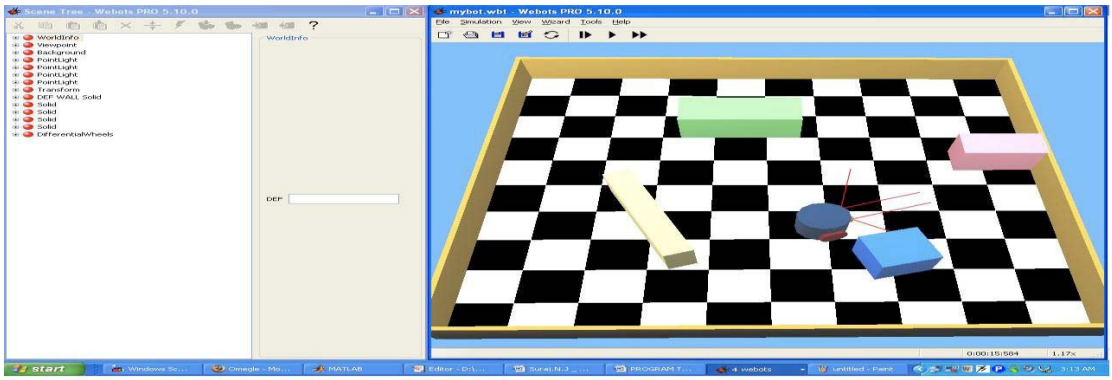


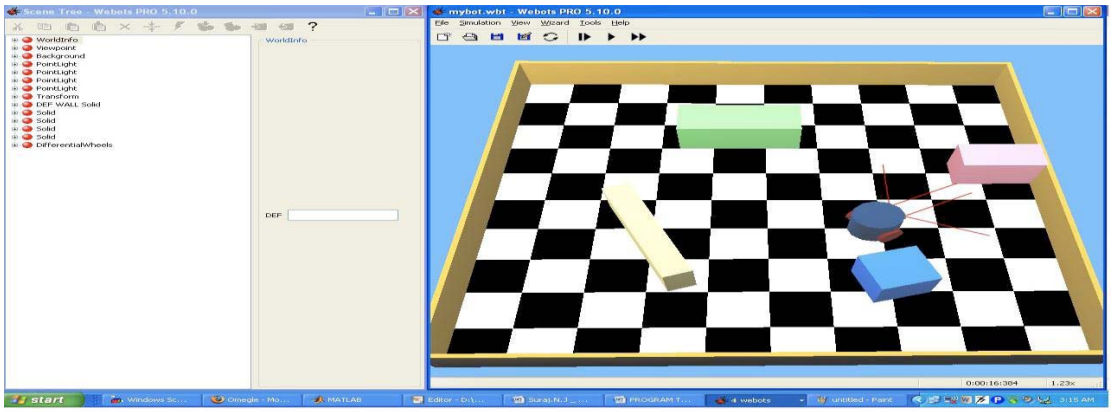
Σχήμα 4: Το ρομπότ αποφεύγει το τέταρτο εμπόδιο (κίτρινο)

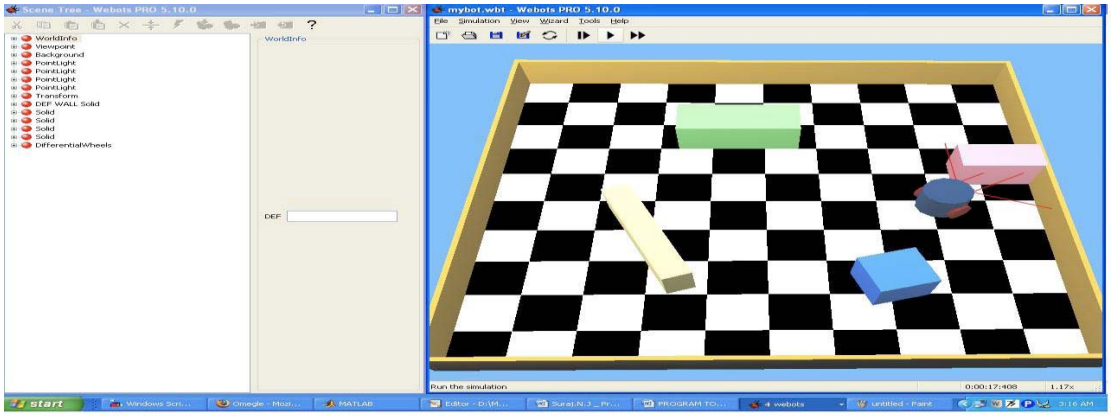


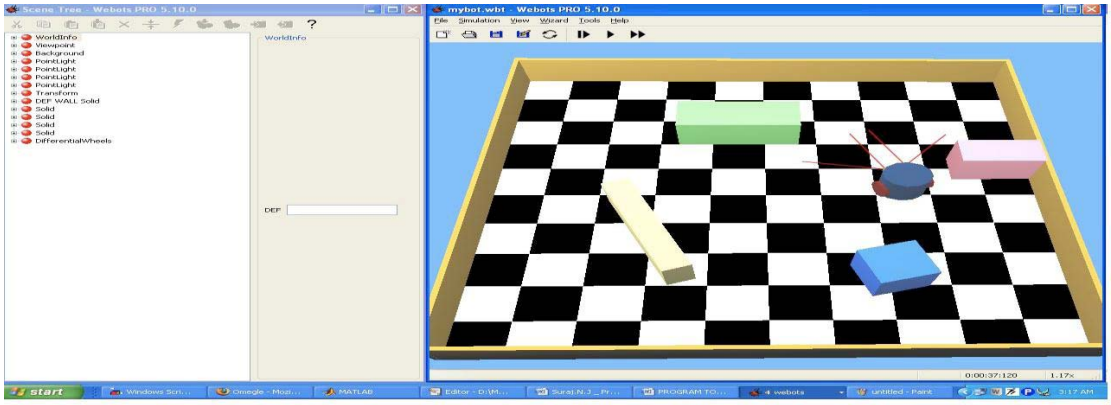
Σχήμα 5: Το ρομπότ στην τελική του θέση

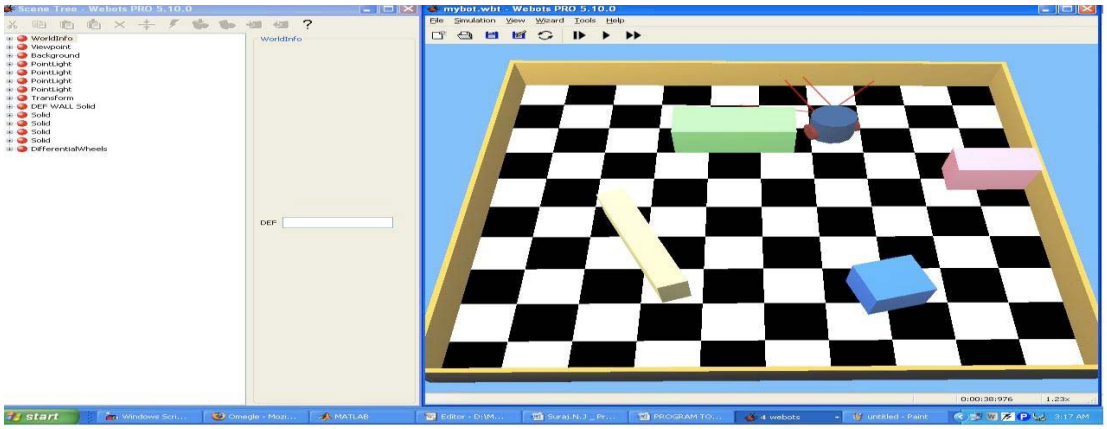
12.5.4 Πορεία του ρομπότ μετά την από την εφαρμογή του Fuzzy-Logic κώδικα

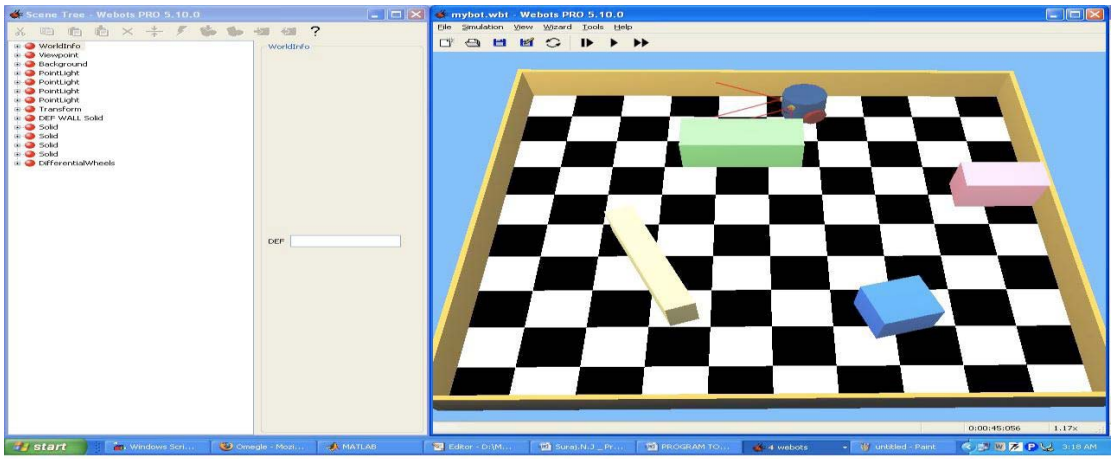


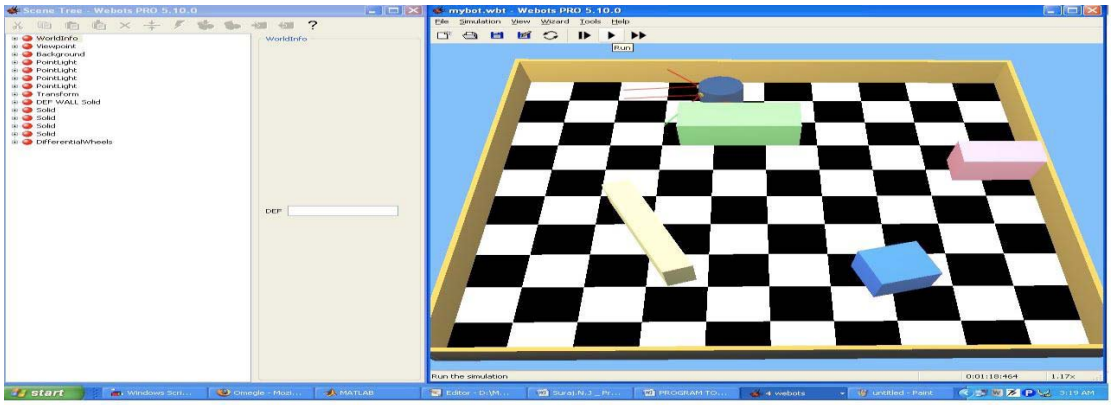


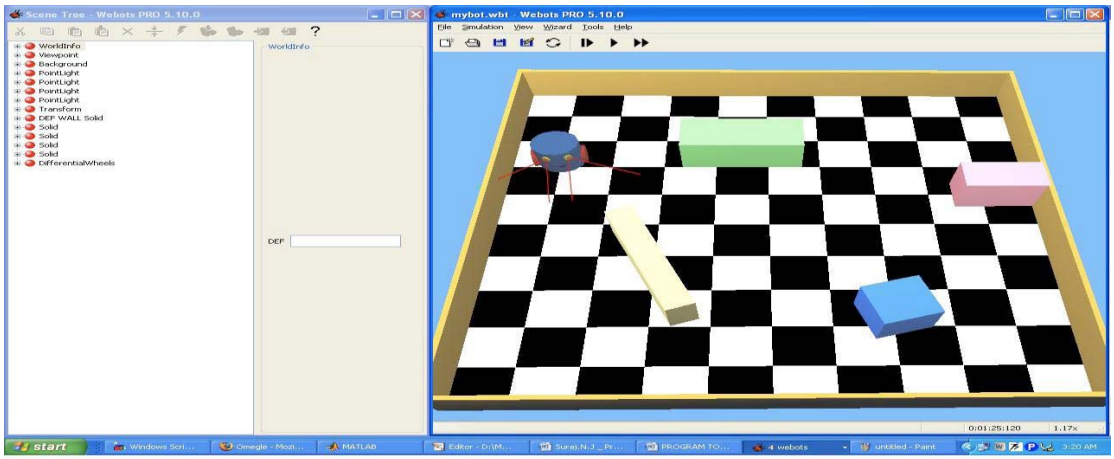

Scene Tree - Webots PRO 5.10.0
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
WorldInfo
Viewpoint
Background
PointLight
Transform
DEF WALL Solid
Solid
DifferentialWheels
DEF
Run the simulation
start


Scene Tree - Webots PRO 5.10.0
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
WorldInfo
Viewpoint
Background
PointLight
Transform
DEF WALL Solid
Solid
DifferentialWheels
DEF
start


Scene Tree - Webots PRO 5.10.0
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
WorldInfo
Viewpoint
Background
PointLight
Transform
DEF WALL Solid
Solid
DifferentialWheels
DEF
start


Scene Tree - Webots PRO 5.10.0
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
WorldInfo
Viewpoint
Background
PointLight
Transform
DEF WALL Solid
Solid
DifferentialWheels
DEF


Scene Tree - Webots PRO 5.10.0
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
Run
WorldInfo
Viewpoint
Background
PointLight
Transform
DEF WALL Solid
Solid
DifferentialWheels
DEF
Run the simulation


Scene Tree - Webots PRO 5.10.0
WorldInfo
Viewpoint
Background
PointLight
PointLight
PointLight
PointLight
Transform
DEF WALL Solid
Solid
Solid
Solid
Solid
DifferentialWheels
WorldInfo
DEF
mybot.wbt - Webots PRO 5.10.0
File Simulation View Wizard Tools Help
0:01:25:120
1.17x
start

Κεφάλαιο 13

13.1 Χρησιμοποιώντας το Webots με το E-puck

Σε αυτό το κεφάλαιο, θα μάθουμε πώς να χρησιμοποιούμε το Webots με το ρομπότ E-puck (σχήμα 1). Το e-puck είναι ένα μικροσκοπικό κινητό ρομπότ που αναπτύσσεται αρχικά για λόγους διδασκαλίας από τους σχεδιαστές του επιτυχούς ρομπότ Khepera. Το υλικό και το λογισμικό του E-puck είναι πλήρως ανοιχτού κώδικα, που παρέχει τη χαμηλού επιπέδου πρόσβαση σε κάθε ηλεκτρονική συσκευή και προσφέρει απεριόριστες δυνατότητες επέκτασης. Η επίσημη ιστοσελίδα του E-puck παρέχει τις πιο ενημερωμένες πληροφορίες για αυτό το ρομπότ. Το E-puck είναι επίσης διαθέσιμο στην αγορά από την Cyberbotics Ltd.

Σχήμα 1 ρομπότ E-puck

13.2 Λειτουργώντας το ρομπότ E-puck

Το E-puck έχει σχεδιαστεί και καλύπτει πλήρως τις παρακάτω απαιτήσεις :

- Κομψό σχεδιασμό : Η απλή μηχανική δομή του, το σχέδιο ηλεκτρονικής και το λογισμικό του E-puck είναι ένα παράδειγμα ενός σύγχρονου συστήματος.

- Ευελιξία : Το E-puck καλύπτει ένα ευρύ φάσμα των εκπαιδευτικών δραστηριοτήτων,προσφέρουν πολλές δυνατότητες που έχουν σχέση με αισθητήρες ή δύναμη επεξεργασίας και άλλα διάφορα extensions

- Λογισμικό προσομοίωσης : Στο E-puck είναι ενσωματωμένο το λογισμικό προσομοίωσης Webots για τον εύκολο προγραμματισμό, την προσομοίωση και τον τηλεχειρισμό του ρομπότ.

- Φιλικός προς το χρήστη : Το E-puck είναι μικρό και εύκολο στην οργάνωση tabletop δίπλα σε έναν υπολογιστή. Δεν χρειάζεται καλώδια.

- Ευρωστία και συντήρηση : Το e-puck είναι ανθεκτικό για τη χρήση σπουδαστών και είναι απλό να επισκευαστεί.

- Προσιτό: Η τιμή του E-puck είναι καλύτερη για αγοραστές όπως είναι πανεπιστήμια, τεχνολογικά εκπαιδευτικά ιδρύματα .

Τέλος το E-puck είναι εξοπλισμένο με έναν μεγάλο αριθμό συσκευών, όπως συνοψίζεται στον παρακάτω πίνακα.

Feature	Description
Size	about 7 cm in diameter
Battery	about 3 hours with the provided 5Wh LiION rechargeable battery
Processor	Microchip dsPIC 30F6014A @ 60MHz (about 15 MIPS)
Motors	2 stepper motors with 20 steps per revolution and a 50:1 reduction gear
IR sensors	8 infra-red sensors measuring ambient light and proximity of obstacles in a 4 cm range
Camera	color camera with a maximum resolution of 640x480 (typical use: 52x39 or 640x1)
Microphones	3 omni-directional microphones for sound localization
Accelerometer	3D accelerometer along the X, Y and Z axis
LEDs	8 red LEDs on the ring and one green LED on the body
Speaker	on-board speaker capable of playing WAV or tone sounds.
Switch	16 position rotating switch
Bluetooth	Bluetooth for robot-computer and robot-robot wireless communication
Remote Control	infra-red LED for receiving standard remote control commands
Expansion bus	expansion bus to add new possibilities to your robot
Programming	C programming with the GNU GCC compiler system
Simulation	Webots STD or EDU facilitates the programming of e-puck with a powerful simulation, remote control and cross-compilation system.

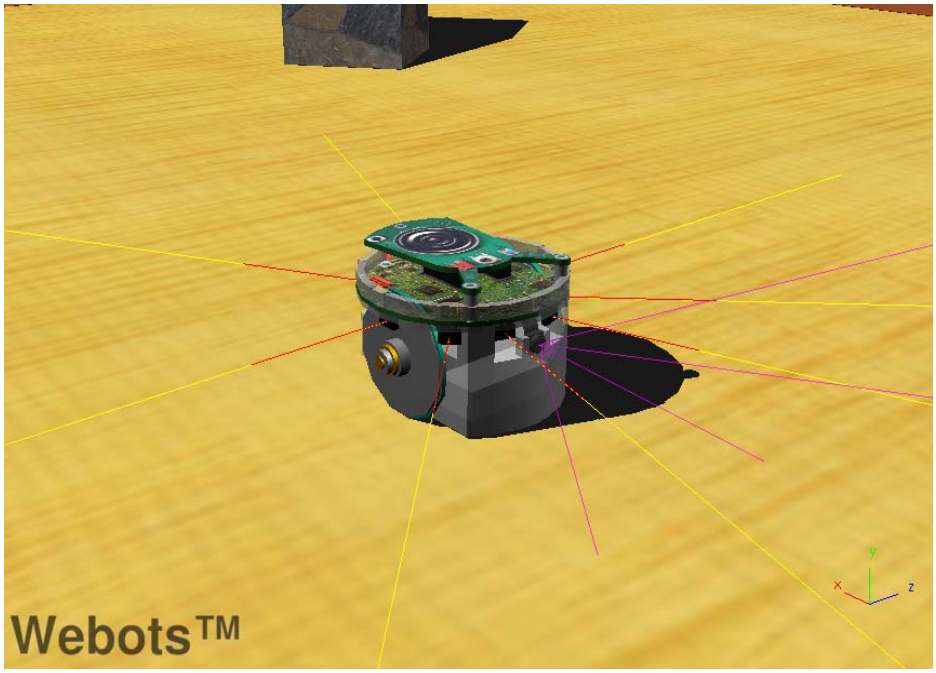
Τα χαρακτηριστικά του e-puck

13.2.1 Τεχνικές Λεπτομέρειες

- Διάμετρος: 70 χιλ.
- Ύψος: 50 χιλ.
- Βάρος: 200 γρ.
- Ανώτατη ταχύτητα: 13 cm/s
- Αυτονομία: Κίνηση 2 ωρών
- dsPIC 30 KME @ 30 MHZ (15 MIPS)
- 8 RAM KB
- 144 Flash KB
- 2 μηχανές βημάτων (differential Wheels)
- 8 υπέρυθρα και φως (TCRT1000)
- Κάμερα χρώματος, 640x480
- 8 LEDs στο δαχτυλίδι οδηγήσεων
- Τρισδιάστατα επιταχύμετρα
- 3 μικρόφωνα
- 1 μεγάφωνο

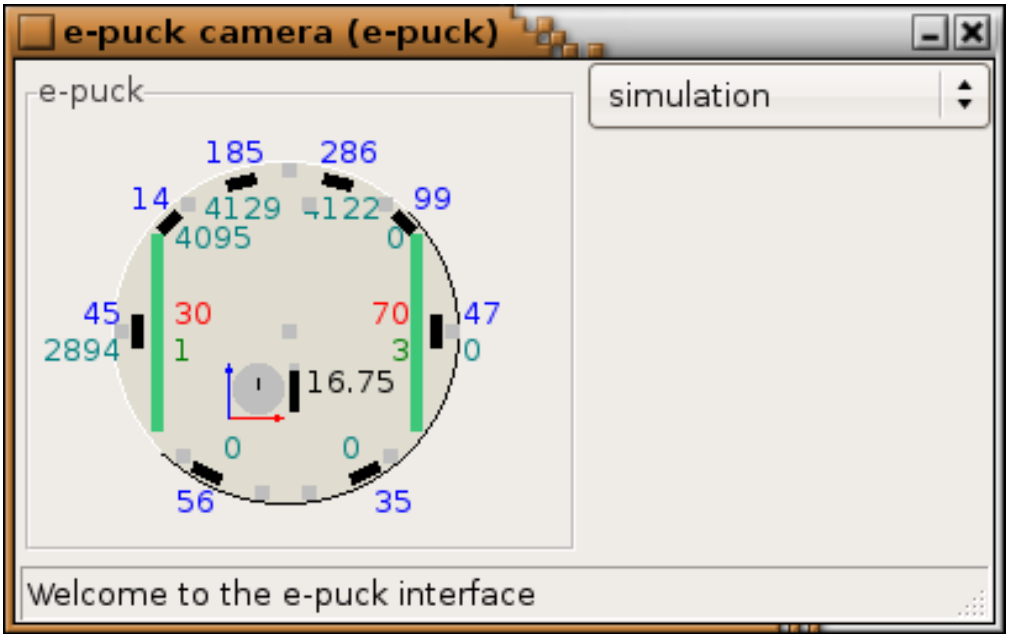
13.3 Simulation model

Το πρότυπο E-puck σε Webots απεικονίζεται στο σχήμα 2.



Σχήμα 2 E-puck

Αυτό το πρότυπο περιλαμβάνει την υποστήριξη για τις διαφορικές μηχανές (differential Wheels) των τροχών (συμπεριλαμβανομένης μιας προσομοίωσης των κωδικοποιητών), τους infra-red sensors, έναν accelerometer (επιταχύμετρο) , την κάμερα καθώς και 8 Led's που περιβάλλονται γύρω από τον E-puck (δείτε σχήμα 3)



Σχήμα 3 Περιγραφή των Led's

Το μοντέλο E-puck παρέχετε από την E-puck.proto prototype. Που ο φάκελος βρίσκετε στο projects/default/protos/robots directory της διανομής Webots . Διάφορα ενδιαφέροντα προγράμματα θα μπορούμε να βρούμε και στο φάκελο projects/robots/e-puck/worlds directory της διανομής Webots.

Παρακάτω παραθέτουμε κώδικες που αφορούν βασικές λειτουργίες του e-puck:

1. Types

```
#ifndef ROBOT_TYPES_H
#define ROBOT_TYPES_H

typedef unsigned char WbDeviceTag;      /* Αναγνώριση της συσκευής*/

#endif /* ROBOT_TYPES_H */
```

2. Robot

```
#ifndef ROBOT_H
#define ROBOT_H
#include "types.h"

int         wb_robot_init();
int         wb_robot_step(int ms);       /* ms = το χρονικό βήμα σε milliseconds */
void         wb_robot_cleanup()
  {
    }
double       wb_robot_get_time();
```

```
WbDeviceTag  wb_robot_get_device(const char *name);

   #define     wb_robot_get_name() "e-puck"
   #define     wb_robot_get_mode() 1
   #define     printf(s,...)
     {
  }
   #endif /* ROBOT_H */
```

3. Differential wheels

```
#ifndef DIFFERENTIAL_WHEELS_H
#define DIFFERENTIAL_WHEELS_H

void   wb_differential_wheels_set_speed(double left, double right);
void   wb_differential_wheels_enable_encoders(int ms);
void   wb_differential_wheels_disable_encoders();
double wb_differential_wheels_get_left_encoder();
double wb_differential_wheels_get_right_encoder();
void   wb_differential_wheels_set_encoders(double left,double right);

#endif /* DIFFERENTIAL_WHEELS_H */
```

4. Camera

```
#ifndef CAMERA_H
#define CAMERA_H

        #include "types.h"
        #include <float.h>

void        wb_camera_enable(WbDeviceTag,int ms);
void        wb_camera_disable(WbDeviceTag);

        #define     wb_camera_get_width(dt) 52
        #define     wb_camera_get_height(dt) 39
        #define     wb_camera_get_fov(dt) 0.7
        #define     wb_camera_get_type(dt) 'c'
        #define     wb_camera_get_near(dt) 0.0
        #define     wb_camera_get_far(dt) DBL_MAX

// Ή επιστρεφόμενη εικόνα έχει περιστραφή κατά 90° και έχει αποκωδικοποιειθη σε
RGB_565

// Αυτό γίνετε χρησιμοποιωντας το  camera_image_get_* functions below.
unsigned char *wb_camera_get_image(WbDeviceTag);

        unsigned char wb_camera_image_get_red(const unsigned char* image,int
width,int x,int y);

        unsigned char wb_camera_image_get_green(const unsigned char* image,int
```

```
width,int x,int y);

        unsigned char wb_camera_image_get_blue(const unsigned char* image,int
width,int x,int y);

#define wb_camera_image_get_grey(image,w,x,y)
        (wb_camera_image_get_red(image,w,x,y)/3 +
        wb_camera_image_get_green(image,w,x,y)/3 +
        wb_camera_image_get_blue(image,w,x,y)/3)

#endif /* CAMERA_H */
```

5. Distance sensor

```
#ifndef DISTANCE_SENSOR_H
#define DISTANCE_SENSOR_H

#include "types.h"

void   wb_distance_sensor_enable(WbDeviceTag,int ms);
void   wb_distance_sensor_disable(WbDeviceTag t);
double wb_distance_sensor_get_value(WbDeviceTag t);
#endif /* DISTANCE_SENSOR_H */
```

Κεφάλαιο 14

ΠΑΡΑΡΤΗΜΑ ΜΑΘΗΜΑΤΩΝ ΚΩΔΙΚΑ ΣΕ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Κωδικας του κεφαλαιου 5

Bumper.wbt

```
#include  <webots/robot.h>
#include  <webots/differential_wheels.h>
#include  <webots/touch_sensor.h>

#define  SPEED  40
#define  TIME_STEP  64

int main()

{
  WbDeviceTag bumper;
  int movement_counter=0;
  int left_speed, right_speed;

  wb_robot_init();

  bumper = wb_robot_get_device("bumper");
  wb_touch_sensor_enable(bumper, TIME_STEP);


  while(wb_robot_step(TIME_STEP)!=-1)
  {
    if (wb_touch_sensor_get_value(bumper) > 0)
   {
       movement_counter = 15;
    }

    if (movement_counter == 0)
      {
        left_speed = SPEED;
        right_speed = SPEED;
       }

 else if (movement_counter >= 7)

       {
       left_speed = -SPEED;
       right_speed = -SPEED;
       movement_counter--;
```

```
      }
else
{
      left_speed = -SPEED / 2;
      right_speed = SPEED;
      movement_counter--;
   }

   wb_differential_wheels_set_speed(left_speed, right_speed);
 }

 wb_robot_cleanup();

 return 0;
}
```

Κωδικας του κεφαλαιου 6

Camera.wbt

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/camera.h>

#define SPEED 40
#define TIME_STEP 64
#define COLOR_BLOB_VALUE 4500000
#define GREY_BLOB_VALUE 4350000

int main()
 {

  WbDeviceTag camera;
  int width, height;
  int pause_counter=0;
  int left_speed, right_speed;
  int i, j, centering_weight;
  int red, blue, green;
  unsigned char *image;

  wb_robot_init();

  camera = wb_robot_get_device("camera");

  wb_camera_enable(camera, TIME_STEP);

  wb_camera_move_window(camera, 0, 0);

  width = wb_camera_get_width(camera);

  height = wb_camera_get_height(camera);


       while(wb_robot_step(TIME_STEP)!=-1)

{

              image = wb_camera_get_image(camera);

                left_speed = 0;
                 right_speed = 0;

            if (pause_counter == 0) {
               red = 0;
             green = 0;
             blue = 0;


           for (i = 0; i < width; i++)
```

```
        {

            centering_weight = (i < (width / 2)) ? i - 10 : (width - i - 10);

               for (j = (height / 2); j < height; j++)

            {
                      red += wb_camera_image_get_red(image, width, i, j) *
centering_weight;
                      blue += wb_camera_image_get_blue(image, width, i, j) *
centering_weight;
                       green += wb_camera_image_get_green(image, width, i, j) *
centering_weight;

               }

         }

        if (blue >= COLOR_BLOB_VALUE)
    {
              printf("Looks like I found a blue blob !\n");
                  pause_counter = 20;
    }
         else if (green >= COLOR_BLOB_VALUE)
    {
               printf("Looks like I found a green blob !\n");
                    pause_counter = 20;
                        wb_camera_save_image(camera,"green_blob.png",100);
    }
              else if (red >= COLOR_BLOB_VALUE)
    {
               printf("Looks like I found a red blob !\n");
                    pause_counter = 20;
                         wb_camera_save_image(camera,"red_blob.jpg",100);

    }
               else if (blue >= GREY_BLOB_VALUE && green >=
GREY_BLOB_VALUE&& red >= GREY_BLOB_VALUE)

    {
                    printf("Looks like I found a grey blob !\n");
                      pause_counter = 20;
    }
                        else

     {
                          left_speed = -SPEED;
                          right_speed = SPEED;
    }
        }

 Else
       {
        pause_counter--;
```

```
                if (pause_counter <= 3)
{
                        left_speed = -SPEED;
                         right_speed = SPEED;

}

   }


  wb_differential_wheels_set_speed(left_speed, right_speed);

    }

          wb_robot_cleanup();

                  return 0;
}
```

Κωδικας του κεφαλαιου 7

Distance Sensor.wbt

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

#define NB_SENSORS 8
#define TIME_STEP  64
#define RANGE (1024 / 2)

int main(void)

{

  int i,j;
  WbDeviceTag ps[NB_SENSORS];
  char name[] = "ir0";
  double speed[2];
  double sensor_value[NB_SENSORS];
  double matrix[2][NB_SENSORS] = {
   {11, 12, 8, -2, -3, -5, -7, -9},
   {-9, -8, -5, -1, -2, 6, 12, 11}
  };

     wb_robot_init();

   for(i = 0; i < NB_SENSORS; i++)

  {
         ps[i] = wb_robot_get_device(name);

   wb_distance_sensor_enable(ps[i], TIME_STEP);

       name[2]++;

   }

     while (wb_robot_step(TIME_STEP)!=-1)
    {
          for (i = 0; i < NB_SENSORS; i++)
    {
           sensor_value[i] = wb_distance_sensor_get_value(ps[i]);
     }
           for (i = 0; i < 2; i++)
          {
              speed[i] = 0;

                for (j = 0; j < NB_SENSORS; j++) {

                      speed[i] += matrix[i][j] * (1 - (sensor_value[j] / RANGE));
     }
```

```
 }

  wb_differential_wheels_set_speed(2 * speed[0], 2 * speed[1]);
}

  wb_robot_cleanup();

    return 0;

  }
```

Κωδικας του κεφαλαιου 8

Force Sensor.wbt

```
#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/touch_sensor.h>

#define SPEED 40
#define TIME_STEP 64

int main()
{
  WbDeviceTag force;
  int movement_counter=0;
  int left_speed, right_speed;
  double force_value;

  wb_robot_init();

  force = wb_robot_get_device("force");

wb_touch_sensor_enable(force, TIME_STEP);

  while(wb_robot_step(TIME_STEP)!=-1)

{

    force_value = wb_touch_sensor_get_value(force);

       if (force_value > 0.01)
{
     printf("Detecting a collision of %g N\n", force_value);
     movement_counter = 15;
}

    if (movement_counter == 0)
{
     left_speed = SPEED;
     right_speed = SPEED;

}

 else if (movement_counter >= 7)

{
        left_speed = -SPEED;
        right_speed = -SPEED;
        movement_counter--;
    }
```

```
Else

{
      left_speed = -SPEED / 2;
      right_speed = SPEED;
      movement_counter--;

 }

   wb_differential_wheels_set_speed(left_speed, right_speed);

 }

  wb_robot_cleanup();

  return 0;

}
```

Matlab API

Οι ακόλουθοι πίνακες περιγράφουν τις λειτουργίες Matlab.

Accelerometer :

```
wb_accelerometer_enable(tag, ms)
wb_accelerometer_disable(tag)
[x y z] = wb_accelerometer_get_values(tag)
```

Camera :

```
WB CAMERA COLOR
WB CAMERA BLACK AND WHITE
WB CAMERA RANGE FINDER
wb_camera_enable(tag, ms)
wb_camera_disable(tag)
fov = wb_camera_get_fov(tag)
wb_camera_set_fov(tag, fov)
width = wb_camera_get_width(tag)
height = wb_camera_get_height(tag)
near = wb_camera_get_near(tag)
far = wb_camera_get_far(tag)
type = wb_camera_get_type(tag)
image = wb_camera_get_image(tag)
wb_camera_move_window(tag, x, y)
image = wb_camera_get_range_image(tag)
wb_camera_save_image(tag, 'filename', quality)
```

Compass :

```
wb_compass_enable(tag, ms)
wb_compass_disable(tag)
[x y z] = wb_compass_get_values(tag) % Connector :
wb_connector_enable_presence(tag, ms)
wb_connector_disable_presence(tag)
presence = wb_connector_get_presence(tag)
wb_connector_lock(tag)
wb_connector_unlock(tag)
```

DifferentialWheels :

wb_differential_wheels_set_speed(left, right)
wb_differential_wheels_enable_encoders(ms)
wb_differential_wheels_disable_encoders()
left = wb_differential_wheels_get_left_encoder()
right = wb_differential_wheels_get_right_encoder()
wb_differential_wheels_set_encoders(left, right)

Display :

width = wb_display_get_width(tag)
height = wb_display_get_height(tag)
wb_display_set_color(tag, [r g b])
wb_display_set_alpha(tag, alpha)
wb_display_set_opacity(tag, opacity)
wb_display_draw_pixel(tag, x, y)
wb_display_draw_line(tag, x1, y1, x2, y2)
wb_display_draw_rectangle(tag, x, y, width, height)
wb_display_draw_oval(tag, cx, cy, a, b)
wb_display_draw_polygon(tag, [x1 x2 ... xn], [y1 y2 ... yn])
wb_display_draw_text(tag, 'txt', x, y)
wb_display_fill_rectangle(tag, x, y, width, height)
wb_display_fill_oval(tag, cx, cy, a, b)
wb_display_fill_polygon(tag, [x1 x2 ... xn], [y1 y2 ... yn])
image = wb_display_image_copy(tag, x, y, width, height)
wb_display_image_paste(tag, image, x, y)
image = wb_display_image_load(tag, 'filename')
wb_display_image_save(tag, image, 'filename')
wb_display_image_delete(tag, image)

DistanceSensor :

wb_distance_sensor_enable(tag, ms)
wb_distance_sensor_disable(tag)
value = wb_distance_sensor_get_value(tag)

Emitter :

WB CHANNEL BROADCAST
wb_emitter_send(tag, data)
wb_emitter_set_channel(tag, channel)
channel = wb_emitter_get_channel(tag)
range = wb_emitter_get_range(tag)
wb_emitter_set_range(tag, range)
size = wb_emitter_get_buffer_size(tag)

GPS :

wb_gps_enable(tag, ms)
wb_gps_disable(tag)
[x y z] = wb_gps_get_values(tag)

Gripper :

wb_gripper_set_position(tag, position)
wb_gripper_enable_position(tag, ms)
wb_gripper_disable_position(tag)
position = wb_gripper_get_position(tag)
wb_gripper_enable_resistivity(tag, ms)
wb_gripper_disable_resistivity(tag)
resistivity = wb_gripper_get_resistivity(tag)

Gyro :

wb_gyro_enable(tag, ms)
wb_gyro_disable(tag)
[x y z] = wb_gyro_get_values(tag)

LED :

wb_led_set(tag, state)

LightSensor :

wb_light_sensor_enable(tag, ms)
wb_light_sensor_disable(tag)
value = wb_light_sensor_get_value(tag)

Motion :

motion = wbu_motion_new('filename')
wbu_motion_delete(motion)
wbu_motion_play(motion)
wbu_motion_stop(motion)
wbu_motion_set_loop(motion, loop)
wbu_motion_set_reverse(motion, reverse)
over = wbu_motion_is_over(motion)
duration = wbu_motion_get_duration(motion)
time = wbu_motion_get_time(motion)
wbu_motion_set_time(motion, time)

Node:

WB NODE NO NODE, WB NODE APPEARANCE,WB NODE BACKGROUND,
WB NODE BOX, WB NODE COLOR, WB NODE CONE,
WB NODE COORDINATE, WB NODE CYLINDER,
WB NODE DIRECTIONAL LIGHT, WB NODE ELEVATION GRID,
WB NODE EXTRUSION, WB NODE GROUP, WB NODE FOG,
WB NODE IMAGE TEXTURE, WB NODE INDEXED FACE SET,
WB NODE INDEXED LINE SET, WB NODE MATERIAL,
WB NODE POINT LIGHT, WB NODE SHAPE, WB NODE SPHERE,
WB NODE SWITCH, WB NODE TEXTURE COORDINATE,
WB NODE TEXTURE TRANSFORM, WB NODE TRANSFORM,
WB NODE VIEWPOINT, WB NODE WORLD INFO, WB NODE ROBOT,
WB NODE SUPERVISOR, WB NODE DIFFERENTIAL WHEELS,
WB NODE SOLID, WB NODE PHYSICS, WB NODE HYPER GATE,
WB NODE CHARGER, WB NODE DISTANCE SENSOR,
WB NODE CAMERA,
WB NODE EMITTER, WB NODE RECEIVER, WB NODE SERVO,
WB NODE TOUCH SENSOR, WB NODE LIGHT SENSOR,
WB NODE GRIPPER, WB NODE LED, WB NODE PEN, WB NODE GPS,
WB NODE RADIO, WB NODE CONNECTOR, WB NODE SPEAKER,
WB NODE MICROPHONE, WB NODE ACCELEROMETER, WB NODE GYRO,
WB NODE COMPASS

Pen :

wb_pen_write(tag, write)
wb_pen_set_ink_color(tag, [r g b], density)

Receiver :

WB CHANNEL BROADCAST
wb_receiver_enable(tag, ms)
wb_receiver_disable(tag)
length = wb_receiver_get_queue_length(tag)
wb_receiver_next_packet(tag)
size = wb_receiver_get_data_size(tag)
data = wb_receiver_get_data(tag)
strength = wb_receiver_get_signal_strength(tag)
[x y z] = wb_receiver_get_emitter_direction(tag)
wb_receiver_set_channel(tag, channel)
channel = wb_receiver_get_channel(tag)

Robot :

WB ROBOT KEYBOARD END
WB ROBOT KEYBOARD HOME
WB ROBOT KEYBOARD LEFT
WB ROBOT KEYBOARD UP
WB ROBOT KEYBOARD RIGHT
WB ROBOT KEYBOARD DOWN
WB ROBOT KEYBOARD PAGEUP
WB ROBOT KEYBOARD PAGEDOWN
WB ROBOT KEYBOARD NUMPAD HOME
WB ROBOT KEYBOARD NUMPAD LEFT
WB ROBOT KEYBOARD NUMPAD UP
WB ROBOT KEYBOARD NUMPAD RIGHT
WB ROBOT KEYBOARD NUMPAD DOWN
WB ROBOT KEYBOARD NUMPAD END
WB ROBOT KEYBOARD KEY
WB ROBOT KEYBOARD SHIFT
WB ROBOT KEYBOARD CONTROL
WB ROBOT KEYBOARD ALT
wb_robot_step(ms)
tag = wb_robot_get_device('name')
wb_robot_battery_sensor_enable(ms)
wb_robot_battery_sensor_disable()
value = wb_robot_battery_sensor_get_value()
step = wb_robot_get_basic_time_step()
mode = wb_robot_get_mode()
name = wb_robot_get_name()
path = wb_robot_get_project_path()
sync = wb_robot_get_synchronization()
time = wb_robot_get_time()
wb_robot_keyboard_enable(ms)
wb_robot_keyboard_disable()
key = wb_robot_keyboard_get_key()

Servo :

WB SERVO INFINITY
wb_servo_set_position(tag, position)
wb_servo_set_velocity(tag, vel)
wb_servo_set_acceleration(tag, acc)
wb_servo_set_motor_force(tag, force)
wb_servo_set_control_p(tag, p)
wb_servo_enable_position(tag, ms)
wb_servo_disable_position(tag)
position = wb_servo_get_position(tag)
wb_servo_enable_motor_force_feedback(tag, ms)
wb_servo_disable_motor_force_feedback(tag)
force = wb_servo_get_motor_force_feedback(tag)
wb_servo_set_force(tag, force)

Supervisor :

WB SF BOOL, WB SF INT32, WB SF FLOAT, WB SF VEC2F,
WB SF VEC3F, WB SF ROTATION, WB SF COLOR, WB SF STRING,
WB SF NODE, WB MF, WB MF BOOL, WB MF INT32, WB MF FLOAT,
WB MF VEC2F, WB MF VEC3F, WB MF ROTATION, WB MF COLOR,
WB MF STRING, WB MF NODE
wb_supervisor_export_image('filename', quality)
node = wb_supervisor_node_get_root()
node = wb_supervisor_node_get_from_def('def')
wb_supervisor_set_label(id, 'text', x, y, size, [r g b], transparency)
wb_supervisor_simulation_quit()
wb_supervisor_simulation_revert()
wb_supervisor_simulation_physics_reset()
wb_supervisor_start_animation('filename')
wb_supervisor_stop_animation()
wb_supervisor_start_movie('filename', width, height, type, quality)
wb_supervisor_stop_movie()
type = wb_supervisor_field_get_type(field)
name = wb_supervisor_field_get_type_name(field)
count = wb_supervisor_field_get_count(field)
b = wb_supervisor_field_get_sf_bool(field)
i = wb_supervisor_field_get_sf_int32(field)
f = wb_supervisor_field_get_sf_float(field)
[x y] = wb_supervisor_field_get_sf_vec2f(field)
[x y z] = wb_supervisor_field_get_sf_vec3f(field)
[x y z alpha] = wb_supervisor_field_get_sf_rotation(field)
[r g b] = wb_supervisor_field_get_sf_color(field)
s = wb_supervisor_field_get_sf_string(field)
node = wb_supervisor_field_get_sf_node(field)
b = wb_supervisor_field_get_mf_bool(field, index)
i = wb_supervisor_field_get_mf_int32(field, index)
f = wb_supervisor_field_get_mf_float(field, index)
[x y] = wb_supervisor_field_get_mf_vec2f(field, index)
[x y z] = wb_supervisor_field_get_mf_vec3f(field, index)
[x y z alpha] = wb_supervisor_field_get_mf_rotation(field, index)
[r g b] = wb_supervisor_field_get_mf_color(field, index)
s = wb_supervisor_field_get_mf_string(field, index)
node = wb_supervisor_field_get_mf_node(field, index)
wb_supervisor_field_set_sf_bool(field, value)
wb_supervisor_field_set_sf_int32(field, value)
wb_supervisor_field_set_sf_float(field, value)
wb_supervisor_field_set_sf_vec2f(field, [x y])
wb_supervisor_field_set_sf_vec3f(field, [x y z])
wb_supervisor_field_set_sf_rotation(field, [x y z alpha])
wb_supervisor_field_set_sf_color(field, [r g b])
wb_supervisor_field_set_sf_string(field, 'value')
wb_supervisor_field_set_mf_bool(field, index, value)
wb_supervisor_field_set_mf_int32(field, index, value)
wb_supervisor_field_set_mf_float(field, index, value)
wb_supervisor_field_set_mf_vec2f(field, index, [x y])
wb_supervisor_field_set_mf_vec3f(field, index, [x y z])
wb_supervisor_field_set_mf_rotation(field, index, [x y z alpha])
wb_supervisor_field_set_mf_color(field, index, [r g b])
wb_supervisor_field_set_mf_string(field, index, 'value')

```
wb_supervisor_field_import_mf_node(field, position, 'filename')
type = wb_supervisor_node_get_type(node)
name = wb_supervisor_node_get_name(node)
field = wb_supervisor_node_get_field(node, 'field name')
% TouchSensor :
wb_touch_sensor_enable(tag, ms)
wb_touch_sensor_disable(tag)
value = wb_touch_sensor_get_value(tag)
```

Βιβλιογραφία

Για τη δημιουργία αυτού του εγγράφου αντλήθηκαν πληροφορίες από ιστοσελίδες καθώς και από συγγράμματα τα οποία σας παραθέτουμε παρακάτω.

Συγγράμματα :

1. Webots User Guide Release 6.1.0 Copyright c 2009 Cyberbotics Ltd.
2. Webots User Guide Release 6.1.3 Copyright c 2009 Cyberbotics Ltd.
3. Webots User Guide Release 6.1.4 Copyright c 2009 Cyberbotics Ltd.
4. Webots Refearence Manual Release 6.1.0 Copyright c 2009 Cyberbotics Ltd.

Ιστοσελίδες :

1. http://www.cyberbotics.com/
2. http://www.cyberbotics.com/cdrom/common/doc/webots/guide/guide.html
3. http://en.wikipedia.org/wiki/Webots
4. https://www.iterasi.net/public/archive/1SlAGHhlTkqJJCnxVpZq-Q
5. http://www.gostai.com/urbi4webots.html
6. http://www.e-puck.org/
7. http://www.epuck.com/
8. http://www.activrobots.com
9. http://en.wikibooks.org/wiki/Cyberbotics%27_Robot_Curriculum/E-puck_and_Webots
10. http://en.wikipedia.org/wiki/Pioneer_2
11. http://www.activrobots.com/
12. http://www.pioneer2.net/