Κουρήτες, Nao Simulation Team

Ευστάθιος Βαφειάς

Κουρήτες, Nao Simulation Team

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευστάθιος Βαφειάς

Α.Μ: 2002010096, Email: svaf001@gmail.com

Εξεταστική Επιτροπή:

Ν. Βλάσσης, Επίκουρος Καθηγητής, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης (Επιβλέπων)

Β.Κουϊκόγλου, Καθηγητής, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης

Μ. Λαγουδάκης, Επίκουρος Καθηγητής, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

© 2009 Ευστάθιος Βαφειάς.

Εικόνα εξωφύλλου: Το Ναο της Aldebaran.

Κουρήτες, Nao Simulation Team

Περίληψη

Στην παρούσα εργασία εξετάζουμε την δημιουργία ενός συνόλου πρακτόρων που έχουν σκοπό να συντελέσουν μια ομάδα ποδοσφαίρου. Οι πράκτορες λειτουργούν εντός του περιβάλλοντος προσομοίωσης που παρέχει η Microsoft, το Microsoft Robotics Studio. Στην εργασία παρουσιάζουμε τον τρόπο που λειτουργεί μια service based εφαρμογή μέσα στο Robotics Studio και πως υλοποιείται ο παράλληλος προγραμματισμός που είναι απαραίτητος στις ρομποτικές εφαρμογές. Στη συνέχεια αναλύουμε τους αλγόριθμους που χρησιμοποιεί ο πράκτορας για να υλοποιήσει την όραση, την κίνηση και την λήψη αποφάσεων.

Περιεχόμενα

Περιεχόμενα **iii**

1 Εισαγωγή **1**

2 Robocup **5**
2.1 Simulation 7
2.2 Small Size League 9
2.3 Middle Size League 10
2.4 Standard Platform League 11
2.5 Humanoid League 12
2.6 Το Όραμα 12

3 Παρουσίαση του Microsoft Robotics Studio **13**
3.1 CCR-Concurrency and Coordination Runtime 13
3.2 DSS-Decentralized Software Services 14
3.3 Simulation 18

4 Κουρήτες, η ομάδα **23**
4.1 Συμπεριφορά 27
4.2 Cognition-Γνώση του περιβάλλοντος 28
4.3 Motion-Κίνηση 32
4.4 Decision-Απόφαση 38
4.5 Επικοινωνία 42

5 Αποτελέσματα – Προοπτικές **45**

Βιβλιογραφία **49**

Κεφάλαιο 1

Εισαγωγή

Η σκέψη είναι ένα από τα σημαντικά χαρακτηριστικά του ανθρώπου και η ποιότητα της σκέψης μας είναι αυτή που μας έδωσε την δυνατότητα να ξεχωρίσουμε από τους υπόλοιπους οργανισμούς στον πλανήτη. Για χιλιάδες χρόνια η διαδικασία της σκέψης συναρπάζει τους εκάστοτε πολιτισμούς και προσπαθούμε να αναλύσουμε την διαδικασία της σκέψης ώστε να καταλάβουμε το πως λειτουργεί. Η τεχνητή νοημοσύνη σαν επιστήμη δεν προσπαθεί απλά να καταλάβει το πως λειτουργεί η σκέψη προσπαθεί να την αναπαράγει με τεχνητά μέσα ώστε να δημιουργήσει σκεπτόμενες οντότητες.

Η τεχνητή νοημοσύνη σε συνδυασμό με την επιστήμη της μηχανικής απαρτίζει μια σύνθετη νέα επιστήμη, την ρομποτική. Η ρομποτικής με την μορφή που την αντιλαμβανόμαστε σήμερα είναι σχετικά μια νέα επιστήμη. Ο όρος ρομπότ χρησιμοποιήθηκε πρώτη φόρα λογοτεχνικά το 1938 στο έργο R.U.R. (Rossum's Universal Robots) του Dr Karel Čapek, αλλά τα πρώτα ρομπότ δεν εμφανίστηκαν πριν την δεκαετία του '50. Το πρώτο ρομπότ ήταν το Unimate από τον George Devol. Έκτοτε η τεχνολογία έχει εξελιχθεί τόσο σε επίπεδο μηχανικής όσο και σε επίπεδο τεχνητής νοημοσύνης.

Πλέον τα ρομπότ είναι σε θέση να αναγνωρίζουν αντικείμενα , να περπατάνε μιμούμενα κάθε μορφής οργανισμό, να αναγνωρίζουν φωνητικές εντολές και γενικά να εκτελούν πολύπλοκες εργασίες με μεγάλη ακρίβεια αλλά και να βρίσκουν λύσεις σε πολύπλοκα προβλήματα (πχ σκάκι). Παρότι η επιτυχία σε πολλά επίπεδα είναι εμφανής κρυφός σκοπός μεγάλων επιστημόνων του κλάδου είναι να δημιουργήσουν ένα αντίγραφο ανθρώπου, να έχει βούληση να μαθαίνει και να προσαρμόζεται αυτόνομα. Σε αυτά τα πλαίσια η διπλωματική αυτή περιγράφει την προσπάθεια να δημιουργήσουμε έναν αυτόνομα πράκτορα που θα μπορεί να ανταγωνιστεί σε ένα από τα πιο δημοφιλή αθλήματα, το ποδόσφαιρο.

Η διαδικασία του ποδοσφαίρου απαιτεί συνδυασμό πολλών ικανοτήτων από το ρομπότ ώστε να εκτελέσει επιτυχώς την λειτουργία αυτή. Χαρακτηριστικά αναφέρω τις βασικές λειτουργίες: όραση, κίνηση, αναγνώριση της θέσης, συμπερι-

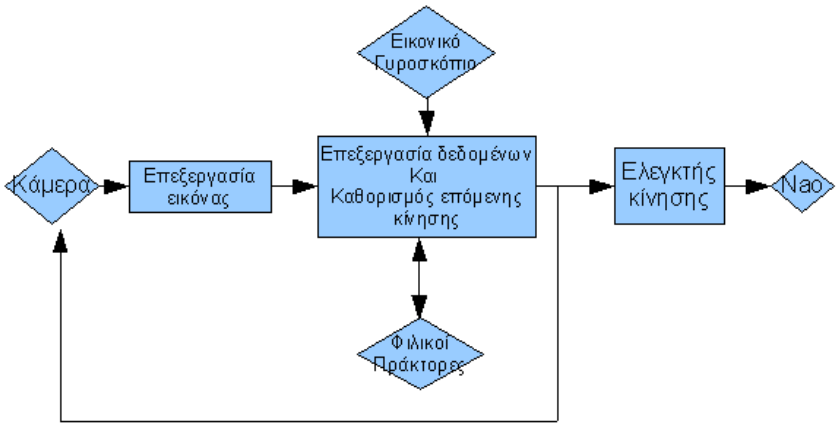
Σχήμα 1.1: Aldebaran's Nao

φορά, επικοινωνία με άλλους πράκτορες- συμπαίχτες. Ακόμα και οι πιο απλές κινήσεις ενός ανθρώπου είναι δύσκολο να γίνουν από ένα ρομπότ ειδικότερα όταν οι συνθήκες του περιβάλλοντος μεταβάλλονται συνεχώς.

Το ρομπότ που προσομοιώσαμε για την εργασία είναι το NAO της Aldebaran. Το NAO είναι ένα ανθρωπόμορφο ρομπότ, όπως φαίνεται στην εικόνα 1. Έχει 1 κάμερα στο μέτωπο, 25 βαθμούς ελευθερίας, ενσωματωμένο επεξεργαστή, μικρόφωνο, ηχεία και ασύρματη δικτύωση. Για τις ανάγκες της προσομοίωσης χρησιμοποιήθηκε το Microsoft Robotics Studio, το οποίο είναι μια πλατφόρμα για την δημιουργία ρομποτικών εφαρμογών. Ένα από τα πιο δυνατά εργαλεία του Microsoft Robotics Studio είναι η μηχανή προσομοίωσης, καταφέρνει να προσομοιώσει εύκολα και ρεαλιστικά τρισδιάστατα μοντέλα δίνοντας μάλιστα την δυνατότητα να εφαρμοστούν με ακρίβεια η πλειονότητα των φυσικών δυνάμεων που διέπουν τον πραγματικό κόσμο.

Θα κάνουμε μία μικρή περιγραφή για το πώς λειτουργεί το σύστημα πριν περιγράψουμε αναλυτικά την αρχιτεκτονική του πράκτορα και την επεξήγηση των

αλγορίθμων. Ο πράκτορας για να αντιληφθεί το περιβάλλον χρησιμοποιεί δύο εισόδους και την επικοινωνία με τους άλλους φιλικούς πράκτορες που είναι διαθέσιμοι. Η βασική είσοδος που αντλεί τις περισσότερες πληροφορίες είναι η κάμερα στο κεφάλι του ρομπότ, γύρω από την είσοδο αυτή είναι σχεδιασμένες οι λειτουργίες του πράκτορα. Κάθε ανανέωση του καρέ της κάμερας ενεργοποιεί τον αλγόριθμο της όρασης ώστε να ξεχωρίσει αντικείμενα στον χώρο, έπειτα ο πράκτορας επεξεργάζεται αυτά τα δεδομένα ώστε να προσεγγίσει αποστάσεις και γωνίες μεταξύ των αντικειμένων και του σώματος του ρομπότ. Έχοντας τα απαραίτητα δεδομένα τότε ο πράκτορας ελέγχει από το εικονικό γυροσκόπιο αν βρίσκεται σε όρθια θέση ή αν είναι σε θέση οριζόντια πράγμα που δεν του επιτρέπει να κάνει κινήσεις. Επόμενο βήμα του πράκτορα είναι να καθορίσει την συμπεριφορά του, αυτό έχει να κάνει με τον ρόλο του στο γήπεδο πχ τερματοφύλακας, αμυντικός επιθετικός αλλά και την συνεργασία του με τους άλλους παίχτες. Ακολουθεί ένα γράφημα που δείχνει την ροή των δεδομένων μέσα στο σύστημα.



Σχήμα 1.2: Διαγραμμα λειτουργίας

Η εργασία χωρίζεται σε τέσσερα βασικά κεφάλαια, το πρώτο κεφάλαιο είναι εισαγωγικό και αφορά την διοργάνωση Robocup και τα είδη διαγωνισμών που υπάρχουν. Στο δεύτερο κεφάλαιο παρουσιάζουμε το Microsoft Robotics Studio το οποίο αποτελεί και την βάση για το σύστημα μας. Στο κεφάλαιο για το MSRS εξηγούμε το μοντέλο εφαρμογών που υιοθετούμε αλλά και τα βασικά συστατικά για την δημιουργία ρομποτικών εφαρμογών εντός του MSRS. Στο τρίτο κεφάλαιο δίνουμε συγκεκριμένες πληροφορίες για την ομάδα των Κουρητών, το κεφάλαιο περιέχει επεξήγηση των αλγορίθμων που χρησιμοποιήθηκαν και αναλύουμε τον τρόπο όρασης, κίνησης και απόφασης. Στο τελευταίο μέρος παρουσιάζονται τα

αποτελέσματα της ομάδας στο Robocup 20008 αλλά και μελλοντικούς στόχους για την βελτίωση της.

Κεφάλαιο 2

Robocup

Το RoboCup είναι μια διεθνής οργάνωση που έχει ως σκοπό την εκπαίδευση και την έρευνα σε θέματα τεχνητής νοημοσύνης και ρομποτικής. Για την επίτευξη του στόχου αυτού χρησιμοποιήθηκε ένα τυποποιημένο πρόβλημα το οποίο μπορεί να προσεγγιστεί με μία μεγάλη γκάμα τεχνολογιών και μπορεί να αποτελέσει βάση για μάθηση μέσω έργων και πειραματισμού.

Για τον σκοπό αυτό το πρώτο πρόβλημα που επιλέχτηκε από την ομοσπονδία του RoboCup ήταν το παιχνίδι του ποδοσφαίρου. Το ποδόσφαιρο είναι απαιτητικό παιχνίδι σε διάφορα επίπεδα και οι ομάδες καλούνται να λύσουν προβλήματα του τύπου: συνεργασία πολλαπλών πρακτόρων, δημιουργία στρατηγικής, λήψη αποφάσεων σε πραγματικό χρόνο, αντιμετώπιση του θορύβου στους αισθητήρες, κινηματικά προβλήματα ρομποτικής και πολλά άλλα. Η φύση του προβλήματος είναι τέτοια που κάθε λύση μεταφέρει το πρόβλημα σε ένα επόμενο επίπεδο δυσκολίας.

Στον ετήσιο διαγωνισμό του RoboCup εκτός από το ποδόσφαιρο ομάδες διαγωνίζονται στο RoboCupRescue και RoboCup@Home. Το RoboCupRescue είναι ένας νέος τομέας όπου έχει σκοπό τις αποστολές εντοπισμού και διάσωσης σε μεγάλης κλίμακας καταστροφές και σε περιβάλλοντα που δεν είναι προσβάσιμα στον άνθρωπο. Κατά την διάρκεια του διαγωνισμού τα ρομποτικά οχήματα τοποθετούνται σε ειδικά διαμορφωμένες πίστες που μιμούνται δυσπρόσιτα σημεία και απαιτούν από τα οχήματα να εντοπίσουν και να διασώσουν "θύματα".

Το RoboCupRescue πραγματοποιείτε και σε simulation μορφή δίνοντας την δυνατότητα σε ομάδες που δεν έχουν τους οικονομικούς πόρους να κατασκευάσουν ένα όχημα ώστε να συμμετάσχουν στον διαγωνισμό.

Ο πιο πρόσφατος διαγωνισμός που προστέθηκε στο RoboCup είναι το @home Competition. Το @Home έχει στόχο να εισάγει την ρομποτική τεχνολογία στο σπίτι εκτελώντας βοηθητικές για τον άνθρωπο εργασίες. Με διάφορες δοκιμασίες βαθμολογείται η ικανότητα των ρομπότ να κινείται σε άγνωστο περιβάλλον κάνοντας κάποια λειτουργία σχετική με την καθημερινότητα διατηρώντας φιλική προσέγγιση

Σχήμα 2.1: Robocup Rescue Competition

Σχήμα 2.2: Robocup@home: Pattern Recognition

προς τον άνθρωπο.

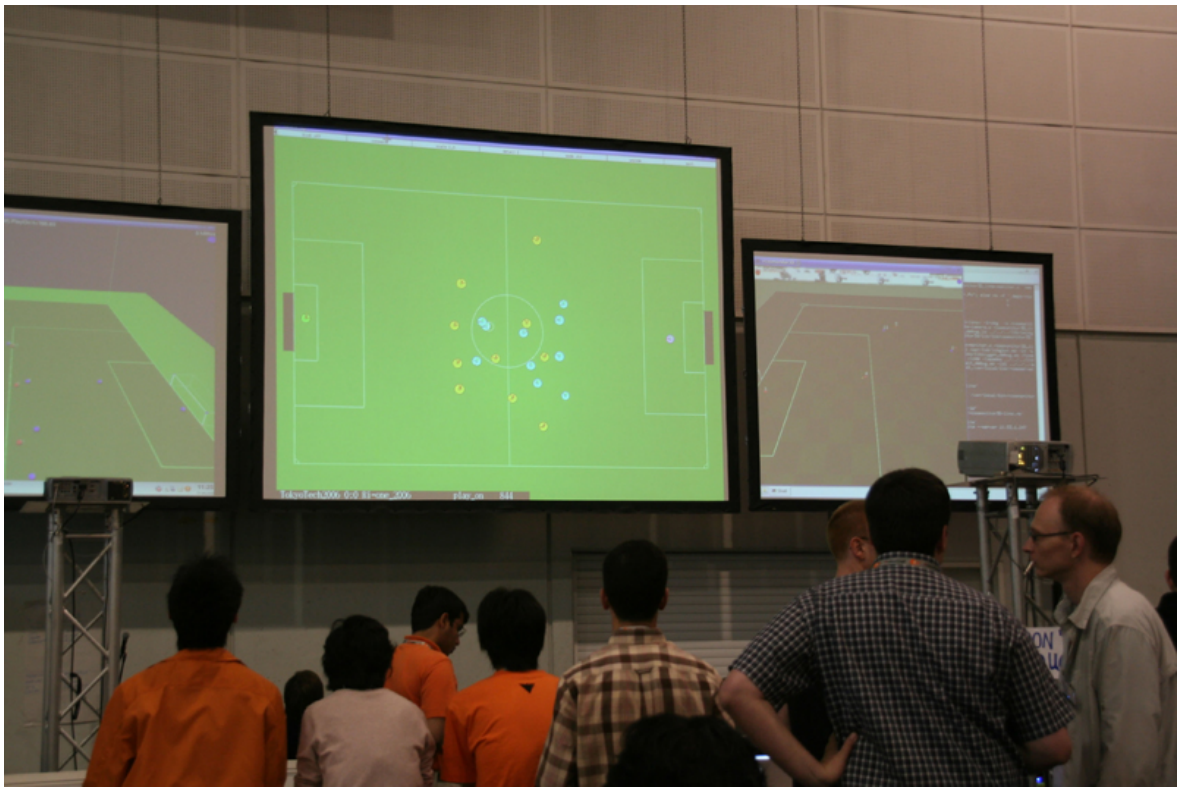
Η βασική κατηγορία του διαγωνισμού, το ποδόσφαιρο, χωρίζεται σε επιμέρους υποκατηγορίες όπου η κάθε μία επικεντρώνεται σε λίγο διαφορετικούς τομείς. Οι κατηγορίες είναι οι εξής:

- Simulation League
  - 2d
  - 3d
  - 3d development
  - Mixed reality
- Small Size Robot League
- Middle Size Robot League
- Standard Platform League
- Humanoid League

2.1 Simulation

Στην κατηγορία αυτή οι αγώνες γίνονται σε εικονικό περιβάλλον που ακολουθώντας δυο διαφορετικές προσεγγίσεις. Στις κατηγορίες 2d/mixed reality τα εικονικά ρομπότ είναι όμοια με αυτά των κατηγοριών Small size robot league & middle size robot league και κάθε ομάδα αποτελείται από 11 πράκτορες που βρίσκονται σε επικοινωνία με έναν server. Ο server γνωρίζει τις θέσεις των παιχτών και τις μπάλας και καθορίζει την λειτουργία του παιχνιδιού. Η ευκολία της κίνησης σε αυτού του είδους τα ρομπότ λύνει προβλήματα ισορροπίας και δίνει την δυνατότητα να δοθεί έμφαση σε ομαδικές στρατηγικές όπως αυτές που πραγματοποιούνται στο πραγματικό ποδόσφαιρο.

Στην κατηγορία του 3d simulation τα μοντέλα είναι αντίγραφα του NAO που χρησιμοποιείται και στο Standard Platform League, ενώ κάθε ομάδα αποτελείται από 4 ρομπότ (3 επιθετικούς, 1 τερματοφύλακα). Αντίθετα με το 2d simulation περισσότερο βάρος δίνεται σε χαμηλού επιπέδου λειτουργίες όπως περπάτημα, οπτική αναγνώριση περιβάλλοντος, κλωτσιά, αποκρούσεις κλπ. Γενικά οι ομάδες καλούνται να επιλύσουν τα ίδια προβλήματα με τα πραγματικά ρομπότ και αυτό καθιστά τους αλγόριθμους εφαρμόσιμους και στην πραγματικότητα.



Σχήμα 2.3: Αριστερά: Αγώνας 2d simulation. Δεξιά: 3d simulation

Σχήμα 2.4: 3d simulation.

2.2 Small Size League

Σε αυτήν την κατηγορία τα ρομπότ είναι κυλινδρικά με διάμετρο 18εκ και ύψος 15 εκ τα ρομπότ είναι τροχοφόρα ενώ ενδιαφέρον προκαλεί ότι δεν έχουν αισθητήρια όργανα και η αναγνώριση της θέσης τους και της θέσης της μπάλας υπολογίζεται από την εικόνα που λαμβάνει μια κάμερα πάνω από το γήπεδο. Για αναγνώριση των ρομπότ από την κάμερα κάθε ένα έχει ένα διαφορετικό συνδυασμό 4 χρωμάτων όπου αποτελεί την ταυτότητα του. Η κάθε ομάδα έχει 5 ρομπότ συμπεριλαμβανομένου του τερματοφύλακα, ενώ ο αγώνας διεξάγεται σε γήπεδο 6m x 4m

Σχήμα 2.5: Small Size Robot League

2.3 Middle Size League

Δυο ομάδες των 5 ρομπότ παίζουν σε ένα γήπεδο 18mx12m. Κάθε ρομπότ είναι εξοπλισμένο με ένα υπολογιστή και αισθητήρες και λειτουργεί χωρίς καμία καθοδήγηση από κάποιο κεντρικό σύστημα. Η επικοινωνία μεταξύ των παιχτών γίνεται ασύρματα αλλά απαγορεύεται η επικοινωνία με ανθρώπους της ομάδας. Τα ρομπότ αναπτύσσουν μεγάλες ταχύτητες και έχουν μεγάλη δύναμη για "κλώτσημα" της μπάλας πράγμα που κάνει το παιχνίδι πιο θεαματικό αλλά και πιο απαιτητικό από πλευράς κατασκευής και προγραμματισμού των ρομπότ.

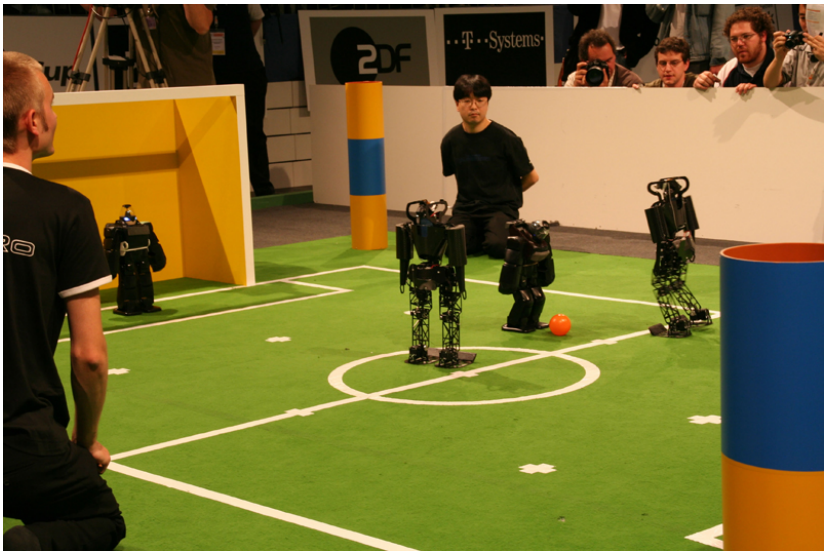
Σχήμα 2.6: Middle Size League

Αξίζει να σημειωθεί ότι μέχρι και το 2008 για την αναγνώριση του γηπέδου είχαν χρησιμοποιηθεί μόνο διαφορετικού χρώματος τέρματα ενώ από το 2009 και έπειτα τα τέρματα θα είναι λευκά, κάτι που φέρνει τον διαγωνισμό πιο κοντά σε πραγματικές συνθήκες.

2.4 Standard Platform League

Σε αυτήν την κατηγορία όλες οι ομάδες χρησιμοποιούν τον ίδιο τύπο ρομπότ το οποίο κατασκευάζεται από τρίτη ανεξάρτητη εταιρία. Μέχρι το 2008 επίσημο ρομπότ ήταν το Aibo της Sony(τετράποδο) ενώ το 2008 χρησιμοποιήθηκε το ανθρωπόμορφο ρομπότ της Aldebaran με την ονομασία Nao. Οι ομάδες επικεντρώνονται αποκλειστικά στο κομμάτι του προγραμματισμού, η τυποποίηση των ρομπότ που χρησιμοποιούνται εξασφαλίζει την νίκη σε αυτούς που έχουν γράψει τους πιο αποτελεσματικούς αλγόριθμους.

Σχήμα 2.7: SPL: Nao,Kouretes Team 2008



Σχήμα 2.8: Humanoid League

2.5 Humanoid League

Στο humanoid league υπάρχουν 2 υποκατηγορίες με βάση το μέγεθος των ρομπότ, από 30-60 εκ για Kid Size και από 100-160 εκ για Teen Size. Τα ρομπότ όπως δηλώνει το όνομα της κατηγορίας είναι ανθρωπόμορφα με τα ίδια αισθητήρια όργανα που είναι διαθέσιμα και στον άνθρωπο. Κάθε ομάδα κατασκευάζει τα δικά της ρομπότ δίνοντας βαρύτητα τόσο στην μηχανική υπεροχή των ρομπότ όσο και στην υπεροχή από πλευράς προγραμματισμού.

2.6 Το Όραμα

Το όραμα του RoboCup είναι:

> "***μέχρι το έτος 2050 να δημιουργηθεί μια ρομποτική ποδοσφαιρική ομάδα αυτόνομων πρακτόρων που θα μπορέσει να κερδίσει σε ένα αγώνα την εκάστοτε πρωταθλήτρια κόσμου.***"

Κεφάλαιο 3

Παρουσίαση του Microsoft Robotics Studio

Το Microsoft Robotics Studio ή MSRS είναι μια πλατφόρμα για την ανάπτυξη ρομποτικών εφαρμογών που συνεργάζονται με ένα μεγάλο εύρος hardware και παρέχει ένα πλήθος εργαλείων για την διευκόλυνση του προγραμματιστή. Το MSRS στηρίζεται πάνω σε δυο βασικές τεχνολογίες, η μία είναι η CCR(Concurrency and Coordination Runtime) και την DSS (Decentralized Software Services) που θα αναλύσουμε στην συνέχεια ώστε να καταλάβουμε πως λειτουργεί μια εφαρμογή πάνω στην πλατφόρμα MSRS. Ένα ακόμα δυνατό εργαλείο του MSRS είναι ο τρισδιάστατος εξομοιωτής του όπου παρέχει την δυνατότητα να εξομοιώνουμε καταστάσεις και να δούμε τις εφαρμογές να λειτουργούν σε εικονικό περιβάλλον πριν περάσουμε στην πραγματική εφαρμογή. Το περιβάλλον παρέχει μια σύγχρονη μηχανή φυσικής δίνοντας μας την δυνατότητα να προσεγγίσουμε συμπεριφορές που θα αντιμετωπίζαμε στον πραγματικό κόσμο. Η καλή προσέγγιση της πραγματικότητας κάνει το περιβάλλον προσομοίωσης ακόμα πιο χρήσιμο.

3.1 CCR-Concurrency and Coordination Runtime

Μια εφαρμογή όπως τα ρομπότ ακολουθεί ένα μοντέλο λειτουργίας που απαιτεί εισόδους, επεξεργασία των δεδομένων εισόδου, δράση και ανατροφοδότηση.

Το μοντέλο αυτό λειτουργεί όταν υπάρχει μια θύρα εισόδου, όταν όμως το σύστημα γίνεται πιο πολύπλοκο και το ρομπότ έχει δυο ή τρία διαφορετικά είδη εισόδου τότε καλούμαστε να λύσουμε το πρόβλημα του προγραμματισμού του χρόνου. Κάθε αισθητήρας οφείλει να μένει ενεργός τροφοδοτώντας πληροφορία χρήσιμη ή άχρηστη, αυτός ο τρόπος δημιουργεί διάφορα προβλήματα. Για παράδειγμα πρέπει να κρατάμε ένα βρόγχο ανοιχτό ώστε να ελέγχουμε συνεχώς τα νέα δεδομένα, από την στιγμή που προσθέσουμε ένα δεύτερο σένσορα πρέπει κάνουμε και ένα δεύτερο έλεγχο για τα δεδομένα της νέας εισόδου. Για κάθε νέο αισθη-

Σχήμα 3.1: Μοντέλο λειτουργίας

τήρας απαιτήται ένας νέος έλεγχος για τα δεδομένα που δεχόμαστε από αυτόν. Η βιβλιοθήκη CCR λύνει προβλήματα που αντιμετωπίζουμε σε συστήματα με πολλαπλούς αισθητήρες και διαδικασίες. Το CCR δίνει την δυνατότητα κάθε έλεγχο να τον καταχωρήσουμε σαν μια ξεχωριστή διαδικασία και έπειτα αναλαμβάνει τον διαμερισμό των υπολογιστικών πόρων ώστε οι διεργασίες αυτές να γίνονται παράλληλα.

CCR σημαίνει συγχρονισμός και συντονισμός του κώδικα που εκτελείται την δεδομένη στιγμή και αποτελείται από μια βιβλιοθήκη κλάσεων που αναλαμβάνουν τέτοιου είδους λειτουργίες. Οι βιβλιοθήκες περιέχουν τριών ειδών κλάσεις, τα Ports, τα arbiters, τα Dispatcher/DispatcherQueue και Task. Τα Ports είναι σειρές αντικειμένων τύπου FIFO(first in first out) και αποτελούν τους αποδέκτες των μηνυμάτων μέσα στο MSRS. Οι arbiters είναι μια κλάση που μας δίνει την δυνατότητα να εκτελέσουμε κώδικα όταν υπάρχουν συγκεκριμένες προϋποθέσεις, για παράδειγμα έχουμε ένα σύστημα από ένα σένσορ που ανταποκρίνεται σε επαφή εμείς ορίζουμε ένα arbiter στο σύστημα όπου τον προγραμματίζουμε να ενεργοποιηθεί με το που δεχτεί το μήνυμα επαφής. Από μέρους μας δεν χρειάζεται να πραγματοποιούμε τον έλεγχο του μηνύματος,το arbiter class του MSRS αναλαμβάνει να εκτελέσει την λειτουργία που έχουμε ορίσει όταν δεχτεί το μήνυμα στο συγκεκριμένο Port. Το Dispatcher / DispatcherQueue και Task classes είναι το μέρος του CCR που διαχειρίζεται τον υπολογιστικό φόρτο εργασίας. Ο χρήστης ορίζει την διεργασία που θέλει να εκτελέσει την στέλνει στην κλάση Dispatcher Queue και το Dispatcher είναι υπεύθυνο να διαμοιράσει τους πόρους του συστήματος. Το Dispatcher class μπορεί να συγχρονίζει πολλαπλές διεργασίες με επιτυχία.

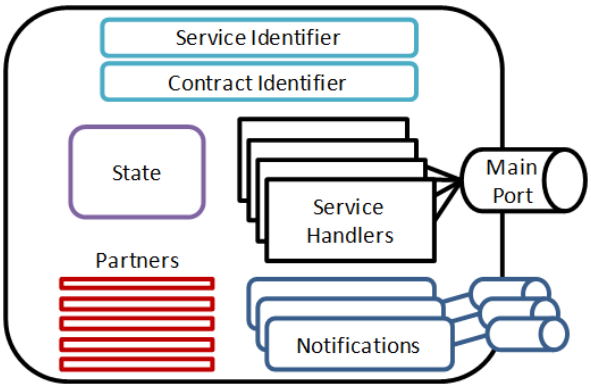
3.2 DSS-Decentralized Software Services

Τα services είναι ο ακρογωνιαίος λίθος του MSRS και του DSS μοντέλου, τα πάντα σε μία ρομποτική εφαρμογή αντιπροσωπεύουν ένα service χωρίς περιορισμό.

Ένα service μπορεί να είναι:

1. hardware οποιουδήποτε είδους
2. software όπως interfaces, αποθηκευτικός χώρος και άλλα
3. μια συλλογή αντικειμένων

Ένα service τρέχει σε ένα κόμβο και έχει την δυνατότητα επικοινωνίας με άλλα services του ίδιου ή άλλου κόμβου μέσω το γνωστού πρωτοκόλλου HTTP κάνοντας την διασύνδεση απομακρυσμένων services εφικτή σε μεγάλη κλίμακας αποστάσεις. Κάθε service επικοινωνεί χρησιμοποιώντας την class Port του CCR και έχει μορφή αντίστοιχη με το ακόλουθο σχήμα.



Σχήμα 3.2: Δομή Service

Το σχήμα[1] μας δείχνει τα μέρη ενός service και θα αναλύσουμε την λειτουργία και χρησιμότητα του κάθε μέρους ξεχωριστά.

Service Identifier Όταν δημιουργούμε ένα service στον κόμβο αυτό πρέπει να έχει μια ξεχωριστή ταυτότητα ώστε να διαχωρίζεται από τα άλλα service, και αυτό για ευνόητους λόγους. Κατά την εκκίνηση του service, το σύστημα αναθέτει δυναμικά ένα URI(Universal Resource Identifier) το οποίο αποτελεί την ταυτότητα του service.

Contract Identifier Το συγκεκριμένο πεδίο αποθηκεύει τον τύπο του συγκεκριμένου service ώστε ανατρέχοντας έπειτα στην βάση των καταχωρημένων Service γνωρίζουμε σε ποιο συγκεκριμένο είδος service μιλάμε και πώς μπορούμε να αλληλεπιδράσουμε με αυτό. Γνωρίζουμε τι είδους δεδομένα δέχεται και τι αποτελέσματα δίνει.

[1]Πηγή: Microsoft Robotics Studio Documentation

State Περιγράφει την κατάσταση του service, όπως είναι γνωστό από την θεωρία του προγραμματισμού μια κλάση αποτελείται από μεθόδους και μεταβλητές συγκεντρωμένες σε ένα αντικείμενο. Στα service θα λέγαμε ότι με το state αναφερόμαστε σε όλες της λειτουργικές μεταβλητές του αντικειμένου και τις τιμές που παίρνουν οι μεταβλητές αυτές. Πχ σε ένα service που εκπροσωπεί ένα κινητήρα θα έχουμε πληροφορίες όπως στροφές/λεπτό, θερμοκρασία, κατανάλωση καυσίμου και άλλες χρήσιμες πληροφορίες. Γενικά το state αποθηκεύει οποιαδήποτε πληροφορία μας είναι σημαντική.

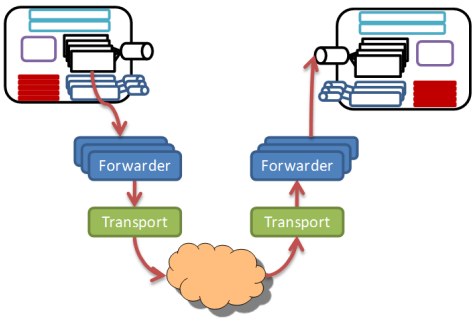
Partners Όπως είναι αναμενόμενο ένα service δεν λειτουργεί μόνο του, συνήθως συνεργάζεται με άλλα service ώστε να επιτευχθεί ο στόχος. Η λίστα των partners δηλώνει με ποία άλλα services αναμένεται να έχει συνεργασία η τρέχουσα υπηρεσία. Πολλές φορές μια υπηρεσία απαιτεί την ύπαρξη κάποιας άλλης ώστε να λειτουργήσει, τέτοιου είδους απαιτήσεις δηλώνονται στο partner list και ο κόμβος αναλαμβάνει την εκκίνηση των προαπαιτούμενων υπηρεσιών.

Main Port Το main port είναι ένα CCR port, είναι το κομμάτι που δέχεται μηνύματα από άλλες υπηρεσίες. Το main port δέχεται default μηνύματα του συστήματος όπως Lookup, Drop, Get, Replace, Post και πέρα από αυτό είναι ικανό να διαχειριστεί τύπους που έχουμε ορίσει εμείς.

Service Handlers Για κάθε τύπο μηνύματος που μπορεί να δεχτεί το main port το service έχει και ένα αντίστοιχο handler, κώδικα που δέχεται το μήνυμα και διαχειρίζεται την πληροφορία κατάλληλα.

Η σύνδεση των υπηρεσιών γίνεται με δύο διαφορετικούς τρόπους μηνύματος:

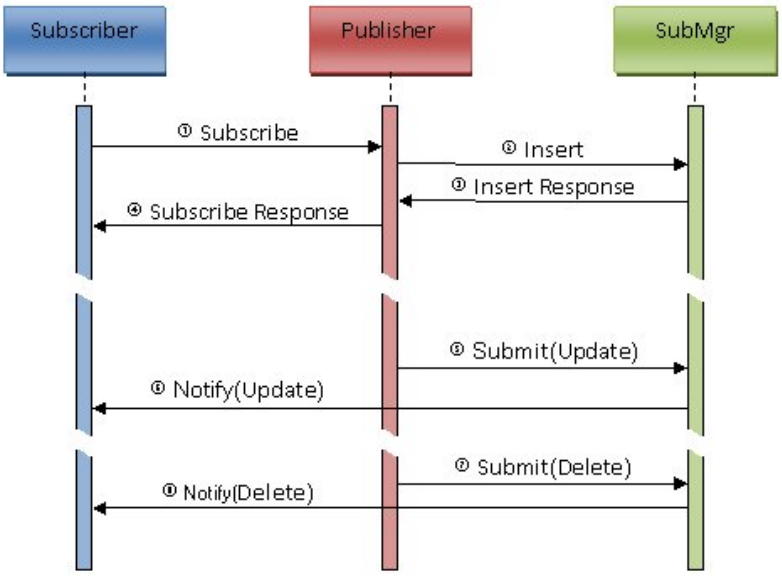
1. Απλό request μηνύματος που αποστέλλεται σε ένα service
2. Μήνυμα που ακολουθείται έπειτα από αλλαγή κατάστασης, όπου είναι μηνύματα ειδοποίησης (notification) έπειτα από εγγραφή στην συγκεκριμένη λειτουργία της υπηρεσίας (service subscription).



Σχήμα 3.3: Notification Forward

Event Notifications

Η εγγραφή σε υπηρεσίες είναι ένα ευέλικτος τρόπος να ενημερωνόμαστε μόνο όταν υπάρχει κάποια μεταβολή στην κατάσταση χωρίς να παίρνουμε περιττά μηνύματα ή να χρειάζεται να ανησυχούμε για το τι γίνεται στα συνεργαζόμενα service. Ουσιαστικά το ίδιο το service αναλαμβάνει να ενημερώσει τον όποιο ενδιαφερόμενο για την αλλαγή κατάστασης του, έτσι αναλαμβάνει να κάνει μια λίστα με τα σημαντικά events στα οποία μπορεί να εγγραφεί κάποια υπηρεσία.

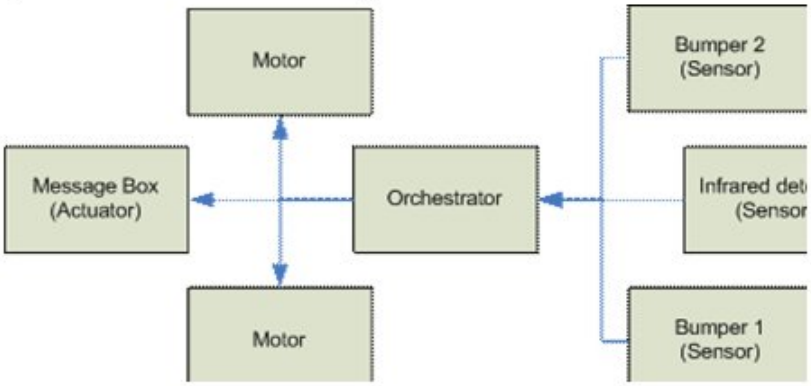


Σχήμα 3.4: Ροή των Subscriptions

Ο SubMgr (subscription manager) αναλαμβάνει να ρυθμίσει τα events που δημοσιεύονται στους αντίστοιχους εγγεγραμμένους. Ο τρόπος αυτός λειτουργίας κάνει ευκολότερη την παρακολούθηση πολλών services ταυτόχρονα και επομένως η ενορχήστρωση των services είναι απλούστερη και πιο αποτελεσματική.

3.2.1 Μοντέλο Εφαρμογών

Αφού γνωρίσαμε τα βασικά συστατικά μιας ρομποτικής εφαρμογής στο MSRS θα επεξηγήσουμε το μοντέλο μιας εφαρμογής. Ένα σύνολο από services συμμετέχουν και ενορχηστρώνονται ώστε να αποτελέσουν ένα λειτουργικό ρομπότ. Στο παραπάνω σχέδιο βλέπουμε ένα διάγραμμα που επιδεικνύει μια εφαρμογή με 2 αισθητήρες επαφής, 2 κινητήρες, 1 αισθητήρα υπέρυθρων, 1 οθόνη μηνυμάτων και τον βασικό ενορχηστρωτή όπου αναλαμβάνει την λήψη αποφάσεων και είναι ο πυρήνας της λογικής του συστήματος μας. Κάθε αισθητήρας είναι ένα service που εκκινείται από τον ενορχηστρωτή αλλά και ο ίδιος ο ενορχηστρωτής είναι ένα service.



Σχήμα 3.5: Μοντέλο Εφαρμογής

Όταν γίνεται η εκκίνηση του βασικού service ενορχηστρωτή αναλαμβάνει να εκκινήσει με την σειρά του τα απαραίτητα services και να δημιουργήσει συνάψεις μέσω partner lists και subscriptions στις υπηρεσίες. Το βασικό service κάνει εγγραφή στις υπηρεσίες των αισθητήρων ούτως ώστε να έχει update των καταστάσεων που τον ενδιαφέρουν ενώ με τους actuators υπάρχει απλή ανταλλαγή μηνυμάτων. Γραφικά περιβάλλοντα εργασίας για χρήστη είναι και αυτά ένα service που επικοινωνεί με την βασική υπηρεσία για να εκτελέσει λειτουργίες. Επειδή όπως προαναφέραμε οι υπηρεσίες επικοινωνούν μέσω HTTP ο έλεγχος μπορεί να γίνεται και απομακρυσμένα χωρίς επιπλέον προγραμματισμό, απλά ορίζοντας την IP του απομακρυσμένου κόμβου που θα συνδεθεί .Αυτός ο τρόπος σύνδεσης των υπηρεσιών δίνει δυνατότητα για διασύνδεση πολλαπλών ρομπότ που απλά συνδέονται σε κοινό δίκτυο κάνοντας το γνωστή την IP τους το ένα στο άλλο.

3.3 Simulation

Το simulation engine του MSRS είναι το τεχνητό τρισδιάστατο περιβάλλον εξομοίωσης που τρέχει η δικιά μας εφαρμογή. Είναι σημαντικό μέρος του robotic studio αφού επιτρέπει την δημιουργία πρωτότυπων εφαρμογών πριν εφαρμοστούν στην πραγματικότητα. Η μηχανή προσομοίωσης αποτελείται από 4 βασικά συστατικά:

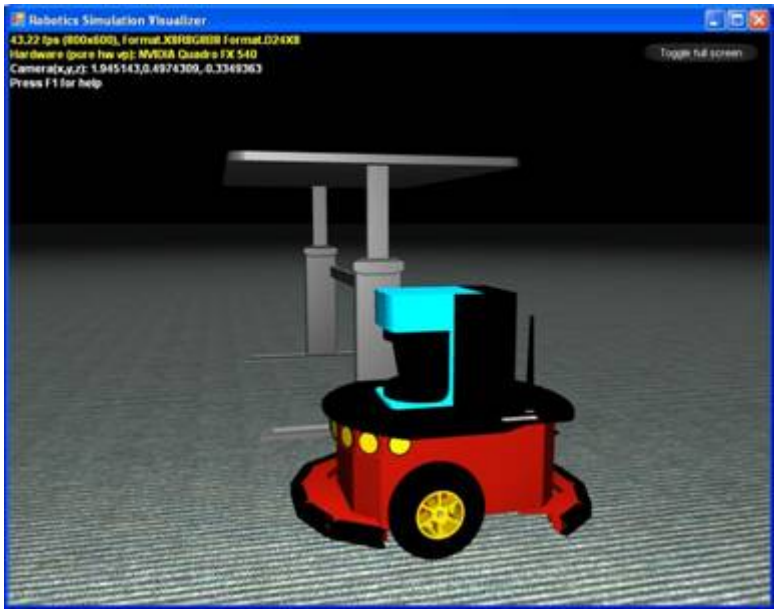
1. Simulation Service Engine: Είναι η υπηρεσία που είναι υπεύθυνη για την απεικόνιση των μονάδων και κρατάει τον χρονισμό της προσομοίωσης για την μηχανή φυσικής του συστήματος. Ουσιαστικά καταγράφει την κατάσταση κάθε σώματος που αναπαριστάται και αποτελεί την διασύνδεση με πρωτόκολλα DSS για τον προγραμματιστή.

2. Managed Physics Engine Wrapper: είναι ένα interface για τις χαμηλού επιπέδου διαδικασίες που πραγματοποιεί η μηχανή φυσικής.

3. Native Physics Engine Library: Αναλαμβάνει την επιτάχυνση της μηχανής φυσικής για υπολογιστές που διαθέτουν ειδικευμένο hardware.
4. Entities: Κάθε αντικείμενο στην προσομοίωση είναι ένα entity, μονάδες ρομποτικού hardware είναι και αυτές entities.

Για να εισάγουμε ένα λειτουργικό αντικείμενο στο περιβάλλον προσομοίωσης όπως παραδείγματος χάριν ένα ρομπότ, δημιουργούμε δυο αντικείμενα:

1. ένα entity το οποίο θα αντιπροσωπεύει το αντικείμενο στην προσομοίωση και είναι ο τρόπος που επικοινωνεί η μηχανή φυσικής και γραφικών με το αντικείμενο μας ώστε να το σχεδιάσει στην οθόνη αλλά και να εφαρμόσει σε αυτό τις απαραίτητες δυνάμεις.
2. Ένα Service που αναπαριστά τις λειτουργίες του εξαρτήματος που προσομοιώνουμε και ουσιαστικά είναι η διασύνδεση μεταξύ του χρήστη και του προσομοιωμένου μοντέλου.

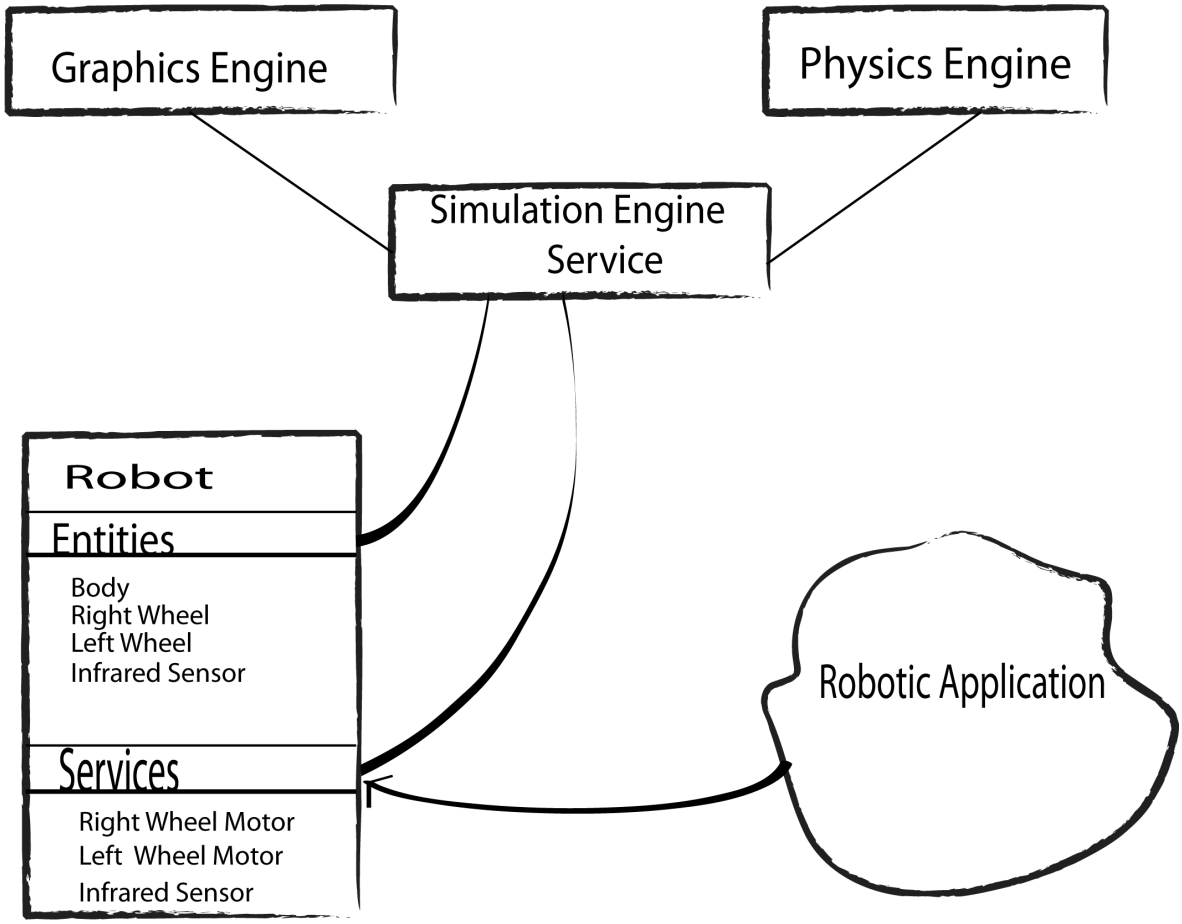


Σχήμα 3.6: Περιβάλλον προσομοίωσης

Για να κατανοήσουμε καλλίτερα το πώς λειτουργεί το MSRS στην παραπάνω εικόνα φαίνεται ένα ρομποτικό όχημα και ένα τραπέζι. Για την δημιουργία της συγκεκριμένης σκηνής έχουμε χρησιμοποιήσει ένα entity για το τραπέζι και ένα σύνολο entities για το ρομπότ, το ρομπότ το παρουσιάζουμε σαν σύνολο πολλών entities διότι περιέχει κινούμενα μέλη που θέλουμε να ελέγχουμε ξεχωριστά. Έτσι έχουμε το βασικό entity με δυο υπό μονάδες για τον δεξί και αριστερό τροχό, επίσης ένα entity ακόμα είναι για το infrared αισθητήρα πού υπάρχει πάνω στο

ρομπότ. Παρότι έχουμε entity για κάθε τι που εμφανίζεται στην οθόνη δεν είναι απαραίτητο να δημιουργήσουμε και το αντίστοιχο service για κάθε μέρος, εφόσον αυτό δεν έχει λειτουργική ιδιότητα. Έτσι στην συγκεκριμένη σκηνή έχουμε ένα service για κάθε τροχό(εννοώντας τον κινητήρα που είναι συνδεδεμένος στον τροχό) και ένα service για τον αισθητήρα υπερύθρων. Με τα service μπορούμε να δίνουμε εντολές προς εκτέλεση και να λαμβάνουμε δεδομένα από το περιβάλλον προσομοίωσης.

Για το περιβάλλον προσομοίωσης μας παρέχεται μια μεγάλη γκάμα προϋπαρχόντων αντικειμένων που κυκλοφορούν στο εμπόριο. Μας παρέχεται η δυνατότητα να εισάγουμε σαν αισθητήρα κάποιον του εμπορίου και αμέσως μας παρέχεται ένα service που μας επιτρέπει την επικοινωνία με αυτόν δίνοντάς μας δεδομένα που θα λαμβάναμε από τον πραγματικό αισθητήρα. Ουσιαστικά το service δεν γνωρίζει ότι συνδέεται με ένα εικονικό μέρος του ρομπότ, αυτού του είδους η λειτουργία είναι εξαιρετικά χρήσιμη και ευέλικτη αφού με τον ίδιο τρόπο ελέγχουμε και το πραγματικό και το εικονικό ρομπότ.

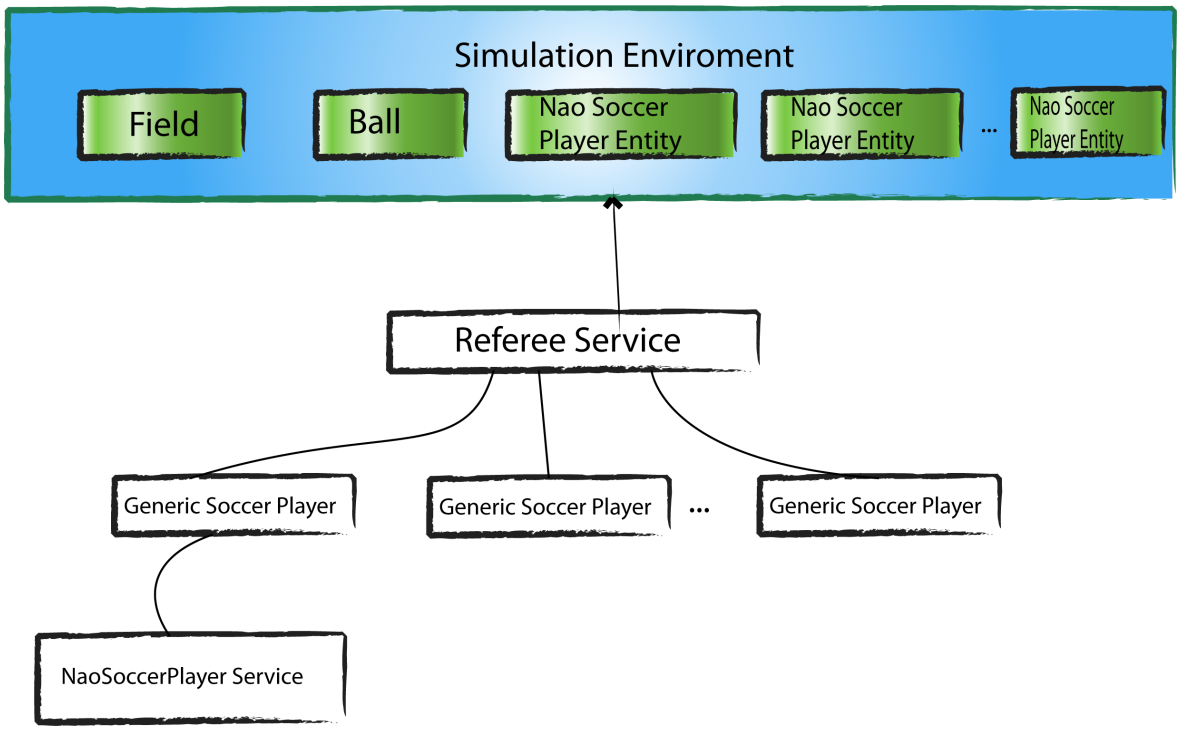


Σχήμα 3.7: Μοντέλο διασύνδεσης

Όπως φαίνεται και στο σχήμα η ρομποτικής μας εφαρμογή επικοινωνεί αποκλειστικά με τα services, και τα services με την σειρά τους επικοινωνούν με κάποια συσκευή που τρέχει στο περιβάλλον προσομοίωσης. Στην θέση του simulation engine service θα μπορούσε να ήταν πραγματικές συσκευές χωρίς διαφορές στο βασικό αλγόριθμο.

Φυσικά η προσομοίωση έχει και μειονεκτήματα τα οποία αν και σημαντικά δεν αναιρούν την χρησιμότητα του συστήματος. Βασικό μειονέκτημα είναι φυσικά η έλλειψη θορύβου στα δεδομένα εισόδου, για παράδειγμα μια φωτογραφία από κάμερα έχει καθαρά χρώματα με συγκεκριμένο φωτισμό σε περιορισμένο περιβάλλον, πράγμα που είναι φυσικά αδύνατο στον πραγματικό κόσμο. Αν δεν προβλέψουμε τέτοιου είδους θόρυβο τα δεδομένα εισόδου θα είναι πολύ μπερδεμένα ώστε να λάβουμε σωστή απόφαση. Το δεύτερο σημαντικό μειονέκτημα είναι ότι ο αληθινός κόσμος έχει ένα σύνολο απρόβλεπτων παραγόντων που μπορούν να επηρεάσουν το σύστημα. Αυτό καθιστά την προσομοίωση σαν μεθοδολογία για να ερευνηθούν μέρη του προβλήματος που καλούμαστε να λύσουμε αλλά και για την σχεδίαση πρωτότυπων εφαρμογών.

Για τους αγώνες ποδοσφαίρου σε περιβάλλον προσομοίωσης χρησιμοποιείται μια πιο σύνθετη διασύνδεση των παιχτών-services και του διαιτητή ώστε να διασφαλίζεται ότι κανείς παίχτης δεν έχει σύνδεση με το simulation service και έτσι δεν μπορεί να πάρει δεδομένα άμεσα από την μηχανή προσομοίωσης. Ο referee service είναι αυτός που αναλαμβάνει τον ρόλο αυτό.



Σχήμα 3.8: Referee service

Το referee service είναι μέρος του πακέτου της Microsoft για να υποστηρίζει τους αγώνες Robocup. Ο διαιτητής είναι το service που εκκινείται πρώτο από όλα τα άλλα και αναλαμβάνει την εκκίνηση και τον υπολοίπων. Αρχικά διαβάζει από ένα XML αρχείο τις παραμέτρους του παιχνιδιού όπως ο αριθμός των παιχτών, ο χρόνος του αγώνα κ.α. Όταν λάβει τις απαραίτητες πληροφορίες τότε αρχίζει το Simulation Service και εισάγει μέσα στο περιβάλλον όλα τα απαραίτητα entities(τέρματα, μπάλα, παίχτες, τάπητα). Για κάθε παίχτη που δημιουργεί ο διαιτητής εκκινεί και ένα NaoOperationsService το οποίο είναι ένα service που παρέχεται μαζί με το NAO ώστε να μπορούμε να ελέγχουμε ρομπότ. Έπειτα ο διαιτητής εκκινεί το NaoSoccerPlayerService που αποτελεί το service που ελέγχει την λογική του παίχτη και συνδέει το service αυτό με το αντίστοιχο NaoOperationsService που ελέγχει ένα NAO. Ξεχωριστά για κάθε ρομπότ δημιουργείται και ένα entity κάμερας και συνδέεται μέσω ενός WebCamOperations service με το NaoSoccerPlayer Service(ο παίχτης μας).

Πέρα από το έργο να δημιουργήσει τις οντότητες στην προσομοίωση και να συνδέσει τον κάθε παίχτη-service με τους αισθητήρες και τους σερβοκινητήρες του κάθε ρομπότ , έχει και την ευθύνη για την διεξαγωγή του αγώνα. Ο διαιτητής καταγράφει τον αγώνα και θέτει την κατάσταση του κάθε παίχτη, ο παίχτης μπορεί να μπει σε τρεις διαφορετικές καταστάσεις:

1. κατάσταση παιχνιδιού
2. κατάσταση τιμωρίας, ο παίχτης είναι σε φάση ακινησίας εκτός γηπέδου μέχρι να τον επαναφέρει ο διαιτητής
3. κατάσταση αναμονής

Επίσης ο διαιτητής ελέγχει την θέση της μπάλας ώστε να προσδιορίζει πότε μπαίνει γκολ ή πότε η μπάλα βγαίνει εκτός γηπέδου, και να την επαναφέρει στην θέση που ορίζουν οι κανονισμοί.

Από την στιγμή που ο διαιτητής δώσει το εναρκτήριο σήμα στους παίχτης-services τότε κάθε service αναλαμβάνει τον ρόλο του για να παίξει ποδόσφαιρο. Στο επόμενο μέρος θα δούμε από τι αποτελείται ένας παίχτης και τι αλγορίθμους χρησιμοποιεί η ομάδα των Κουρητών.

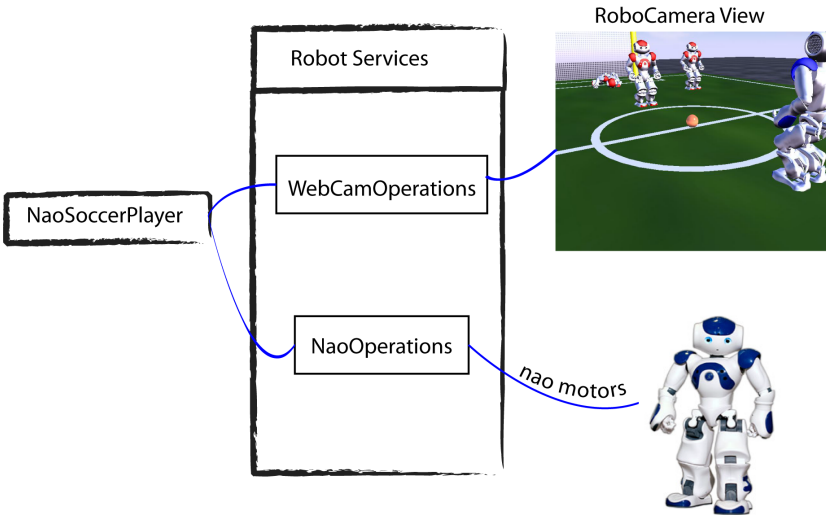
Σημείωση: Στο κεφάλαιο χρησιμοποιήθηκαν σχήματα απο το επίσημο documentation του Microsoft Robotics Studio.(http://msdn.microsoft.com/en-us/library/bb881626.aspx)

Κεφάλαιο 4

Κουρήτες, η ομάδα

Στο προηγούμενο κεφάλαιο περιγράψαμε πως διαρθρώνεται μια εφαρμογή στο MSRS ώστε να έχουμε μια εικόνα για το πώς λειτουργεί το σύστημα συνολικά. Σε αυτό το κεφάλαιο θα παρουσιάσουμε την δομή του NaoSoccerPlayer service και στην συνέχεια θα αναλύσουμε τους αλγόριθμους που χρησιμοποιήσαμε για την ομάδα των Κουρητών.

Μια οντότητα NAO στο σύστημα προσομοίωσης παρέχετε με 2 services για τον έλεγχο της, το πρώτο service, που δημιουργήθηκε από την Aldebaran, είναι το NaoOperations που μας δίνει τον έλεγχο του ρομπότ. Το δεύτερο service είναι ένα WebCamera Service που παρέχεται από Microsoft για την επικοινωνία με διαφόρων τύπου web cameras. Το WebCam service χρησιμοποιείται μόνο για να λάβουμε εικόνα από το ρομπότ, ενώ το NaoOperations service είναι πολύ πλουσιότερο σε λειτουργίες.

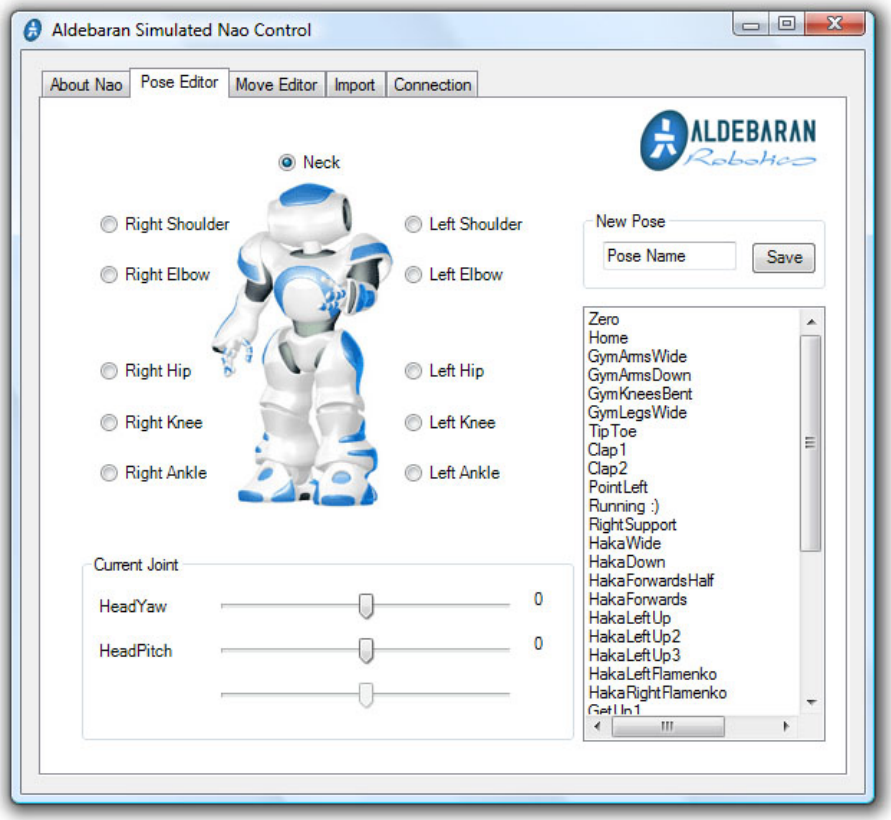


Σχήμα 4.1: Η διασύνδεση των services μέσα στον παίχτη

Ο έλεγχος των αρθρώσεων του ρομπότ γίνεται με τρεις διαφορετικούς τρόπους:

1. Άμεση ανάθεση κάποιας γωνίας σε μια συγκεκριμένη άρθρωση.
2. Την δημιουργία μιας πόζας κατά την οποία σε μια λίστα αρθρώσεων αντιστοιχούμε μια συγκεκριμένη τιμή. Αυτή την πόζα την αποθηκεύουμε σε μια βιβλιοθήκη που περιέχει κινήσεις και πόζες και έχουμε την δυνατότητα να την καλέσουμε στο μέλλον.
3. Την δημιουργία μιας ολοκληρωμένης κίνησης η οποία αποτελείται από μια σειρά διαδοχικών frames(πόζες). Η κίνηση αποθηκεύεται στην βιβλιοθήκη κινήσεων και εκτελείται κατά βούληση.

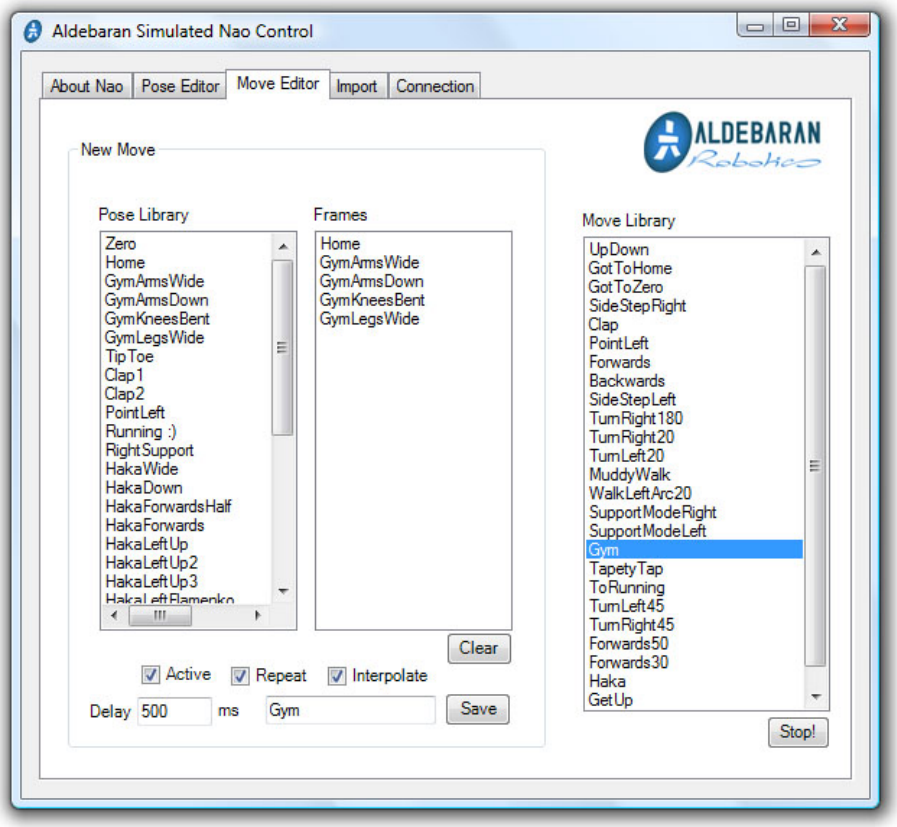
Μπορούμε να δημιουργήσουμε πόζες και κινήσεις μέσα στον κώδικα αλλά για την συγκεκριμένη εργασία μας παρέχεται ένα πιο εύχρηστο εργαλείο από την Aldebaran ο pose/move editor που φαίνεται στην παρακάτω εικόνα.



Σχήμα 4.2: Pose/Move editor

Στην καρτέλα "Pose Editor" μας επιτρέπεται να δώσουμε τιμές γωνίας σε κάθε γωνία ενώ παράλληλα βλέπουμε το αποτέλεσμα τις κίνησης στο περιβάλλον προ-

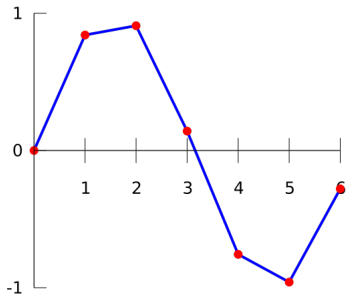
σομοίωσης. Δίνουμε ένα ξεχωριστό όνομα σε κάθε πόζα και πατώντας το save αποθηκεύεται στην βιβλιοθήκη του NAO.



Σχήμα 4.3: Move Editor

Αντίστοιχα στην "Move editor" καρτέλα που δείχνει η προηγούμενη φωτογραφία μας επιτρέπεται να συνθέσουμε της πόζες ώστε να δημιουργήσουμε μια νέα κίνηση. Από την βιβλιοθήκη κινήσεων εισάγουμε πόζες στην λίστα "frames" με την σειρά που θέλουμε να διαδεχτεί η μια πόζα την άλλη. Ο editor εκτός από την λίστα των "frames" δέχεται 4 ιδιότητες τις εξής: repeat, interpolate, delay και όνομα. Το repeat είναι μεταβλητή τύπου boolean και σημειώνει αν μια κίνηση είναι κίνηση που επαναλαμβάνεται άπειρες φορές η απλώς εκτελείται μια φορά. Η δεύτερη επιλογή είναι και αυτή τύπου boolean και ορίζει αν θα γίνει παρεμβολή μεταξύ των διαδοχικών γωνιών, αυτή η λειτουργία είναι και η πιο σημαντική και πάντα την θέλουμε ενεργοποιημένη. Η διαδικασία της παρεμβολής είναι ότι δίνουμε μια αρχική τιμή γωνίας και μία τελική τιμή γωνίας και τον χρόνο(delay) που από την μία τιμή πηγαίνει στην άλλη και το NaoOperations αναλαμβάνει να δημιουργήσει μια σειρά διαδοχικών τιμών. Αυτό μας επιτρέπει να ορίζουμε θέσεις κλειδιά που όταν συνδυάζονται γίνονται μια αρμονική κίνηση. Προς το παρών ο τύπος παρεμβολής που

χρησιμοποιείται είναι γραμμική παρεμβολή όπως φαίνεται στο παρακάτω σχήμα[1].

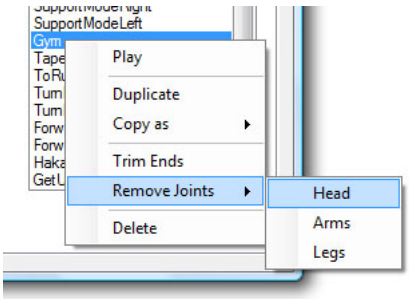


Στον άξονα y είναι η γωνία σε RAD και στον άξονα x ο χρόνος, οι ενδιάμεσες τιμές που παίρνει η άρθρωση περιγράφονται από την εξίσωση ευθείας:

$$y = y_a + (x - x_a)\frac{y_b - y_a}{x_b - x_a}$$

όπου x_a,y_a και x_b,y_b οι τιμές του πρώτου και του δεύτερου frame αντίστοιχα.Αργότερα θα μιλήσουμε και για εξυπνότερες μεθόδους που θα μπορούσαμε να εφαρμόσουμε για τον ίδιο σκοπό.

Μια σημαντική λειτουργία του "move editor" είναι η δυνατότητα του διαχωρισμού της κίνησης με κάποιες αρθρώσεις. Για παράδειγμα στην κίνηση του περπατήματος θέλουμε να έχουμε την ελευθερία να κουνάμε το κεφάλι ανεξάρτητα από το σώμα ώστε να μπορούμε να έχουμε οπτική επαφή με την μπάλα κάθε στιγμή. Για να το πραγματοποιήσουμε αυτό ορίζουμε ποιο σύνολο αρθρώσεων θέλουμε να εξαιρέσουμε όπως δείχνει η φωτογραφία. Τα σύνολα είναι: αρθρώσεις κεφαλιού, χεριών, ποδιών.

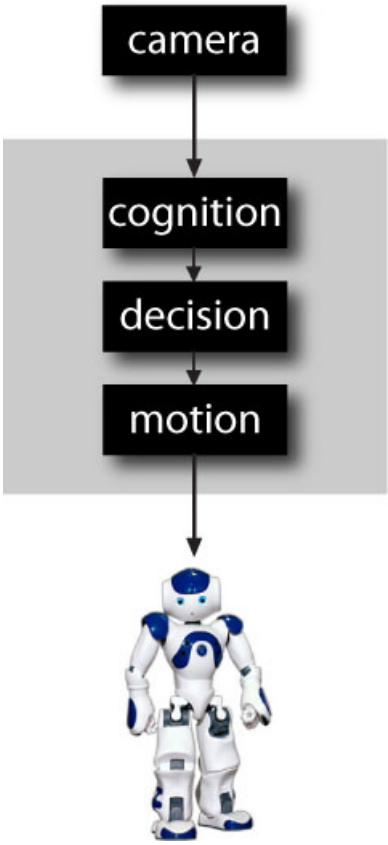


Σχήμα 4.4: Λειτουργία αφαίρεσης ομάδων αρθρώσεων από μια κίνηση.

[1]Πηγή: http://en.wikipedia.org/wiki/Interpolation

4.1 Συμπεριφορά

Με τον όρο συμπεριφορά ορίζουμε τις αποφάσεις που λαμβάνει το ρομπότ με βάση τα δεδομένα εισόδου. Η διαδικασία της απόφασης χωρίζεται σε πολλά διαφορετικά μέρη, στο δικό μας σύστημα η συμπεριφορά μας ορίζεται γύρω από τα δεδομένα που δίνει ο αισθητήρας τις κάμερας. Με κάθε νέο καρέ που στέλνει το WebCam service καλείται μια σειρά από ρουτίνες ώστε να προσδιορίσουν την δράση του ρομπότ. Ακολουθεί το συνοπτικό διάγραμμα αποφάσεων:



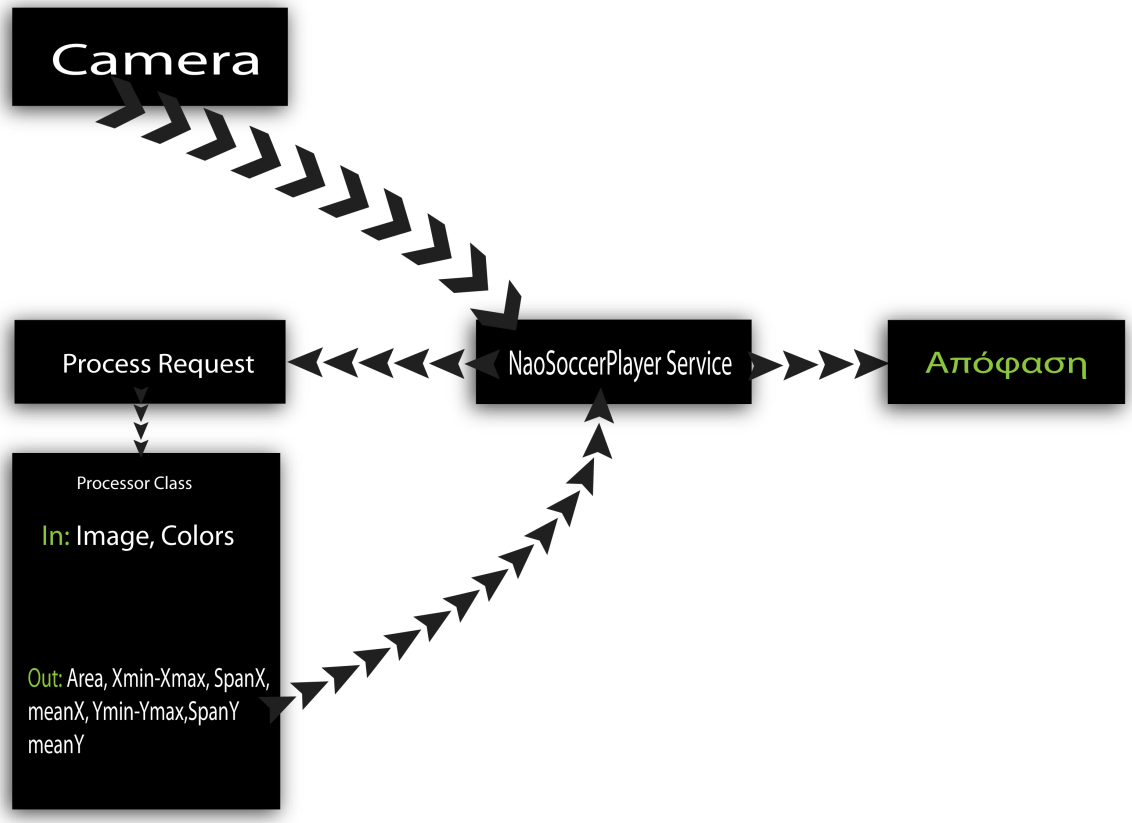
Σχήμα 4.5: Algorithm Flow

Διακρίνουμε ότι τα τρία βασικά μέρη του αλγορίθμου είναι:

1. cognition, είναι η αναγνώριση του περιβάλλοντα χώρου
2. decision, η απόφαση με βάση τα δεδομένα που έχουμε για το περιβάλλον αλλά και τον στόχο μας
3. motion ο προγραμματισμός για το πώς θα κινηθεί το ρομπότ αφού έχουμε λάβει την απόφαση για τι πρέπει να γίνει

4.2 Cognition-Γνώση του περιβάλλοντος

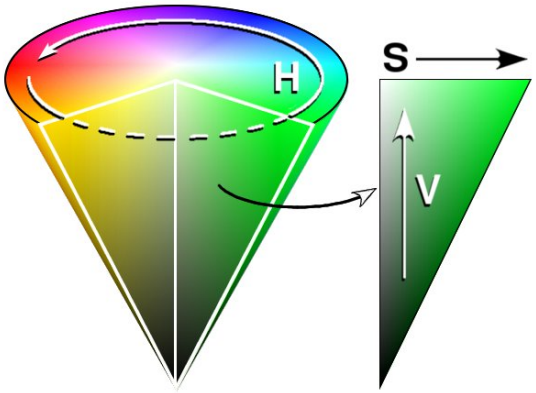
Η τρόπος με τον οποίο αποκτούμε γνώση για το περιβάλλον είναι μέσω της ψηφιακής κάμερας του NAO. Λαμβάνουμε ένα βίντεο το οποίο επεξεργαζόμαστε καρέ καρέ για να εντοπίσουμε αντικείμενα-κλειδιά μέσα στον χώρο. Ο εντοπισμός των αντικειμένων γίνεται με βάση τον χρωματικό τους διαχωρισμό. Σε πρώτη φάση τα ρομπότ είναι προγραμματισμένα να εντοπίζουν την μπάλα και τα τέρματα, και έπειτα προσπαθούν να υπολογίσουν την απόσταση και γωνία με την μπάλα.



Σχήμα 4.6: Διάγραμμα ροής για την όραση.

Το service μας λαμβάνει την εικόνα από την κάμερα, με το που συμβεί αυτό δημιουργεί ένα αίτημα(request) το οποίο στέλνει στον επεξεργαστή εικόνας για να αρχίσει την επεξεργασία. Το αίτημα είναι ένα πακέτο πληροφοριών που περιέχει την εικόνα που θέλουμε να επεξεργαστούμε και πληροφορίες για το πλάτος / ύψος της εικόνας και σε μια λίστα ζητάμε τα χρώματα που θέλουμε να εντοπίσουμε μέσα στην εικόνα.

Για να διακρίνουμε τα χρώματα χρησιμοποιούμε την μορφή χρωματικής κωδικοποίησης Hue/Saturation. Στο hue/saturation τα χρώματα παράγονται από τον συνδυασμό των ιδιοτήτων, το Hue δηλώνει το χρώμα πχ μπλε, κίτρινο, κόκκινο



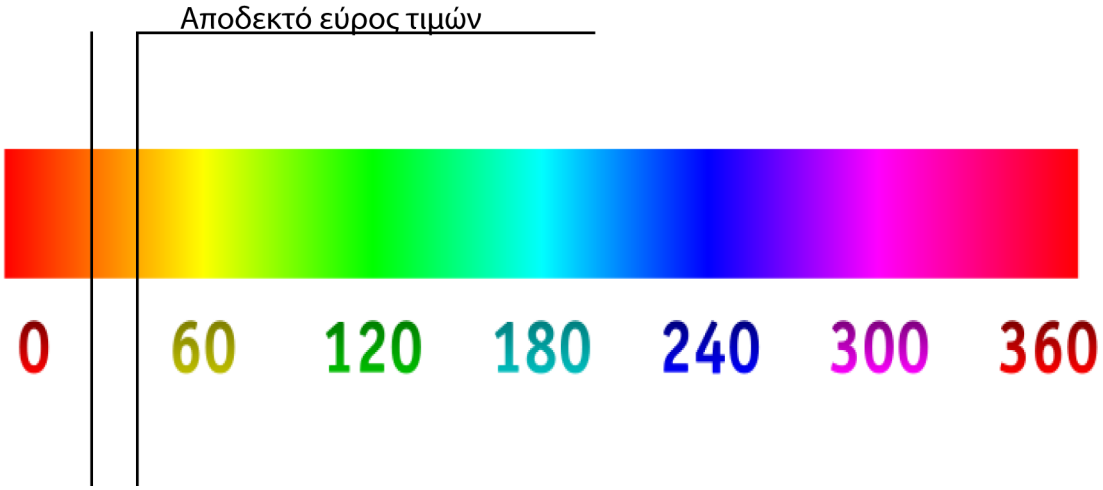
Σχήμα 4.7: Κώνος Hue/Saturation

κλπ ενώ το saturation δηλώνει το πόσο έντονο είναι το συγκεκριμένο χρώμα δηλαδή το πόσο κοντά είναι το χρώμα στο γκρι. Ο τρισδιάστατος κώνος παραπάνω μας δίνει να καταλάβουμε καλύτερα τον διαχωρισμό των χρωμάτων.

Όπως φαίνεται στην εικόνα[2] τα χρώματα μπαίνουν στην σειρά σε μορφή "ουράνιου τόξου" και έπειτα χωρίζονται σε 360 τιμές. Για κάθε τιμή hue αντιστοιχεί ένα ξεχωριστό χρώμα. Μπορούμε να παρατηρήσουμε ότι τα χρώματα που διαδέχονται το ένα το άλλο περνάνε σε μια φάση μίξης , το κόκκινο ακολουθείται από το κίτρινο έχοντας μια κατάσταση με ενδιάμεσες τιμές. Αυτού του είδους η κατάταξη είναι εξαιρετικά χρήσιμη για την διάκριση χρωμάτων που επιχειρούμε να κάνουμε στον αλγόριθμο μας. Ένα χρώμα ποτέ δεν μένει το ίδιο , εξαρτάται από τον φωτισμό του χώρου και τον φωτισμό που προέρχεται από την αντανάκλαση του φωτός στις γύρω επιφάνειες. Έτσι όταν μας ζητείται να εντοπίσουμε ένα χρώμα σε πολύ λίγες περιπτώσεις θα έχουμε ακριβώς τις ίδιες τιμές hue/saturation στο αντικείμενο που αναζητούμε με αποτέλεσμα να θεωρούμε ότι δεν υπάρχει. Συγκεκριμένα όταν θέλουμε να εντοπίσουμε την μπάλα η οποία έχει χρώμα πορτοκαλί , κάνουμε μια αναζήτηση για τα χρώμα που μπορεί να καταλήξει η μπάλα υπό διάφορες συνθήκες φωτισμού,σε κάποια σημεία μπορεί να είναι πιο κοκκινωπή ή κιτρινωπή. Ο τρόπος που είναι δομημένο το σύστημα HS μας βοηθάει, μπορούμε να ψάξουμε τιμές θέτοντας σαν κεντρική τιμή το ακριβές χρώμα που αναζητούμε και δίνοντας ένα περιθώριο στην τιμή hue, το ίδιο και στην ιδιότητα saturation θέτουμε μια ένταση συγκεκριμένη σαν κεντρική τιμή και δεχόμαστε τιμές μέσα σε ένα εύρος που ορίζουμε.

Καθώς ξεκινά η αναζήτηση για ένα χρώμα ο επεξεργαστής εικόνας ξεκινά τον έλεγχο από την πάνω αριστερή γωνία και και κινείται δεξιά μέχρι το τέλος διαβάζοντας το χρώμα κάθε pixel, φτάνοντας στο τέλος επαναλαμβάνει την διαδικασία για την επόμενη γραμμή και όλες τις υπόλοιπες. Την στιγμή που διαβάζει το χρώμα

[2]Πηγή: http://en.wikipedia.org/wiki/Hue



Σχήμα 4.8: Hue map

από την κάμερα το μετατρέπει από μορφή RGB σε HS και πραγματοποιεί έλεγχο για το αν ανήκει σε κάποιο χρώμα που αναζητούμε. Η πρώτη ενέργεια κατά τον έλεγχο είναι να ελέγξουμε αν το χρώμα είναι στο επιτρεπόμενο εύρος του hue και του saturation με ένα if :

```
If (( pixelHue-Hue < HueEpsilon)&(pixelSaturation - Saturation < SaturationEpsilon))
```

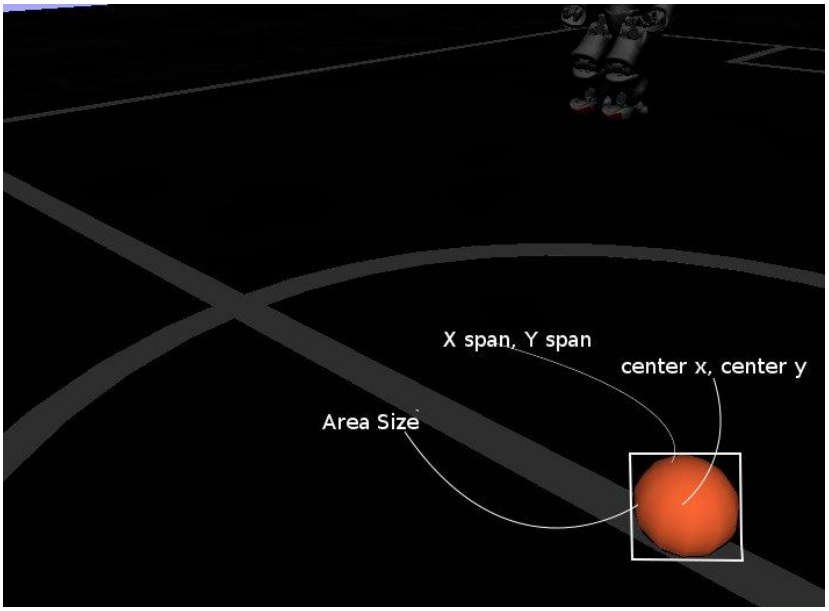
Η σύγκριση αυτή εξασφαλίζει ότι το χρώμα πλησιάζει αυτό που ψάχναμε αρχικά. Εφόσον η συνθήκη αποδειχθεί σωστή προχωράμε στην εξής διαδικασία:

1. Αυξάνουμε μια μεταβλητή με όνομα Area κατά μια μονάδα, η μεταβλητή αυτή καταγράφει τον αριθμό των pixel που έχουν βρεθεί θετικά στον έλεγχο.
2. Έπειτα βρίσκουμε και αποθηκεύουμε το μέγιστο και ελάχιστο X στα όποια έχουμε εντοπίσει χρώμα.
3. Το ίδιο κάνουμε για τον άξονα Y.

Η εικόνα 1.8 μας δείχνει τα στοιχεία που λαμβάνουμε μετά την επεξεργασία για το κάθε χρώμα. Παίρνουμε το πλήθος των pixel (μεταβλητή area) το πλάτος της περιοχής (x-span), το ύψος της περιοχής (y-span) και το μέσο της χρωματισμένης περιοχής (center-x, center-y).Τα δεδομένα αυτά χρησιμοποιούμε έπειτα ώστε να υπολογίσουμε την απόσταση προς κάθε αντικείμενο και την σχετική γωνία.

Γνωρίζοντας το κέντρο της χρωματισμένης περιοχής και με βάση αυτό υπολογίζουμε την γωνία(pitch) μεταξύ της κάθετης γωνίας που περνάει από το μέσο του άξονα χ και το κέντρο x της χρωματισμένης περιοχής. Υπολογίζουμε την γωνία αυτή με βάση τον τύπο:

$$CameraAngleX = 22.5 * \frac{centerx - (0.5 * ImageWidth)}{ImageWidth}$$

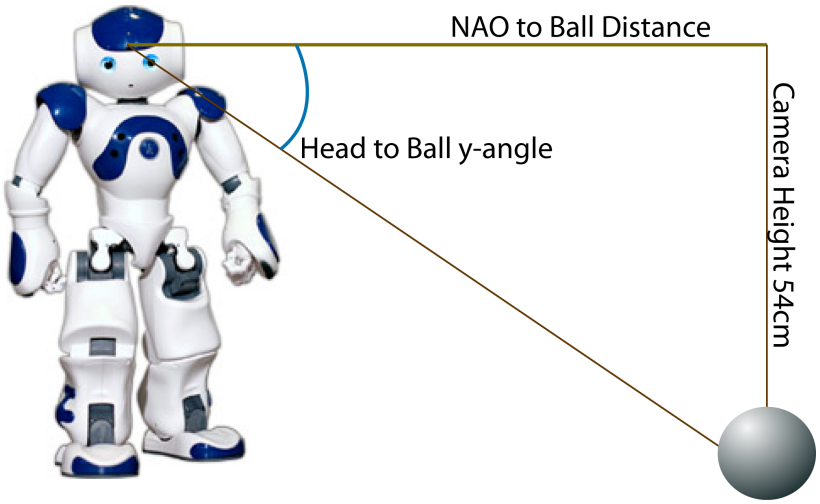


Σχήμα 4.9: Εξαγωγή του χρώματος που μας ενδιαφέρει.

Έπειτα για να βρούμε την γωνία(yaw) που σχηματίζεται μεταξύ της οριζόντιας ευθείας και την y συντεταγμένη του κέντρου της χρωματισμένης περιοχής.

$$CameraAngleY = 16.8 * \frac{centerY - (0.5 * ImageHeight)}{ImageHeight}$$

Οι δύο αυτές γωνίες είναι σημαντικές πληροφορίες για την θέση της μπάλας σχετικά με το σώμα του παίχτη. Όμως δεν είναι αρκετές, εξίσου σημαντική πληροφορία για την θέση της μπάλας είναι η απόσταση από το σώμα του παίχτη.



Σχήμα 4.10: Ball Distance

Μέσω αυτού του απλού τριγώνου μετράμε την απόσταση της μπάλας από τον παίχτη. Γνωρίζουμε το ύψος που βρίσκεται η κάμερα που είναι 54 εκατοστά και γνωρίζουμε και την γωνία του κεφαλιού προς την μπάλα, οπότε έχουμε:

$$distance = \frac{0.54}{\tan(yAngle)}$$

Η απόσταση ισούται με το ύψος προς την εφαπτομένη της γωνίας μεταξύ κεφαλιού και μπάλας. Να σημειώσουμε ότι η γωνία κεφαλιού – μπάλας δεν είναι η γωνία που βρήκαμε προηγουμένως, η γωνία που είχαμε βρει αφορούσε την κάμερα προς την μπάλα. Η συνολική γωνία είναι το άθροισμα της γωνίας μπάλας – κάμερας με την pitch γωνία στην άρθρωση του κεφαλιού. Το ίδιο ισχύει και για την γωνία στην οριζόντια κατεύθυνση x. Η γωνία είναι το άθροισμα της γωνίας της μπάλα – κάμερα συν τη γωνία YAW της άρθρωσης του κεφαλιού.

Με την ίδια διαδικασία εντοπίζουμε την απόσταση και την κατεύθυνση για τα τέρματα που αποτελούνται από διαφορετικά χρώματα, κίτρινο και γαλάζιο.

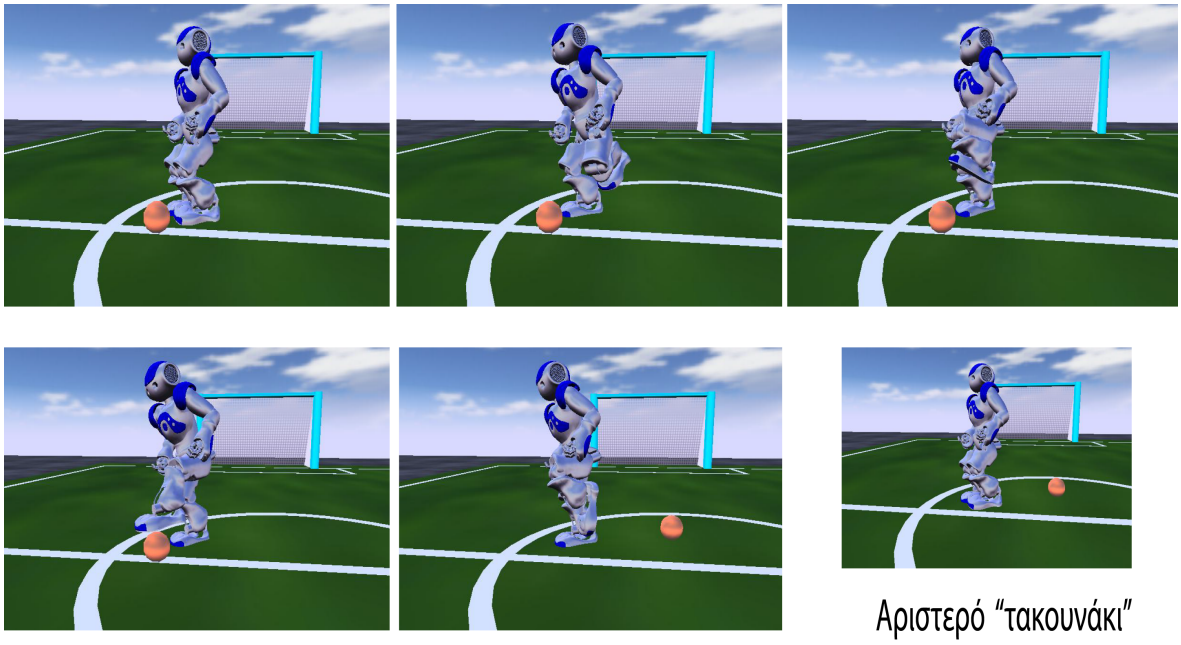
4.3 Motion-Κίνηση

Η κίνηση του NAO είναι ένας συνδυασμός προσχεδιασμένων κινήσεων και απευθείας ανάθεσης τιμών σε αρθρώσεις. Λόγω περιορισμένου χρόνου για την προετοιμασία της ομάδας να συμμετάσχει στο Robocup χρησιμοποιήθηκαν αποκλειστικά προσχεδιασμένες κινήσεις. Οι κινήσεις που φτιάξαμε χωρίζονται σε περπατήματα διαφόρων ειδών και σε ειδικές κινήσεις για να πραγματοποιούν λειτουργίες όπως κλώσημα της μπάλας, απόκρουση ή να σηκωθούν από το έδαφος μετά από πτώση. Ακολουθεί μια λίστα με τις κινήσεις του NAO:
Περπάτημα:

1. Μπροστά σε 3 διαφορετικές αποστάσεις 15εκ, 30εκ, 50εκ.
2. Αριστερή στροφή 20 μοιρών και 45 μοιρών
3. Δεξιά στροφή 20μοιρών και 45 μοιρών
4. Πίσω περπάτημα
5. Πλάγιος βηματισμός αριστερά
6. Πλάγιος βηματισμός δεξιά
7. Περιστροφή 180 μοιρών

Ειδικές κινήσεις:

1. Αριστερό και δεξί σουτ

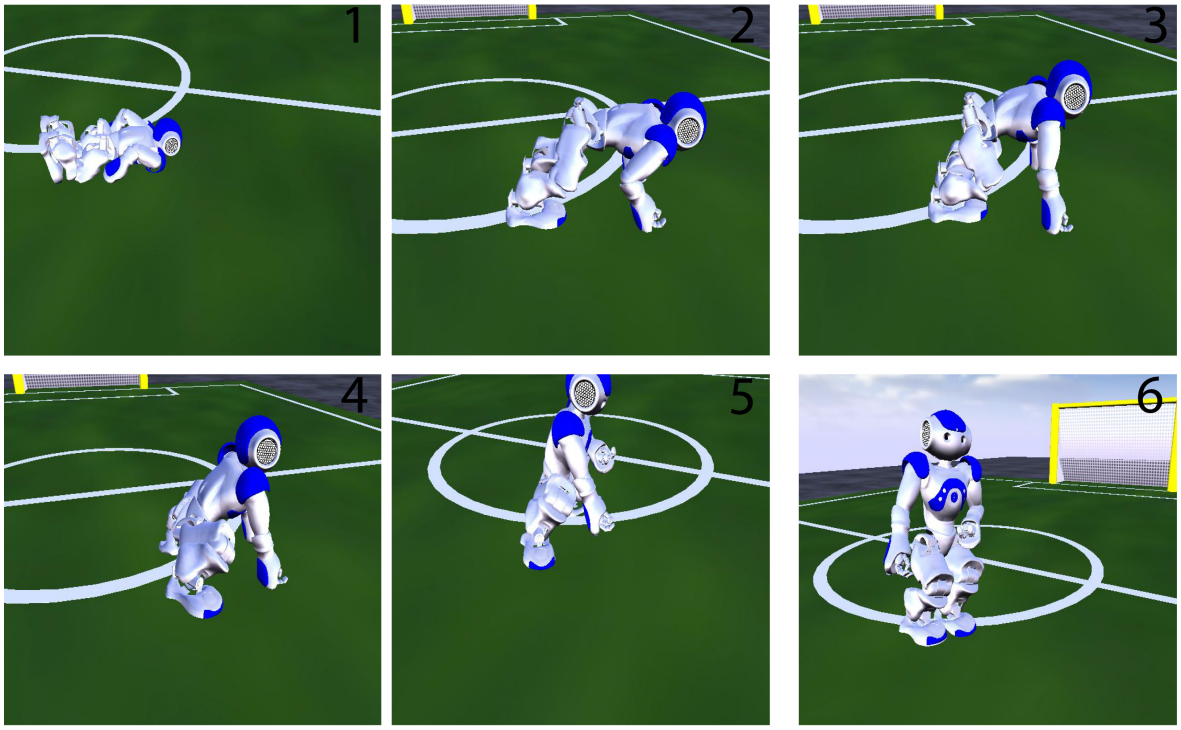


Σχήμα 4.11: Ειδική κίνηση 'Τακουνάκι'

2. Σήκωμα από το έδαφος από μπροστινή πλευρά
3. Γύρισμα από πίσω πλευρά σε μπροστινή ενώ είναι στο έδαφος
4. Απόκρουση τερματοφύλακα
5. Σήκωμα μετά την απόκρουση
6. Δεξί και αριστερό τακουνάκι, μια κίνηση για δίνει κατεύθυνση στην μπάλα αντίθετη προς αυτήν που κοιτάει το NAO

Για την δημιουργία των περπατημάτων χρησιμοποίησα σαν βάση το περπάτημα του NAO που παρείχε η Aldebaran βελτιώνοντας κυρίως την ταχύτητα των κινήσεων. Η βελτίωση της ταχύτητας των κινήσεων ήταν μια σχετικά απλή διαδικασία, με ένα script επεξεργάστηκα το XML αρχείο που αποθηκεύονται οι κινήσεις/πόζες του NAO. Η ιδέα ήταν να αφαιρέσω κάποια από τα frames που εμπεριέχονταν στην κίνηση χωρίς όμως να δημιουργώ αστάθεια στο περπάτημα. Δοκιμάζοντας διάφορες περιοδικότητες για την αφαίρεση των frame και καταγράφοντας τις διαφορές ταχυτήτων κατέληξα σε μια μείωση των frame γύρω στο 35%για τις εμπρόσθιες κινήσεις ενώ οι περιστροφικές είχαν μείωση γύρω στο 25%. Επειδή τα frame έχουν κοινό χρόνο εκτέλεσης αυτό συνεπάγεται ότι το περπάτημα έγινε 35%πιο γρήγορο ενώ οι περιστροφές 25%πιο γρήγορες.

Όλες οι ειδικές κινήσεις δημιουργήθηκαν από το μηδέν χρησιμοποιώντας τον Pose/Move editor της Aldebaran για να φτιαχτούν οι πόζες και να μπούνε στην



Σχήμα 4.12: Κίνηση Getup

κατάλληλη σειρά. Επειδή όμως ο Pose/Move Editor θέτει κοινή καθυστέρηση μεταξύ διαδοχικών ποζών οι τιμές καθυστέρησης επεξεργάστηκαν χειροκίνητα, μέσω ενός απλού XML editor, για να επιτευχθούν ομαλότερες και αποτελεσματικότερες κινήσεις.

4.3.1 Σύγκριση

Το 2008 ήταν η πρώτη χρονιά που έγινε Simulation League με NAO και όπως είναι φυσικό οι ομάδες δεν είχαν αναπτύξει πολύπλοκες στρατηγικές και το παιχνίδι στηριζόταν περισσότερο στην τακτική να φτάσεις πρώτος στην μπάλα και να κάνεις δυνατά σουτ προς την σωστή κατεύθυνση. Σε μια τέτοια κατάσταση ο πιο σημαντικοί παράγοντες είναι η ταχύτητα και η ακρίβεια των κινήσεων. Κατά την διάρκεια του διαγωνισμού οι Κουρήτες είχαν το γρηγορότερο περπάτημα από όλες τις ομάδες, ενώ τα σηκώματα από το έδαφος είχαν το μεγαλύτερο ποσοστό επιτυχίας. Μια ακόμα μεγάλη επιτυχία ήταν η αποτελεσματική απόκρουση του τερματοφύλακα, ο τερματοφύλακας έπαιρνε σωστή θέση προς την μπάλα ενώ όταν αυτή πλησίαζε εκτελούσε μια κίνηση που κάλυπτε περίπου το ½ της περιοχής του τέρματος κάνοντας δύσκολο το σκοράρισμα από την αντίπαλη ομάδα. Χαρακτηριστικά αναφέρουμε ότι η ομάδα στο τουρνουά δέχτηκε μόνο ένα γκολ. Όμως η κίνηση που ίσως έδωσε το μεγαλύτερο πλεονέκτημα στους Κουρήτες ήταν το

τακουνάκι καθώς επέτρεπε στην ομάδα να περάσει από την άμυνα στην επίθεση γρηγορότερα από κάθε άλλη ομάδα, ήταν μια κίνηση που έδινε πολύ γρήγορα αντίθετη κατεύθυνση στην μπάλα. Αποτέλεσμα της χρήσης της κίνησης είναι να αναγκάσει τους αντιπάλους να ανατρέχουν πάντα στην άμυνα τους ώστε να ανακτήσουν την μπάλα ενώ έφερνε τους Κουρήτες σε θέση επίθεσης.

4.3.2 Εκτέλεση Κίνησης

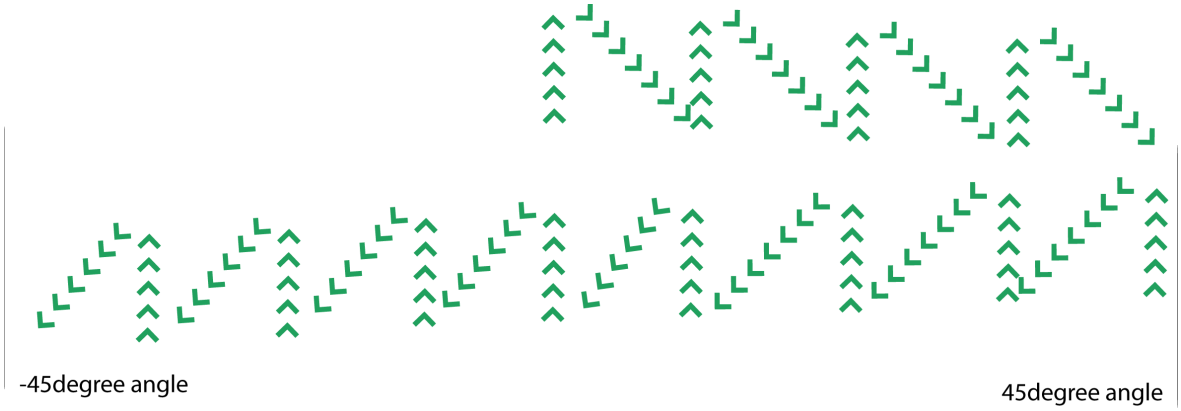
Για την εκτέλεση μιας κίνησης χρησιμοποιούμε μια συνάρτηση που λέγεται SendWalkMove. Η συνάρτηση έχει σαν όρισμα το όνομα της κίνησης που θέλουμε να εκτελέσουμε:

```
SendWalkMove("MoveName");
```

Η συνάρτηση αυτή σαν πρώτο βήμα σημειώνει στην κατάσταση του service ότι το NAO βρίσκεται σε κίνηση και έπειτα καταγράφει το όνομα της κίνησης. Αυτές οι δύο μεταβλητές βοηθούν κατά την διάρκεια λήψης αποφάσεων να ξέρουμε αν βρίσκεται σε κίνηση το ρομπότ και τι είδους κίνηση πραγματοποιεί δίνοντας μας την δυνατότητα να την τερματίσουμε αν κρίνουμε ότι δεν είναι κατάλληλη. Αφού ανανεώσει το state του service η συνάρτηση δημιουργεί ένα αίτημα κίνησης που περιέχει το όνομα της κίνησης, το να στείλουμε όμως απλά το μήνυμα στο NAO δεν αρκεί, χρειαζόμαστε να ξέρουμε το αποτέλεσμα του αιτήματος χωρίς να σταματήσουμε την υπόλοιπη λειτουργία του NAO. Για να στείλουμε λοιπόν το μήνυμα ενεργοποιούμε ένα Arbiter class τύπου choice, μέσω του Arbiter class μας επιτρέπεται να δημιουργήσουμε μια νέα διεργασία(task) το οποίο στέλνεται στον CCR task Manager όπου του επιτρέπει να τρέχει ταυτόχρονα με το υπόλοιπο μέρος του κώδικα. Το arbiter.choice class στέλνει το πακέτο του μηνύματος και παραμένει ενεργή διαδικασία περιμένοντας απάντηση από το NAO, το NAO απαντά με "success" σε περίπτωση που το αίτημα είναι επιτυχές ενώ σε περίπτωση αποτυχίας επιστρέφει μήνυμα λάθους. Το arbiter εκτελεί κώδικα που του έχουμε αναθέσει ανάλογα την απάντηση που θα δεχτεί από το NAO. Με αυτό τον τρόπο είμαστε σε θέση να γνωρίζουμε αν εκπληρώθηκε η κίνηση και να αλλάξουμε την κατάσταση του service σε ακίνητο .

Ενώ το σώμα κινείται μέσω της ρουτίνας SendWalkMove και προσχεδιασμένων κινήσεων, η κίνηση του κεφαλιού γίνεται με διαφορετικό τρόπο. Καθώς το κεφάλι αποτελεί το σημείο που βρίσκεται η κάμερα, απαιτήται να είναι περισσότερο κινητικό ώστε να διευρύνει το οπτικό πεδίο και να εντοπίζει ευκολότερα την μπάλα, αλλά και να είναι σε θέση μετά τον εντοπισμό να την κρατήσει στο στόχαστρο τις κάμερας.

Για κάθε ένα από τους σκοπούς αυτούς υπάρχει και μια ρουτίνα που καλείται ανάλογα με την περίσταση. Η συνάρτηση PanHead() έχει ρόλο να περιστρέφει το κεφάλι κάνοντας ένα σκανάρισμα τις περιοχής μέχρι να εντοπιστεί η μπάλα. Ενώ η συνάρτηση LookHead() αναλαμβάνει να κρατάει την μπάλα μέσα στο κάδρο της



Σχήμα 4.13: Pan Head Trail

κάμερας αφού αυτή εντοπιστεί.

Η εικόνα δείχνει την κίνηση που ακολουθεί το κεφάλι. Περιστρέφεται μέχρι $\pm 45^o$ γωνία (yaw) ενώ η κίνηση πάνω-κάτω (pitch) έχει μόνο 3° ελευθερία στην κίνηση. Για να εκτελέσουμε την περιοδική κίνηση αυτή υπολογίζουμε την γωνία με τον τύπο:

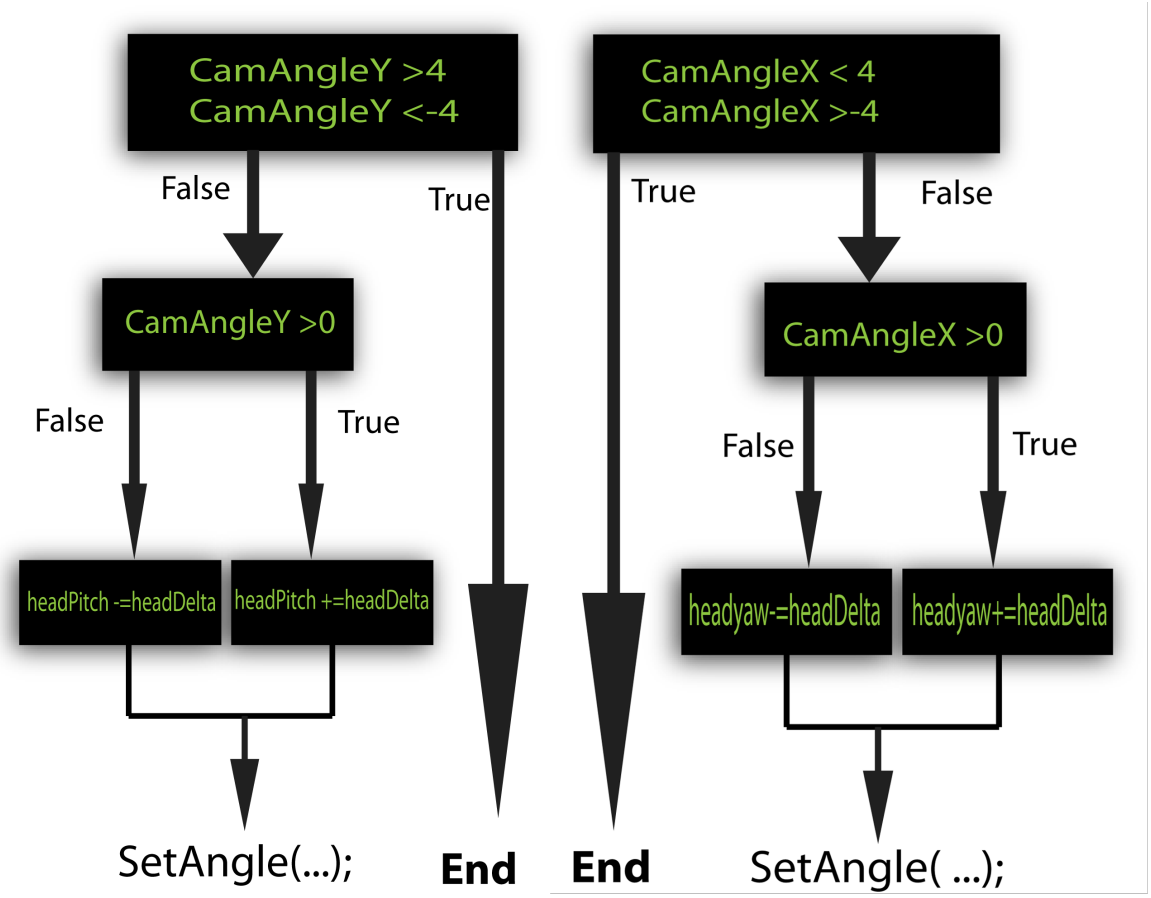
$$\text{HeadYaw} = \text{HeadYaw} + \text{HeadDelta} * \text{Direction}$$

YawDelta είναι το βήμα της αλλαγής γωνίας και Direction είναι η κατεύθυνση της κίνησης . Όταν το κεφάλι φτάσει στις 45° τότε το Direction αλλάζει από 1 σε -1 και αρχίζουμε και αφαιρούμε το YawDelta από την γωνία μέχρι να αγγίξουμε το άλλο άκρο όπου θα ξαναλλάξουμε την μεταβλητή Direction. Αφού υπολογίζουμε τις επιθυμητές γωνίες Yaw/Pitch της άρθρωσης του κεφαλιού της της θέτουμε στο NaoOperations service μέσω της ρουτίνας:

```
SetAngles (name, motorlist, angleslist);
```

Το motorlist είναι μια λίστα με της ονομασίες των αρθρώσεων που στοχεύουμε ενώ στο angleslist τοποθετούμε σε αντιστοιχία την τιμή που θέλουμε να πάρουν.

Η συνάρτηση LookAtBall() καλείται όταν έχουμε εντοπίσει πλέον την μπάλα και κινεί κατάλληλα το κεφάλι ώστε η μπάλα να βρίσκεται διαρκώς σε αυτό που ονομάζουμε center zone. Η μπάλα είναι στο center zone όταν η μεταβλητή cameraAngleY & cameraAngleX έχει τιμή από -4 ως +4. Προχωρούμε με την εξής δράση για να υπολογίσουμε το HeadYaw και HeadPitch, ελέγχουμε αν το cameraAngleX είναι μεγαλύτερο του 4, αν αυτό ισχύει τότε προσθέτουμε στο HeadYaw το headDelta, αντίθετα αν το cameraAngleX είναι μικρότερο του -4 τότε αφαιρούμε το headDelta από το HeadYaw. Την ίδια διαδικασία ακολουθούμε με το CameraAngleY για να



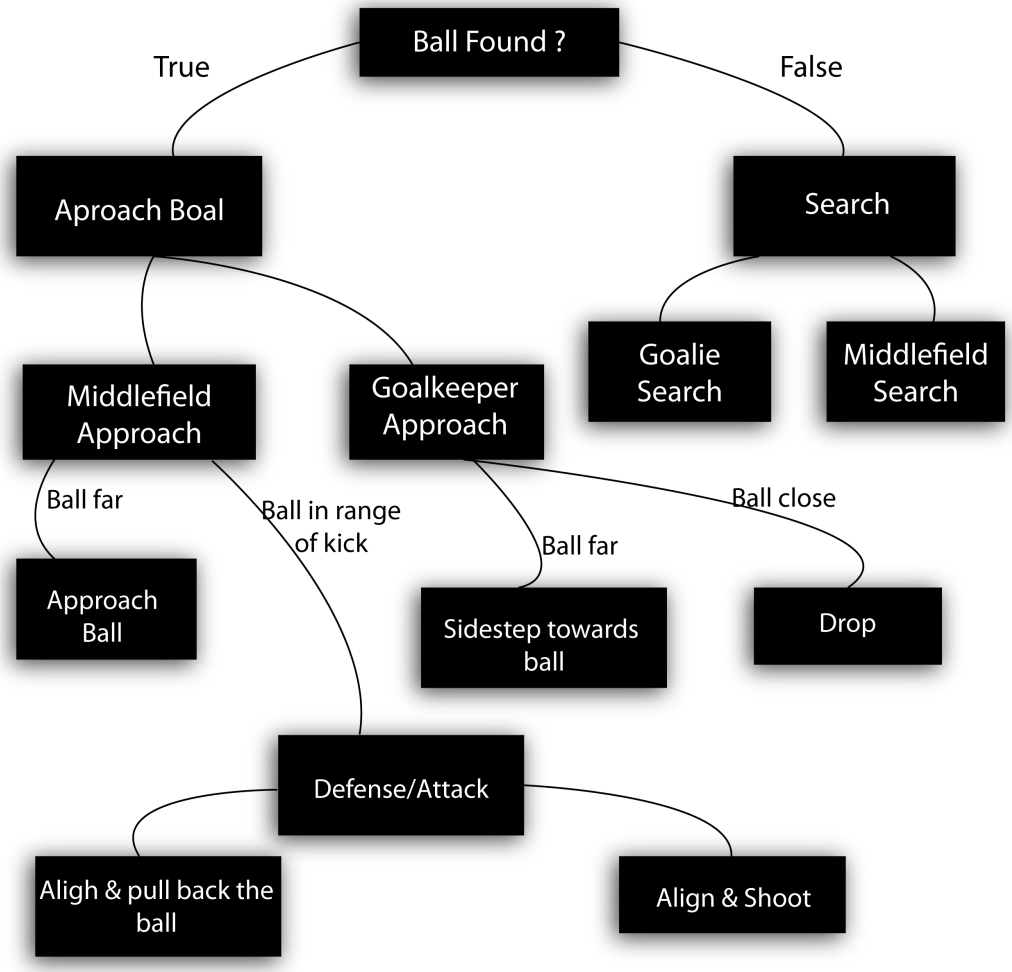
Σχήμα 4.14: Pitch and Yaw movement

υπολογίσουμε το HeadPitch. Όταν υπολογίσουμε της νέες τιμές με την συνάρτηση setAngles(...); στρέφουμε το κεφάλι σταδιακά προς την μπάλα.

Οι ρουτίνες LookAtBall() και PanHead() καλούνται σε κάθε ανανέωση του frame της κάμερας, αυτό συνεπάγεται περίπου 20 $\sim$ 25 frames ανά δευτερόλεπτο. Θα μπορούσαμε σε κάθε κάλεσμα της LookAtBall() να στοχεύουμε απευθείας την μπάλα αλλά αυτό δεν θα ήταν αντιπροσωπευτικό στα πραγματικά ρομπότ όπου οι κινητήρες έχουν περιορισμούς. Αντί αυτού επιλέγουμε μικρές αλλαγές στις γωνίες ώστε να έχουμε ομαλότερη κίνηση το κεφαλιού.

4.4 Decision-Απόφαση

Αφού επιλύσαμε το πρόβλημα της όρασης μέσω της κάμερα και τις κίνησης του ρομπότ, αποκτήσαμε τα βασικά εργαλεία ώστε να σχεδιάσουμε τον τρόπο λήψης αποφάσεων στο ρομπότ αλλά και την υλοποίηση της απόφασης. Το επόμενο στάδιο στην ανάπτυξη του ρομποτικού-παίχτη είναι να του δώσουμε τακτική παιχνιδιού. Ο αλγόριθμος ξεκινάει με το να πάρει τα δεδομένα τις ανάλυσης εικόνας, από εκεί παίρνει πληροφορίες για την θέση της μπάλας και την θέση των δυο τερμάτων. Αφού ανανεώσει τα δεδομένα για την μπάλα και τα τέρματα προσπαθούμε να προσδιορίσουμε αν είμαστε σε κατάσταση επίθεσης, κατάσταση άμυνας ή κατάσταση αναζήτησης. Για κάθε μια από τις καταστάσεις που μπορεί να βρεθεί το ρομπότ έχουμε ορίσει και ένα τρόπο αντίδρασης. Το σχέδιο μας δείχνει το δένδρο αποφάσεων του NAO μας.



Σχήμα 4.15: Δένδρο απόφασης.

Από την επεξεργασία εικόνας λαμβάνουμε πληροφορίες για τρία αντικείμενα με διαφορετικό χρώμα, την μπάλα (πορτοκαλί) και δυο τέρματα (γαλάζιο – κίτρινο). Τα χρώματα αυτά είναι μοναδικά στον κόσμο της προσομοίωσης και επειδή ο θόρυβος στα δεδομένα μας είναι μηδαμινός, είμαστε σίγουροι για τα αποτελέσματα της όρασης. Τα αποτελέσματα της όρασης είναι να μας δώσει αν έχουμε εντοπίσει κάποιο χρώμα και σε τη θέση βρίσκεται αυτό το χρώμα σχετικά με το σώμα του NAO.

Το πρώτο δεδομένο που εξετάζει ο αλγόριθμος είναι αν έχει εντοπιστεί κάποιο από τα τέρματα. Σε περίπτωση που κάποιο τέρμα είναι ορατό τότε πρέπει να αποφασίσει αν είναι αντίπαλο τέρμα η φιλικό, η επικοινωνία με τον διαιτητή κατά την φάση αρχικοποίησης της υπηρεσίας δίνει στο NAO την πληροφορία για το τι χρώμα ομάδα αντιπροσωπεύει. Η κόκκινη ομάδα σουτάρει στο γαλάζιο τέρμα ενώ η μπλε ομάδα στο κίτρινο, έτσι όταν εντοπιστεί ένα από τα χρωματιστά τέρματα γίνεται ο έλεγχος για το αν είναι φιλικό η αντίπαλο. Σε περίπτωση πού βρεθεί αντίπαλο τέρμα τότε το NAO θεωρεί ότι βρίσκεται σε επιθετικό προσανατολισμό αποθηκεύοντας την συνθήκη σε μία boolean μεταβλητή. Η μεταβλητή αυτή ορίζει ότι το NAO είναι ελεύθερο να κλωτσήσει την μπάλα χωρίς να υπάρχει φόβος να την στείλει στο δικό του τέρμα. Η μεταβλητή isAttacking αλλάζει κατάσταση σε ψευδής όταν στο οπτικό πεδίο εμφανιστεί το φιλικό τέρμα.

Σε αυτό το σημείο ο αλγόριθμος εξετάζει αν έχει βρεθεί η μπάλα στο οπτικό πεδίο και με βάση αυτό αποφασίζει τι εντολές θα εκτελέσει. Στο στάδιο που βρισκόμαστε ο αλγόριθμος παίρνει μόνο αποφάσεις για το τι θα ακολουθήσει αλλά δεν εκτελούνται στην ρουτίνα αυτή. Για να εκτελεστεί μια ρουτίνα πρέπει να δημιουργήσουμε ένα μήνυμα εντολής το οποίο θα εμπεριέχει τον τύπο δράσης που θέλουμε να εφαρμόσουμε και να το στείλουμε στο mainPort του ίδιου του NAO. Εκ πρώτης όψεως ίσως ο τρόπος αυτός φαίνεται περιττός και πολύπλοκος, στην ουσία όμως διευκολύνει κάποια πράγματα που ίσως θέλουμε να εφαρμόσουμε στο μέλλον.

Με το να ορίσουμε την εκτέλεση κάποιας εντολής σαν handler μηνύματος που έρχεται στο mainPort μας δίνεται η ευκαιρία να επεκτείνουμε την συνεργασία των ρομπότ ένα βήμα παρά πέρα, με αυτή την αρχιτεκτονική μπορεί το ένα ρομπότ να στείλει απευθείας την οδηγία του σε κάποιον "συμπαίχτη" και αυτός να την εκτελέσει χωρίς να είναι απαραίτητο να γνωρίζει όλα τα δεδομένα της απόφασης που πάρθηκε. Για παράδειγμα ένα ρομπότ που έχει εντοπίσει την μπάλα και κάνει κίνηση να κλωτσήσει την μπάλα προς τα μέσα μπορεί παράλληλα να στείλει την εντολή στον συμπαίχτη του να κινηθεί προς το αντίπαλο τέρμα ώστε να βρεθεί σε πλεονεκτική θέση. Η απόφαση αυτή δεν απαιτεί το ρομπότ no2 να έχει κάποια γνώση για το πού είναι η μπάλα έτσι μπορούμε να παρακάμψουμε την διαδικασία επεξεργασίας που θα ακολουθούσε στέλνοντας του κατευθείαν την κίνηση που πρέπει να κάνει,αποφεύγοντας την πιθανότητα να πάρει κακή απόφαση λόγω ελλιπών δεδομένων.

Το δεύτερο προτέρημα της αρχιτεκτονικής αυτής για την εκτέλεση της απόφασης είναι ότι εκμεταλλευόμαστε τις δυνατότητες του CCR, το ExecuteCommand handler που ορίζουμε όπως κάθε handler τρέχει σε δικό του thread σαν ξεχωριστή διαδικασία επιτρέποντας έτσι το ρομπότ να περάσει στην διαδικασία εκτέλεσης της απόφασης που έχει παρθεί χωρίς να μπλοκάρει την επεξεργασία των νέων δεδομένων που θα δεχτεί από την κάμερα. Παρόλα αυτά όταν βρεθεί στην μέση της εκτέλεσης και επικαλεστεί κάποιο δεδομένο όπως το που βρίσκεται η μπάλα τότε θα λάβει τα στοιχεία από την πιο πρόσφατη ανανέωση της κατάστασης του service αφού οι δυο ρουτίνες τρέχουν παράλληλα. Με τον τρόπο αυτό η επεξεργασία των δεδομένων δεν σταματά για να εκτελεστεί μια απόφαση και εγγυόμαστε την συνεχή ροή των πληροφοριών.

Αν η μπάλα δεν έχει εντοπιστεί τότε στέλνουμε την εντολή FindBall, στην κατάσταση FindBall το NAO μπαίνει στην διαδικασία να σκανάρει όλο το γήπεδο μέχρι να εντοπίσει την μπάλα. Τα βήματα που ακολουθούνται είναι τα εξής:

1. Κοιτάνε αριστερά
2. Κοιτάνε δεξιά
3. Ένα βήμα πίσω
4. Κοιτάνε αριστερά
5. Κοιτάνε δεξιά
6. Στροφή 180 μοιρών

Η διαδικασία αυτή ακολουθείται μέχρι να βρεθεί η μπάλα. Ο τερματοφύλακας κάνει διαφορετική αναζήτηση, εφόσον δεν πρέπει να φύγει από την περιοχή του για το μόνο που κάνει για να εντοπίσει την μπάλα είναι να περιστρέφει το κεφάλι του αριστερά δεξιά παραμένοντας στην ίδια θέση.

Στην αντίθετη περίπτωση, αν έχουμε δηλαδή εντοπίσει την μπάλα τότε οι παίχτες αναλαμβάνουν να κινηθούν προς αυτήν για να κερδίσουν την κατοχή της. Οι παίχτες στο γήπεδο δεν προσεγγίζουν την μπάλα με τον ίδιο τρόπο, οι δύο παίχτες που είναι ορισμένοι σαν offense1 και offense2 κινούνται προς την μπάλα επιθετικά ενώ ο offense3 και ο τερματοφύλακας ακολουθούν διαφορετικές τακτικές στην περίπτωση αυτή.

Ο παίχτης 1 & 2 είναι οι επιθετικοί της ομάδας όταν λάβουν την εντολή να πλησιάσουν την μπάλα τότε γίνεται έλεγχος αν κάποια άλλη κίνηση είναι υπό εκτέλεση. Εφόσον δεν εκτελεί άλλη κίνηση το NAO ,ελέγχει αν η γωνία yaw μεταξύ μπάλας παίχτη, αν η γωνία είναι μεγαλύτερη από 20° τότε το NAO στρίβει δεξιά κατά 20°, ενώ αν η γωνία είναι μικρότερη των -20° τότε το NAO στρίβει αριστερά κατά 20°. Σε περίπτωση που η μπάλα βρίσκεται εντός του λεγόμενου center zone μεταξύ

-20 και 20 μοιρών τότε το NAO εκτελεί την κίνηση "Forward50" για απόσταση μεγαλύτερη των 75 εκατοστών ενώ εκτελεί την "forward30" για απόσταση 75cm και μικρότερη.

Ο τερματοφύλακας παρακολουθεί την μπάλα με πλάγιο βηματισμό αριστερά η δεξιά ώστε η μπάλα να βρίσκεται εντός του center zone, φυσικά ο τερματοφύλακας δεν πρέπει να φύγει από την εστία του, έτσι πριν γίνει κάθε πλάγιος βηματισμός υπολογίζουμε πόσο έχει φύγει από το κέντρο του τέρματος. Ο τερματοφύλακας επιτρέπεται να κάνει απόσταση 5 βηματισμών δεξιά ή αριστερά. Για να το επιτύχουμε αυτό ανανεώνουμε μια μεταβλητή με όνομα offset , προσθέτοντας +1 για κάθε δεξί βήμα ενώ αφαιρούμε 1 για κάθε αριστερό βήμα και φροντίζουμε η μεταβλητή να είναι εντός της περιοχής [-5,5].

Ο παίχτης 3 που λειτουργεί ως αμυντικός στο γήπεδο έχει παρόμοια λογική με τους παίχτες 1&2. Όταν η μπάλα βρίσκεται εκτός center zone κάνει την απαραίτητη περιστροφή για να βρεθεί η μπάλα στο κέντρο του οπτικού πεδίου αλλά δεν πλησιάζει την μπάλα σε απόσταση πάνω από 1.2 μέτρα. Ενώ όταν η μπάλα τύχει να τον πλησιάσει σε πολύ κοντινή απόσταση 35 εκατοστά και λιγότερο και βρίσκεται σε επιθετική κατάσταση (isAttacking==true) τότε εκτελεί την κίνηση drop για να εμποδίσει τα αντίπαλα ρομπότ να έχουν ελεύθερο πεδίο για σουτ.

Οι παίχτες 1, 2 είναι αυτοί που βρίσκονται πιο κοντά στην μπάλα και εφόσον έχουν πιο άμεση προσέγγιση της μπάλας είναι τα NAO που κάνουν τα περισσότερα σουτ . Το πρόβλημα που αντιμετώπισα με το σουτ ήταν ότι όταν το ρομπότ ήταν σε θέση να κλωτσήσει χάνει από το οπτικό του πεδίο την μπάλα. Για να λυθεί το πρόβλημα αυτό θέτουμε τα ρομπότ σε μια κατάσταση που ονομάζεται Align, είναι μια κατάσταση που το ρομπότ προσπαθεί να τοποθετηθεί κοντά στην μπάλα πριν ξεκινήσει το σουτ και θεωρεί ότι η μπάλα βρίσκεται μπροστά του ενώ δεν έχει οπτική επαφή. Το κριτήριο για να μπει σε κατάσταση ευθυγράμμισης το NAO είναι το εξής:

$$\texttt{if}\ (_ballArea > 1250 \& _state.BallHeading < 16 \& _state.BallHeading > -16)$$

Όπως φαίνεται στον κώδικα το αν η μπάλα βρίσκεται σε σωστή απόσταση δεν το υπολογίζουμε με βάση την απόσταση που μας δίνει η επεξεργασία εικόνας. Μετά από πολλές δοκιμές διαπίστωσα ότι σε κοντινές αποστάσεις κάτω του 0.3 μέτρα ο υπολογισμός είναι ανακριβής δίνοντας ακόμα και αρνητικές τιμές. Οπότε με βάση πειραματισμούς όταν το _ballArea $>$ 1250 pixels τότε η μπάλα είναι σε απόσταση που με ένα μικρό βήμα μπροστά είναι ακριβώς μπροστά στα πόδια του NAO χωρίς αυτό να την βλέπει. Όταν λοιπόν η συνθήκη βρεθεί αληθινή τότε ελέγχουμε αν πραγματοποιείται κάποια κίνηση από το ρομπότ εκείνη την στιγμή, αν το ρομπότ κάνει κάποια κίνηση τότε αυτή ακυρώνεται και στέλνει μήνυμα στο Main Port να εκτελέσει την εντολή Align. Οι κινήσεις περιστροφής και κλωτσιάς για ευνόητους λόγους δεν διακόπτονται. Όταν το ExecuteCommand handler παραλάβει το μήνυμα Align τότε αυτό κάνει δύο ενέργειες, πρώτα εκτελεί ένα βηματισμό στο NAO

και έπειτα θέτει την μεταβλήτη isAligned==True. Στον επόμενο κύκλο επεξεργασίας αν η μεταβλητή βρεθεί αληθής τότε το NAO είναι έτοιμο να σουτάρει χώρις να μπεί στην διαδικασία να ανιχνεύσει την μπάλα που θα έχει χαθεί απο το οπτικό του πεδίο.

Από την στιγμή που στην συνάρτηση ελέγχου βρεθεί το σήμα ότι το ρομπότ έχει ευθυγραμμιστεί (isAligned) για σουτ, τότε στέλνουμε στο ρομπότ την εντολή Kick, η εντολή Kick αναλαμβάνει να επιλέξει αν είμαστε σε επίθεση ή άμυνα και και να χρησιμοποιήσει το κατάλληλο πόδι είτε για τακουνάκι είτε για κλωτσιά. Η επίθεση και άμυνα όπως προαναφέραμε ορίζεται από την μεταβλητή isAttacking ενώ για την επιλογή ποδιού ελέγχει την μεταβλητή BallHeading που είναι η γωνία yaw μεταξύ μπάλας – NAO, αν η γωνία είναι θετική τότε χρησιμοποιεί το δεξί πόδι, αν είναι αρνητική χρησιμοποιεί το αριστερό. Η μεταβλητή BallHeading αλλάζει τιμή μόνο κατά τον εντοπισμό της μπάλας, έτσι ακόμα και όταν δεν είναι ορατή η μπάλα χρησιμοποιεί σαν δεδομένο την τελευταία της εμφάνιση αν αυτή ήταν δεξιά ή αριστερά. Προς το παρών τα ρομπότ δεν έχουν αντιμετωπίσει κάποιο πρόβλημα με αυτού του είδους την τακτική αφού είναι πιο γρήγορη από το να ελέγχει κάθε φορά αν η μπάλα είναι στα πόδια του και οι περιπτώσεις λάθους κατά την διάρκεια του τουρνουά ήταν ελάχιστες.

Μία επιπλέον λειτουργία είναι η πτώση των ρομπότ που διαχειρίζεται με διαφορετικό τρόπο. Κατά την πτώση του NAO στο γήπεδο ο παίχτης δέχεται ένα μήνυμα το οποίο εμπεριέχει την γωνία που σχηματίζει το ρομπότ με το έδαφος γύρω από τον άξονα Z. Όταν η τιμή είναι μεγαλύτερη του 1 rad τότε το νάο βρίσκεται με την πλάτη στο έδαφος, ενώ όταν είναι μικρότερη του -1 rad τότε είναι με το στήθος στο έδαφος. Όταν το ειδοποιηθεί το NAO ότι βρίσκεται στο έδαφος με το μπροστινό μέρος τότε εκτελεί την κίνηση "getup" για να σηκωθεί, σε περίπτωση που βρίσκεται με την πλάτη στο έδαφος εκτελεί την κίνηση "flip" όπου χρησιμοποιώντας τα χέρια και τα πόδια περιστρέφεται για να μπορέσει να σηκωθεί με την κίνηση "getup".

4.5 Επικοινωνία

Η επικοινωνία των NAO μεταξύ τους είναι μια λειτουργία που υπάρχει αλλά δυστυχώς δεν συμπεριλήφθηκε στην διαδικασία των αποφάσεων. Τα NAO έχουν την δυνατότητα να ανταλλάσσουν δεδομένα μεταξύ τους για την θέση τους αλλά και να διατάσει το ένα το άλλο να εκτελέσει μια λειτουργία. Κατά την εκκίνηση της υπηρεσίας μια ρουτίνα με όνομα DetectPlayers εντοπίζει τα mainPorts των συμπαιχτών και δημιουργεί ένα δίαυλο μεταξύ τους. Στέλνοντας πακέτα σε αυτά τα Ports επικοινωνούν τα ρομπότ μεταξύ τους. Τα πακέτα μπορούν να είναι δυο ειδών, μήνυμα πληροφορίας ή εντολή προς εκτέλεση. Τα μηνύματα εντολές είναι ακριβώς τα ίδια μηνύματα που στέλνει το NAO στον εαυτό του πριν εκτελέσει μια κίνηση, εμπλουτίζοντας το ExecuteCommand handler με επιπλέον επιλογές

μπορούμε να διευρύνουμε τον τρόπο συνεργασίας με ειδικές περιπτώσεις.

Τα μηνύματα πληροφοριών είναι έχουν την ίδια μορφή αλλά είναι σχεδιασμένα ειδικά για να μεταφέρουν πληροφορία που θα επεξεργαστεί από τον συμπαίχτη NAO. Για την διαχείριση των μηνυμάτων αυτών έχουμε ένα ReceivePeerMessage handler που αναλαμβάνει να κρατάει ένα κανάλι ανοιχτό που δέχεται μηνύματα τύπου PeerMessage. Ο Handler αυτός τρέχει παράλληλα με το υπόλοιπο σύστημα και ενεργοποιείται μόνο όταν έχει κάποιο μήνυμα να επεξεργαστεί.

Η αρχική ιδέα για την συνεργασία των ρομπότ είναι να υπάρχει ένα είδος δημοπρασίας για το ποιο NAO θα πλησιάσει πρώτο την μπάλα ώστε να μην δημιουργείται συνωστισμός. Έτσι κάθε ρομπότ θα έστελνε την απόσταση που έχει από την μπάλα και το κοντινότερο θα έπαιρνε προτεραιότητα ενώ το δεύτερο κοντινότερο θα έκανε προωθητική κίνηση προς το αντίπαλο τέρμα. Το σύστημα δεν δοκιμάστηκε αφού η επικοινωνία μέσω μηνυμάτων για κάποιο αδιευκρίνιστο λόγο δεν γινόταν να υλοποιηθεί στους υπολογιστές που διεξαγόταν το ROBOCUP, καθιστώντας άχρηστο το όλο σχέδιο.

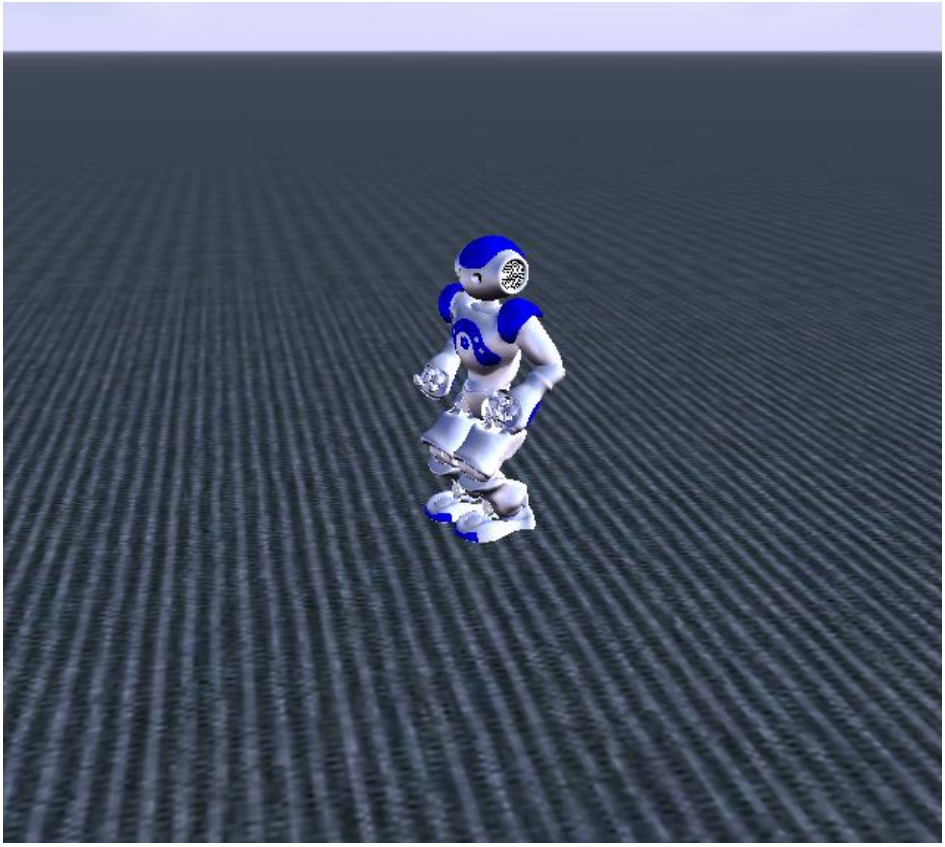
Κεφάλαιο 5

Αποτελέσματα – Προοπτικές

Το project των Κουρητών ξεκίνησε για να συμμετάσχει στο Robocup που διεξάχθηκε στο Suzhou της Κίνας τον Ιούνιο του 2008. Η ομάδα είχε μια πολύ καλή πορεία στον διαγωνισμό κερδίζοντας όλα τα παιχνίδια και κατακτώντας την πρώτη θέση. Οι αλγόριθμοι που χρησιμοποιήθηκαν είναι απλοί ωστόσο αποδείχθηκαν αρκετά αποτελεσματικοί. Βασιζόμενοι σε γρήγορη επεξεργασία οπτικών δεδομένων, γρήγορη και έξυπνη κίνηση έχουμε δημιουργήσει μια αρκετά καλή ομάδα σε πρώτο επίπεδο. Το μοντέλο αρχιτεκτονικής που έχουν στηθεί οι παίκτες, είναι ένα μοντέλο στα πρότυπα του Microsoft Robotics Studio με υψηλές δυνατότητες επεκτασιμότητας. Οι βελτιώσεις που μπορούν να γίνουν στο μέλλον είναι πολλές.

Κυριότερη βελτίωση θα αποτελούσε η βελτίωση της κίνησης του NAO. Παρότι η κίνηση είναι γρήγορη, το ρομπότ κινείται μόνο σε ευθείες τροχιές και εκτελεί περιστροφές γύρο από τον εαυτό του. Η ευθύγραμμη προσέγγιση σημείων στον χώρο δεν είναι αρκετά γρήγορη όταν χρειάζεται να έχουμε συγκεκριμένο προσανατολισμό όταν πλησιάσουμε την μπάλα. Επίσης η ευθύγραμμη κίνηση καθιστά δυσκολότερη την αποφυγή εμποδίων κατά την διάρκεια του περπατήματος. Μια μηχανή που θα παρήγαγε τα νέα βήματα κατά την διάρκεια της κίνησης θα ήταν η ιδανική λύση για το συγκεκριμένο πρόβλημα. Με την παραγωγή νέων βημάτων κατά την διάρκεια της κίνησης έχουμε την δυνατότητα να εκτελούμε μανούβρες άμεσα. Επίσης η κίνηση σε ελλειπτικές τροχιές είναι πολύ καλός τρόπος να πηγαίνεις σε συγκεκριμένες θέσεις με συγκεκριμένο προσανατολισμό, κάτι που θα βοηθούσε πολύ της ομαδικές στρατηγικές.

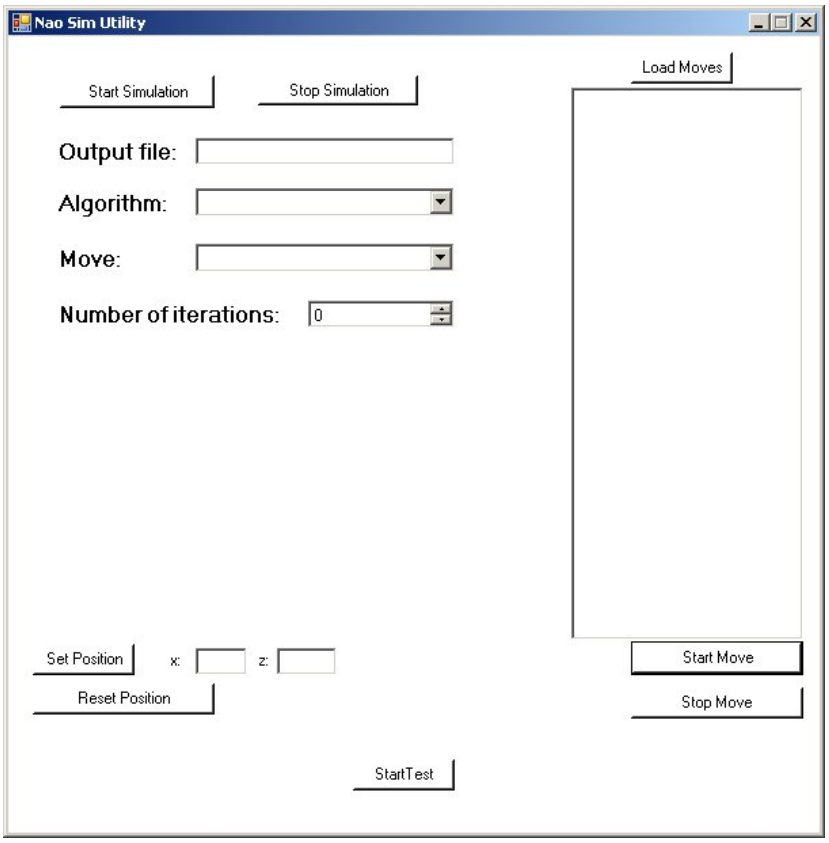
Στον τομέα τις κίνησης συγκαταλέγονται και οι ειδικές κινήσεις του ρομπότ όπως περιγράψαμε στο προηγούμενο κεφάλαιο. Οι κινήσεις αυτές δημιουργήθηκαν με τον move editor χειροκίνητα αλλά ίσως δεν είναι οι βέλτιστες. Εφαρμόζοντας αλγόριθμους reinforcement learning θα μπορούσαμε να χρησιμοποιήσουμε σαν πρότυπο αυτές τις κινήσεις για να δημιουργήσουμε νέες πιο αποδοτικές. Reinforcement learning είναι μια μέθοδος που αναλαμβάνει μέσω της επανάληψης να βελτιώνει σταδιακά τον τρόπο που εκτελείτε μια διαδικασία. Σε ένα αλγόριθμο



Σχήμα 5.1: Simulation Enviroment

RL το εκπαιδευόμενο σύστημα δεν δέχεται εντολές για τι να κάνει αλλά προσπαθεί μέσα από μια διαδικασία επαναλήψεων να βρει ποιες κινήσεις δίνουν καλύτερο αποτέλεσμα. Τα αποτελέσματα δεν κρίνονται βραχυπρόθεσμα αλλά μακροπρόθεσμα, σκοπός είναι στο τέλος της διαδικασίας να έχουμε την μέγιστη ανταμοιβή. Πιο συγκεκριμένα για την κίνηση της κλωτσιάς το πρόβλημα έχει σαν ανταμοιβή την απόσταση που διανύει η μπάλα, την ταχύτητα την οποία εκτελεί την κίνηση ενώ σαν τιμωρία (αρνητική τιμή ανταμοιβής) θα δίναμε σε περιπτώσεις που δεν χτυπούσε την μπάλα ή το NAO θα έχανε την ισορροπία του.

Για να δημιουργήσουμε και να δοκιμάσουμε έναν αλγόριθμο RL πρέπει λοιπόν να στήσουμε ένα σύστημα προσομοίωσης που επιτρέπει την αξιολόγηση της κίνησης και την ελεγμένη επανάληψη της διαδικασίας. Για την επίτευξη του σκοπού αυτού στο μέλλον άρχισα την δημιουργία ενός ξεχωριστού εργαλείου με όνομα NaoSim. Το εργαλείο αυτό είναι σα αρχικό στάδιο και μας επιτρέπει να στήσουμε το περιβάλλον για την επαναληπτική αυτή διαδικασία. Το NaoSim δημιουργεί ένα περιβάλλον προσομοίωσης που περιέχει μέσα ένα NAO σε αρχική θέση στην αρχή του κόσμου (σημείο 0,0,0), επίσης το NaoSim μας παρέχει ένα port επικοινωνίας με το NAO. Πέρα από αυτό είμαστε σε θέση να ελέγχουμε την θέση του NAO ανά πάσα στιγμή, να συνθέτουμε νέες κινήσεις και να τις εκτελούμε, να χρονομε-



Σχήμα 5.2: Tool Gui

τρούμε την διάρκεια τις κίνησης αλλά και να μας ανακοινώνει το πότε τέλειωσε η κίνηση. Επίσης μπορούμε να εισάγουμε ότι άλλο αντικείμενο θέλουμε στην προσομοίωση όπως πχ μια μπάλα. Αφού έχουμε διαθέσιμες όλες της πληροφορίες του αντικειμένου μέσα στην προσομοίωση μας είναι εύκολο να δίνουμε μια ανταμοιβή στο τέλος κάθε κίνησης και έπειτα να την επανασχεδιάζουμε και να την εκτελούμε όσες φορές κρίνουμε απαραίτητο. Το πρόγραμμα επίσης διαθέτει ένα γραφικό περιβάλλον για τον έλεγχο της διαδικασίας και την ανταλλαγή πληροφοριών μεταξύ προσομοίωσης και χρήστη.

Επόμενος τομέας βελτίωσης είναι σίγουρα το localization, αυτή την στιγμή δεν υπάρχει ακριβής υπολογισμός της θέσης αλλά αρκούμαστε στον προσανατολισμό τους ρομπότ. Ένα σύστημα localization που θα στηριζόταν στην όραση θα έδινε πολύ καλύτερα αποτελέσματα. Μέσω της όρασης το NAO θα μπορούσε να αναγνωρίζει της γραμμές του γηπέδου και σε συνδυασμό με κάποιο σύστημα μνήμης των τροχιών που έχει ακολουθήσει να προσδιορίζει την θέση του στο γήπεδο.

Στο κομμάτι της όρασης θα μπορούσαμε να προσθέσουμε την αναγνώριση εμποδίων, όπως τα αντίπαλα ρομπότ. Αυτό μπορούμε να το επιτύχουμε προσθέτοντας επλέον επιλέον χρώματα αναζήτησης στον υπάρχον αλγόριθμο για την επεξεργασία εικόνας. Αυτό δεν θα ήταν όμως λειτουργικό από την στιγμή που δεν

θα είχαμε κάποιο αποτελεσματικό τρόπο να χειριστούμε αυτό τα δεδομένα που θα δέχεται το NAO. Σε συνδυασμό με μια εξελιγμένη μηχανή περπατήματος και ένα σύστημα localization αυτή η πληροφορία θα ήταν πολύ πιο χρήσιμη. Θα ήταν σε θέση το NAO να υπολογίσει τροχιές για να αποφύγει τα εμπόδια που θα παρομοιαζόντουσαν ενώ παράλληλα να μένει σε ευνοϊκή θέση προς την μπάλα.

Από την στιγμή που ρομπότ έχει ένα βασικό "skill set" (δυνατότητες του συστήματος) είναι φυσικό να ακολουθήσει ο τομέας τις «λογικής» των παιχτών. Με περισσότερα εργαλεία διαθέσιμα, περνάμε στον τομέα της δημιουργίας πιο εξελιγμένων αλγορίθμων για τον έξυπνο έλεγχο των ρομπότ. Όσα περισσότερα δεδομένα συλλέγονται για τον περιβάλλον και όσο πιο περίπλοκες εργασίες μπορούν να εκτελεστούν από το ρομπότ τόσο πιο πολύπλοκες συμπεριφορές μπορούν να παραχθούν. Δεν θα είχε νόημα να παραχθούν πολύπλοκα σενάρια συνεργασίας από την στιγμή που τα ρομπότ δεν θα ήταν σε θέση να αναγνωρίσουν που βρίσκονται και που θα πάνε. Η κίνηση, η όραση και η κατανόηση του χώρου είναι το βασικό σκαλοπάτι που πρέπει να ξεπεραστεί για να προχωρήσουμε στα επόμενα στάδια ανάπτυξης.

Το ποδόσφαιρο σαν παιχνίδι είναι μια πρόκληση που απαιτεί διαρκή εξέλιξη σε αντίθεση με παιχνίδια όπως σκάκι και checkers όπου αλγόριθμοι έχουν εξελιχθεί σε τέτοιο βαθμό ώστε να κερδίζουν παγκόσμιους πρωταθλητές. Η πρόκληση του ποδοσφαίρου είναι πολύπλευρη και με πολλά περιθώρια ανάπτυξης. Αυτό είναι που διατηρεί το ενδιαφέρον για όλους όσους έχουν ασχοληθεί, αλλά και ο λόγος που επιλέχθηκε από το Robocup σαν βασικός τομέας ανταγωνισμού.

Βιβλιογραφία

Kyle Johns and Trevor Taylor (2008). Professional Microsoft®Robotics Developer Studio

Jesse Liberty and Donald Xie(2008). Programming C-sharp 3.0.

Peter Norvig and Stuart Russell, (2003). Artificial Intelligence *A Modern Approach*