Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρονικών Μηχανικών και
Μηχανικών Υπολογιστών

Διαδραστικό Ρομποτικό Gaming

Διπλωματική εργασία: Κωνσταντίνος Γ. Χαλδέζος

Επιβλέπων καθηγητής: Μιχαήλ Γ. Λαγουδάκης

Εξεταστική επιτροπή: Μιχαήλ Γ. Λαγουδάκης
Ευριπίδης Πετράκης
Βασίλειος Σαμολαδάς

Χανιά 2009

Περίληψη

Τα αυτόνομα ρομποτικά συστήματα αρχίζουν και γίνονται μέρος της καθημερινής μας ζωής, είτε ως βοηθοί στις οικιακές εργασίες (π.χ. η ρομποτική σκούπα Roomba), είτε ως ψυχαγωγική συντροφιά (π.χ. το ρομποτικό τετράποδο AIBO). Κατά πόσον όμως μπορούν να αλληλεπιδράσουν με το περιβάλλον που έχουμε διαμορφώσει εμείς οι άνθρωποι για τις ανάγκες μας; Εξετάζουμε αυτό το ερώτημα σε ένα πολύ συγκεκριμένο πλαίσιο: μπορεί ένα ρομπότ να παίξει ένα παιχνίδι στον υπολογιστή, όπως ακριβώς θα έπαιζε ένας άνθρωπος, δηλαδή παρατηρώντας την οθόνη και πιέζοντας πλήκτρα στο πληκτρολόγιο; Στα πλαίσια της παρούσας εργασίας, κατ' αρχήν σχεδιάστηκε ένα απλό ηλεκτρονικό παιχνίδι τύπου Arcanoid, όπου ο παίκτης πρέπει να χειριστεί μια ρακέτα για να αποτρέψει την πτώση της μπάλλας που αναπηδά στην οθόνη, και αφ' ετέρου προγραμματίστηκε ένα τετράποδο ρομπότ AIBO να παίζει αυτόνομα αυτό το παιχνίδι στον υπολογιστή χωρίς καμία εσωτερική επικοινωνία, αλλά μόνο χρησιμοποιώντας την έγχρωμη κάμερα που διαθέτει και τις αρθρώσεις του κεφαλιού για αντίληψη, καθώς και τις αρθρώσεις των μπροστινών ποδιών για δράση. Το AIBO αποδείχθηκε ικανός παίκτης για το συγκεκριμένο παιχνίδι, ωστόσο στην πορεία αναφάνηκαν διάφορα πρακτικά προβλήματα τέτοιων εγχειρημάτων (φωτισμός, τοποθέτηση, ρύθμιση, προσαρμογή) που πρέπει μελλοντικά να αντιμετωπίζονται δυναμικά ώστε να ευελπιστούμε σε πραγματικά αυτόνομα ρομποτικά συστήματα ικανά να συνυπάρξουν στο περιβάλλον μας.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Ο άνθρωπος χρησιμοποιεί διάφορα τεχνάσματα και μηχανήματα για την ευκολότερη εκτέλεση μια εργασίας. Ένα από αυτά τα μηχανήματα είναι ο ηλεκτρονικός υπολογιστής. Η χρήση ένος ηλεκτρονικού υπολογιστή απαιτεί κάποιες γνώσεις από τον χρήστη. Είναι δυνατόν όμως ένα ρομπότ να χρησιμοποιήσει ενα Η/Υ για μια απλή εργασία, όπως ένα παιχνίδι;

Με την κατασκευή των τετραπόδων ρομπότ AIBO από την Ιαπωνική Sony, μία από τις βασικές επιδιώξεις της εταιρείας και κάποιων προγραμματιστικών ομάδων, ήταν ο προγραμματισμός των AIBO για την χρήση τους σε διάφορα παιχνίδια. Η τεχνητή νοημοσύνη και η μηχανική μάθηση αποτελούν τα σημαντικότερα εργαλεία στην προσπάθεια για την επίτευξη αυτού του στόχου.

Ένα ρομπότ, για να συμμετάσχει σε ένα παιχνίδι, πρέπει να διαθέτει αισθητήρες για να ανιχνεύει τις αλλαγές στο παιχνίδι καθώς και επενεργητές που να μπορούν να επιφέρουν με τον τρόπο τους τις δικές τους αλλαγές. Μία κάμερα σε συνδυασμό με την ανάπτυξη κατάλληλων εφαρμογών μηχανικής όρασης, αποτελούν ίσως το σημαντικότερο αισθητήριο για ένα ρομπότ, κυρίως λόγω του μεγάλου όγκου δεδομένων, που μπορεί να περιέχεται σε μια εικόνα. Το AIBO διαθέτει κάμερα καθώς και μέλη-επενεργητές που του επιτρέπουν την εκτέλεση απλών παιχνιδιών, για αυτό και επιλέχθηκε σε αυτήν τη διπλωματική εργασία.

1.2 Περιγραφή της εργασίας

Στα πλαίσια της εργασίας αυτής σχεδιάσαμε ένα παιχνίδι. Στο παιχνίδι αυτό ο παίκτης χειρίζεται μια πράσινη μπάρα και προσπαθεί να κυνηγήσει με αυτήν μια ροζ μπάλα (αναλυτική περιγραφή του παιχνιδιού στο κεφάλαιο 4). Το κάναμε αρκετά απλό σε όλους τους τομείς, ειδικά σ' αυτόν των γραφικών και της ταχύτητας, έτσι ώστε το AIBO να προλαβαίνει να συλλέγει την πληροφορία μέσα από την εικόνα που λαμβάνει από την κάμερα. Ο προγραμματισμός του ρομπότ έγινε με τα παρακάτω βήματα:

1. Παρακολούθηση της ροζ μπάλας από το AIBO με ταυτόχρονη κίνηση του κεφαλιού, ώστε η μπάλα να βρίσκεται στο κέντρο του οπτικού πεδίου της κάμερας.

2. Αναγνώριση της πράσινης μπάρας και αποθήκευση των συντεταγμένων της όποτε εντοπίζεται.

3. Κίνηση των μπροστινών ποδιών ώστε να πιέζουν τα πλήκτρα του πληκτρολογίου.

4. Εύρεση αλγόριθμου για πρόβλεψη της κίνησης της μπάλας μετά από κάποια αναπήδηση.

5. Συνδυασμός των παραπάνω ώστε το AIBO να πιέζει τα πλήκτρα, όταν η μπάλα βρίσκεται μακριά από την μπάρα και απαιτείται κίνηση της μπάρας, ώστε η μπάλα να προσκρούσει πάνω της και να αναπηδήσει.

Αφού προγραμματίσαμε το AIBO ώστε να προβλέπει που πρέπει να μετακινηθεί η μπάρα, για να προσκρούσει η μπάλα πάνω της και να αναπηδήσει, μετακινούμε τη μπάρα στη θέση αυτή με την πίεση των κατάλληλων πλήκτρων. Έτσι η μπάρα έχει μετακινηθεί όσο πιο σύντομα γίνεται στο κατάλληλη θέση πριν η μπάλα φτάσει στο ύψος της μπάρας. Αυτό φυσικά εξαρτάται από την ταχύτητα της μπάλας, την ταχύτητα μετακίνησης της μπάρας και το χρόνο ανταπόκρισης και κίνησης του AIBO.

1.3 Οργάνωση της εργασίας

Η παρούσα εργασία έχει χωριστεί σε οκτώ κεφάλαια. Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή, όπου περιγράφονται η αλληλεπίδραση ρομπότ και παιχνιδιού, ο σκοπός της εργασίας, καθώς και η μέθοδος που ακολουθήσαμε για να την επίτευξη του στόχου μας. Στο δεύτερο κεφάλαιο περιγράφονται το ρομπότ που χρησιμοποιήσαμε, τα τεχνικά του χαρακτηριστικά και το διαθέσιμο λογισμικό. Στο επόμενο κεφάλαιο περιγράφεται αναλυτικά το εργαλείο-λογισμικό που χρησιμοποιήσαμε για
τον προγραμματισμό του ρομπότ. Στο τέταρτο κεφάλαιο περιγράφονται το παιχνίδι, το λογισμικό που αναπτύξαμε, καθώς και μια ανάλυση του κώδικα μας για την περαιτέρω αξιοποίηση του. Συνεχίζουμε στο πέμπτο κεφάλαιο, όπου γίνεται μια περιγραφή του προβλήματος, της όρασης του AIBO και της κλάσης-συμπεριφοράς που δημιουργήσαμε. Στο επόμενο κεφάλαιο παρουσιάζουμε τα αποτελέσματα μας και στο έβδομο κεφάλαιο προτείνουμε πιθανές μελλοντικές προεκτάσεις. Φτάνουμε στο όγδοο κεφάλαιο,όπου παρουσιάζουμε σχετικές εργασίες. Τέλος στον επίλογο καταγράφονται τα συμπεράσματά μας από αυτή την εργασία.

Κεφάλαιο 2

AIBO

2.1 Περιγραφή AIBO

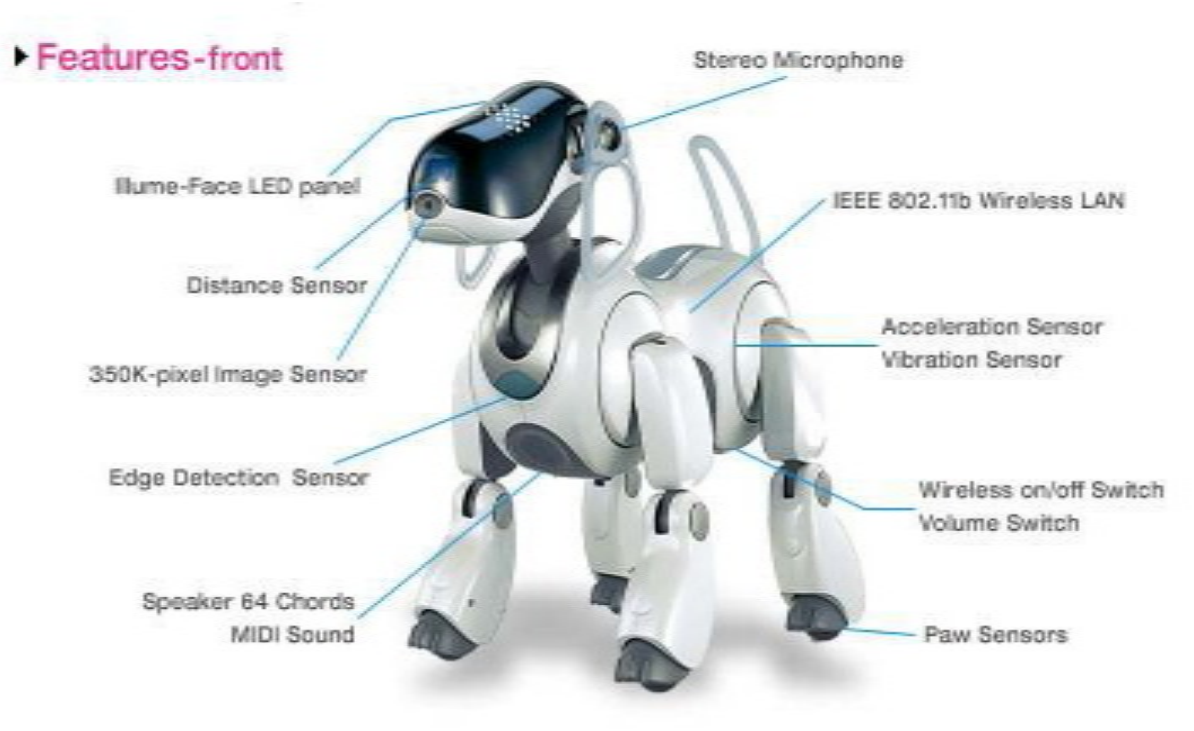
Το AIBO είναι ένα τετράποδο ρομπότ της Sony που μοιάζει με σκύλο. Το όνομά του προέρχεται από τις λέξεις A.I. (Artificial Intelligence) και robot. Επίσης, στα Ιαπωνικά AIBO σημαίνει φίλος. Το πρώτο AIBO, που κυκλοφόρησε το Μάιο του 1999, μπορούσε να μαθαίνει από το περιβάλλον του και να εκφράζει τα συναισθήματά του. Με την πάροδο των ετών, οι ικανότητες του AIBO βελτιώθηκαν και έφτασε να μπορεί να επικοινωνεί με τους ανθρώπους και να συνδέεται ασύρματα με το Διαδίκτυο. Αυτό φαίνεται και στην εξέλιξη των διαφόρων μοντέλων του AIBO που κατασκευάστηκαν ανά καιρούς, τόσο στον υλικό εξοπλισμό (hardware) όσο και στις δυνατότητές τους.

2.2 Τεχνικά Χαρακτηριστικά AIBO

Το AIBO επικοινωνεί με το περιβάλλον με τη βοήθεια διαφόρων αισθητήρων και επενεργητών. Η έγχρωμη κάμερα που διαθέτει, προσφέρει εικόνα του χώρου που βρίσκεται το AIBO. Για τον εντοπισμό της θέσης του AIBO και την αποφυγή των εμποδίων, το AIBO είναι εξοπλισμένο με 3 αισθητήρες υπέρυθρων. Επίσης, υπάρχουν αισθητήρες αφής σε πολλά σημεία του AIBO, όπως στο κεφάλι και στην πλάτη, καθώς και μικρόφωνα στα αυτιά. Το λογισμικό για το AIBO αποθηκεύεται σε ένα memory stick, που τοποθετείται στο κάτω μέρος του AIBO. Υπάρχουν memory stick ήδη προγραμματισμένα από την Sony, που προσομοιώνουν τη συμπεριφορά ενός σκύλου. Επίσης με τη χρήση ενός υπολογιστή, κατάλληλων εργαλείων λογισμικού και ασύρματης επικοινωνίας υπολογιστή-ρομπότ, εντολές μπορούν να δωθούν και μέσω του υπολογιστή.

Η κίνηση του AIBO βασίζεται στην ποικιλία αρθρώσεων που διαθέτει: 3 σε κάθε πόδι (4x3) , 3 στο κεφάλι, 1 στο στόμα, 2 στην ουρά και 1 σε κάθε αυτί (2x1). Κάθε άρθρωση μπορεί να κινηθεί ανεξάρτητα από τις υπόλοιπες, δίνοντας συνολικά 20 βαθμούς ελευθερίας στο AIBO. Επίσης το AIBO διαθέτει LEDs σε πολλά σημεία του σώματος του, καθώς και μεγάφωνο στο θώρακα του, κάνοντας έτσι την επικοινωνία του AIBO με τον άνθρωπο και το περιβάλλον πιο ευχάριστη. Ο ενσωματωμένος υπολογιστής που διαθέτει το AIBO περιέχει επεξεργαστή 64-bit RISC στα 576MHz και RAM 64Mb. Οπότε το AIBO διαθέτει όλα τα απαραίτητα εργαλεία για την εκτέλεση παιχνιδιών, γι' αυτό και επιλέχτηκε σ' αυτή τη διπλωματική εργασία.

Τα τεχνικά χαρακτηριστικά του ERS-7 , που είναι το μοντέλο του AIBO που χρησιμοποιήθηκε σε αυτήν την διπλωματική εργασία, φαίνονται στην εικόνα 2.1 και στην εικόνα 2.2.



Εικόνα 2.1 Τεχνικά χαρακτηριστικά μπροστινής όψης AIBO

Εικόνα 2.2 Τεχνικά χαρακτηριστικά πίσω όψης AIBO

Επεξεργαστής	576 Mhz 64bit RISC
Μνήμη	64 Mb SD RAM
Δίκτυο	802.11b wireless LAN 2.4GHz
Κάμερα	color CCD 350k pixels 30fps 56.9 wide 45.2 high
Υπέρυθρες	50-500mm, 100-900mm, 200-1500mm
Μπαταρία	lithium-ion διάρκειας 1.5 ώρας
Βάρος	1.6kg περιλαμβανομένου μπαταρίας και memory stick
Διαστάσεις	180mm (w) x 278mm (h) x 319mm (l)

Πίνακας 2.1 Τεχνικά Χαρακτηριστικά AIBO

2.3 Λογισμικό AIBO

Υπάρχουν διαφόρων ειδών λογισμικά και εργαλεία για τον προγραμματισμό του AIBO. Βέβαια η αρχική πολιτική της Sony ήταν να μην προσφέρει την απαραίτητη διασύνδεση με το hardware, με αποτέλεσμα όλα τα λογισμικά του AIBO να προέρχονται μόνο από την Sony. Το 2002 όμως αποφάσισε να δώσει στο ευρύ κοινό ένα πακέτο ανάπτυξης λογισμικού για το AIBO, το OPEN-R SDK. Έτσι το AIBO από ένα απλό παιχνίδι έγινε και μέσο έρευνας και διδασκαλίας σε διάφορα πανεπιστήμια, όπως το Πολυτεχνείο της Κρήτης.

Τα πιο σημαντικά προγραμματιστικά περιβάλλοντα που υπάρχουν αυτή τη στιγμή για τον προγραμματισμό του AIBO και είναι διαθέσιμα στο internet ως ελεύθερα λογισμικά είναι:

Pyro

URBI

OPEN-R SDK

R-Code

Tekkotsu

2.3.1 Pyro

Το Pyro [1] είναι ένα προγραμματιστικό περιβάλλον, που επιτρέπει εύκολη χρήση εξελιγμένων δυνατοτήτων ενός ρομπότ, χωρίς ο χρήστης να ανησυχεί για τις "χαμηλού-επιπέδου" λεπτομέρειες. Το Pyro είναι γραμμένο σε Python και η ονομασία του προέρχεται από τις λέξεις Python Robotics. Το Pyro εκτός από προγραμματιστικό περιβάλλον είναι και μια συλλογή από αντικείμενα-κλάσεις.

Το Pyro είναι ένα εύχρηστο εργαλείο εφόσον μπορεί να χρησιμοποιηθεί με διάφορα μοντέλα-ρομπότ. Έτσι ο χρήστης δεν αναγκάζεται να μάθει άλλα εργαλεία, εφόσον το Pyro καλύπτει πολλά ρομπότ. Η τελευταία έκδοση του Pyro καλύπτει τα εξής μοντέλα: Pioneer, Pioneer2, PeopleBot, Khepera, Khepera2, Hermisson, AIBO, IntelliBrain-Bot και Roomba.

2.3.2 URBI

Το URBI [2] είναι μία πλατφόρμα γραμμένη σε μια νέα scripted language και βασίζεται σε επικοινωνία client-server με το ρομπότ. Το URBI μπορεί να χρησιμοποιηθεί με όλα τα ρομποτικά μοντέλα, όπως φαίνεται από την ονομασία του Universal Real-time Behavior Interface. Βασικό στοιχείο του URBI είναι η απλότητά του και μπορεί να μαθευτεί πολύ εύκολα χωρίς ιδιαίτερες προγραμματιστικές γνώσεις. Οι εντολές του URBI δίνονται διαδραστικά και μπορούν να "τρέξουν" παράλληλα, σειριακά, είτε με χρήση των events. Τέλος, το URBI μπορεί να χρησιμοποιηθεί μαζί με C++, Python, Matlab με την προσθήκη κατάλληλων βιβλιοθηκών, και τρέχει σε οποιοδήποτε από τα λειτουργικά συστήματα Windows, Mac OSX και Linux.

2.3.3 OPEN-R SDK

Το Open-R [3] αποτελεί το πιο στοιχειώδες περιβάλλον για την επεξεργασία λογισμικού για το AIBO. Ο χρήστης έχει τον πλήρη έλεγχο του ρομπότ, για αυτό και το Open-R είναι εργαλείο για "χαμηλού" επιπέδου προγραμματισμό. Το Open-R λειτουργεί σε Unix ή Linux λειτουργικά συστήματα και σε Windows μέσω του cygwin. Το Open-R SDK αποτελεί το πρώτο προγραμματιστικό εργαλείο για AIBO, το οποίο κυκλοφόρησε η Sony ύστερα από χρόνια κυκλοφορίας των ρομπότ.

Τα Open-R προγράμματα αποτελούνται από διάφορα Open-R αντικείμενα, που τρέχουν ταυτόχρονα μέσω ενός συστήματος επεξεργασίας μηνυμάτων στο λειτουργικό σύστημα του AIBO, το Aperios. Τα αντικείμενα Open-R δημιουργούνται με βάση τα C++ αντικείμενα, αλλά αυτές οι δύο έννοιες δεν πρέπει να μπερδεύονται μεταξύ τους. Κάθε αντικείμενο Open-R τρέχει σε ένα ανεξάρτητο thread και επικοινωνεί με άλλα αντικείμενα Open-R μέσω μηνυμάτων. Για τον συγχρονισμό των μηνυμάτων υπάρχει ένα πρωτόκολλο επικοινωνίας, το οποίο βασίζεται σε ένα ειδικό μήνυμα ASSERT_READY (AR). Κάθε δέκτης στέλνει στον πομπό ένα AR όταν έχει τελειώσει την επεξεργασία του προηγούμενου μηνύματος και είναι έτοιμος να δεχτεί ένα καινούριο μήνυμα.

2.3.4 R-Code

Το R-Code [4] είναι ακόμα μια scripting γλώσσα προγραμματισμού για τα τετράποδα ρομπότ AIBO και παρέχεται από τη Sony μαζί με το OPEN-R. Ένα απλό αρχείο κειμένου ή λίγες γραμμές εντολών αρκούν για την εκτέλεση πολύπλοκων συμπεριφορών, εφόσον πρόκειται για γλώσσα υψηλού επιπέδου. Παρόλα αυτά ο χρήστης μπορεί να εκμεταλλευτεί όλες τις δυνατότητες του AIBO, όπως θα είχε με ένα άλλο προγραμματιστικό περιβάλλον. Ένα πρόγραμμα γραμμένο σε R-Code είναι μία λίστα εντολών για το AIBO. Η σειρά εκτέλεσης των εντολών υποστηρίζεται από την χρήση μιας στοίβας, η οποία λειτουργεί παρόμοια με αυτήν της assembly.

2.3.5 Tekkotsu

Το Tekkotsu [5] αναπτύχθηκε στο CMU και αποτελεί ένα εύχρηστο πλαίσιο προγραμματισμού, καθώς και μια πλούσια βιβλιοθήκη από ρουτίνες για το χειρισμό του ρομπότ. Είναι γραμμένο σε C++, ενώ ταυτόχρονα στηρίζεται στο Open-R, και διαθέτει μια αντικειμενοστρεφή αρχιτεκτονική βασισμένη στην κληρονομικότητα και στα events. Το Tekkotsu διαθέτει στοιχεία γλώσσας “υψηλού” και “χαμηλού” επιπέδου. Το Tekkotsu, ενώ αρχικά σχεδιάστηκε αποκλειστικά για το AIBO, εξελίχθηκε για να χρησιμοποιείται και με άλλα ρομπότ.

Κεφάλαιο 3

Tekkotsu

3.1 Περιγραφή Tekkotsu

Το Tekkotsu [5] αναπτύχθηκε στο CMU και σημαίνει “σιδερένιο κόκκαλο” στα Ιαπωνικά. Το Tekkotsu αποτελεί ένα εύχρηστο πλαίσο προγραμματισμού, καθώς και μια πλούσια βιβλιοθήκη από ρουτίνες για το χειρισμό του ρομπότ. Επίσης είναι γραμμένο σε C++, ενώ ταυτόχρονα στηρίζεται στο Open-R, και διαθέτει μια αντικειμενοστρεφή αρχιτεκτονική βασισμένη στην κληρονομικότητα και στα events. Ο χρήστης μπορεί να δημιουργήσει δικές του κλάσεις, που κληρονομούν από κάποια βασική κλάση του Tekkotsu. Έτσι ο χρήστης δεν χρειάζεται να πειράξει τον κώδικα του Tekkotsu. Παρόλο που ο χρήστης δεν θα χρειαστεί πραγματικά να κάνει μια άμεση κλήση από OPEN-R, το Tekkotsu του δίνει αυτήν την δυνατότητα. Έτσι, το Tekkotsu διαθέτει στοιχεία γλώσσας “υψηλού” και “χαμηλού” επιπέδου. Το Tekkotsu, ενώ αρχικά σχεδιάστηκε αποκλειστικά για το AIBO, εξελίχθηκε για να χρησιμοποιείται και με άλλα ρομπότ. Το Tekkotsu τρέχει κυρίως σε Linux, αλλά μπορεί να τρέξει και σε Windows μέσω του Cygwin [6]. Το Tekkotsu επιλέχθηκε ως το εργαλείο για το χειρισμό του AIBO σε αυτή τη διπλωματική εργασία, λόγω της ευελιξίας της C++. Η κληρονομικότητα σε συνδυασμό με την πληθώρα κλάσεων και μεθόδων, που διατίθενται με το Tekkotsu, το κάνουν ένα πολύ δυνατό εργαλείο.

3.2 ControllerGUI

Το controlergui αποτελεί το βασικό intreface του Tekkotsu για το χειρισμό του ρομπότ από τον υπολογιστή.

Το controllergui καλείται είτε με:

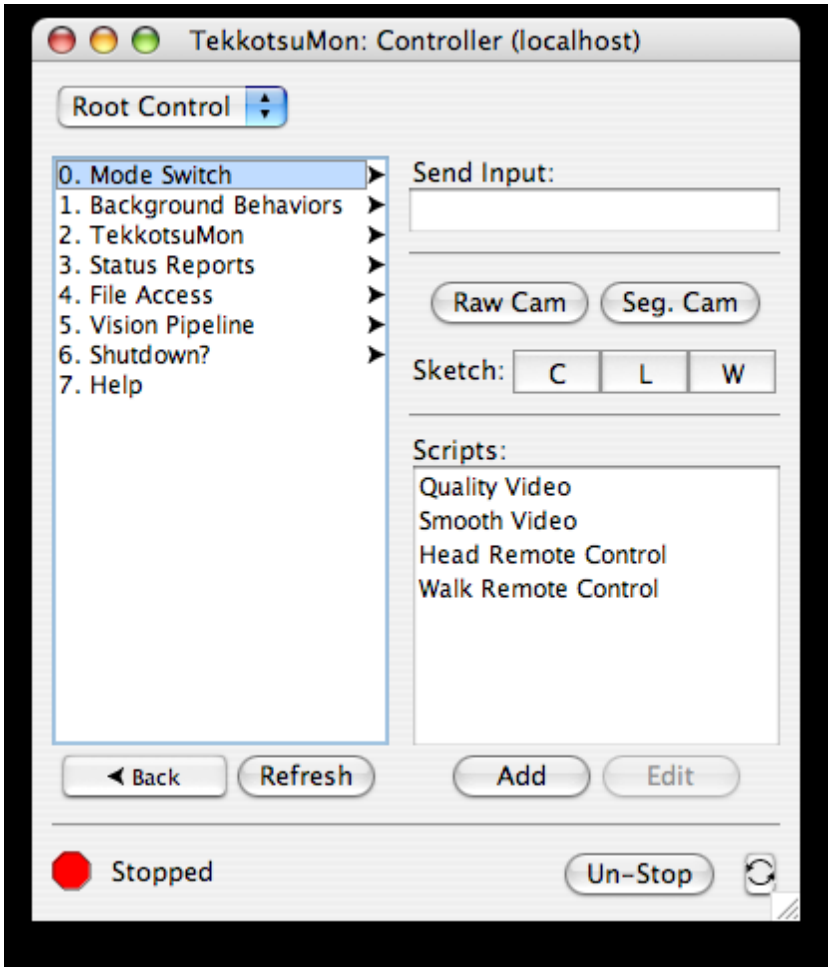
./controllergui (ip του ρομπότ)
από το φάκελο του Tekkotsu:
/Tekkotsu/tools/mon
είτε με:

telnet (ip του ρομπότ) **10001**

Επίσης με telnet στο port 59000 ο χρήστης μπορεί να δει τα μηνύματα που στέλνει το ρομπότ. Τα μηνύματα αυτά μπορεί να είναι μηνύματα λάθους (π.χ. joint conflict, δηλαδή η κίνηση του AIBO σταμάτησε επειδή δεν μπορούσε να συνεχιστεί λόγω κάποιου εμποδίου ή δυσκολίας της κίνησης από την συγκεκριμένη “στάση του σώματος” του AIBO) ή μηνύματα cout που μπορεί να προσθέσει ο χρήστης στον κώδικά του. Και στις δύο περιπτώσεις πάντως είναι πολύ χρήσιμο να παρακολουθούνται αυτά τα μηνύματα, εφόσον διευκολύνουν το debugging σε μεγάλο βαθμό.

Το controllergui αποτελεί έναν τρόπο χρήσης του AIBO. Ο τελικός κώδικας μπορεί να τρέξει και μόνος του χωρίς επικοινωνία.

Στο Mode Switch διαλέγουμε τη συμπεριφορά που θέλουμε να τρέχει στο ρομπότ. Εκτός από την δική μας συμπεριφορά υπάρχουν έτοιμες συμπεριφορές-tutorials για την εξοικείωση του χρήστη με το Tekkotsu. Το interface προσφέρει στο χρήστη την δυνατότητα να αλλάξει τις τιμές οποιασδήποτε άρθρωσης του AIBO και την αποθήκευση διαφόρων postures για μετέπειτα χρήση. Posture στο Tekkotsu είναι μια συλλογή από τιμές για κάθε άρθρωση. Αυτές οι δυνατότητες βρίσκονται στο File Access (Εικόνα 3.1).



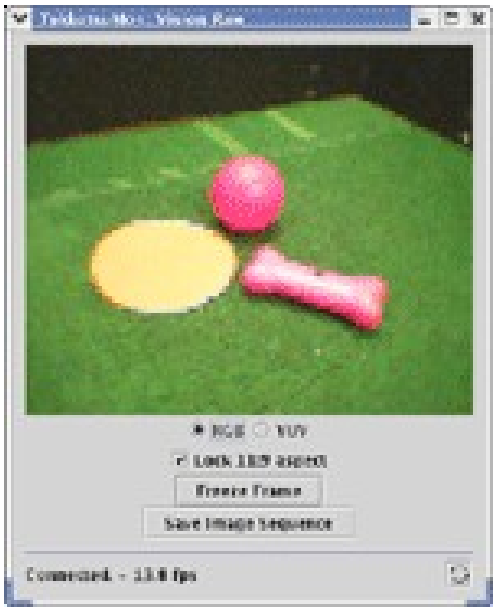
Εικόνα 3.1 Το interface του Tekkotsu, controllergui

Αρχικά το AIBO είναι σταματημένο και δεν εκτελείται καμία συμπεριφορά, μέχρι να επιλεχθεί μία από αυτές και να πατηθεί και το κουμπί Stop/Unstop. Όσο το κουμπί είναι κόκκινο, το AIBO είναι σταματημένο και δεν εκτελεί καμία κίνηση. Το AIBO προειδοποιεί τον χρήστη με ένα γάβγισμα, στην περίπτωση που του δοθεί εντολή ενώ είναι σταματημένο. Όταν πιεστεί το κουμπί, η κόκκινη βούλα Stopped τότε πρασινίζει, αλλάζοντας κατάσταση και επιτρέποντας την κίνηση του AIBO έως ότου σταματηθεί.

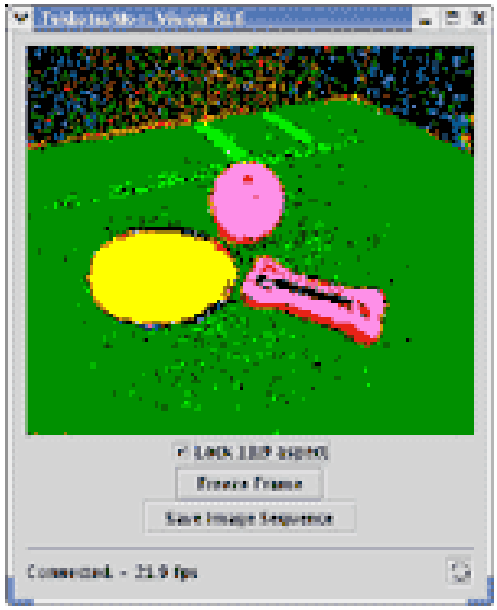
Χρησιμοποιώντας το Walk Remote Control Interface (Εικόνα 3.4), το AIBO μπορεί να καθοδηγηθεί και να μετακινηθεί στον χώρο τριγύρω του. Σε συνδυασμό με την realtime κάμερα που διαθέτει το AIBO, ο χρήστης έχει πλήρη εικόνα του χώρου όπου κινείται και βλέπει το AIBO. Στο controllergui υπάρχουν δύο επιλογές για εμφάνιση της εικόνας της κάμερας:

RawCam, που εμφανίζει την "καθαρή" εικόνα της κάμερας (Εικόνα 3.2)

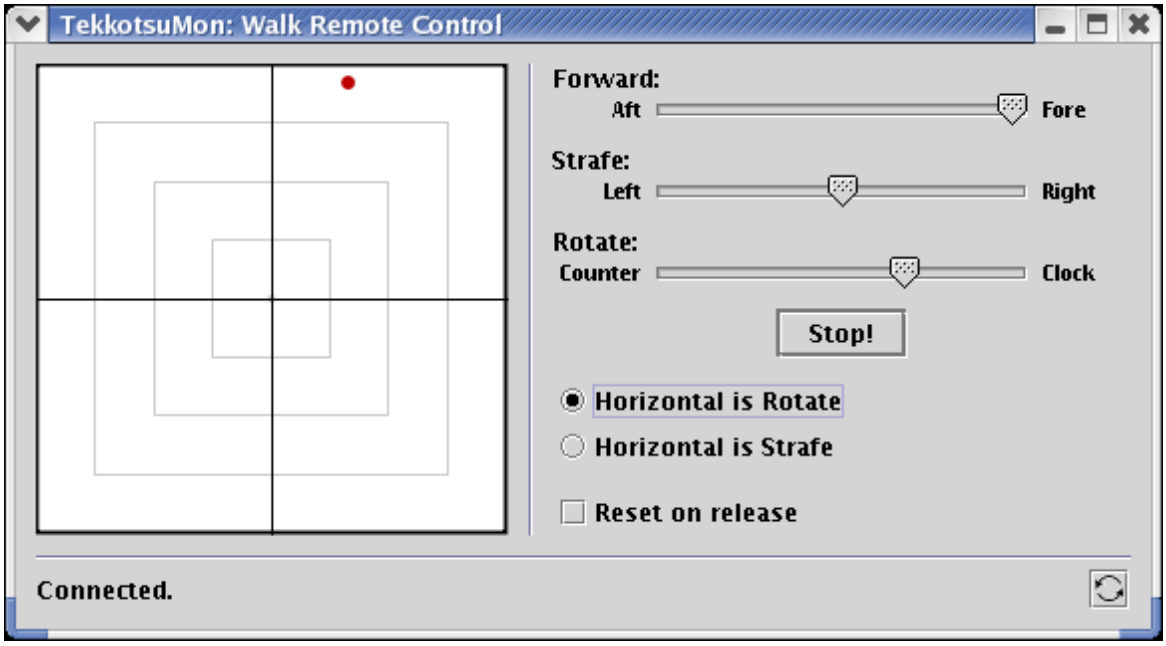
SegCam, που εμφανίζει μια color segmented μορφή της εικόνας (Εικόνα 3.3)



Εικόνα 3.2 RawCam του controllergui



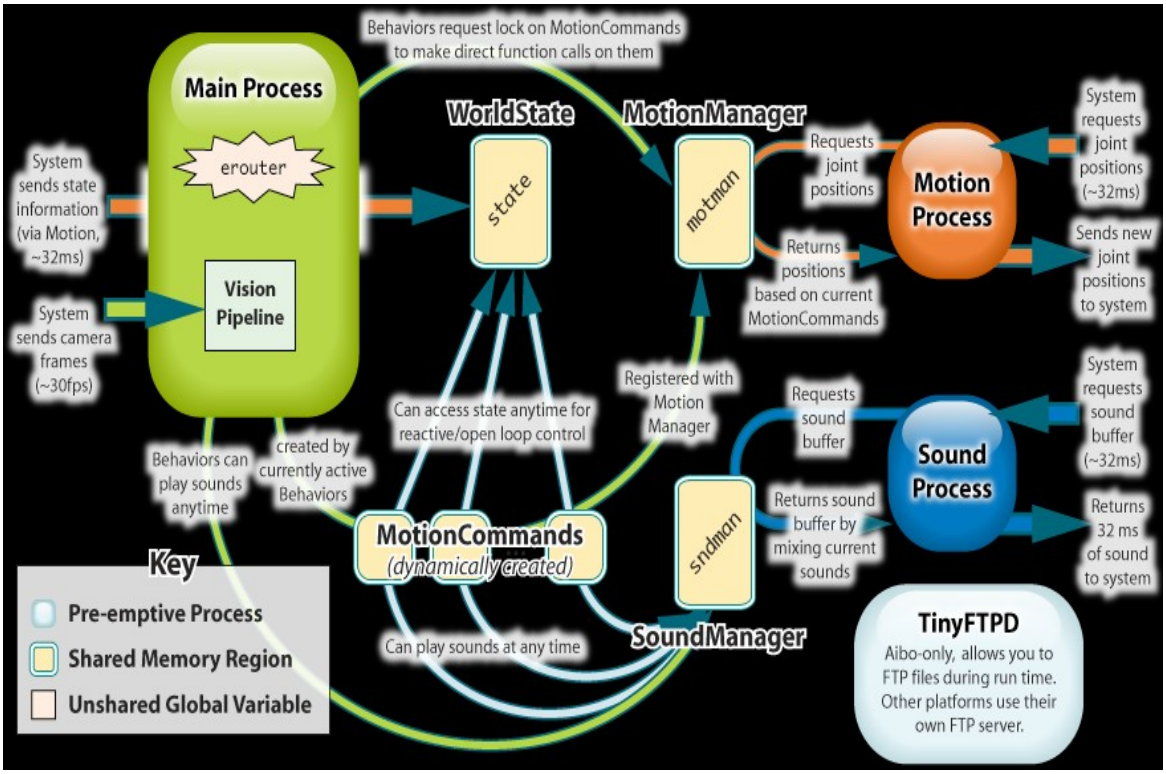
Εικόνα 3.3 SegCam του controllergui



Εικόνα 3.4 Walk Remote Control nterface του controllergui.

3.3 Κλάσεις στο Tekkotsu

Μία εφαρμογή στο Tekkotsu αποτελείται από ένα σύνολο κλάσεων, Behaviors και Motion Commands. Οι συναρτήσεις-μέλη των κλάσεων τρέχουν σε δύο διαφορετικές διαδικασίες την "Main" και την "Motion", οι οποίες επικοινωνούν μεταξύ τους. Η Main χειρίζεται την αντίληψη και τη λήψη αποφάσεων, που απαιτούν παραπάνω χρόνο και επεξεργασία. Η Motion χειρίζεται τον έλεγχο των επενεργητών σε πραγματικό χρόνο. Όλες οι λειτουργίες στην Motion πρέπει να εκτελούνται σε μικρό χρονικό διάστημα (32ms). Η καθυστέρηση της εκτέλεσης κάποιας τέτοιας λειτουργίας μπορεί να προκαλέσει τερματισμό της λειτουργίας του ρομπότ ή σπασμωδική κίνηση του ρομπότ. Μια τρίτη διαδικασία, η SoundPlay, επεξεργάζεται τον ήχο και ο τρόπος λειτουργίας της είναι ίδιος με αυτόν της Motion.



Σχήμα 3.1 Αρχιτεκτονική του Tekkotsu

Τα behaviors δημιουργούνται και τρέχουν στην Main, όπως φαίνεται στην πράσινη περιοχή του παραπάνω διαγράμματος (Σχήμα 3.1). Τα MotionCommands δημιουργούνται σε ένα κοινό κομμάτι μνήμης (μπλε περιοχή στο διάγραμμα), αλλά τρέχουν στην Motion (πορτοκαλί περιοχή στο διάγραμμα). Ένας έξυπνος μηχανισμός εξαίρεσης, που ονομάζεται MMAccessor, επιτρέπει την ασφαλή εκτέλεση των Behaviors στην Main, ώστε να μπορούν να ενημερώνονται και να ενημερώνουν τα κοινά κομμάτια μνήμης, ενώ αυτά χρησιμοποιούνται από τον Motionmanager, που τρέχει στην Motion.

Τα behaviors είναι κλάσεις στο Tekkotsu που μπορούν να κληρονομηθούν για την δημιουργία καινούριων συμπεριφορών για το AIBO. Κάθε behavior κληρονομεί από την βασική behavior κλάση του Tekkotsu : BehaviorBase. Όταν μια συμπεριφορά είναι να χρησιμοποιηθεί δημιουργείται ένα “instance” της κλάσης, το οποίο καταστρέφεται όταν η συμπεριφορά παύει να χρησιμοποιείται. Κάθε συμπεριφορά-κλάση που γράφει ο χρήστης πρέπει να κάνει “override” τις συναρτήσεις που κληρονομεί από την βασική κλάση BehaviorBase. Οι βασικότερες από αυτές είναι:

DoStart() ,όπου αρχικοποιείται το “instance” της κλάσης

DoStop() ,όπου καταστρέφεται το “instance” της κλάσης

proccessEvent(const EventBase &event) , όπου είναι και το βασικό κομμάτι της κλάσης εφόσον χειρίζεται τα events που προκύπτουν.

Τα behaviors εκτελούνται από το RootControl στο controllergui και υπάρχουν έτοιμες συμπεριφορές που μπορεί να εκτελέσει ο χρήστης. Προσθέτοντας απλά μια γραμμή κώδικα και κάνοντας #include το header file στο StartupSetupModeSwitch.cc ο χρήστης προσθέτει την δική του κλάση-συμπεριφορά στο μενού του RootControl.

Event μπορεί να είναι είτε ένα μήνυμα του χρήστη από το controllergui, είτε η ανίχνευση ενός αντικειμένου από την κάμερα του AIBO, καθώς και πολλά άλλα. Για να μπορεί να αντιληφθεί όμως η proccessEvent ένα event, πρέπει πρώτα η συμπεριφορά να παρακολουθεί αυτού του είδους τα events. Αυτό γίνεται μέσω του EventRouter, που καλείται από την global μεταβλητή erouter. Γράφοντας erouter->addListener(this,”το event που παρακολουθούμε”) η proccessEvent() θα καλείται όποτε προκύπτει ένα τέτοιο event.

Κάθε event χαρακτηρίζεται από μια τριπλέτα αριθμών που αναδεικνύουν πως παράχθηκε το event (π.χ. κάποιο κουμπί , η αναγνώριση αντικειμένου από την κάμερα...), την πηγή (π.χ. ποιο κουμπί πατήθηκε, ποιο αντικείμενο αναγνωρίστηκε...) και τον τύπο του event (π.χ. το κουμπί πατήθηκε ή αφέθηκε ελεύθερο). Οι τιμές αυτές μπορούν να προσπελασθούν με τη χρήση των συναρτήσεων getGeneratorID( ), getSourceID( ) και getTypeID( ). Κάθε event είναι υποκλάση της EventBase. Το σύστημα όρασης περιέχει ένα μηχανισμό εντοπισμού αντικειμένων, ο οποίος παράγει ένα σύνολο VisionObjectEvents. Τα VisionObjectEvents είναι υποκλάσεις της Event και περιέχουν επιπλέον δεδομένα και συναρτήσεις για τον καθορισμό των x και y συντεταγμένων του κέντρου του αντικειμένου, όπως αυτό που φαίνεται από την SegCam. Οι συντεταγμένες κανονικοποιούνται στο εύρος τιμών -1.0 έως +1.0. Διαφορετικά αντικείμενα αναγνωρίζονται από το sourceID, που βασίζεται στο χρώμα του αντικειμένου.

Ένας ακόμα τύπος event χρησιμοποιείται για την εκτέλεση εντολών, που γράφονται από τον χρήστη σε ένα ειδικό κουτάκι κειμένου στο controllergui. Οι εντολές αυτές πρέπει να ξεκινούν με θαυμαστικό. Κάθε TextMsgEvent είναι υποκλάση της Event και περιέχει μια επιπλέον συνάρτηση getText( ), που επιστρέφει την εντολή του χρήστη σε string.

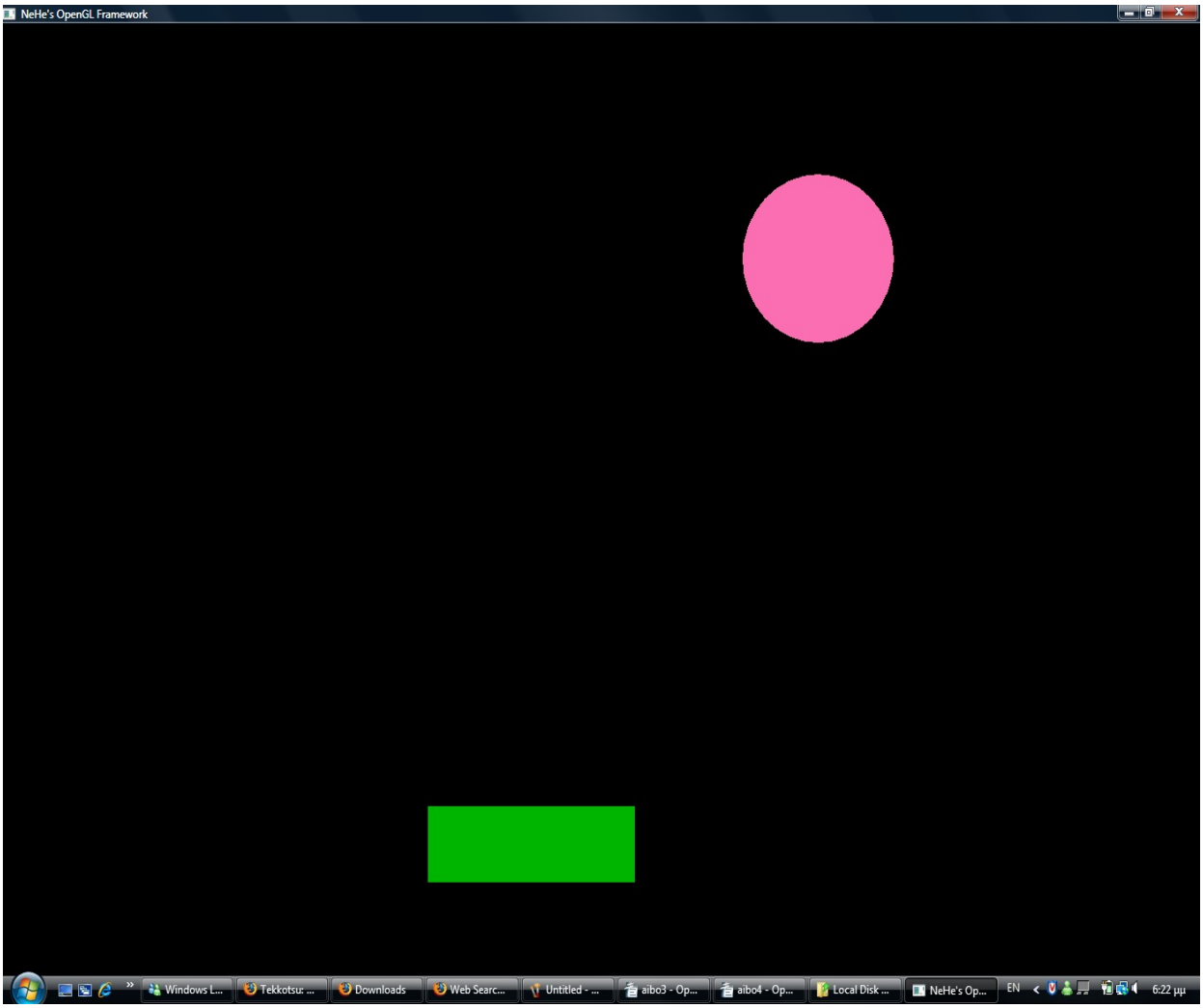
Κεφάλαιο 4

Το Παιχνίδι

4.1 Περιγραφή του Παιχνιδιού

Το παιχνίδι, που επιλέξαμε για να παίξει το AIBO, είναι ένα αρκετά απλό παιχνίδι τύπου Arcanoid. Βασικά στοιχεία του παιχνιδιού είναι η πράσινη μπάρα και η ροζ μπάλα (Εικόνα 4.1). Σκοπός του παιχνιδιού είναι ο χρήστης να μετακινεί την μπάρα, έτσι ώστε η μπάλα που περιφέρεται στην οθόνη να προσκρούσει πάνω σ' αυτήν και να αναπηδήσει.

Οι κανόνες του παιχνιδιού είναι αρκετά απλοί. Η μπάλα δεν πρέπει να φτάσει την κάτω άκρη. Εφόσον γίνει αυτό, ο παίκτης χάνει μία ζωή και το παιχνίδι ξεκινάει από την αρχή. Όταν η μπάλα προσκρούσει σε κάποια άλλη άκρη, αναπηδάει αλλάζοντας κατεύθυνση. Για να μην φτάσει η μπάλα στην κάτω άκρη, πρέπει ο χρήστης να μετακινήσει την μπάρα με την χρήση του πληκτρολογίου, ώστε η μπάλα να προσκρούσει πάνω στην μπάρα και να αναπηδήσει. Η μπάρα μετακινείται προς τα δεξιά με την πίεση των πλήκτρων “Y”, “H”, “B” και οποιοδήποτε κουμπί βρίσκεται πιο δεξιά στο πληκτρολόγιο από τα παραπάνω (δεξί μισό του πληκτρολογίου). Το υπόλοιπο πληκτρολόγιο (δηλαδή τα κουμπιά που βρίσκονται στην αριστερή μεριά) μετακινούν την μπάρα προς τα αριστερά. Η μπάλα μετακινείται αρχικά βορειοανατολικά λόγω παράδοσης στα παιχνίδια τύπου arcanoid. Όποτε η μπάλα αναπηδάει, η γωνία πρόσκρουσης και η ταχύτητά της μεταβάλλονται με τυχαίο τρόπο. Έτσι το παιχνίδι δεν παραμένει το ίδιο με κάθε εκτέλεσή του και δυσκολεύει με το πέρασμα του χρόνου. Το παιχνίδι έχει υλοποιηθεί σε C++ [7] χρησιμοποιώντας βιβλιοθήκες γραφικών OpenGL. Πρέπει να τονιστεί ότι το παιχνίδι εκτελείται στον υπολογιστή ανεξάρτητα από τον κώδικα που εκτελείται στο AIBO και χωρίς καμία επικοινωνία μεταξύ τους.



Εικόνα 4.1 Το παιχνίδι σε windowed mode μεγιστοποιημένο

4.2. OpenGL

Η OpenGL [8] δημιουργήθηκε το 1992 από την Silicon Graphics ως ένα εργαλείο για την επεξεργασία των γραφικών και χρησιμοποιείται μέχρι σήμερα ειδικά στον προγραμματισμό των ηλεκτρονικών παιχνιδιών, όπως στο Quake 3.

Η OpenGL αποτελείται από εκατοντάδες συναρτήσεις και προσφέρει πρόσβαση σε όλες τις δυνατότητες της κάρτας γραφικών. Όλες οι ρουτίνες της OpenGL είναι σχεδιασμένες για χαμηλού-επιπέδου rendering της εικόνας. Φυσικά, αυτό δεν σημαίνει ότι η OpenGL δεν προσφέρει υψηλού-επιπέδου δυνατότητες στον χρήστη. Η βιβλιοθήκη GLU

(OpenGL Utility Library), που περιέχεται στις περισσότερες εκδόσεις της OpenGL, χρησιμοποιεί αυτές τις ρουτίνες για υψηλού-επιπέδου rendering. Μια μηχανή καταστάσεων υποδεικνύει στην OpenGL τι ακριβώς να κάνει.

Για τη χρήση μενού παραθύρων και μηνυμάτων εισόδου-εξόδου προσφέρεται το εργαλείο GLUT (OpenGL Utility Toolkit), εφόσον η OpenGL δεν διαθέτει έτοιμες ρουτίνες για τις παραπάνω δυνατότητες. Το GLUT είναι ένα χρήσιμο και απλό εργαλείο, αλλά δεν προτείνεται για την δημιουργία πολύπλοκων εφαρμογών. Παρόλο που θα ήταν δυνατή η δημιουργία του παιχνιδιού με την χρήση του GLUT, προτιμήσαμε να μην το χρησιμοποιήσουμε.

4.3. Ανάλυση του κώδικα

Η κατανόηση του κώδικα του παιχνιδιού απαιτεί κυρίως γνώσεις μαθηματικών και γεωμετρίας. Το προγραμματιστικό κομμάτι παραμένει σχεδόν ίδιο για την δημιουργία μιας οποιαδήποτε Win32 εφαρμογής εφόσον δημιουργεί το rendering context (hRC) και το device context (hDC). Για να μπορέσουμε να σχεδιάσουμε πάνω σε ένα παράθυρο χρειαζόμαστε το device context, που συνδέει το παράθυρο με το GDI (Graphics Device Interface). Το rendering context συνδέει την OpenGL με το device context. Στην αρχή του παιχνιδιού (Σχήμα 4.1) εμφανίζεται ένα μήνυμα παραθύρου windows, το οποίο ρωτάει τον χρήστη αν θέλει να "τρέξει" το παιχνίδι σε windowed mode ή fullscreen. Για την εκτέλεση του κώδικα προτείνεται το fullscreen, ενώ για debugging το windowed mode είναι πιο χρήσιμο εφόσον μπορούμε να έχουμε σε διπλανό παράθυρο την RawCam του controllergui, εφόσον δεν χρησιμοποιούμε δύο οθόνες. Η μπάλα ξεκινάει την κίνησή της προς τα πάνω δεξιά (αυτό ισχύει από παράδοση στα περισσότερα παιχνίδια τύπου arcanoid). Η μπάλα προσκρούει στις άκρες του παραθύρου (ή της οθόνης για fullscreen) και αλλάζει κατεύθυνση. Οι μεταβλητές, που περιγράφονται παρακάτω, χρησιμοποιούνται στον κώδικα του παιχνιδιού και έχουν κάποιες αρχικές τιμές. Αυτές επηρεάζουν βασικά στοιχεία του παιχνιδιού και οι τιμές τους μπορούν να μεταβληθούν από τον χρήστη.

float angle = 45.0f;	Η αρχική γωνία μεταξύ του διανύσματος κατεύθυνσης της μπάλας και του x άξονα.
float xd = 1;	Η xd προσδιορίζει την αρχική κατεύθυνση της μπάλας στον x άξονα. Το xd παίρνει αποκλειστικά τιμές 1 ή -1 (1 για την κίνηση της μπάλας αρχικά προς τα δεξιά και -1 για την κίνηση της μπάλας αρχικά προς τα αριστερά).

float yd = 1;	Η yd προσδιορίζει την αρχική κατεύθυνση της μπάλας στον y άξονα. Όπως και το xd παίρνει τιμές 1 ή -1. Συνιστάται το 1 για την αρχική κατεύθυνση της μπάλας προς τα πάνω για την ευκολότερη εξέλιξη του παιχνιδιού.
float speed = 0.03f ;	Η speed προσδιορίζει την αρχική ταχύτητα της μπάλας, η οποία αλλάζει στη διάρκεια του παιχνιδιού (όπως εξηγήσαμε στους κανόνες του παιχνιδιού παραπάνω). Προτείνονται θετικές τιμές εφόσον η κατεύθυνση προσδιορίζεται από άλλες μεταβλητές. Επίσης, η τιμή της speed δεν πρέπει να ξεπερνάει κατά πολύ την τιμή της barspeed. Το αντίστροφο όμως επιτρέπεται αν και θα κάνει το παιχνίδι πολύ εύκολο.
float barspeed = 0.04f;	Η barspeed προσδιορίζει την ταχύτητα της μπάρας (ταχύτητα της μπάρας όσο κάποιο πλήκτρο παραμένει πιεσμένο).
int lifes = 10;	Η lifes προσδιορίζει τις ζωές που διαθέτει ο χρήστης.

Η συνάρτηση ReSizeGLScene( ) καλείται όποτε το παράθυρο του παιχνιδιού γίνεται resize, εφόσον δεν τρέχουμε το παιχνίδι σε fullscreen. Η συνάρτηση τρέχει μία φορά τουλάχιστον είτε δουλεύουμε σε πλήρη οθόνη είτε σε παράθυρο. Αυτό γίνεται επειδή η συγκεκριμένη συνάρτηση καθορίζει και την προοπτική της εικόνας (γωνία, βάθος κλπ.).

Η συνάρτηση InitGL( ) καλείται εφόσον έχει δημιουργηθεί το OpenGL παράθυρο. Δουλειά της είναι να αρχικοποιήσει κάποιες μεταβλητές για το OpenGL παράθυρο. Αρχικοποιεί το χρώμα του background σε μαύρο (RGB 0,0,0) και επιτρέπει smooth shading και blending (καλύτερος φωτισμός και μίξη των χρωμάτων μεταξύ τους). Οι υπόλοιπες γραμμές του κώδικα έχουν να κάνουν με την αλλαγή του παιχνιδιού σε 3D. Το παιχνίδι είχε σχεδιαστεί αρχικά σε 3D, αλλά για λόγους απλότητας και εύκολου πειραματισμού με το AIBO προτιμήθηκαν οι 2 διαστάσεις.

Το πιο βασικό κομμάτι του παιχνιδιού είναι η DrawGLScene( ), όπου γίνεται και η απεικόνιση του παιχνιδιού στην οθόνη. Αρχικά δημιουργούμε τα όρια του παραθύρου μας αλλάζοντας την κατεύθυνση της μπάλας, όποτε αυτή φτάσει σε κάποιο από αυτά τα όρια. Ταυτόχρονα αλλάζει τυχαία και η γωνία πρόσκρουσης της μπάλας. Έπειτα λαμβάνουμε την περίπτωση που η μπάλα συγκρούεται με την μπάρα, όπως είχαμε περιγράψει προηγουμένως. Μετά φροντίζουμε ώστε η γωνία πρόσκρουσης να παραμένει μεταξύ 10 και 80 μοιρών. Αν σε κάποια αναπήδηση η γωνία ξεφύγει από το όριο 10-80, θέτουμε την τιμή της στην πλησιέστερη εκ των δύο αυτών τιμών. Η κίνηση της μπάλας καθορίζεται από τις παρακάτω εξισώσεις:

x = (xd * speed * (cos(radians))) + x;

y = (yd * speed * (cos(radians))) + y;

xd και yd	η κατεύθυνση της μπάλας στον x και υ άξονα αντίστοιχα
speed	η ταχύτητα της μπάλας
radians	η γωνία μεταξύ του διανύσματος κατεύθυνσης της μπάλας και του x άξονα σε radians.

Όπως δημιουργήσαμε τα όρια της μπάλας κάνουμε το ίδιο και για την πράσινη μπάρα. Έτσι, δεν επιτρέπουμε στη μπάρα να χαθεί από την οθόνη. Στην μπάρα έχουμε όρια μόνο στον x άξονα, εφόσον αυτή κινείται μόνο εκεί. Μετά ακολουθεί το σχεδιαστικό κομμάτι, όπου δίνουμε μορφή στην ροζ μπάλα και στην πράσινη μπάρα. Θέτουμε το χρώμα σε πράσινο και το σημείο που θα ζωγραφιστεί η μπάρα. Η x συντεταγμένη αυτού του σημείου δεν είναι σταθερή και μεταβάλλεται κάθε φορά που πιέζεται κάποιο από τα πλήκτρα του πληκτρολογίου. Σχεδιάζουμε ένα ορθογώνιο με κέντρο αυτό το σημείο και άκρα (-3, 1) (3,1) (3,-1) (-3,-1), αφού έχουμε θέσει το χρώμα σε πράσινο (τιμές RGB 0, 0.8, 0) και έχουμε την επιθυμητή πράσινη μπάρα.

Για την σχεδίαση της ροζ μπάλας θέτουμε το χρώμα σε ροζ (τιμές RGB 0.99, 0.43, 0.70) και το σημείο που θα σχεδιαστεί η μπάλα. Το σημείο αυτό έχει μεταβλητές x και y συντεταγμένες, που μεταβάλλονται από τις παραπάνω εξισώσεις για την κίνηση της μπάλας. Τέλος, σχεδιάζουμε ένα δίσκο με κέντρο το σημείο αυτό και ακτίνα 2.2 και έχουμε την ροζ μπάλα μας. Χρησιμοποιούμε αυτά τα χρώματα, γιατί είναι “ευδιάκριτα” από το AIBO, εφόσον υπάρχουν συναρτήσεις στο Tekkotsu για ανίχνευση αντικειμένων αυτών των χρωμάτων.

Προσκρούοντας στην άκρη, η γωνία πρόσκρουσης μεταβάλλεται με μια συνάρτηση, προσθέτοντας στην παλιά γωνία μία τυχαία τιμή από -5 έως 5 . Έτσι η γωνία μπορεί να μεταβληθεί είτε θετικά είτε αρνητικά ή και καθόλου. Επίσης, η ταχύτητα της μπάλας αυξάνεται με κάθε πρόσκρουση της μπάλας, δυσκολεύοντας έτσι το παιχνίδι όσο περνάει ο χρόνος. Η μπάλα με το που θα προσκρούσει στην μπάρα θα αλλάξει κατεύθυνση, όπως θα έκανε αν χτυπούσε σε κάποια άκρη του παραθύρου.

Στην μπάρα προσθέτουμε όμως μια επιπλέον μεταβλητή για την πρόσκρουση. Αν η μπάλα χτυπήσει στο κέντρο της μπάρας η πρόσκρουση γίνεται κανονικά (αλλάζει μόνο η y κατεύθυνση). Ενώ αν η μπάλα χτυπήσει στην άκρη της μπάρας θα μεταβληθεί και η x και η y κατεύθυνση της μπάλας. Τέλος, αν η μπάλα προσκρούσει στα πλαϊνά της μπάρας αλλάζει η x κατεύθυνσή της, αλλά η y παραμένει σταθερή, δηλαδή η μπάλα συνεχίζει να κινείται προς τα κάτω μέχρι να φτάσει στην κάτω άκρη, όπου και ο παίκτης χάνει μια ζωή. Το παιχνίδι προειδοποιεί τον χρήστη για αυτό το γεγονός με άλλο ένα μήνυμα παραθύρου. Μετά από αυτό το μήνυμα επανέρχεται το μήνυμα που ρωτάει τον χρήστη αν είναι έτοιμος.... και το παιχνίδι ξεκινάει από την αρχή με μία χαμένη ζωή. Οι ζωές του χρήστη είναι 10, αριθμός που μπορεί να μεταβληθεί από τον κώδικα του παιχνιδιού, όπως και άλλες μεταβλητές που θα έχουμε παρουσιάσει προηγουμένως. Αν οι ζωές του παίκτη μειωθούν κάτω από το μηδέν το παιχνίδι τελειώνει με την εμφάνιση του μηνύματος “Game Over”.

Η συνάρτηση KillGLWindow καλείται εφόσον το πρόγραμμα τερματιστεί. Φροντίζει να απελευθερώσει το rendering context, το device context και οποιοδήποτε κομμάτι μνήμης είχαμε δεσμεύσει στον κώδικά μας.

Η συνάρτηση CreateGLWindow( ) δημιουργεί ένα OpenGL παράθυρο. Εδώ καθορίζεται το ύψος, το πλάτος του παραθύρου, καθώς και η ανάλυση. Αν κάτι δεν πάει καλά στην δημιουργία του παραθύρου θα εμφανιστεί και το αντίστοιχο μήνυμα λάθους. Ο κώδικας της δημιουργίας του παραθύρου είναι πολύπλοκος, αλλά μπορεί να χρησιμοποιηθεί αυτούσιος για τη δημιουργία οποιουδήποτε OpenGL παραθύρου σε συνδυασμό με τις παρακάτω συναρτήσεις. Οι συναρτήσεις αλληλεπιδρούν μεταξύ τους, γι' αυτό και η μία καλεί τις άλλες.

Η συνάρτηση WndProc( ) χειρίζεται όλα τα μηνύματα παραθύρου που εμφανίζονται στην οθόνη. Επίσης, ανιχνεύει ποιά κουμπιά του πληκτρολογίου πιέζονται ή παραμένουν πατημένα.

Η συνάρτηση WinMain( ) δημιουργεί το instance του OpenGL παραθύρου καλώντας την CreateGLWindow με τις επιθυμητές τιμές των παραμέτρων. Επίσης, ανιχνεύει οποιαδήποτε αλληλεπίδραση του χρήστη ή μήνυμα λάθους και ενεργεί κατάλληλα. Όταν κάποιο πλήκτρο πατηθεί (αφού έχουμε ανιχνεύσει ποιο πατήθηκε προηγουμένως) η WinMain( ) προσδιορίζει την μετακίνηση της πράσινης μπάρας κατάλληλα (εκτός βέβαια αν το πλήκτρο ήταν το ESC οπότε το παιχνίδι τερματίζεται).

Κεφάλαιο 5

Προσέγγιση του Προβλήματος

5.1 Ορισμός του προβλήματος

Στόχος της εργασίας αυτής είναι η ανάπτυξη λογισμικού για το AIBO, έτσι ώστε να μπορεί παρακολουθώντας την κίνηση κάποιων αντικειμένων μέσω της κάμερας του (στην περίπτωση μας την πράσινη μπάρα και την ροζ μπάλα) να αντιδράει κατάλληλα (στην περίπτωση μας να κουνάει τα μπροστινά του πόδια με τέτοιον τρόπο, ώστε να πιέζει τα κατάλληλα πλήκτρα στο πληκτρολόγιο ενός υπολογιστή).

5.2. Όραση του AIBO

Το AIBO είναι εξοπλισμένο με μία έγχρωμη κάμερα, η οποία μπορεί να λαμβάνει 30 εικόνες το δευτερόλεπτο διαστάσεων 208x160. Η κάμερα χρησιμοποιεί αισθητήρα τεχνολογίας CMOS μεγέθους 0.25in και ευρυγώνιο φακό διαφράγματος f/2,8 που της επιτρέπει να καταγράφει πεδίο 56,9 μοιρών στον οριζόντιο άξονα και 45,2 μοιρών στον κατακόρυφο. Οι προκαθορισμένες τιμές της εξισορρόπησης λευκού και της ταχύτητας φωτοφράχτη είναι 5000 βαθμοί Kelvin και 1/100 του δευτερολέπτου αντίστοιχα. Ανάλογα με τις συνθήκες φωτισμού, η εξισορρόπηση λευκού μπορεί να ρυθμιστεί στους 2856 ή στους 6500 βαθμούς Kelvin. Αντίστοιχα, ανάλογα με την ένταση του φωτός, η ταχύτητα φωτοφράχτη μπορεί να ρυθμιστεί στο 1/50 ή 1/200 του δευτερολέπτου. Έτσι οι δυνατότητες παρακολούθησης αντικειμένων δεν περιορίζονται μόνο στην αναγνώριση χρωμάτων, αν και η εργασία μας χρησιμοποιεί κυρίως αυτή τη μέθοδο για μεγαλύτερη ευκολία επεξεργασίας των αποτελεσμάτων και ταχύτερη αναγνώριση των αντικειμένων, που είναι πολύ σημαντικά για την ροή της εργασίας.

Η όραση του AIBO επηρεάζεται πολύ από τον φωτισμό. Ειδικότερα για να μπορέσει το AIBO να διακρίνει εύκολα την ροζ μπάλα και την πράσινη μπάρα, και μετά από εύρεση των κατάλληλων τιμών RGB για ευκολότερη ανίχνευση των παραπάνω από το AIBO, ανακαλύψαμε ότι απαιτείται χαμηλός φωτισμός.

Αυτό συμβαίνει διότι η οθόνη του υπολογιστή είναι πολύ φωτεινή και με δυνατό φωτισμό είναι αδύνατη η ανίχνευση χρωμάτων και αντικειμένων σ' αυτή. Επίσης, στο controllergui ο χρήστης πρέπει να προσθέσει δυο γραμμές εντολών στη γραμμή εισόδου Send Input για την ρύθμιση της κάμερας:

```
!set vision.shutter_speed=mid
!set vision.gain=low
```

Κυριότερο πρόβλημα στην όραση του AIBO ήταν η εύρεση των κατάλληλων ρυθμίσεων της κάμερας, καθώς και των κατάλληλων χρωμάτων (τιμές RGB) για την μπάρα και την μπάλα, έτσι ώστε να γίνεται γρήγορη και εύκολη ανίχνευση των αντικειμένων αυτών.

Το AIBO αρχικά ψάχνει να εντοπίσει την ροζ μπάλα κουνώντας το κεφάλι του, κάνοντας μία κυκλική κίνηση. Όταν η μπάλα βρεθεί, το AIBO κεντράρει την μπάλα στην κάμερα του, παρακολουθώντας έτσι την κίνησή της. Όταν η μπάλα για κάποιο λόγο χαθεί από την οθόνη το AIBO ξαναεκτελεί μια κυκλική κίνηση με το κεφάλι του, αναζητώντάς την ξανά. Έτσι, το AIBO πρέπει να βρίσκεται κοντά στην οθόνη του υπολογιστή για να κεντράρει στην μπάλα, αφού πρώτα έρθει στην αρχική του θέση "startup.pos" (το posture file που έχει αποθηκευμένο τις τιμές των μελών του AIBO). Εφόσον σκοπός μας δεν ήταν η ανίχνευση της οθόνης και του πληκτρολογίου από το AIBO, ούτε η κατάλληλη τοποθέτησή του πάνω από τα πλήκτρα του πληκτρολογίου, ο χρήστης τοποθετεί το AIBO πάνω από τα πλήκτρα.

Έπειτα προκύπτει το πρόβλημα του ταυτόχρονου εντοπισμού της μπάλας και της μπάρας. Αν και το AIBO "βλέπει" το μεγαλύτερο μέρος της οθόνης και τα αντικείμενα είναι αρκετά μεγάλα για να μην εντοπιστούν, προσθέσαμε μεταβλητές που αποθηκεύουν τις συντεταγμένες των αντικειμένων αυτών, όποτε αυτά ανιχνεύονται. Παρ' όλα αυτά με την κίνηση των ποδιών του AIBO πάνω στα πλήκτρα του πληκτρολογίου, η μπάρα μετακινείται χωρίς να ανανεώνονται οι συντεταγμένες της μπάρας (έως ότου αυτή εντοπιστεί). Γι' αυτό το λόγο επιτρέψαμε στο AIBO να πιέζει τα πλήκτρα για να μετακινήσει τη μπάρα, μόνο εφόσον η μπάλα έχει αναπηδήσει και κινείται προς τα κάτω. Επίσης, περιμένουμε να βρεθεί η κατάλληλη θέση που πρέπει να μετακινηθεί η μπάρα για να αναπηδήσει η μπάλα πάνω της, πρωτού δώσουμε εντολή στο AIBO να πιέσει κάποιο πλήκτρο. Για να επιβεβαιώσουμε ότι η μπάρα έχει μετακινηθεί στη κατάλληλη θέση, πρέπει το AIBO να εντοπίσει τη μπάρα. Εφόσον το κεφάλι του AIBO, οπότε και η κάμερά του, ακολουθούν τη κίνηση της μπάλας, η μπάλα πρέπει να κατέβει σε ένα ύψος όπου θα είναι ορατή και η μπάρα από το AIBO. Με αυτόν τον τρόπο είμαστε σίγουροι ότι η μπάρα βρίσκεται στην κατάλληλη θέση, ενώ η μπάλα βρίσκεται αρκετά ψηλά (όπου το AIBO δεν κινδυνεύει να χάσει). Το AIBO υπολογίζει την τελική θέση της μπάλας και πιέζει τα κατάλληλα πλήκτρα του πληκτρολογίου έως ότου η μπάρα φτάσει σ' αυτήν τη θέση. Έτσι δεν γίνεται σπατάλη κινήσεων από το AIBO.

5.3 Αντιμετώπιση του προβλήματος

Η κλάση-συμπεριφορά που χρησιμοποιούμε ονομάζεται PlayBall και είναιυποκλάση της BehaviorBase. Οι μεταβλητές-μέλη της είναι οι ακόλουθες:

private:

float ball_x, ball_y ,όπου αποθηκεύονται οι x και y συντεταγμένες της μπάλας.

float bar_x, bar_y ,όπου αποθηκεύονται οι x και y συντεταγμένες της μπάρας

float select_x, past_select ,που χρησιμοποιούνται για την λήψη αποφάσεων, δηλαδή ποιο πλήκτρο πρέπει να πατηθεί, ώστε η μπάρα να μετακινηθεί προς τη σωστή κατεύθυνση και να εμποδίσει τη μπάλα απ' το να φτάσει στο κάτω άκρο (για να είμαστε και πιο ακριβείς ποιο πόδι πρέπει να μετακινηθεί).

protected:

MotionManager::MC_ID head_id ,που αποθηκεύει το motion command που καθορίζει την κίνηση του κεφαλιού του AIBO.

MotionManager::MC_ID mseq_id ,που αποθηκεύει το motion command που καθορίζει την κίνηση του υπόλοιπου σώματος του AIBO (κυρίως των μπροστινών του ποδιών)

MotionManager::MC_ID track_id ,που αποθηκεύει το motion command που καθορίζει την κίνηση του κεφαλιού του AIBO, όποτε η μπάλα δεν βρίσκεται στο οπτικό πεδίο του AIBO.

Οι συναρτήσεις-μέλη της PlayBall δεν είναι άλλες από τις:

public:

virtual void Dostart( )

virtual void DoStop( )

virtual void processEvent(const EventBase& event)

Στην DoStart αρχικοποιούμε τις μεταβλητές-μέλη της κλάσης σε 0 και δημιουργούμε τα MotionCommands, που θα χρειαστούν. Το head_id χρησιμοποιείται για την κίνηση του κεφαλιού του AIBO, έτσι ώστε να παρακολουθεί την ροζ μπάλα. Το mseq_id είναι υπεύθυνο για την κίνηση του σώματος του AIBO στην αρχική του στάση και για την κίνηση των ποδιών. Το track_id, που προστέθηκε αργότερα, κουνάει το κεφάλι του AIBO ψάχνοντας για την ροζ μπάλα στο οπτικό του πεδίο κάνοντας μια κυκλική κίνηση με το κεφάλι του. Αυτή η κίνηση σταματάει εφόσον εντοπιστεί η ροζ μπάλα. Τέλος, προσθέτουμε τα events που θέλουμε να παρακολουθούμε, που δεν είναι άλλα από τον εντοπισμό της ροζ μπάλας και της πράσινης μπάρας.

Στην Dostop απελευθερώνουμε τα MotionCommands που είχαμε χρησιμοποιήσει και καλούμε την BehaviorBase::DoStop( ).

Στην processEvent ασχολούμαστε μόνο με τα event τύπου VisionObjectEvent. Τα VisionObjectEvent που δημιουργούνται, όποτε εντοπίζεται η ροζ μπάλα ή η πράσινη μπάρα, διαφέρουν στο sourceID. Τα events της ροζ μπάλας έχουν sourceID ίσο με 0, ενώ της πράσινης μπάρας έχουν sourceID ίσο με 2. Όταν εντοπιστεί η ροζ μπάλα αποδεσμεύουμε το track_id. Μετά θέτουμε σε κίνηση το κεφάλι του AIBO, ώστε η ροζ μπάλα να βρίσκεται στο κέντρο του οπτικού πεδίου της κάμερας. Αυτό γίνεται μέσω του head_id και των παρακάτω εξισώσεων:

```
ball_horiz_angle = visev.getCenterX( ) * CameraFOV/2;

ball_vert_angle = visev.getCenterY( ) * CameraFOV/2;

head_pan_angle = state->outputs[ HeadOffset + PanOffset ];

head_tilt_angle = state->outputs[ HeadOffset + TiltOffset ];

new_pan = head_pan_angle  - ball_horiz_angle;

new_tilt = head_tilt_angle – ball_vert_angle;
```

CameraFOV είναι το οπτικό πεδίο της κάμερας.

HeadOffset + PanOffset καθορίζει την θέση του κεφαλιού του AIBO στον x άξονα.

HeadOffset + TiltOffset καθορίζει την θέση του κεφαλιού του AIBO στον y άξονα.

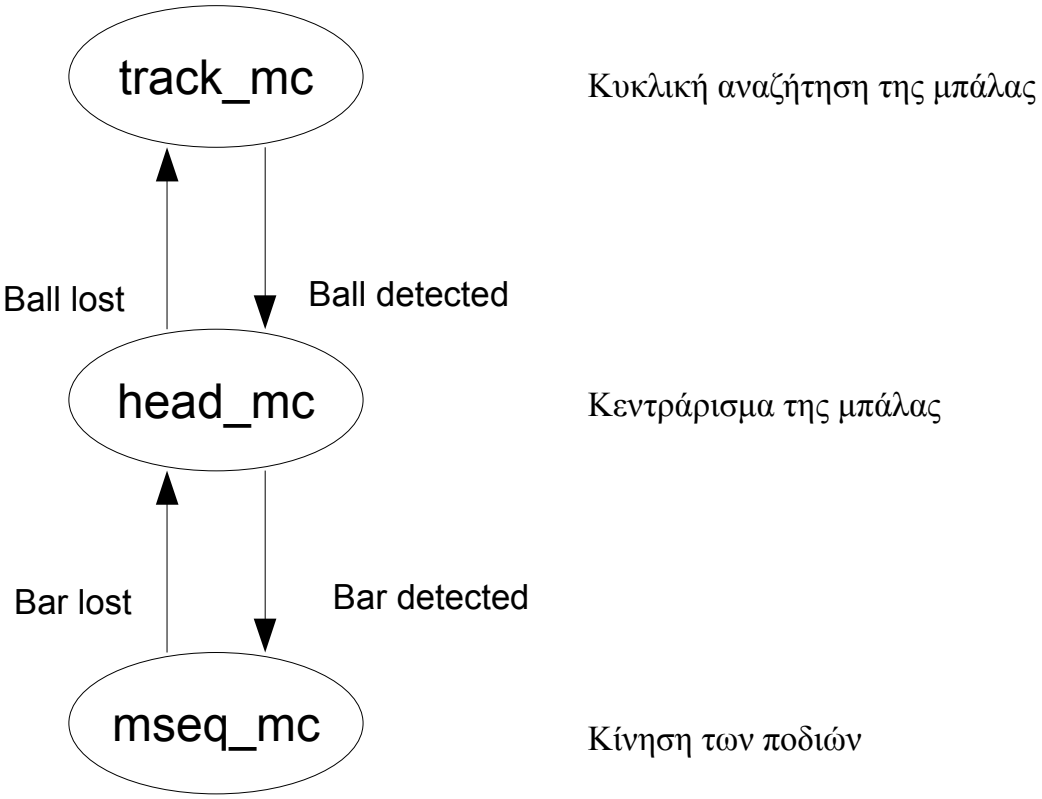
new_pan η νέα θέση του κεφαλιού του AIBO στον x άξονα.

new_tilt η νέα θέση του κεφαλιού του AIBO στον y άξονα.

Έπειτα αποθηκεύουμε τις συντεταγμένες της μπάλας πριν την κίνηση του κεφαλιού του AIBO. Αυτό γίνεται κυρίως για λόγους debugging, εφόσον μετά την κίνηση του κεφαλιού του AIBO η μπάλα βρίσκεται στο κέντρο του οπτικού πεδίου της κάμερας.

Με την εντολή "MMAccessor<HeadPointerMC>(head_id)->setJoints(new_tilt, new_pan, 0.6);" κινούμε το κεφάλι στην επιθυμητή θέση. Ο MMAccesor φροντίζει ώστε να γίνουν αλλαγές στις τιμές των αρθρώσεων του AIBO, ενώ γίνεται η κίνηση του κεφαλιού. Το 0.6 χρησιμοποιείται για τη διατήρηση της θέσης του λαιμού σταθερή.

Όταν εντοπιστεί η πράσινη μπάρα αποθηκεύουμε τις συντεταγμένες της. Αργότερα υπολογίζουμε την κίνηση της μπάλας εφόσον αυτή κατεβάινει, έχοντας υπόψιν μας την τωρινή θέση και μία προηγούμενή της. Προβλέπουμε τη θέση (τη χ-συντεταγμένη που θα χει η μπάλα όταν φτάσει στο ύψος που βρίσκεται η μπάρα) που πρέπει να μετακινηθεί η μπάρα, για να προσκρούσει πάνω της η μπάλα και να αναπηδήσει. Ανάλογα αν η μπάρα βρίσκεται δεξιά ή αριστερά από την προβλεπόμενη θέση της, κινούμε τα πόδια του AIBO με τη χρήση του mseq_id. Έτσι, αν η μπάρα πρέπει να μετακινηθεί δεξιά σηκώνουμε το αριστερό πόδι και κατεβάζουμε το δεξιό (αυτό γίνεται για να μην ανεβοκατεβάζουμε συνέχεια τα πόδια του AIBO και να έχουμε σπασμωδικές κινήσεις). Το ανεβοκατέβασμα των μπροστινών ποδιών έχει ως στόχο την πίεση των πλήκτρων του πληκτρολογίου (εφόσον το AIBO έχει τοποθετηθεί σωστά από τον χρήστη). Αναλόγως πράττουμε και στην περίπτωση που η μπάρα πρέπει να μετακινηθεί αριστερά. Σε οποιαδήποτε άλλη περίπτωση σηκώνουμε και τα δύο πόδια.



Κεφάλαιο 6

Αποτελέσματα

6.1 Αποτελέσματα

Τα αποτελέσματα της συγκεκριμένης εφαρμογής εξαρτώνται σε μεγάλο βαθμό από το ποσοστό επιτυχίας της διαδικασίας ανίχνευσης. Επίσης απαιτείται κατάλληλη τοποθέτηση του AIBO μπροστά στην οθόνη και πάνω από το πληκτρολόγιο (Εικόνα 6.1). Δοκιμάστηκαν διαφορετικά χρώματα μπάλας και μπάρας, καθώς και διαφορετικός φωτισμός σε συνδυασμό με διαφορετικές ρυθμίσεις της κάμερας του AIBO. Όσο πιο φωτεινή είναι μια οθόνη τόσο χαμηλότερος φωτισμός απαιτείται.

Το AIBO παρακολουθεί την μπάλα και κουνάει το κεφάλι του, έτσι ώστε η μπάλα να βρίσκεται πάντα στο κέντρο της κάμερας του. Όταν η μπάλα αρχίσει να κινείται προς τα κάτω, υπολογίζουμε τη θέση που πρέπει να μετακινηθεί η μπάρα (γνωρίζοντας τη θέση της και μία προηγούμενή της), ώστε η μπάλα να προσκρούσει πάνω της και να αναπηδήσει.. Η πρόβλεψη της κίνησης της μπάλας γίνεται εφόσον η ταχύτητα και η κατεύθυνση της παραμένουν σταθερές κατα τη διάρκεια της κίνησης. Η ταχύτητα και η κατεύθυνση της μπάλας αλλάζει μετά από κάθε αναπήδηση. Γι' αυτό προβλέπουμε την κίνηση της μπάλας μετά από κάθε αναπήδηση. Αν η τελική θέση που πρέπει να μετακινηθεί η μπάρα βρίσκεται προς τα δεξιά , το AIBO αφού εντοπίσει την μπάρα για να παρακολουθεί την κίνηση της, σηκώνει το αριστερό του πόδι (εφόσον αυτό είναι σηκωμένο) και κατεβάζει το δεξιό του. Μόλις η μπάρα φτάσει στην κατάλληλη θέση σηκώνει και τα δύο του πόδια. Ανάλογα πράττουμε αν η τελική θέση που πρέπει να μετακινηθεί η μπάρα βρίσκετα προς τα αριστερά (Εικόνα 6.2). Όση ώρα το AIBO παίζει, το παιχνίδι δυσκολεύει, εφόσον η ταχύτητα της μπάλας αυξάνεται χωρίς να αυξάνεται η ταχύτητα της μπάρας. Έτσι ύστερα από ένα χρονικό διάστημα το παιχνίδι καταντάει αρκετά δύσκολο για το AIBO. Παρ'όλα αυτά ο χρόνος αυτός είναι αρκετά μεγάλος, εφόσον το παιχνίδι δεν δυσκολεύει απότομα. Όμως, εφόσον το AIBO έχει ορισμένες ζωές και το παιχνίδι γίνεται reset όποτε χάνεται μια ζωή, το AIBO θα παίζει αρκετή ώρα πριν χάσει (το παιχνίδι καταντάει κουραστικό μιας και είναι αρκετά απλό, ενώ το AIBO σπαταλάει την μπαταρία του!).

Στην περίπτωσή μας το AIBO πρέπει να προλαβαίνει να υπολογίσει την κατάλληλη θέση που πρέπει να μετακινηθεί η μπάρα, ώστε η μπάλα να προσκρούσει πάνω της και να αναπηδήσει, πρωτού η μπάλα φτάσει στη θέση αυτή. Απαιτείται γρήγορη πρόβλεψη της θέσης αυτής καθώς και γρήγορη κίνηση των ποδιών του AIBO, ώστε η μπάρα να μετακινηθεί στη θέση αυτή. Έτσι το AIBO με έναν καλύτερο αλγόριθμο για να προβλέπει νέες θέσεις της μπάλας και έναν αλγόριθμο που θα επιτρέπει μετακίνηση της μπάρας με την ελάχιστη δυνατή κίνηση του AIBO, θα μπορεί να τρέχει και να εκτελεί το παιχνίδι όσο του επιτρέπει η μπαταρία και με καλύτερη αξιοποίηση της ενέργειας αυτής. Οπότε το AIBO μπορεί να παίζει το παιχνίδι αξιόπιστα! Διαπιστώσαμε ότι ο χρόνος ανταπόκρισης και κίνησης του AIBO δεν του επιτρέπει να αντεπεξέλθει σε παιχνίδια υψηλών ταχυτήτων. Παρόμοιος προγραμματισμός μπορεί να επιτρέψει στο AIBO να παίξει ένα πιο πολύπλοκο παιχνίδι, αφού πρώτα βρεθεί κατάλληλος αλγόριθμος για το παιχνίδι αυτό από την πλευρά του AIBO.

Εικόνα 6.1 Τοποθέτηση του AIBO πριν ξεκινήσει το παιχνίδι

Εικόνα 6.2 Το AIBO μετακινεί τη μπάρα προς τα αριστερά, όπου κατευθύνεται η μπάλα

Κεφάλαιο 7

Μελλοντικές προεκτάσεις

7.1. Αναβάθμιση της συμπεριφοράς

Επιπρόσθετες λειτουργίες στο AIBO μπορούν να προγραμματιστούν επιτρέποντας έτσι την ανεξαρτησία του AIBO από τον χρήστη. Το AIBO δεν είναι ανάγκη να επιβλέπεται από τον χρήστη για να μπορεί να παίξει σωστά το παιχνίδι. Αφού τοποθετηθεί το AIBO κατάλληλα και ο χρήστης συνδεθεί με το ρομπότ, τρέχοντας το controllergui στο ip του ρομπότ, ο χρήστης πρέπει να εκκινήσει το παιχνίδι στον υπολογιστή σε fullscreen.

Με τη δημιουργία μιας συμπεριφοράς, που τρέχει πριν την εκτέλεση του παιχνιδιού, το AIBO μπορεί να εντοπίζει την οθόνη του υπολογιστή και το πληκτρολόγιο και να παίρνει την κατάλληλη θέση μόνο του πάνω από αυτά. Το AIBO δεν πρέπει όμως να μετακινείται κατα τη διάρκεια του παιχνιδιού. Εφόσον γίνει αυτό, το AIBO θα πρέπει να ψάξει για το πληκτρολόγιο και να επανατοποθετηθεί κατάλληλα. Αυτή η διαδικασία απαιτεί την στροφή της κάμερας στο πληκτρολόγιο και την κίνηση του AIBO ώστε κάποιο από τα μπροστινά του πόδια να βρίσκεται ακριβώς πάνω από αυτό. Αυτό φυσικά δεν μπορεί να γίνεται κατά τη διάρκεια του παιχνιδιού, γιατί όσο γίνεται αυτή η κίνηση του AIBO, η μπάλα συνεχίζει να κινείται ενώ το AIBO ψάχνει για κάποιο πλήκτρο. Για να ανταποκρίνεται στο παιχνίδι και να μην χάνεται χρόνος στην εύρεση του πληκτρολογίου, πρέπει το AIBO να παραμένει στην αρχική του θέση.

Μία τέτοια "αναβάθμιση" της συμπεριφοράς, θα επέτρεπε στον χρήστη να αφήνει το AIBO να παίζει το παιχνίδι μόνο του. Ο χρήστης απλά ανοίγει τον υπολογιστή, τρέχει το παιχνίδι, ανοίγει το AIBO και επιλέγει τη συμπεριφορά από το controllergui.

7.2 Μελλοντικές προεκτάσεις

Όπως προαναφέραμε το παιχνίδι που παίζει το AIBO είναι αρκετά μονότονο και αργό. Επίσης ο άνθρωπος δεν παίζει με το AIBO, απλά παρατηρεί την επίδοση του σε αυτό. Ένα πιο ενδιαφέρον ηλεκτρονικό παιχνίδι, για τον άνθρωπο και το ρομπότ, πρέπει να επιτρέπει την ύπαρξη δύο παικτών. Είτε οι δύο παίκτες παίζουν σαν ομάδα είτε ανταγωνιστικά, το παιχνίδι προσφέρει περισσότερη διασκέδαση. Επίσης οι επιδόσεις των δύο παικτών βελτιώνονται με ταχύτερο ρυθμό, εφόσον υπάρχει ένα μέτρο σύγκρισης. Ένα ακόμα μέτρο σύγκρισης, όπως ένας βαθμολογικός πίνακας, στον οποίο θα αποθηκεύονται οι υψηλότερες επιδόσεις, δημιουργεί ένα πιο έντονο ανταγωνιστικό κλίμα. Φυσικά για να γίνει εφικτή η ικανοποιητική εκτέλεση ενός τέτοιου παιχνιδιού από το AIBO ή κάποιο άλλο ρομπότ, απαιτείται τρομερή αξιοποίηση των δυνατοτήτων του ρομπότ και μαθηματική προμελέτη για την εύρεση κατάλληλου αλγόριθμου εκτέλεσής του. Συνήθως, όπως συμβαίνει στη δική μας περίπτωση, απαιτείται ο σχεδιασμός και ο προγραμματισμός του παιχνιδιού. Αυτό εξαρτάται από τις δυνατότητες του ρομπότ… ειδικότερα της κάμεράς του και τη δυνατότητα αναγνώρισης κινούμενων αντικειμένων.

Με λίγα λόγια, τα περισσότερα παιχνίδια σχεδιάζονται για να εκτελούνται από τον άνθρωπο και όχι από ένα ρομπότ (π.χ. η ύπαρξη μιας μικροσκοπικής μπάλας, που κινείται ταχύτατα, είναι απίθανο να αναγνωριστεί εγκαίρως από την κάμερα του ρομπότ). Μια καλή εφαρμογή των παραπάνω θα ήταν η χρήση ενός κατάλληλου ρομπότ για την εκτέλεση του πασίγνωστου παιχνιδιού Pac-Man. Ευτυχώς υπάρχει σχετική εργασία (κεφάλαιο 8.3), η οποία έχει αναλάβει τη δημιουργία ενός αυτόνομου πράκτορα για την κίνηση του κεντρικού ήρωα. Η δυνατότητα χρήσης κάποιου ρομπότ, για την κίνηση του ήρωα, περιορίζεται μόνο από τα τεχνικά χαρακτηριστικά του ρομπότ που θα επιλέξουμε.

Κεφάλαιο 8

Σχετικές Εργασίες

8.1 Γενικές Αναφορές

Η εργασία που ενέπνευσε την ανάπτυξη αυτής της διπλωματικής εργασίας υπάρχει στο site του Tekkotsu [12]. Το AIBO παίζει ένα τυχερό παιχνίδι k - bandit (Σχήμα 7.1) προσπαθώντας να μεγιστοποιήσει τις πιθανότητες νίκης του. Πιο συγκεκριμένα υπάρχουν δύο πιθανοτικές μηχανές, με διαφορετικές πιθανότητες η κάθε μια. Το AIBO πιέζει τα πλήκτρα που βρίσκονται στο αριστερό μέρος του πληκτρολογίου για τη μηχανή με την μικρότερη πιθανότητα, και τα υπόλοιπα πλήκτρα για τη μηχανή με τη μεγαλύτερη πιθανότητα. Το AIBO δεν γνωρίζει τις πιθανότητες αυτές, αλλά παίζοντας το παιχνίδι "μαθαίνει" ποια μηχανή έχει μεγαλύτερη πιθανότητα και στη συνέχεια επιλέγει μόνο αυτή. Στη δική μας εργασία το παιχνίδι είναι πιο πολύπλοκο, αφού απαιτεί πιο γρήγορη λήψη αποφάσεων και κινήσεων από το AIBO.

Εικόνα 8.1 Στιγμιότυπο από βίντεο, όπου το AIBO παίζει το k-bandit

Εργασίες που χρησιμοποιούν το AIBO για την αναγνώριση κάποιων αντικειμένων μέσω της κάμερας υπάρχουν πολλές. Αρκετές από αυτές χρησιμοποιούνται για τον προγραμματισμό του AIBO, ώστε να μπορέσει να "παίξει" κάποιο παιχνίδι (ένας σκύλος για να είναι αξιαγάπητος πρέπει να είναι υπάκουος, πιστός και παιχνιδιάρης). Το πιο διάσημο από αυτά τα παιχνίδια είναι το ποδόσφαιρο. Γι' αυτό δημιουργήθηκε ένα ετήσιο διεθνές τουρνουά ποδοσφαίρου, το RoboCup, όπου σε ένα από τα πιο διάσημα πρωταθλήματα οι παίκτες δεν είναι άλλοι από τα τετράποδα ρομπότ AIBO. Κάθε ομάδα αποτελείται από τέσσερα ρομπότ AIBO και την ερευνητική-προγραμματιστική ομάδα. Προκειμένου να διαγωνισθεί επιτυχώς, κάθε ρομπότ καλείται να επιλύσει μια σειρά προβλημάτων: οπτική αναγνώριση αντικειμένων, μετακίνηση με βηματισμό, εντοπισμός θέσης στο γήπεδο και ομαδική συντονισμένη συμπεριφορά. Πολλές από αυτές τις ερευνητικές ομάδες πραγματοποιούν τις διπλωματικές τους εργασίες στα πλαίσια του RoboCup. Στο Πολυτεχνείο Κρήτης υπάρχει ομάδα που συμμετέχει στο RoboCup, με το όνομα Κουρήτες.

8.2 Οπτική Αναγνώριση Αντικειμένων

Μια από αυτές τις διπλωματικές εργασίες επικεντρώνεται στο πρόβλημα της οπτικής αναγνώρισης ορισμένων αντικειμένων στο γήπεδο στο RoboCup 2007 [9]: τα δύο τέρματα, τα δύο beacons (χρωματιστά κολωνάκια) και η μπάλα. Το μοντέλο που προτείνεται για επεξεργασία εικόνας επεξεργάζεται την χρωματικά τμηματοποιημένη εικόνα της κάμερας (σαν αυτή που φαίνεται στην εικόνα 3.2 τραβηγμένη από τη SegCam του controllergui) και δίνει ως έξοδο τον τύπο του αντικειμένου που αναγνωρίστηκε, καθώς και μία εκτίμηση της απόστασης στην οποία βρίσκεται και της γωνίας σε σχέση με το ρομπότ. Αυτή η επεξεργασία της εικόνας, επαναλαμβάνεται συνέχεια επιτρέποντας στο AIBO να έχει πλήρη ιδέα του χώρου όπου βρίσκεται (προς τα που βρίσκονται τα τέρματα, οι παίκτες και η μπάλα). Η επεξεργασία αυτή βασίζεται σε μια σειρά από διεργασίες οι οποίες χρησιμοποιούνται και για τους τρεις τύπους αντικειμένων. Οι διεργασίες αυτές κάνουν οριζόντια και κάθετη σάρωση ολόκληρης της εικόνας, προσδιορισμό μεγάλων χρωματικών περιοχών μέσω ενός πεπερασμένου αυτομάτου, συγκέντρωση των χρωματικών περιοχών με τη βοήθεια ιστογραμμάτων και σχηματισμό ενός περιγράμματος του αντικειμένου. Τα αντικείμενα που εντοπίζονται δεν είναι αναγκαστικά κάποια από τα ενδιαφερόμενα αντικείμενα. Οπότε, ύστερα από ένα προσαρμοσμένο φιλτράρισμα, τα αντικείμενα που δεν ανταποκρίνονται στους περιορισμούς του φιλτραρίσματος (περιορισμός σε σχήματα και σε χρώματα) δεν λαμβάνονται υπόψη. Η προσέγγιση αυτή έχει χρησιμοποιηθεί με επιτυχία από την ομάδα Κουρήτες του Πολυτεχνείου Κρήτης κατά τη διάρκεια διαφόρων διοργανώσεων RoboCup.

Άλλη μια διπλωματική εργασία είχε στόχο την οπτική αναγνώριση προσώπων [10], εστιάζοντας στην αλληλεπίδραση ανθρώπου με ρομπότ. Όπως είναι γνωστό η επικοινωνία μεταξύ ανθρώπων, χωρίς τη χρήση της τεχνολογίας, γίνεται πρόσωπο με πρόσωπο. Στόχος της εργασίας ήταν η ανάπτυξη λογισμικού για το ρομποτικό τετράποδο AIBO ώστε να επιδεικνύει ανάλογη συμπεριφορά στην αλληλεπίδρασή του με ανθρώπους. Συγκεκριμένα, το ρομπότ προσπαθεί να ανιχνεύσει κάποιο ανθρώπινο πρόσωπο στο οπτικό του πεδίο και να εστιάσει το βλέμμα του σ' αυτό, ώστε τα μάτια του ανθρώπου να φαίνονται στην κάμερά του. Στην προσπάθειά του αυτή καλείται να χρησιμοποιήσει όχι μόνο τις κινήσεις του κεφαλιού, αλλά και τη δυνατότητα μετακίνησής του στο χώρο. Έτσι, εάν το πρόσωπο φαίνεται από κάποια άλλη οπτική γωνία (όπου δεν φαίνεται το μέτωπο, αλλά αναγνωρίζεται το ανθρώπινο πρόσωπο) να αναγνωρίζει την κατάλληλη κατεύθυνση μετακίνησής του σώματος και του κεφαλιού του, ώστε να βλέπει τον άνθρωπο κατάματα. Στη δική μας εργασία η αναγνώριση αντικειμένων έγινε καθαρά βάση χρώματος, ωστόσο σε άλλα παιχνίδια θα έπρεπε να λαμβάνονται υπόψη και άλλα χαρακτηριστικά του αντικειμένου (σχήμα, διάνυσμα κίνησης). Εμείς βασιστήκαμε στο χρώμα για την αναγνώριση αντικειμένων, διότι δεν υπάρχουν αντικείμενα ίδιου χρώματος και διαφορετικού σχήματος στο παιχνίδι. Έτσι το χρώμα αρκεί για τη διαφοροποίηση των αντικειμένων μεταξύ τους. Επίσης το χρώμα αποτελεί και τον πιο εύκολο και έμπιστο τρόπο διαφοροποίησης των αντικειμένων, ειδικά στο Tekkotsu.

8.3 Προγραμματισμός Παιχνιδιού

Μία τέτοια διπλωματική εργασία είχε ως αντικείμενο το σχεδιασμό ενός αυτόνομου πράκτορα-παίκτη του παιχνιδιού Pac-Man, καθώς και την παραμετρική υλοποίηση του ίδιου του παιχνιδιού [11]. Πιο συγκεκριμένα, επινοήθηκε ένας αλγόριθμος, ο οποίος ευθύνεται για την κίνηση του κεντρικού ήρωα (pacman), έχοντας ως πληροφορία τη συνολική κατάσταση που επικρατεί οποιαδήποτε στιγμή μέσα στο λαβύρινθο (τοποθεσία των φαντασμάτων, των υπολειπόμενων κουκίδων και των υπολειπόμενων δυναμωτικών κουκίδων). Η βασική ιδέα έγκειται στην ομοιόμορφη αναπαράσταση όλων των στοιχείων του λαβυρίνθου ανά πάσα στιγμή. Έτσι έχοντας ένα "χάρτη" του λαβυρίνθου, ο πράκτορας αποφασίζει για την καταλληλότερη κίνηση του ήρωα. Ως αποτέλεσμα, ο αλγόριθμος λειτουργεί εξίσου καλά ανεξάρτητα από τη μορφολογία του λαβυρίνθου. Η παραμετρική υλοποίηση επιτρέπει στον εκάστοτε σχεδιαστή να διαμορφώσει εύκολα όλα τα χαρακτηριστικά του παιχνιδιού σύμφωνα με τις προτιμήσεις του, επιτρέποντας έτσι την δημιουργία ενός ακόμα παιχνιδιού-κλώνου του Pac-Man. Σε αντίθεση με την περίπτωση μας, εδώ η επικοινωνία του πράκτορα με το παιχνίδι δε γίνεται μέσω του φυσικού κόσμου, αλλά μέσω του κοινού λογισμικού.

Κεφάλαιο 9

Επίλογος

Από αυτή την εργασία συμπεραίνουμε ότι είναι δυνατόν ένα ρομπότ να παίξει ένα παιχνίδι όπως ένας άνθρωπος. Παιχνίδια που βασίζονται στην ταχύτητα και μεθοδικότητα του παίκτη μπορεί εύκολα να τα παίξει ένα ρομπότ, με την εύρεση του κατάλληλου αλγόριθμου. Αυτό δε συμβαίνει με όλα τα παιχνίδια. Πολλά από αυτά απαιτούν παραπάνω ικανότητες από τον παίκτη όπως τη φαντασία, τη πονηριά και τον αυτοσχεδιασμό. Ικανότητες που δε διαθέτει κανένα ρομπότ. Εκτός αυτού πολλά ρομπότ δε διαθέτουν τα κατάλληλα τεχνικά χαρακτηριστικά για την εκτέλεση ακόμα και των πιό απλών παιχνιδιών. Για αυτό η επιλογή του κατάλληλου ρομπότ είναι σημαντική. Το ρομπότ πρέπει να διαθέτει κάμερα και ένα υπολογιστή, που να επεξεργάζεται γρήγορα τα δεδομένα που λαμβάνει από αυτήν. Το AIBO διαθέτει τα παραπάνω, οπότε με την εύρεση ενός αλγόριθμου που προβλέπει την κίνηση της μπάλας, καταφέραμε το AIBO να παίξει το παιχνίδι μας.

Ένα ρομπότ μπορεί να παίξει ένα παιχνίδι, γιατί έχει προγραμματιστεί έτσι ώστε να αναγνωρίζει τα βασικά στοιχεία του παιχνιδιού. Ένα ρομπότ μπορεί να αξιοποιήσει μονάχα αντικείμενα στο περιβάλλον του, τα οποία έχει προηγουμένως αναγνωρίσει. Αντίθετα ένας άνθρωπος μπορεί να χρησιμοποιήσει το περιβάλλον τριγύρω του, ακόμα και αν δεν έχει παραβρεθεί ξανά σε παρόμοιο περιβάλλον. Οι πέντε αισθήσεις, η φαντασία και η ευφυία του ανθρώπου του επιτρέπουν να χρησιμοποιεί και να διαμορφώνει το περιβάλλον προς όφελος του.

Βιβλιογραφία

[1] http://pyrorobotics.org (Pyro Official Website)

[2] http://www.urbiforge.com (URBI Offcial Website)

[3] "How to Get Started with Tekkotsu and Open-R on the AIBO"

[4] http://www.ouroboros.org/rcode_tutorial_1v2.pdf

[5] http://www.tekkotsu.org (Tekkotsu Ofiicial Website)

[6] http://www.cygwin.com (Cygwin-Platform for Windows)

[7] http://www.bloodshed.net (Dev C++ Compiler)

[8] http://nehe.gamedev.net (Nehe's OpenGL Tutorial)

[9] Σουζάνα Βολιώτη, "Αναγνώριση αντικειμένων για το Τουρνουά Robocup 2007 με χρήση ιστογράμματος", Διπλωματική εργασία, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης, Ιούνιος 2007

[10] Κωνσταντίνος Μακαντάσης, "Αναγνώριση ανθρώπινου προσώπου και παρακολούθησή του από το τετράποδο ρομπότ AIBO", Διπλωματική εργασία, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης, Φεβρουάριος 2008

[11] Κωνσταντίνος Καπουράκης, "Pac-man: Παραμετρική Υλοποίηση του παιχνιδιού και σχεδιασμός αυτόνομου πράκτορα-παίκτη", Διπλωματική εργασία, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης, Ιούνιος 2007

[12] www.tekkotsu.org/Samples.html