ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ
ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Διπλωματική εργασία

«Βιομιμητική συμπεριφορά ρομποτικών διατάξεων: Η περίπτωση του κυνηγετικού μοντέλου λύκου»

Κονδυλάκης Γεώργιος

Επιβλέπων καθηγητής: **Τσουρβελούδης Νικόλαος**

Τριμελής Επιτροπή:

Τσουρβελούδης Νικόλαος

Ιωαννίδης Ευστράτιος

Πιπερίδης Σάββας

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου και κοσμήτορα της σχολής, Καθηγητή Νικόλαο Τσουρβελούδη για την άψογη συνεργασία που είχαμε καθ'όλη τη διάρκεια της εκπόνησης της διπλωματικής μου. Θα ήθελα να τον ευχαριστήσω ακόμα που μου ανέθεσε το συγκεκριμένο θέμα, το οποίο μου "ξεκλείδωσε" μια μεγάλη αγάπη για ένα αντικείμενο με το οποίο δεν είχα έρθει σε επαφή έως τώρα.

Επίσης, θα ήθελα να ευχαριστήσω τον Διδάκτορα του τμήματος Μηχανικών Παραγωγής και Διοίκησης, Σάββα Πιπερίδη για τις τεχνικές γνώσεις που μου μετέφερε γύρω από το πρόγραμμα της προσομοίωσης, τον προγραμματισμό και τον κόσμο της Ρομποτικής γενικότερα.

Ευχαριστώ επίσης τον κύριο Ευστράτιο Ιωαννίδη, Επίκουρο Καθηγητή της σχολής Μηχανικών Παραγωγής και Διοίκησης για τις γνώσεις που μου μετέδωσε στα πλαίσια του προπτυχιακού μαθήματος της Ρομποτικής.

Ευχαριστώ τους φίλους και κολλητούς μου για την υπομονή τους το διάστημα αυτό που με "έχασαν" λόγω διαβάσματος (Σταυρή και Λυτώ χρωστώ πολλές "εξόδους" ακόμα, το ξέρω!). Ευχαριστώ το φίλο μου το Γιάννη που με παρότρυνε να ασχοληθώ με αυτό το θέμα και να κυνηγήσω το όνειρο μου να ασχοληθώ με την Ρομποτική. Ένα μεγάλο ευχαριστώ θα ήθελα ακόμη να δώσω σε όλους τους φίλους μου από το συγκρότημα μας, τους Yar Aman, για όλες τις απίστευτες στιγμές μουσικής και έμπνευσης που μου χάρισαν όλα αυτά τα χρόνια. Ευχαριστώ την οικογένεια μου που έμπρακτα με στήριξαν όλα αυτά τα χρόνια και ήταν πάντα δίπλα μου σε καθετί, όμορφο και άσχημο.

Περίληψη

Η κυνηγετική δεινότητα των λύκων είναι αυτή που ευνόησε την επιβίωση τους και τους καθιέρωσε στις κορυφαίες θέσεις των θηρευτών στην πυραμίδα της τροφικής αλυσίδας. Η επιτυχία τους εν μέρει οφείλεται στην ικανότητα τους να καταναλώνουν διαφορετικά είδη θηραμάτων, αλλά και στην γενικευμένη συνεργατική συμπεριφορά που επιδεικνύουν κατά το κυνήγι. Η κυνηγετική συμπεριφορά των λύκων αποτέλεσε και αποτελεί πεδίο έμπνευσης για τον ραγδαίως αναπτυσσόμενο τομέα της ρομποτικής που, με τη βοήθεια της Βιολογίας, κατασκευάζει μοντέλα και αρχιτεκτονικές που μιμούνται τη συμπεριφορά και παρουσιάζουν εξαιρετικό ενδιαφέρον από άποψη εφαρμογών στην καθημερινότητα.

Η παρούσα εργασία αφορά την σχεδίαση και κατασκευή δύο διαφορετικών μοντέλων/αρχιτεκτονικών πολύ-ρομποτικών συστημάτων εμπνευσμένων από δύο διαφορετικές βιομιμητικές συνεργατικές συμπεριφορές που απαντώνται στη φύση και πιο συγκεκριμένα στο κυνήγι των γκρι λύκων. Τα μοντέλα αναπτύχθηκαν στο περιβάλλον προσομοίωσης Webots, ενώ οι κυνηγετικές συμπεριφορές που μελετήθηκαν, στηρίζονται σε μεγάλο βαθμό στις εργασίες των *Alfredo Weitzenfeld, Alberto Vallesa and Horacio Flores: "* A Biologically-Inspired Wolf Pack Multiple Robot Hunting Model" και *John D. Madden, Ronald C. Arkin and Daniel R. MacNulty*: "Multi-robot System Based on Model of Wolf Hunting Behavior to Emulate Wolf and Elk Interactions", οι οποίες στηρίζονται σε ηθολογικές μελέτες δεκαετιών γύρω από τη δομή, οργάνωση και ιεραρχία του κυνηγιού των γκρι λύκων.

Το πρώτο μοντέλο που κατασκευάστηκε, υιοθετεί την έως πρότινος επικρατούσα αντίληψη για το πώς κυνηγάνε οι γκρι λύκοι, η οποία προβλέπει την ύπαρξη ιεραρχίας μεταξύ της αγέλης, την κατανομή εργασιών και την συντονισμένη προσπάθεια κατά το κυνήγι. Σε αυτήν την αρχιτεκτονική το πολύ-ρομποτικό σύστημα αποτελούνταν από ρομπότ που είχαν τον ρόλο που έχουν οι λύκοι στη φύση, δηλαδή κυνηγοί, και από ρομπότ που έχουν το ρόλο των θηραμάτων. Στα πρώτα διακρίναμε δύο διαφορετικούς ρόλους, όπως αναφέραμε και παραπάνω, τους alpha και τους beta.

Το δεύτερο μοντέλο που κατασκευάστηκε υιοθετεί την πλέον επικρατούσα άποψη που ήρθε να διαψεύσει την έως τότε πιο διαδεδομένη από βιολογικής σκοπιάς, ότι οι λύκοι στη φύση δεν κυνηγάνε με κανενός είδους οργάνωση ή ιεραρχία. Αντιθέτως κυνηγούν προσπαθώντας τρόπον τινά να βελτιστοποιήσουν σε κάθε περίπτωση την πιθανότητα αιχμαλωσίας του θηράματος και να εξασφαλίσουν την τροφή τους. Είτε αυτό γίνεται κυνηγώντας κατά μόνας ,είτε συνεργαζόμενοι μεταξύ τους. Η παραπάνω γνώση αξιοποιήθηκε στο δεύτερο μοντέλο που κατασκευάστηκε όπου στην προσομοίωση συμπεριελήφθησαν μόνο ρομπότ-"λύκοι" και "θηράματα".

Όλα τα ρομπότ και στα δύο μοντέλα/σενάρια έχουν προγραμματιστεί ώστε να δρουν σύμφωνα με τις πληροφορίες που έχουν λάβει από τα αισθητήρια όργανα που υπάρχουν στα ρομπότ και κυρίως τις κάμερες χρωματικού μοντέλου τύπου RGB οι οποίες εντοπίζουν χρωματικούς στόχους στο περιβάλλον του ρομπότ. Χρησιμοποιούνται επίσης πομπό-δέκτης για να επιτυγχάνεται η επικοινωνία μεταξύ της αγέλης των ρομπότ ενώ αισθητήρες αφής αξιοποιούνται στην περίπτωση πρόσκρουσης των ρομπότ.

Thesis Overview

Hunting superiority of wolves is the reason of their wide spread in the natural environment and their dominance in the food chain. Hunting success has to do partly with their ability to consume a great range of preys but also with the cooperative behavior they demonstrate during hunting. Hunting behavior of wolves guides Robotics to build models and architectures, inspired by Biology, that emulate animal behavior and address them to deal with every-day problems.

This thesis is about the design of two biomimetic multi-robot systems that emulate the hunting behavior of gray wolves. The models were designed with the aid of Webots robotic simulator, while the hunting behaviors are based partially in the papers of *Alfredo Weitzenfeld, Alberto Vallesa and Horacio Flores: "* A Biologically-Inspired Wolf Pack Multiple Robot Hunting Model" and *John D. Madden, Ronald C. Arkin and Daniel R. MacNulty*: "Multi-robot System Based on Model of Wolf Hunting Behavior to Emulate Wolf and Elk Interactions". These papers demonstrate ethological studies of centuries on the structure, organization and hierarchy in the wolf pack.

The first model encapsulates the notion that was established until recently, that there is a hierarchy in the wolf pack, division of labor and distribution of roles. In this architecture the multi-robot system is comprised of robots that simulate the behavior of wolves in nature and of robots that simulate the preys. In this architecture we distinguish two different roles during hunting, alpha and betas.

The second model encapsulates the current notion of how grey wolves hunt, that wolves hunt without any hierarchy and coordination trying to maximize their own utility and the chance to captivate a prey. In this model we used robots that simulate independent wolves, and preys.

Every robot is programmed to act upon sensory information acquired mostly from the RGB color type camera. Emitter and receiver are also used to accomplish the communication between the members of the pack. Touch sensor is also to avoid collisions.

Περιεχόμενα

1. Εισαγωγή

Η λέξη «ρομπότ» πρωτοχρησιμοποιήθηκε αναφορικά με κάποιο φανταστικό ανθρωποειδές στο θεατρικό έργο R.U.R. του Τσέχου συγγραφέα Karel Čapek, μα στην ουσία ήταν ο αδερφός του Karel, ο Josef Čapek που επινόησε τη λέξη. Η αρχή του έργου εκτυλίσσεται σε ένα εργοστάσιο που κατασκευάζει τεχνητούς ανθρώπους από συνθετική οργανική ύλη οι οποίοι κατονομάζονται roboti (ρομπότ). Η συγκεκριμένη περιγραφή όμως δεν ταιριάζει με τα ρομπότ όπως τα εννοούμε σήμερα αυτά τα όντα είναι πιο κοντά στη σημερινή ιδέα των cyborg, των android, ή ακόμα και των κλώνων, μιας και θα μπορούσαν εύκολα να χαρακτηριστούν σαν άνθρωποι και επίσης έχουν τη δική τους νόηση.

Εικόνα 1.1 Το δημοφιλές θεατρικό έργο R.U.R. του Τσέχου συγγραφέα Karel Čapek στο οποίο πρωτοχρησιμοποιήθηκε η λέξη «ρομπότ»

Αυτά τα όντα αρχικά παρουσιάζονται να είναι ικανοποιημένα με το να υπηρετούν τους ανθρώπους όμως αυτό αλλάζει όταν μια εχθρική εξέγερση των ρομπότ οδηγεί στην εξαφάνιση του ανθρώπινου είδους. Ο Čapek αργότερα προσέγγισε διαφορετικά το ίδιο θέμα στο War with the Newts, στο οποίο οι μη-άνθρωποι έχουν τη θέση του υπηρέτη στην ανθρώπινη κοινωνία. Το θεατρικό έργο του Karel Čapek έχει οπότε την κύρια ευθύνη για το τι θεωρούμε σήμερα ρομπότ, για το ότι τα ρομπότ είναι ανθρωποειδείς μηχανές με προσωπικότητα και Τεχνητή Νοημοσύνη. Αυτή η ιδέα έγινε ακόμα πιο ισχυρή μέσω της Γερμανικής ταινίας κινηματογράφου Metropolis (1926) που παρουσίασε στο κοινό ένα ρομπότ με το όνομα Electro. Οι άνθρωποι μέχρι πρότινος πίστευαν ότι τα ρομπότ είναι κακές τεχνητές οντότητες μέχρι που ο Isaac Asimov τη δεκαετία του ’50 εισήγαγε ως ιδέα τα καλά ρομπότ, ονομάζοντάς τα droid και έκανε τον όρο «ρομποτική» πασίγνωστο. Οι Τρεις Νόμοι της Ρομποτικής (συχνά συναντάται ως Οι Τρεις Νόμοι ή Τρεις Νόμοι αλλά και Οι Νόμοι του Asimov) είναι ένα σύνολο κανόνων που επινόησε ο συγγραφέας βιβλίων επιστημονικής φαντασίας Isaac Asimov. Οι κανόνες πρώτοπαρουσιάστηκαν το 1942 στο διήγημα «Runaround», αν και η ιδέα τους είχε γίνει αισθητή και από προηγούμενες ιστορίες. Οι Τρεις Νόμοι, οι οποίοι σύμφωνα με το κείμενο προέρχονται από το «Εγχειρίδιο της Ρομποτικής, 56η έκδοση, 2058 μ.Χ.» είναι:

1. *Το ρομπότ δε θα κάνει κακό σε άνθρωπο, ούτε με την αδράνειά του θα επιτρέψει να βλαφτεί ανθρώπινο όν*
2. *Το ρομπότ πρέπει να υπακούει τις διαταγές που του δίνουν οι άνθρωποι, εκτός αν αυτές οι διαταγές έρχονται σε αντίθεση με τον πρώτο νόμο*
3. *Το ρομπότ οφείλει να προστατεύει την ύπαρξή του, εφόσον αυτό δεν συγκρούεται με τον πρώτο και τον δεύτερο νόμο*

Η ιδέα των αυτόματων (automata) πηγάζει από τις μυθολογίες πολλών και διαφόρων πολιτισμών σε όλο τον κόσμο. Μηχανικοί και εφευρέτες αρχαίων πολιτισμών, συμπεριλαμβανομένης της αρχαίας Κίνας, της αρχαίας Ελλάδας και της πτολεμαϊκής Αιγύπτου, επιχείρησαν να κατασκευάσουν αυτόματες μηχανές, κάποιες εκ των οποίων έμοιαζαν με ζώα και ανθρώπους. Στις πρώιμες περιγραφές αυτόματων, περιλαμβάνονται τα τεχνητά περιστέρια του Αρχύτα (Αρχύτας ο Ταραντίνος), τα τεχνητά πουλιά των Mozi και Lu Ban, ένα ομιλών αυτόματο από τον Ήρων τον Αλεξανδρινό (Ήρων ο Αλεξανδρεύς), ένα αυτόματο έπιπλο μπάνιου από τον Φίλων του Βυζαντίου (Φίλων ο Βυζάντιος) και ένα ανθρωποειδές αυτόματο το οποίο περιγράφεται στο Lie Zi.

Πολλές αρχαίες μυθολογίες και οι περισσότερες από τις σύγχρονες θρησκείες περιλαμβάνουν τεχνητούς ανθρώπους, όπως οι μηχανικοί υπηρέτες που κατασκευάστηκαν από τον Έλληνα θεό Ήφαιστο (Vulcan, όπως είναι γνωστός στους Ρωμαίους), τα πήλινα γκόλεμ (golem) της Εβραϊκής μυθολογίας, οι πήλινοι γίγαντες της Σκανδιναβικής μυθολογίας, καθώς επίσης και η Γαλατεία, το μυθικό άγαλμα του ο Πυγμαλίων ζωντάνεψε. Από το 400 π.Χ. περίπου, οι μύθοι της Κρήτης περιλαμβάνουν τον Τάλω, έναν μπρούτζινο άντρα που φρουρούσε το νησί της Ευρώπης (την Κρήτη) από τους πειρατές. Μύθοι όπως ο τελευταίος είναι πολύ συνηθισμένοι, καθώς φανερώνουν την αντίληψη των αρχαίων, πως τα ανθρωπόμορφα ρομπότ κατασκευάζονται, για να βρίσκονται στην υπηρεσία των ανθρώπων. Είναι πολύ ενδιαφέρον το ότι το φάσμα των υπηρεσιών που εκτελούσαν τα αρχαία «ρομπότ» είναι ως επί το πλείστων το ίδιο με σήμερα (βαριά και ανθυγιεινά καθήκοντα, κατασκευές, περιφρούρηση, βοήθεια σε ανθρώπους με αναπηρία (όπως ο Ήφαιστος), ανύψωση βαρέων φορτίων κ.τ.λ.).

1.1. Τεχνικός ορισμός του ρομπότ

Ακόμα κι αν η ιστορία και η επιστημονική φαντασία ισχυρίζονται πως τα ρομπότ είναι ανθρωπόμορφες μηχανές αυτό απέχει πολύ από την πραγματικότητα. Ενώ τα ρομπότ είναι μηχανικά, δεν χρειάζεται να είναι ανθρωπομορφικά ή ακόμα να μοιάζουν με ζώα. Αναλογιστείτε τις ρομποτικές σκούπες ή τα αυτόνομα αυτοκίνητα.

Ένα ρομπότ, όπως το ορίζει το Ινστιτούτο Ρομποτικής της Αμερικής, *είναι ένας επαναπρογραμματιζόμενος, πολυλειτουργικός ειδικευμένος χειριστής που σχεδιάστηκε για να μετακινεί υλικά, εξαρτήματα, εργαλεία ή ειδικές συσκευές, μέσα από ποικίλες προγραμματισμένες κινήσεις για την εκτέλεση μιας πληθώρας εργασιών.* Ή όπως δηλώνει η Robin R. Murphy πιο σύντομα και εκλεπτυσμένα στο βιβλίο της "Introduction to AI Robotics" *ένα έξυπνο ρομπότ είναι ένα μηχανικό κατασκεύασμα που μπορεί να λειτουργεί αυτόνομα.*

1.2. Εφαρμογές της ρομποτικής

Λαμβάνοντας υπόψη την ευελιξία και την προσαρμοστικότητα των ρομπότ θα μπορούσαμε να πούμε ότι μπορούν να κάνουν σχεδόν οτιδήποτε μπορούμε να φανταστούμε. Σε μια προσπάθεια να κατηγοριοποιήσουμε τις βασικές τους χρήσεις μέχρι σήμερα θα συνοψίζαμε στις παρακάτω: 1) όταν ο άνθρωπος βρίσκεται σε σημαντικό κίνδυνο (πυρηνικό, διαστημικό, στρατιωτικό), 2) για ασύμφορες ή κουραστικές δουλειές (όπως σε μερικές υπηρεσίες, για παράδειγμα σε επιστατικές εργασίες όπως η συντήρηση δημόσιων αποχωρητηρίων ή το καθάρισμα λουτρών, στη γεωργία για την συγκομιδή μεγάλων χωραφιών και το κλάδεμα ή το άρμεγμα ζώων) και 3) για ανθρωπιστικές χρήσεις (όπως η εκκαθάριση ναρκοπεδίων και η έρευνα και διάσωση ανθρώπων σε μία πόλη σε περίπτωση φυσικών καταστροφών όπως ο σεισμός).

1.3. Κοινωνικές επιπτώσεις της ρομποτικής

Οι κοινωνικές συνέπειες της Ρομποτικής και της αυτοματοποίησης έχουν παρατηρηθεί από την Βιομηχανική Επανάσταση. Κάποιες από αυτές είναι *η μόνιμη ανεργία,* όταν ο απολυμένος υπάλληλος είναι ήδη μεγάλος για να επανεκπαιδευτεί. Άλλη συνέπεια είναι η *ανακατανομή της εργασίας,* όταν ο απολυμένος υπάλληλος εργάζεται σε διαφορετικό πόστο ανεξάρτητα από το εάν έχει επανεκπαιδευτεί.

Συμπερασματικά, ο αντίκτυπος των ρομπότ είναι ασαφής. Ο τρόπος με τον οποίο αλληλεπιδρούν οι άνθρωποι με τα ρομπότ διαφέρει. Σε ορισμένες περιπτώσεις, όπως στην HelpMate Robotics Inc., τα ρομπότ ή πιο συγκεκριμένα τα επιστατικά ρομπότ, φαίνεται ότι ανταγωνίζονται τους ανθρώπους, αλλά τα ρομπότ καλύπτουν θέσεις στις οποίες είναι δύσκολο να βρεις ανθρώπινους εργάτες (ανθρώπους) με οποιαδήποτε τιμή. Το καθάρισμα μεγάλων κτηρίων γραφείων είναι ευτελές και πληκτικό και τα ωράρια είναι κουραστικά. Σε άλλες περιπτώσεις οι πολιτισμικές επιπτώσεις της Ρομποτικής έχουν σοβαρό αντίκτυπο στην χρήση των ρομπότ. Τα ρομπότ που δημιουργήθηκαν για το κούρεμα των προβάτων στην Αυστραλία ήταν επιτυχημένα και έτοιμα για εμπορευματοποίηση, αλλά στο τέλος η βιομηχανία προβάτων τα απέρριψε. Οι εκτροφείς προβάτων δεν θα δέχονταν ένα ρομπότ για

κουρευτή εκτός κι αν είχε 0% ποσοστό θνησιμότητας (πράγμα σχεδόν απίθανο γιατί είναι αρκετά εύκολο να κόψεις μία αρτηρία σε ένα πρόβατο που προσπαθεί να απελευθερωθεί) παρόλο που οι ίδιοι οι άνθρωποι κουρευτές σκοτώνουν κατά λάθος αρκετά πρόβατα, ενώ τα ρομπότ έχουν ένα σίγουρα καλύτερο ποσοστό. Η χρήση των μηχανών εγείρει ένα ηθικό ερώτημα: είναι αποδεκτό να πεθάνει ένα ζώο στα «χέρια» μιας μηχανής απ' ότι στα χέρια ενός ανθρώπου; Τι θα γινόταν αν ένα ρομπότ εκτελούσε μία περίπλοκη χειρουργική επέμβαση σε έναν άνθρω

2. Βιομιμητική

Η βιομιμητική ή βιομίμηση είναι η μίμηση των μοντέλων, των συστημάτων και των στοιχείων της φύσης με σκοπό την επίλυση περίπλοκων ανθρώπινων προβλημάτων. Οι όροι «βιομιμητική» (biomimetics) και «βιομίμηση» (biomimicry) προέρχονται από τα αρχαία ελληνικά: βίος (bios), που είναι η ζωή και μίμησις (mīmēsis), που είναι η μίμηση και προέρχεται από το μῖμος (mimos), δηλαδή ηθοποιός.

Η αναζήτηση στη φύση για νέες λύσεις σε μηχανικά προβλήματα δεν άφησε ανεπηρέαστη την Ρομποτική, κι έτσι μέσα από το "πάντρεμα" επιστημών όπως η Βιολογία, η Ψυχολογία και οι Νευροεπιστήμες με την Ρομποτική προέκυψαν νέα δυναμικά διεπιστημονικά πεδία όπως η Βιομιμητική Ρομποτική.

Βιομιμητική ρομποτική είναι ο κλάδος της ρομποτικής ο οποίος ασχολείται με την μελέτη της μηχανικής συμπεριφοράς και του ελέγχου της κίνησης σύνθετων ρομποτικών συστημάτων, εμπνευσμένων από τη βιολογία .

2.1. Λόγοι αντιγραφής της συμπεριφοράς των ζώων στη Ρομποτική

Όλα τα παραπάνω γεγονότα και η αναζήτηση της ευφυΐας στα ρομπότ ενέπνευσαν τους επιστήμονες να μελετήσουν και να μιμηθούν τη φύση. Οπότε από την παρατήρηση των βιολογικών συστημάτων γίνεται η προσπάθεια να αποκωδικοποιηθούν οι μηχανισμοί και οι λειτουργίες των ζώων και να εφαρμοστεί αυτή η γνώση στο πεδίο της μηχανικής. Τα παραδείγματα στο πεδίο της Ρομποτικής είναι πολλά. Εξαιρετικά σημαντικές εφαρμογές αποτελούν ρομποτικά συστήματα που προσομοιάζουν την κυματοειδή μετακίνηση (undulatory locomotion) των ερπετών, συστήματα ενεργού αντίληψης (active perception) εμπνευσμένα από το ανθρώπινο οφθαλμοκινητικό σύστημα, ρομποτικά συστήματα εμπνευσμένα από το σύστημα ελέγχου της πτήσης ορισμένων εντόμων, που βασίζεται σε οπτική πληροφορία κίνησης (visual motion) κ.α., ικανότητες που μπορούν να συνοψισθούν στις παρακάτω βιολογικές αντιληπτικές ικανότητες:

- Βιο-αισθητήρες (για παράδειγμα το μάτι).
- Βιο- ενεργητές (για παράδειγμα οι μύες).
- Βιοϋλικά (για παράδειγμα ο ιστός της αράχνης).

Εικόνα 2.1: Το ρομπότ Ariel κάτω από το βυθό σε μία λίμνη στη Βοστώνη. Κινείται πλάγια ακριβώς όπως ένα καβούρι

Τα παραδείγματα αληθινών βιομιμητικών ρομπότ που έχουν κατασκευαστεί σε εργαστήρια και ερευνητικά κέντρα είναι αμέτρητα και διαφοροποιούνται ανάλογα με τη χρήση για την οποία προορίζονται. Πολλά από αυτά έχουν κατασκευαστεί με σκοπό την κατανόηση του τρόπου με τον οποίο τα ζώα καταφέρνουν μερικές αξιοθαύμαστες κινητικές δραστηριότητες όπως η μετακίνηση χωρίς άκρατο πήδημα και το σκαρφάλωμα (βλ. ρομπότ φίδια, κατσαρίδες, γρύλλος, φίδι κλπ.).

Εικόνα 2.2: Ρομποτικός δεινόσαυρος Troody

2.2. Λόγοι αντιγραφής της συμπεριφοράς των ζώων σε αλγοριθμικό επίπεδο

Ένα ζήτημα που απασχολεί τους επιστήμονες είναι η απόκτηση της ευφυΐας στη Ρομποτική. Για την κατάκτηση αυτού του στόχου καταφύγαμε στη Βιολογία και στις Νευροεπιστήμες. Ξεκινήσαμε να εξερευνούμε τις βασικές λειτουργίες του νευρικού συστήματος και τους μηχανισμούς μάθησης των ζώων και από αυτά φτιάξαμε αλγόριθμους. Επιχειρήσαμε να αντιγράψουμε επίσης την κοινωνική συμπεριφορά των ζώων και κατασκευάσαμε ρομπότ που μπορούσαν να συνεργαστούν σαν ομάδα (κοπάδια, σμήνη, αγέλες κ.τ.λ.).

Εικόνα 2.3: Η ρομποτική στην αναζήτηση της νοημοσύνης

2.3. Γνωριμία με τον γκρίζο λύκο

Ο Γκρίζος λύκος (κοινά αναφερόμενος ως λύκος) είναι σαρκοφάγο θηλαστικό της οικογενείας των Κυνιδών, της οποίας αποτελεί το μεγαλύτερο και ισχυρότερο μέλος. Απαντά σε εκτεταμένες περιοχές του Βορείου ημισφαιρίου, κυρίως όμως στα μεγάλα γεωγραφικά πλάτη του (εξαιρούνται τα διακριτά υποείδη σκύλος και ντίνγκο). Η επιστημονική ονομασία του είδους (sensu lato) είναι Canis lupus και περιλαμβάνει 36-40 υποείδη, αναλόγως του ταξινομικού φορέα.

Εικόνα 2.4: Λύκος των βόρειων βραχώδων ορέων, (Canis lupus irrmotus)

Ο γκρίζος λύκος αποτελεί το πλέον εξειδικευμένο μέλος του γένους Canis, όπως αποδεικνύεται από προσαρμογές στην μορφολογία του, το κυνήγι μεγάλων θηραμάτων, την πλήρως κοινωνική φύση του, και την εξαιρετικά προηγμένη εκφραστική συμπεριφορά του. Όπως και ο κόκκινος λύκος (Canis rufus), διακρίνεται από άλλα είδη του γένους, από το μεγάλο μέγεθος και τα λιγότερο αιχμηρά δομικά στοιχεία του κεφαλιού του, ιδιαίτερα τα αυτιά και το ρύγχος. Είναι το μόνο είδος του γένους Canis που έχει εξάπλωση τόσο στον Παλαιό, όσο και τον Νέο Κόσμο.

Πρόκειται για κοινωνικότατο ζώο, με πυρήνα την «οικογένεια», η οποία αποτελείται από ένα (1) κυρίαρχο αναπαραγωγικό ζεύγος, συνοδευόμενο από τους ενήλικους απογόνους του συγκεκριμένου ζευγαριού. Τρέφεται κυρίως με μεγάλα οπληφόρα, μικρότερα ζώα κάθε είδους (λαγούς, πουλιά κ.α.), ζώα κτηνοτροφίας, θνησιμαία και απορρίμματα. Ο τρόπος οργάνωσης και επικοινωνίας των μελών του, με κυρίαρχα ηθολογικά στοιχεία το ουρλιαχτό και τις οσμητικές σημάνσεις, προκαλεί τον θαυμασμό ακόμη και στους διώκτες του (βλ. Ηθολογία).

Εικόνα 2.5: Αγέλη λύκων αποτελούμενη από τα μικρά και τα μεγαλύτερα μέλη

2.3.1. Κυνηγετικό μοντέλο γκρίζου λύκου

Ο τρόπος που κυνηγούν οι γκρίζοι λύκοι αποτελεί ένα από τα σημαντικότερα στοιχεία της ηθολογίας και της εξελικτικής τους πορείας, γενικότερα. Έχει μελετηθεί εκτεταμένα από επιστήμονες και ερευνητές, ενώ έχει καταγραφεί πολλάκις από τον φωτογραφικό και τον κινηματογραφικό φακό, ως παράδειγμα οργάνωσης ενός μικρού ή μεγάλου συνόλου για την επίτευξη κοινού στόχου.

Παρόλο που είναι έντονα κοινωνικοί, οι γκρίζοι λύκοι μπορούν να κυνηγούν μοναχικά ή κατά ζευγάρια και, μάλιστα, έχουν συνήθως υψηλότερα ποσοστά επιτυχίας στο κυνήγι από ότι οι μεγάλες αγέλες. Έχουν παρατηρηθεί μοναχικά άτομα να καταβάλλουν μεγάλα θηράματα χωρίς βοήθεια. Φαίνεται περίεργο, αλλά η -οπωσδήποτε ανεπτυγμένη- αίσθηση της όσφρησης στους γκρίζους λύκους είναι, παρόλα αυτά, ασθενέστερη σε σύγκριση με εκείνην κάποιων κυνηγετικών σκύλων. Έτσι, σε αντίθεση με αυτούς, δεν είναι σε θέση να ανιχνεύσουν κάποιο θήραμα που βρίσκεται μακρύτερα από 2-3 χιλιόμετρα. Εξαιτίας αυτού του γεγονότος, ένας γκρίζος λύκος σπάνια καταφέρνει να συλλάβει κρυμμένους λαγούς ή πουλιά αν και, όπως ο σκύλος, μπορεί εύκολα να ακολουθήσει φρέσκα ίχνη. Ωστόσο, η αίσθηση της ακοής του γκρίζου λύκου είναι εξαιρετικά οξεία. Είναι ικανός να ακούσει χαμηλούς (μπάσους) ήχους μέχρι την συχνότητα των 26 Hz, αρκετή για να αντιλαμβάνεται τα φύλλα που πέφτουν, κατά την περίοδο του φθινοπώρου.

Εικόνα 2.6: Λύκοι αναζητούν την τροφή τους σε περιοχή με γρασίδι δίπλα σε ποτάμι

Όταν οι γκρίζοι λύκοι κυνηγούν ομαδικά, διατηρούν αξιοθαύμαστη οργάνωση και πειθαρχία, επιδεικνύοντας υψηλής ευφυΐας στρατηγική. Ειδικά στις μεγάλες, ανοικτές περιοχές, όταν η αγέλη συγκεντρώνεται, ακολουθείται κάποιο είδος τελετουργικού, με την ομάδα να συνεννοείται για την έναρξη του κυνηγιού και τα μέλη της να ακουμπούν τις μύτες τους και να κουνάνε έντονα την ουρά τους. Όταν όλα έχουν «τακτοποιηθεί» αρχίζει το κυνήγι, με «μπροστάρηδες» τα κυρίαρχα ζευγάρια. Η όλη διαδικασία μπορεί να χωριστεί σε 5 στάδια:

- **Εντοπισμός της λείας**: αυτό γίνεται είτε με την ανίχνευση της οσμής του θηράματος στον αέρα -για κοντινές αποστάσεις-, είτε με την καθοδήγηση από τα ίχνη στο έδαφος, είτε με τυχαία συνάντηση. Ειδικά στην πρώτη περίπτωση, όταν ένα ρεύμα αέρα μεταφέρει την μυρωδιά του θηράματος, οι λύκοι τίθενται άμεσα σε εγρήγορση και «φερμάρουν» με τα μάτια, τα αυτιά και τη μύτη τους προς την κατεύθυνση της μυρωδιάς.

- **Κάλυψη και προσέγγιση**: οι λύκοι προσπαθούν να καλυφθούν όσο το δυνατόν καλύτερα, καθώς πλησιάζουν το θήραμα. Όσο η απόσταση θηρευτή και θηράματος μικραίνει, οι λύκοι επιταχύνουν τον ρυθμό τους, κουνάνε την ουρά τους, και κοιτάζουν επίμονα, προσπαθώντας να πλησιάσουν ακόμη περισσότερο το θήραμα, χωρίς ωστόσο να τού επιτρέψουν να διαφύγει.

Εικόνα 2.7: Καταδίωξη θηράματος (βίσονα) από αγέλη λύκων. Εάν το θήραμα -εδώ ένας βίσονας- παραμείνει στην θέση του, αυξάνει τις πιθανότητες επιβίωσής του

- **Εμπλοκή**: Από την στιγμή που το θήραμα αντιληφθεί τους λύκους, έχει τρεις πιθανές επιλογές: να τραπεί σε φυγή, να μείνει στην θέση του ή να στραφεί εναντίον τους. Τα μεγάλα οπληφόρα, όπως οι άλκες, τα *ουαπίτι* (*Cervus canadensis*) και οι *μοσχόβοες* (*Ovibos moschatus*), συνήθως παραμένουν στην θέση τους. Αυτό δημιουργεί αμηχανία στους λύκους, καθώς είναι «προγραμματισμένοι» να επιτίθενται μόλις το θήραμα αρχίζει να τρέχει. Σε αρκετές περιπτώσεις, μάλιστα, οι λύκοι είτε το αγνοούν, είτε προσπαθούν να το εκφοβίσουν και να το αναγκάσουν να τραπεί σε φυγή.

- **Καταδίωξη**: Όπως είναι φυσικό, τα περισσότερα θηράματα τρέπονται σε φυγή μόλις αντιληφθούν τους λύκους, οπότε εκείνοι αρχίζουν να τα καταδιώκουν αμέσως. Αυτό είναι το πιο κρίσιμο στάδιο του κυνηγιού, διότι πολλά θηράματα τρέχουν με ταχύτητα μεγαλύτερη της αγέλης των λύκων και μπορεί να μην τα προλάβουν. Εάν υπάρχει καταδιωκόμενο κοπάδι, οι λύκοι χωρίζονται, με κάποιους από αυτούς να «κόβουν δρόμο» για να βρεθούν μπροστά, άλλοι στήνουν ενέδρα και οι υπόλοιποι καταδιώκουν. Απώτερος σκοπός είναι η απομόνωση ενός μέλους του κοπαδιού για να τού επιτεθούν όλοι μαζί. Όταν κυνηγούν μικρά θηράματα, οι γκρίζοι λύκοι προσπαθούν να καλύψουν την απόσταση, το συντομότερο δυνατόν, ενώ με τα μεγαλύτερα ζώα, το κυνηγητό είναι παρατεταμένο, προκειμένου να εξαντλήσουν το επιλεγμένο θήραμα. Παρά την αντοχή τους, οι λύκοι συνήθως παραιτούνται εάν η καταδίωξη ξεπεράσει τα 1-2 χιλιόμετρα, αν και ένας λύκος καταγράφηκε να καταδιώκει ένα ελάφι για 21 χιλιόμετρα. Τόσο οι γκρίζοι λύκοι της Σιβηρίας όσο και της Βόρειας Αμερικής, έχουν παρατηρηθεί να οδηγούν το θήραμα σε παγωμένες επιφάνειες, γκρεμούς, φαράγγια, πλαγιές και απότομες πλαγιές για να το επιβραδύνουν.

- **Επαφή**: Όταν οι λύκοι φθάσουν το θήραμα, προσπαθούν να επιτεθούν από πίσω ή από τα πλάγια και να τού προκαλέσουν σοβαρό τραύμα. Σπάνια επιτίθενται κατά μέτωπο, ιδιαίτερα όταν πρόκειται για μεγάλα οπληφόρα, για να αποφύγουν την πιθανότητα τραυματισμού.

Οι λύκοι συνήθως στοχεύουν στα μαλακά τμήματα της περιοχής του περινέου, προκαλώντας μαζική απώλεια αίματος. Τέτοια δαγκώματα μπορεί να προκαλέσουν πληγές 10-15 εκατοστών σε μήκος και, τρία από αυτά είναι συνήθως αρκετά για να ρίξουν κάτω ένα υγιές μεγάλο ελάφι. Μεσαίου μεγέθους θηράματα όπως ζαρκάδια ή πρόβατα, αποτελούν εύκολη λεία για τους λύκους, που τα θανατώνουν με δάγκωμα στην περιοχή του λαιμού, αποκόπτοντας νευρώνες και την καρωτιδική αρτηρία και το θήραμα πεθαίνει μέσα σε λίγα δευτερόλεπτα έως ένα λεπτό. Όταν το θήραμα είναι μικρό, λ.χ. λαγοί και τρωκτικά, οι λύκοι προσπαθούν να το αιφνιδιάσουν, επιτιθέμενοι με τοξωτό άλμα και ακινητοποιώντας την λεία με τα μπροστινά τους πόδια.

Μόλις το θήραμα θανατωθεί, οι λύκοι αρχίζουν να τρέφονται με βουλιμία, σκίζοντας και τραβώντας το σφάγιο προς όλες τις κατευθύνσεις και αποσπώντας μεγάλα κομμάτια του. Τυπικά, αρχίζουν να τρέφονται με την κατανάλωση των μεγαλύτερων εσωτερικών οργάνων του θύματος, όπως είναι η καρδιά, το ήπαρ, οι πνεύμονες και το στομάχι. Τα νεφρά και ο σπλήνας τρώγονται από τη στιγμή που εκτίθενται, ενώ τελευταίο ακολουθεί το μυικό σύστημα. Ένας και μόνον γκρίζος λύκος, μπορεί να φάει κρέας που αντιστοιχεί στο 15%-19% του σωματικού του βάρους του, με την μία.

Το κυρίαρχο ζευγάρι της αγέλης διεκδικεί την πρωτιά στην σίτιση και, όταν η τροφή είναι λιγοστή, αυτό γίνεται εις βάρος των άλλων μελών της οικογένειας, ιδιαίτερα στα νεαρά άτομα που δεν είναι πλέον κουτάβια. Βέβαια, το κυρίαρχο ζευγάρι δικαιωματικά τρέφεται πρώτο, διότι έχει κάνει την περισσότερη «δουλειά» στο κυνήγι, αλλά μπορεί και να ξεκουραστεί, αφήνοντας τα υπόλοιπα μέλη της αγέλης να τραφούν πρώτα. Μόλις το κυρίαρχο ζευγάρι τελειώσει το φαγητό, η υπόλοιπη οικογένεια παίρνει τα εναπομείναντα κομμάτια του σφαγίου και τα μεταφέρει σε απομονωμένες περιοχές όπου μπορούν να καταναλωθούν αργότερα.

2.3.2. Διάρθρωση αγέλης

Ο γκρίζος λύκος είναι από τα *κοινωνικότερα ζώα της υφηλίου*. Η βασική μονάδα αποτελείται από ένα κυρίαρχο ζευγάρι, το οποίο συνοδεύεται από τους απογόνους -διαφόρων γενεών- του ζευγαριού που, όλοι μαζί, σχηματίζουν την *αγέλη* (*pack*). Μια «μέση» αγέλη αποτελείται από 5-11 ζώα (1-2 ενήλικες, 3-6 νεαρά άτομα και 1-3 μονοετή κουτάβια), αλλά μερικές φορές απαρτίζεται από δύο ή τρεις οικογένειες, οπότε μπορεί να φθάσει σε μεγάλους αριθμούς (κάποια αγέλη έφθασε τα 42 άτομα). Το κυρίαρχο αρσενικό είναι ο απόλυτος «αρχηγός» της αγέλης, εξουσιάζοντας ακόμη και το κυρίαρχο θηλυκό, το οποίο, με την σειρά του, επιβάλλεται στα νεαρά άτομα. Σε ιδανικές συνθήκες, το κυρίαρχο ζεύγος τεκνοποιεί κάθε χρόνο, με τους απογόνους να παραμένουν στην αγέλη για 10-54 μήνες πριν την διασπορά τους. Το έναυσμα για τον αποχωρισμό και διασπορά των νεαρών ατόμων δίνεται από την έναρξη της σεξουαλικής ωριμότητας και τον ανταγωνισμό για την τροφή. Η απόσταση κατά την οποίαν απομακρύνονται τα άτομα που εγκαταλείπουν την αγέλη, ποικίλλει σε μεγάλο βαθμό. Μερικά μένουν στην ευρύτερη περιοχή της γονικής ομάδας, ενώ άλλα άτομα μπορεί να ταξιδέψουν πολύ μακριά (206-670 χιλιόμετρα) από τον τόπο που γεννήθηκαν.

Εικόνα 2.8: Στην συγκεκριμένη οικογένεια λύκων ο αρχηγός της αγέλης τρώει πρώτος

Νέα αγέλη δημιουργείται συνήθως από «εργένικα» αρσενικά και θηλυκά που περιπλανώνται μαζί προς αναζήτηση περιοχής ελεύθερης από άλλες εχθρικές αγέλες. Κάθε εγκαθιδρυμένη αγέλη σπάνια δέχεται στους κόλπους της ξένους λύκους και συνήθως προσπαθεί να τους εξολοθρεύσει. Στις σπάνιες περιπτώσεις όπου «εγκρίνονται» ξένα άτομα, ο «υιοθετούμενος» είναι σχεδόν πάντοτε ένα ανώριμο ζώο (1-3 ετών), με ελάχιστες πιθανότητες να ανταγωνιστεί τα μέλη του κυρίαρχου ζευγαριού για την αρχηγία της αγέλης. Σε ορισμένες περιπτώσεις, κάποιος μοναχικός λύκος «εισάγεται» στην αγέλη για να αντικαταστήσει έναν απωλεσθέντα

επιβήτορα, χρήσιμο στην αναπαραγωγή της ομάδας. Κατά καιρούς η αφθονία στους πόρους τροφής (μετανάστευση και αναπαραγωγή οπληφόρων) οδηγεί επί μέρους αγέλες να ενώσουν προσωρινά τις δυνάμεις τους στο κυνήγι.

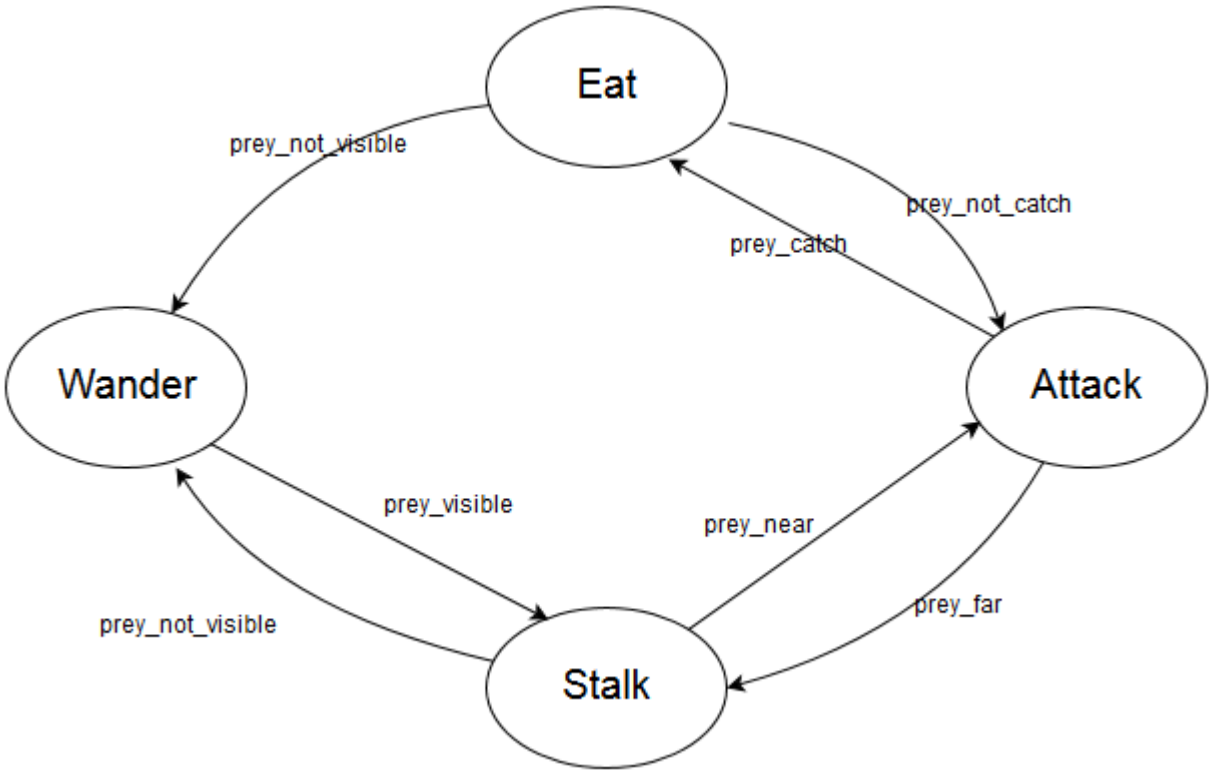
- Στις αγέλες των γκρίζων λύκων, παρατηρείται το φαινόμενο του *ψυχολογικού ευνουχισμού*, που θεωρείται απόρροια της κοινωνικής ιεραρχίας. Το κυρίαρχο αρσενικό ανέχεται μεν την παρουσία άλλων αρσενικών, γεννητικά ωρίμων, αλλά αυτά δεν ζευγαρώνουν λόγω ορμονικής αδρανοποίησης. Αντίστοιχη είναι και η εικόνα των υπολοίπων, πλην του κυρίαρχου, θηλυκών, οι οποίες δεν εμφανίζουν οίστρο.

Στο παρελθόν επικρατούσε η άποψη ότι οι αγέλες των γκρίζων λύκων αποτελούνταν από άτομα που συναγωνίζονταν μεταξύ τους για την κυριαρχία, με το κυρίαρχο ζευγάρι να αποκαλείται ως «άλφα» -αρσενικό και θηλυκό-, και τους υποτασσόμενους λύκους «βήτα» έως «ωμέγα». Αυτή η ορολογία χρησιμοποιήθηκε για πρώτη φορά το 1947 από τον Rudolf Schenkel του Πανεπιστημίου της Βασιλείας, με βάση τα ευρήματά του σε έρευνα συμπεριφοράς των γκρίζων λύκων σε αιχμαλωσία. Αυτή η άποψη για την ιεραρχία των λύκων, αργότερα διαδόθηκε ευρέως από τον L. David Mech, το 1970, μέσω του βιβλίου του *Ο Λύκος*. Όμως, αργότερα, ο ίδιος αποκήρυξε επισήμως αυτή την ορολογία, το 1999, εξηγώντας ότι ήταν σε μεγάλο βαθμό βασισμένη στην συμπεριφορά των γκρίζων λύκων σε αιχμαλωσία που, ωστόσο αποτελούνταν από άσχετα μεταξύ τους άτομα, ένα σφάλμα που αντικατοπτρίζει την άλλοτε επικρατούσα άποψη ότι ο σχηματισμός άγριων αγελών συμβαίνει κατά την διάρκεια του χειμώνα μεταξύ ανεξάρτητων γκρίζων λύκων. Αργότερα, έρευνα για τους άγριους γκρίζους λύκους αποκάλυψε ότι η αγέλη δεν είναι παρά μία οικογένεια που αποτελείται από το κυρίαρχο ζευγάρι αναπαραγωγής και τους απογόνους του από τα προηγούμενα 1-3 χρόνια.

2.3.3. Ιεραρχικό, Οργανωμένο Κυνήγι Λύκων σε Αγέλες

Η πρώτη προσομοίωση που διεξήγαγα βασίστηκε στην εργασία των Alfredo Weitzenfeld, *Alberto Vallesa and Horacio Flores, "A Biologically-Inspired Wolf Pack Multiple Robot Hunting Model"*. Στην εργασία παρουσιάζεται η αρχιτεκτονική ενός πολύ-ρομποτικού συστήματος εμπνευσμένου από ένα μοντέλο κυνηγετικής συμπεριφοράς λύκων σε αγέλες. Το συμπεριφοριστικό μοντέλο που χρησιμοποιείται από τους Weitzenfeld, *Alberto Vallesa and Horacio Flores* βασίζεται στις ηθολογικές μελέτες του γνωστού βιολόγου και ειδικού στους λύκους David L. Mech. Το μοντέλο παρατηρεί μια αγέλη από λύκους, που έχουν συγκεκριμένη (βλ. λύκος άλφα, ο οποίος είναι ο αρχηγός της αγέλης, και κάποιοι υφιστάμενοι βήτα λύκοι-beta wolves). Όταν η αγέλη αιχμαλωτίσει το θήραμα ο ισχυρότερος άλφα λύκος τρώει πρώτος.

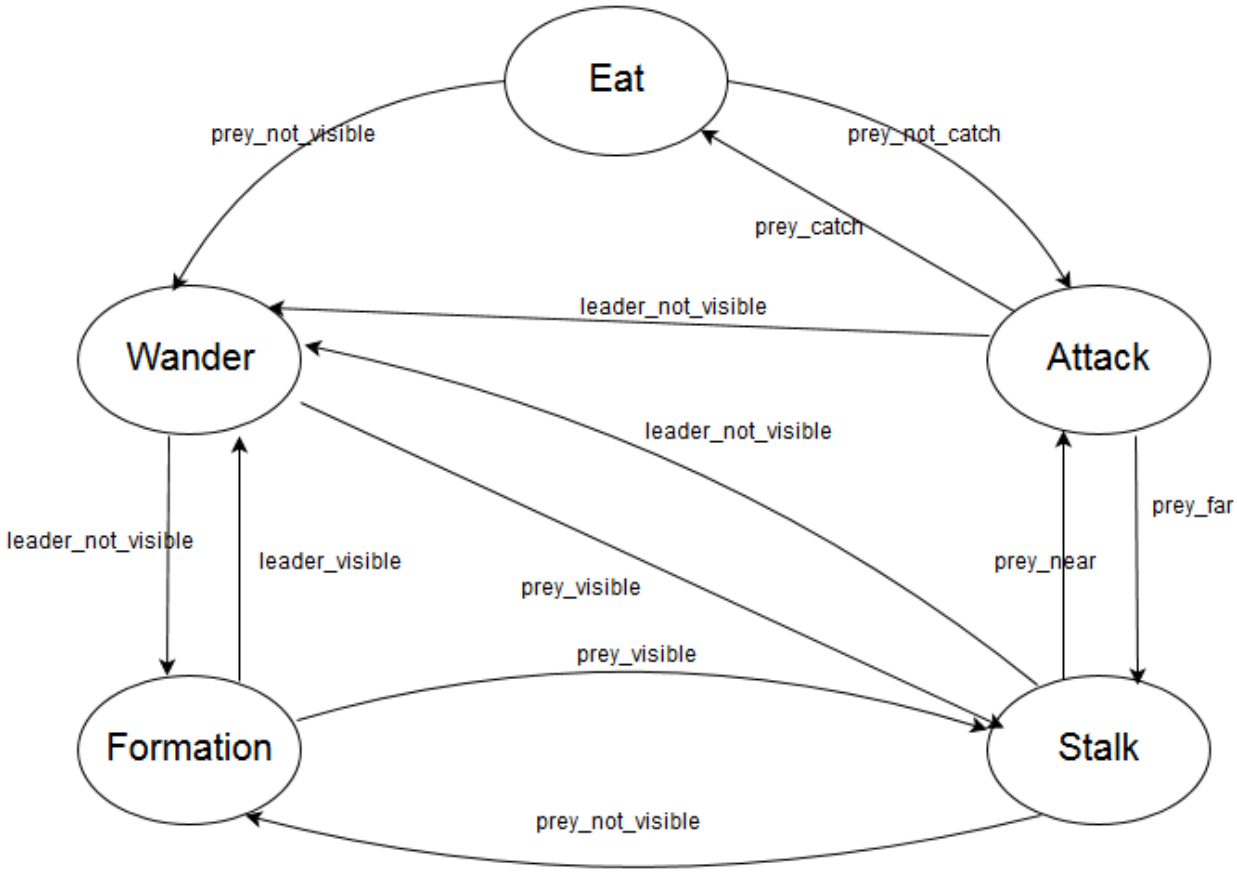
Λύκος άλφα: Το σχήμα 2.1 παρουσιάζει την κυνηγετική συμπεριφορά του λύκου άλφα η οποία καθορίζεται από τέσσερα στάδια, όπως προτείνεται από τους Alfredo Weitzenfeld, Alberto Vallesa, Horacio Flores.



Σχήμα 2.1: Διάγραμμα καταστάσεων για τον alpha

- Περιπλάνηση (Wander): Σε αυτό το στάδιο ο λύκος άλφα εξερευνά το περιβάλλον αναζητώντας ένα θήραμα να φάει. Μόλις εντοπίσει το θήραμα η συνθήκη prey_visible (το θήραμα είναι ορατό) ενεργοποιείται υποδεικνύοντας την αλλαγή στο επόμενο στάδιο, την Παρακολούθηση.
- Παρακολούθηση (Stalk): Σε αυτό το στάδιο μπορούν να συμβούν δύο μεταβάσεις: μία προς το στάδιο της Περιπλάνησης, στην περίπτωση που το θήραμα είναι εκτός ορατότητας, κάτι που προκαλείται από την συνθήκη prey_not_visible (το θήραμα δεν είναι ορατό). Η άλλη μετάβαση προκύπτει όταν το θήραμα εντοπίζεται σε κοντινή απόσταση, όπου και ενεργοποιείται η κατάσταση prey_near (το θήραμα είναι κοντά) η οποία οδηγεί στο επόμενο στάδιο, την Επίθεση.
- Επίθεση (Attack): Σε αυτό το στάδιο ο λύκος πλησιάζει το θήραμα μέχρι να το πιάσει. Όταν συμβεί αυτό ενεργοποιείται η κατάσταση prey_catch (το θήραμα πιάστηκε) και ο λύκος μεταβαίνει στο επόμενο στάδιο, την Κατανάλωση. Εάν ο λύκος χάσει ξαφνικά τον έλεγχο του θηράματος πηγαίνει πίσω στο στάδιο Παρακολούθησης το οποίο ενεργοποιείται από την συνθήκη prey_not_catch (το θήραμα δεν πιάστηκε).
- Επαφή (Eat): Σε αυτό το στάδιο ο λύκος τρώει το θήραμα. Έπειτα, ενεργοποιείται η συνθήκη prey_not_visible υποδεικνύοντας πως το θήραμα έχει καταναλωθεί. Ο λύκος επιστρέφει στο στάδιο της Περιπλάνησης.

Λύκος βήτα: Το σχήμα 2.2 παρουσιάζει την κυνηγετική συμπεριφορά του λύκου βήτα η οποία καθορίζεται από πέντε στάδια, όπως προτείνεται από τους Alfredo Weitzenfeld, Alberto Vallesa, Horacio Flores.



Σχήμα 2.2 : : Διάγραμμα καταστάσεων για τον beta

- Περιπλάνηση (Wander): Ο λύκος βήτα εξερευνά το περιβάλλον αναζητώντας τον αρχηγό της ομάδας ή ένα θήραμα. Εάν ο λύκος βρει τον αρχηγό, ενεργοποιείται η συνθήκη leader_visible (ο αρχηγός είναι ορατός) η οποία οδηγεί στο στάδιο του Σχηματισμού. Εάν ο λύκος βρει το θήραμα ενεργοποιείται η συνθήκη prey_visible η οποία οδηγεί στο στάδιο της Παρακολούθησης. Οι κινήσεις του λύκου είναι παρόμοιες με τις κινήσεις του λύκου άλφα σε αυτό το στάδιο.
- Σχηματισμός (Formation): Όσο ο λύκος βήτα βλέπει τον λύκο άλφα, μένει κοντά στον αρχηγό. Εάν η οπτική επαφή με τον αρχηγό χαθεί ενεργοποιείται η συνθήκη leader_not_visible (ο αρχηγός δεν είναι ορατός) η οποία οδηγεί στο στάδιο Περιπλάνησης. Εάν ο λύκος βήτα εντοπίσει ένα θήραμα, μεταβαίνει στο στάδιο της Παρακολούθησης αφού ενεργοποιείται η συνθήκη prey_visible.
- Παρακολούθηση (Stalk): Σε αυτό το στάδιο μπορούν να συμβούν τρεις μεταβάσεις: μία προς το στάδιο του Σχηματισμού, στην περίπτωση που το θήραμα βρίσκεται

εκτός της ορατότητας του λύκου, όπου και ενεργοποιείται η κατάσταση prey_not_visible. Μια άλλη μετάβαση προκύπτει όταν ο λύκος εντοπίσει ένα θήραμα σε κοντινή απόσταση, όπου και ενεργοποιείται η συνθήκη prey_near, η οποία οδηγεί στο στάδιο της Επίθεσης. Τέλος, η τρίτη μετάβαση συμβαίνει στην περίπτωση που ο λύκος βήτα χάσει τα ίχνη του λύκου άλφα όπου και ενεργοποιείται η συνθήκη leader_not_visible, και οδηγεί στο στάδιο Περιπλάνησης. Ο στόχος αυτού του σταδίου είναι να πλησιάσει το θήραμα χωρίς να χωριστεί ιδιαίτερα από τον αρχηγό της ομάδας.

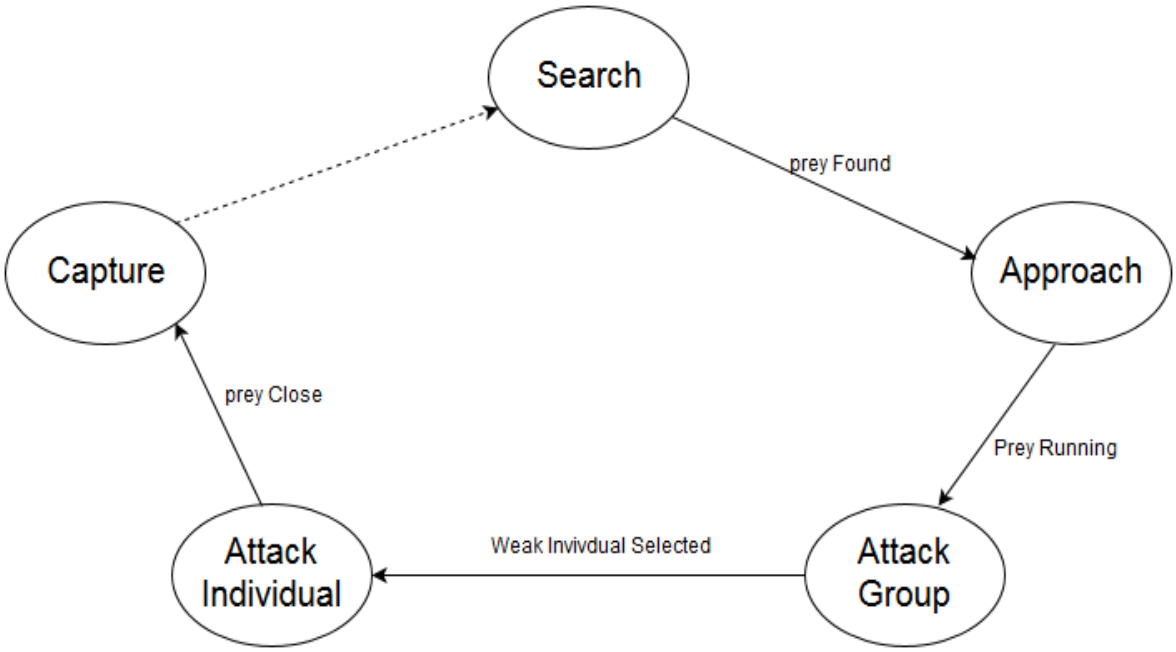
- Επίθεση (Attack): Ο λύκος βήτα θεωρεί τον εαυτό του αρκετά κοντά στο θήραμα αν και ακόμη λαμβάνει υπόψιν τη σχετική θέση του ίδιου και του αρχηγού. Ωστόσο, σε αυτό το στάδιο η κίνηση για την προσέγγιση και την αιχμαλωσία του θηράματος έχει μεγαλύτερη προτεραιότητα από το να ακολουθήσει τον αρχηγό. Παρ' όλα αυτά το θήραμα μπορεί να ξεφύγει επειδή σε πολλές περιπτώσεις είναι πιο γρήγορο. Εάν συμβεί αυτό ενεργοποιείται η συνθήκη prey_far (το θήραμα είναι μακριά) και ο λύκος βήτα επιστρέφει στο στάδιο της Παρακολούθησης. Εάν οι λύκοι καταφέρουν να πιάσουν το θήραμα, ενεργοποιείται η συνθήκη prey_catch που οδηγεί στο στάδιο Κατανάλωσης.
- Επαφή (Eat): Αυτό το στάδιο είναι παρόμοιο με του λύκου άλφα αλλά ο λύκος άλφα είναι αυτός που θα φάει πρώτος.

2.3.4. Μη ιεραρχικό, Ανοργάνωτο Κυνήγι Λύκων σε Αγέλες

Η δεύτερη προσομοίωση που διεξήγαγα βασίστηκε στη δουλειά του *John D. Madden, του Ronald C. Arkin και του Daniel R. MacNulty : "Multi-robot System Based on Model of Wolf Hunting Behavior to Emulate Wolf and Elk Interactions"*. Το project διαφοροποιείται από το προηγούμενο με δύο σημαντικούς τρόπους. Πρωτίστως, ο Weitzenfeld υπέθεσε ότι υπάρχει μια αυστηρή οργάνωση ως προς τον συντονισμό μιας αγέλης λύκων. Ωστόσο, άμεσες παρατηρήσεις λύκων πάνω στο κυνήγι άλκεων στο Εθνικό Πάρκο Yellowstone δεν υποδεικνύουν προφανή μοτίβα οργανωμένης κυνηγετικής συμπεριφοράς. Έπειτα, ο Weitzenfeld επίσης υπέθεσε ότι υπάρχουν ορισμένοι ρόλοι όπως ο λύκος άλφα και ο λύκος βήτα, για να ελέγχονται οι θέσεις και οι πράξεις ατομικά σε όλη τη διάρκεια του κυνηγιού. Οι παρατηρήσεις πάνω στις οποίες έφτιαξαν τα μοντέλα τους φανερώνουν πως πράγματι υπάρχουν ρόλοι αλλά ενδέχεται να αλλάξουν με τυχαίο τρόπο και βασίζονται σε σωματικές δεξιότητες και όχι σε μια προϋπάρχουσα κυριαρχική ιεραρχία. Οι παρατηρήσεις του πεδίου, στο οποίο βασίζεται η κατανόηση της κυνηγετικής συμπεριφοράς των λύκων, υποδεικνύει ότι υπάρχει έλλειψη ρητού συντονισμού μεταξύ των λύκων. Η συμπεριφορά της ομάδας δεν είναι προφανώς ένα καλά δομημένο σύνολο στρατηγικών αλλά μάλλον γενικευμένοι «εμπειρικοί κανόνες» που χρησιμοποιούνται ως αντίδραση στη συμπεριφορά διαφυγής του θηράματος για να ελαχιστοποιηθεί ο κίνδυνος τραυματισμού της ίδιας της ομάδας.

Ωστόσο σε αυτή την έρευνα υιοθετήθηκε μία ηθογραφία με έξι στάδια: έρευνα, προσέγγιση, παρακολούθηση, ομαδική επίθεση, ατομική επίθεση και αιχμαλώτιση. Το στάδιο παρακολούθησης θα παραληφθεί καθώς είναι ένα σπάνιο στάδιο στο οποίο μπαίνουν οι λύκοι όταν επιτίθενται στις άλκες.

Το σχήμα 2.3 δείχνει τα στάδια αναζήτησης τροφής ενός τυπικού κυνηγιού σύμφωνα με τους John D. Madden, του Ronald C. Arkin και του Daniel R. MacNulty:



Σχήμα 2.3: : Διάγραμμα καταστάσεων για οποιονδήποτε λύκο

- Έρευνα (Search): Ταξιδεύουν και κινούνται χωρίς εμμονές.
- Προσέγγιση (Approach): Παθαίνουν εμμονή με το θήραμα και ταξιδεύουν προς αυτό.
- Ομαδική Επίθεση (Attack group): Κυνηγούν μια ομάδα ζώων που τρέπεται σε φυγή ή ορμούν σε μία στάσιμη ομάδα ενώ παράλληλα διατηρούν οπτική επαφή με όλα τα μέλη της ομάδας ξεχωριστά (δηλαδή σκανάρισμα).
- Ατομική επίθεση (Attack individual): Κυνηγούν ή ορμούν σε ένα απομακρυσμένο, μεμονωμένο ζώο ή μόνο σε ένα μέλος μιας ομάδας ενώ παράλληλα αγνοούν όλα τα υπόλοιπα μέλη.
- Αιχμαλώτιση (Capture): Δαγκώνουν και περιορίζουν το θήραμα.

Στο σχήμα 2.3 παρουσιάσαμε τις μεταβάσεις από στάδιο σε στάδιο. Όμως σε αυτή την εργασία, όπως περίπου και στην πραγματικότητα, οι μεταβάσεις δεν συμβαίνουν με προκαθορισμένο τρόπο. Πέρα από την γραμμική και ξεκάθαρη ακολουθία από την έρευνα στην προσέγγιση, μετά στην ομαδική επίθεση, έπειτα στην ατομική επίθεση και τελικά στην αιχμαλώτιση, πολλές άλλες μεταβάσεις είναι πιθανές. Ο MacNulty και οι άλλοι συνέλεξαν τα δικά τους στατιστικά δεδομένα παρατήρησης για τις μεταβάσεις των σταδίων όπου η εκτίμηση σε μορφή πίνακα (πίνακας 2.1) απεικονίζει την πιθανότητα μετάβασης μεταξύ των σταδίων.

Πίνακας 2.1: Πιθανότητες μεταβάσεων από κατάσταση σε κατάσταση

	Following State					
Preceding State	**Search**	**Approach**	**Watch**	**Attack Group**	**Attack Individual**	**Capture**
Search	.00	.68	.00	.31	.01	.00
Approach	.09	.00	.12	.69	.09	.01
Watch	.32	.35	.00	.27	.06	.00
Attack Group	.24	.09	.03	.13	.51	.00
Attack Individual	.16	.06	.02	.16	.08	.52

3. Webots

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

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

3.1. Εφαρμογές του Webots

Συνοπτικά μερικά από τα χαρακτηριστικά του προσομοιωτή:

- Εκτελεί τη μοντελοποίηση και προσομοίωση αυτοκινούμενων, ιπτάμενων και αρθρωτών ρομποτικών εφαρμογών.
- Περιλαμβάνει πληθώρα από εμπορικά διαθέσιμους αισθητήρες και όργανα αλληλεπίδρασης με το περιβάλλον.
- Κάνει χρήση δημοφιλών γλωσσών προγραμματισμού C/C++, Java Python, MATLAB κ.α.
- Κάνει χρήση της βιβλιοθήκης ODE για την αναπαράσταση με ακρίβεια φυσικών φαινομένων.
- Μπορεί να μεταφέρει τους υπό εξέταση ελεγκτές σε πραγματικά ρομποτικά οχήματα όπως τα e-puck, Nao, Katana, Hoap-2 κ.α.
- Δημιουργία αρχείων βίντεο τύπο AVI ή MPEG.
- Προσομοίωση συστημάτων πολλαπλών πρακτόρων

3.2. Ρομποτικός προσομοιωτής Webots

Μια προσομοίωση Webots αποτελείται από τα παρακάτω :

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

2. Προγράμματα ελεγκτών για τα ανωτέρω ρομπότ.

3. Προαιρετική χρήση πρόσθετων φυσικής για να τροποποιήσουμε το περιβάλλον προσομοίωσης του Webots (στην C,C++).

3.3. Ορισμός του ‘world’ στο Webots

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

3.4. Ορισμός του controller

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

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

3.5. Ορισμός των πρόσθετων φυσικής

Το Webots προσφέρει τη δυνατότητα να προσθέσεις πρόσθετα φυσικής σε μια προσομοίωση. Τα πρόσθετα φυσικής είναι μια κοινή βιβλιοθήκη που εφαρμόζει ο χρήστης και η οποία «φορτώνεται» από το Webots την ώρα που το πρόγραμμα βρίσκεται σε εφαρμογή (τρέχει) και του δίνουν πρόσβαση to the low-level API of the ODE physics engine. Τα πρόσθετα φυσικής μπορούν να χρησιμοποιηθούν, για παράδειγμα, στη συλλογή πληροφοριών σχετικά με τα τεχνητά σώματα (θέση, προσανατολισμός, γραμμική ή γωνιακή ταχύτητα, κ.τ.λ.), στην πρόσθεση δυνάμεων και ροπών στρέψης, στην πρόσθεση επιπλέων αρθρώσεων, για παράδειγμα «σφαιροειδής και γωνιώδης» ή «ομοκινητικοί σύνδεσμοι», σε μία προσομοίωση. Για παράδειγμα με τα πρόσθετα φυσικής είναι πιθανό να σχεδιάσουμε ένα αεροδυναμικό μοντέλο για ένα ιπτάμενο ρομπότ, ένα υδροδυναμικό μοντέλο για ένα ρομπότ που κολυμπάει, κ.τ.λ. Επιπλέον, με τα πρόσθετα φυσικής μπορούμε να εφαρμόσουμε το δικό μας σύστημα εντοπισμού πρόσκρουσης και να ορίσουμε τις ανομοιόμορφες παραμέτρους τριβής σε κάποιες επιφάνειες. Σημειώστε ότι τα πρόσθετα φυσικής μπορούν να προγραμματιστούν μόνο σε C ή C++. Το Webots PRO είναι απαραίτητο για τον προγραμματισμό των πρόσθετων φυσικής.

3.6. Supervisor

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

3.7. Λόγοι χρήσης του Webots και της προσομοίωσης

Η χρήση εργαλείων προσομοίωσης για τον σχεδιασμό ρομπότ είναι μια απλή και φτηνή μέθοδος για να κατασκευάσεις πολύπλοκα ρομπότ, ή τουλάχιστον εικονικά ρομπότ. Στα πλεονεκτήματα θα συμπεριελάμβανα το μειωμένο κόστος, ενώ όλα τα εργαλεία προσομοίωσης προσφέρουν τη δυνατότητα προσομοίωσης του ρομπότ σε διαφορετικά σενάρια, ο κώδικας προγραμματισμού μπορεί να ελεγχθεί για να καθορίσει τη συμβατότητα με τις απαραίτητες προδιαγραφές και πολλά ακόμη χαρακτηριστικά.

Στην παρακάτω λίστα παρουσιάζονται πιο επιγραμματικά ορισμένα πλεονεκτήματα της προσομοίωσης:

- *Χαμηλό κόστος στην κατασκευή ενός ρομπότ από την αρχή*
- *Ο κώδικας προγραμματισμού μπορεί να ελεγχθεί σύμφωνα με τις προδιαγραφές*
- *Ο σχεδιασμός των ρομπότ μπορεί να τροποποιηθεί χωρίς κόστος*
- *Οποιοδήποτε μέρος του ρομπότ μπορεί να δοκιμαστεί*
- *Σε ένα πολύπλοκο project το ρομπότ μπορεί να προσομοιωθεί σε στάδια*
- *Μια ολοκληρωμένη προσομοίωση μπορεί να καθορίσει αν το ρομπότ καλύπτει τις προδιαγραφές*
- *Σχεδόν όλες οι προσομοιώσεις λογισμικού είναι συμβατές με ένα ευρύ φάσμα από γλώσσες προγραμματισμού*
- *Το διάστημα που μεσολαβεί από την αρχή του project και μέχρι την ολοκλήρωσή του μπορεί να ελαττωθεί*

4. Παρουσίαση του μοντέλου της βιομιμητικής συνεργατικής συμπεριφοράς στο περιβάλλον του Webots

Στο κεφάλαιο αυτό θα αναλύσουμε τους κυριότερους άξονες της συνεργατικής συμπεριφοράς ενώ θα παρουσιάσουμε την αρχιτεκτονική του συστήματος όπως αναπτύχθηκε στο Webots.

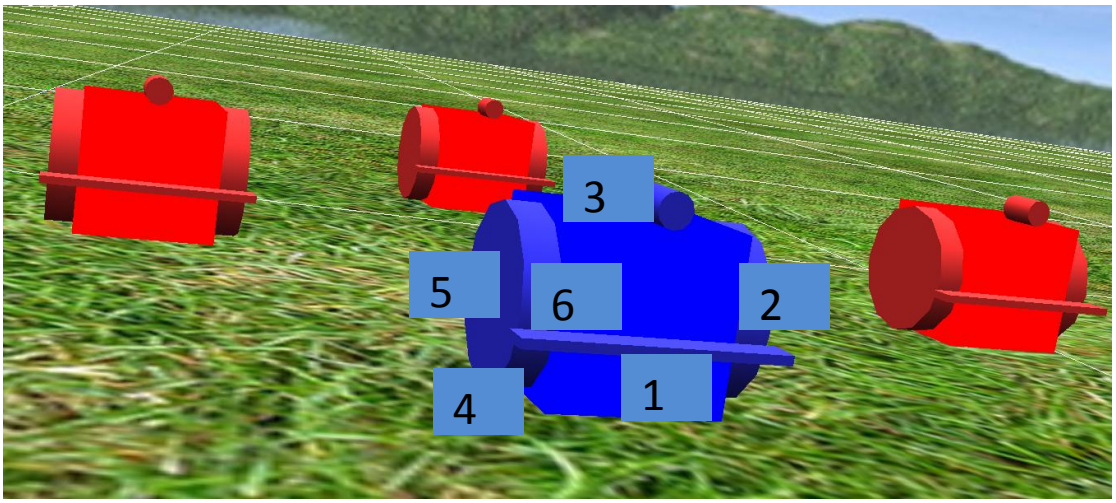
Στην ενότητα 4.1 και 4.2 θα παρουσιαστεί το περιβάλλον της προσομοίωσης, οι προσαρμογές και οι αλλαγές που έγιναν σε αυτό για να καλύπτει τις ανάγκες μας καθώς και τα ρομπότ που χρησιμοποιήθηκαν στα δύο πειράματα, έτσι όπως σχεδιάστηκαν και εξελίχθηκαν από την αρχή εξ ολοκλήρου από το μηδέν

Στην ενότητα 4.3 και 4.4 θα παρουσιάσουμε τα δύο σενάρια έτσι όπως τελικά διαμορφώθηκαν μετά από μήνες πειραματικών δοκιμών. Προκειμένου να κατανοήσουμε πλήρως έννοιες όπως συνεργατική συμπεριφορά, ρόλοι alpha και beta, αποτελεσματικότητα "κυνηγιού" κ.α. θα "σπάσουμε" την προσομοίωση σε στιγμιότυπα. Η παρουσίαση των ενοτήτων αυτών δεν έχει σκοπό να υποσκελίσει την παρουσίαση του ελεγκτή αλλά αποτελεί μια εισαγωγική προσέγγιση όπου χρησιμοποιούνται εικόνες από διάφορα στιγμιότυπα του κυνηγιού. Οι 5 ενότητες του κεφαλαίου αυτού αποτελούν ένα αδιάσπαστο σύνολο για να κατανοήσουμε σε βάθος τη συνεργατική συμπεριφορά.

Τέλος, στην ενότητα 4.5 εξηγείται ο ρόλος και η λειτουργία του Supervisor.

4.1. Παρουσίαση του WheelRobot

Παρακάτω θα παρουσιάσουμε το ρομπότ αμαξίδιο που χρησιμοποιήθηκε στην παρουσίαση προκειμένου να μοντελοποιήσουμε τη συνεργατική συμπεριφορά. Το WheelRobot κατασκευάστηκε μέσα στο περιβάλλον προσομοίωσης του Webots και η κίνηση των δύο τροχών του βασίζεται στις αρχές λειτουργίας της διαφορικής κίνησης. Έχει ενσωματωμένα αισθητήρες αφής, κάμερα, πομπό και δέκτη. Το σώμα απαρτίζεται από ένα κυβικό στερεό με δύο ρόδες εκατέρωθεν. Στην προσομοίωση μπορούμε να διακρίνουμε WheelRobot τριών διαφορετικών χρωμάτων, πανομοιότυπων όμως στην κατασκευή και τα χαρακτηριστικά τους, που το καθένα αντιπροσωπεύει συγκεκριμένο ρόλο στην προσομοίωση του κυνηγετικού μοντέλου. Στην εικόνα 4.1 μπορούμε να δούμε με μπλε και κόκκινο τα ρομπότ που αντιπροσωπεύουν τον alpha και τους beta αντίστοιχα.



Εικόνα 4.1: WheelRobot-alpha αρχηγός της ομάδας με μπλέ χρώμα. Απαρτίζεται από: (1): το σώμα του , (2): τις ρόδες, (3): την κάμερα χρώματος, (4): τον αισθητήρα αφής και (5),(6): τον emitter και τον receiver. Οι beta είναι ακριβή αντίγραφα του alpha.

4.2. Παρουσίαση του περιβάλλοντος της προσομοίωσης

Ο χώρος που φαίνεται στην εικόνα 4.2 είναι ο χώρος όπου λαμβάνει χώρα η προσομοίωση. Αποτελείται από το προσομοιωμένο γρασίδι, τον ουράνιο σφαιρικό θόλο ενώ πολύ σημαντικό ρόλο παίζει ο φωτισμός που χρησιμοποιήθηκε από φώτα τύπου DirectionalLight (τοποθετημένα στις 4 άκρες της παραπάνω περιοχής).

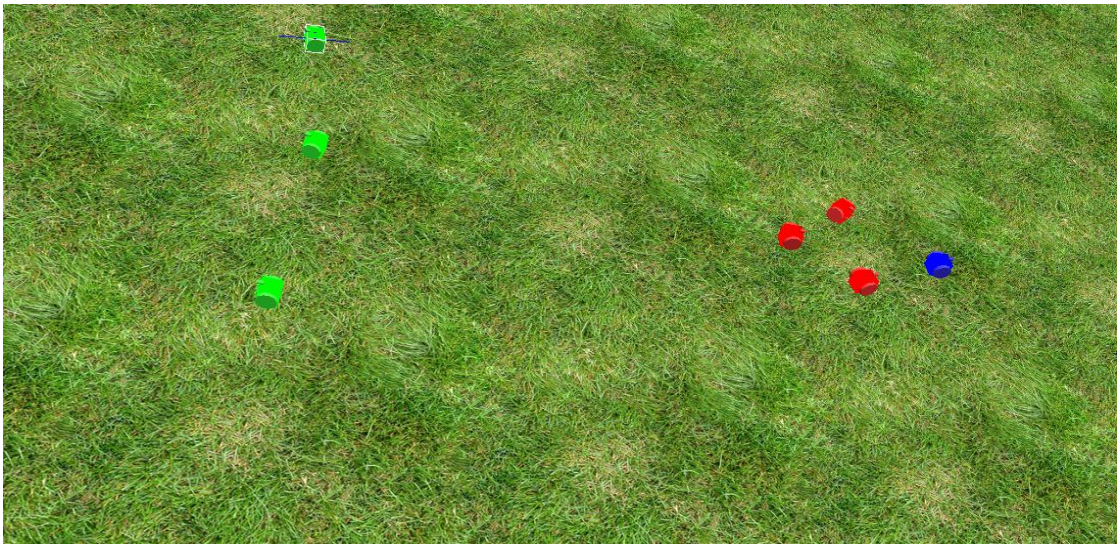
Εικόνα 4.1: Ο χώρος όπου λαμβάνει χώρα η προσομοίωση

4.3. Παρουσίαση οργανωμένου ιεραρχικού κυνηγιού (alpha, beta)

Στην ενότητα αυτή θα παρουσιαστεί η υλοποίηση του σεναρίου του ιεραρχικού κυνηγιού με ρόλους alpha και beta.

4.3.1. Αρχή "κυνηγιού"- αναζήτηση θηράματος

Παρατηρούμε στην εικόνα 4.3 αρχικά την τυχαία διάταξη των ρομπότ στο χώρο. Η ομάδα των ρομπότ-"λύκων" φαίνεται στα δεξιά σε σχηματισμό με τους beta να είναι πίσω από τον alpha. Αριστερά φαίνεται με πράσινο χρώμα το "κοπάδι" των ρομπότ-"θηραμάτων". Αρχικά τα θηράματα δεν έχουν δει κάποιο ρομπότ λύκο να τα καταδιώκει οπότε κινούνται τυχαία στο χώρο. Τα ρομπότ-"λύκοι" ψάχνουν για κάποιο "θήραμα" με επικεφαλής τον alpha διατηρώντας πάντα την διάταξη τους.



Εικόνα 4.2: Διάταξη των ρομπότ στο χώρο

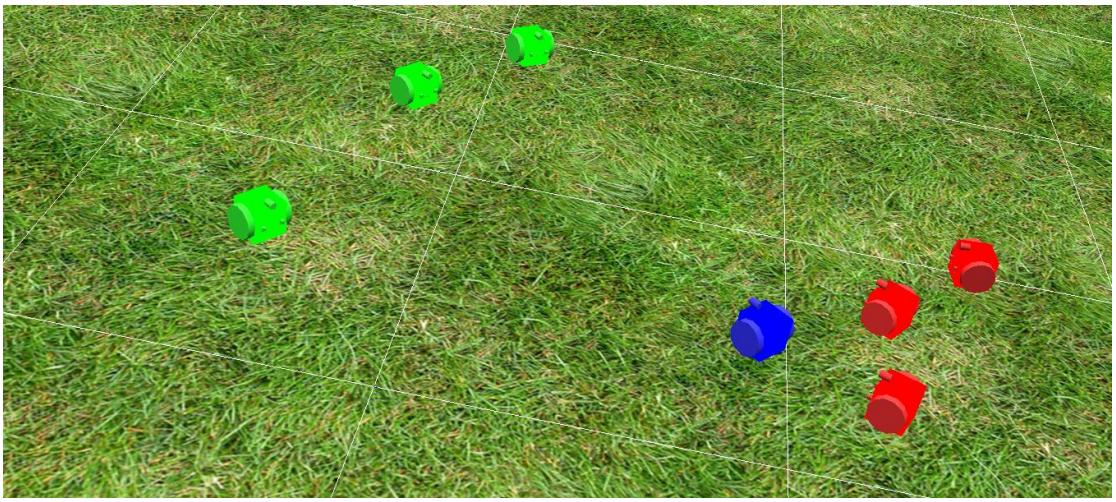
4.3.2. Ο alpha βρίσκει το "θήραμα"

Στη συνέχεια, στην εικόνα 4.4, ο alpha καθώς ψάχνει το χώρο καταφέρνει να "δει" ένα "θήραμα" και γυρίζει προς το μέρος του για να το κυνηγήσει. Οι beta- μη έχοντας δει κάποιο "θήραμα" σε πολύ κοντινή απόσταση έτσι ώστε να το κυνηγήσουν- προσπαθούν όσο πιο γρήγορα γίνεται να σχηματίσουν διάταξη ακολουθώντας τον alpha κατά πόδας.

Εικόνα 4.4: Στιγμιότυπο στο οποίο ο alpha βλέπει θήραμα από μακρινή απόσταση

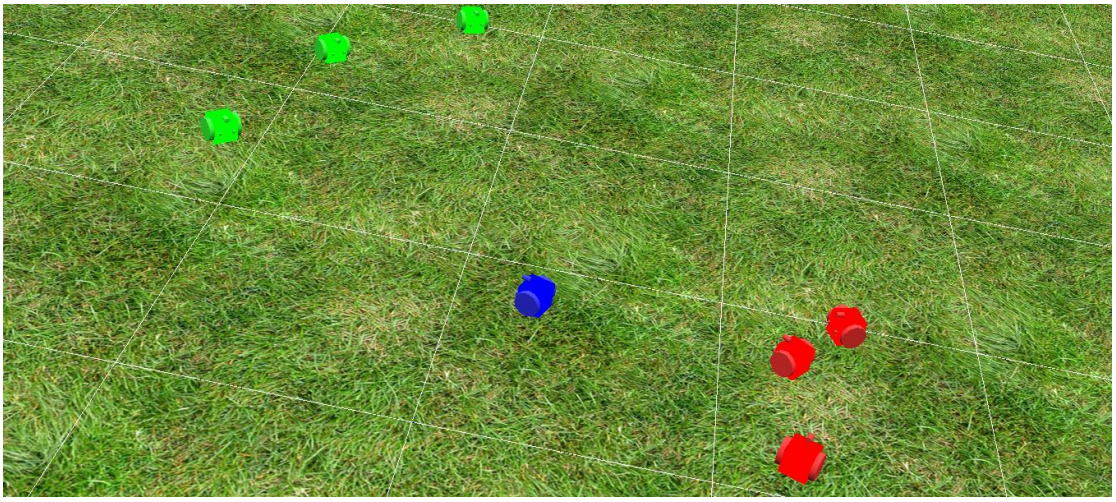
4.3.3. Τα ρομπότ-"θηράματα" βλέπουν ότι οι ρομπότ-"λύκοι" πλησιάζουν προς το μέρος τους

Στην περίπτωση που τα "θηράματα" δουν τους "λύκους" (εικόνα 4.5) να πλησιάζουν τότε αρχίζουν και κινούνται με μεγάλη ταχύτητα προς την αντίθετη κατεύθυνση.



Εικόνα 4.5: Στιγμιότυπο στο οποίο τα ‘’θηράματα” έχουν δει τους κυνηγούς τους

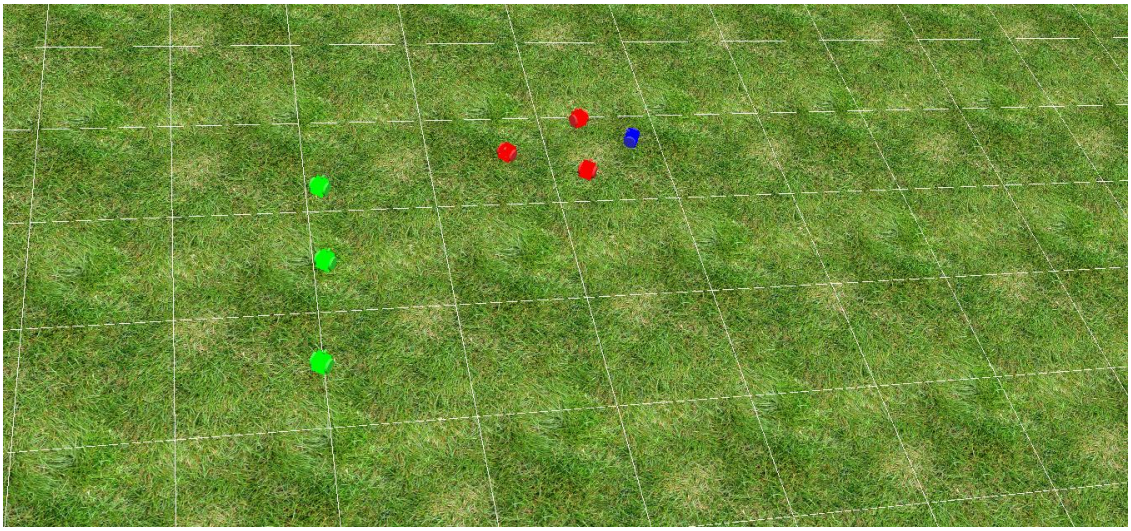
Στην εικόνα 4.6 βλέπουμε τα "θηράματα" να απομακρύνονται από τους "λύκους" (κινούμενοι προς τα πίσω) οι οποίοι συνεχίζουν κανονικά την αναζήτηση τους.



Εικόνα 4.6: Στιγμιότυπο στο οποίο τα θηράματα απομακρύνονται από τους κυνηγούς τους

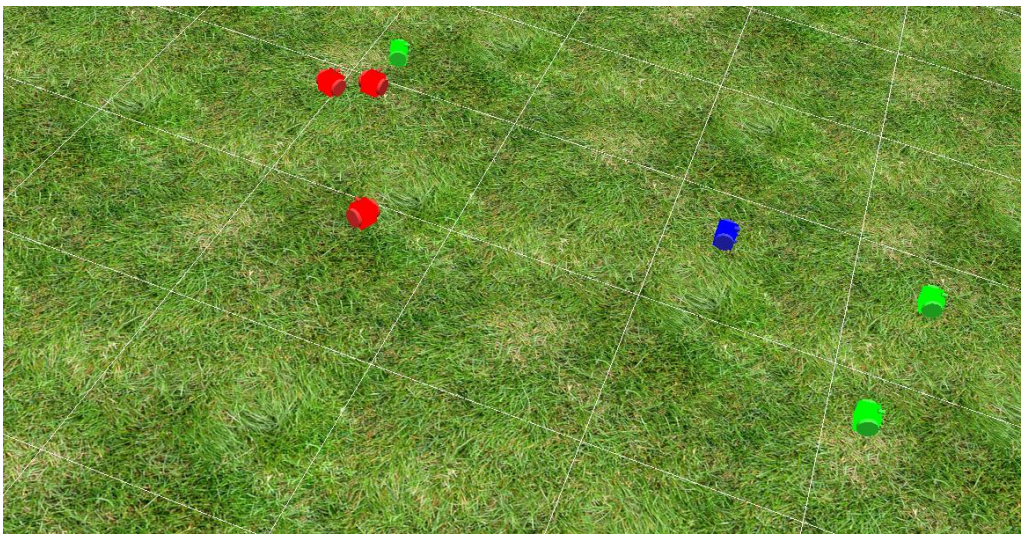
4.3.4. Κάποιος beta " βλέπει το θήραμα" πριν τον alpha

Όπως φαίνεται στο παραπάνω στιγμιότυπο στην εικόνα 4.7 ο beta προσπαθώντας να βρει τον alpha είδε κάποιο "θήραμα". Στη συγκεκριμένη περίπτωση ο beta "ενημερώνει" τον alpha ότι είδε θήραμα και ο alpha, που γνωρίζει τη σχετική διεύθυνση του ως προς τον beta, στρέφεται προς το μέρος του beta προσπαθώντας να βελτιστοποιήσει τις πιθανότητες του να βρει το "θήραμα".



Εικόνα 4.7: Στιγμιότυπο στο οποίο ένας από τους beta έχει δει θήραμα, όντας στη διάταξη της αγέλης

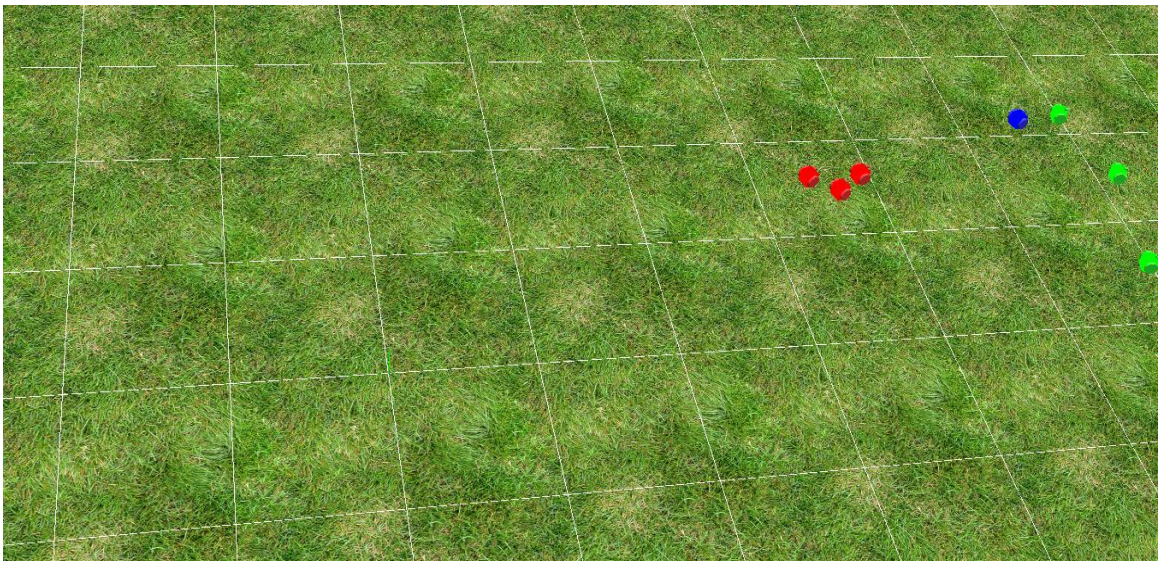
Αν δε, το "θήραμα" βρεθεί για κάποιο λόγο, αρκετά κοντά στον beta τότε και μόνο τότε μπορούν οι beta να αποσπαστούν από τη διάταξη της "αγέλης"(εικόνα 4.8) και να καταδιώξουν το "θήραμα" ανεξάρτητα από τη θέση και τη δραστηριότητα του alpha τη συγκεκριμένη χρονική στιγμή.



Εικόνα 4.8: Στιγμιότυπο στο οποίο οι beta έχουν αποσπαστεί από τη διάταξη της αγέλης για να κυνηγήσουν μόνοι τους ‘’θήραμα’’

4.3.5. Επιτυχής καταδίωξη του "θηράματος"

Ο alpha προσεγγίζει το "θήραμα" αφότου το εντόπισε με την κάμερα. Όπως διαφαίνεται από την εικόνα 4.9, ο alpha θα αιχμαλωτίσει επιτυχώς το θήραμα, το οποίο βρίσκεται στην κατάσταση “βόσκησης” περιπλανώμενο στο χώρο.



Εικόνα 4.9: Στιγμιότυπο στο οποίο ο alpha καταδιώκει το ‘’θήραμα’’

4.4. Παρουσίαση ανοργάνωτου, ανιεραρχικού κυνηγιού

Στην ενότητα αυτή θα παρουσιαστεί η υλοποίηση του σεναρίου του κυνηγιού χωρίς κατανομή εργασιών και χωρίς εμφάνιση ρόλων.

4.4.1. "Αρχή κυνηγιού"- αναζήτηση θηράματος

Εικόνα 4.11: Στιγμιότυπο στο οποίο αρχίζει η προσομοίωση χωρίς κατανομή ρόλων στους "λύκους"

Φαίνεται στην παραπάνω εικόνα 4.11 η διάταξη που έχουν τα ρομπότ στην αρχή της προσομοίωσης. Στο συγκεκριμένο σενάριο υπάρχει παντελής έλλειψη οργάνωσης, συνεννόησης και ιεραρχίας μεταξύ των ρομπότ. Δεν υπάρχουν "δεσμοί" και ρόλοι που να επιφορτίζουν την ομάδα των ρομπότ-"λύκων" και αυτοί δρουν ανεξάρτητοι στην αναζήτηση του πράσινου ρομπότ-"θηράματος".

4.4.2. Προσέγγιση και επίθεση στο "θήραμα"

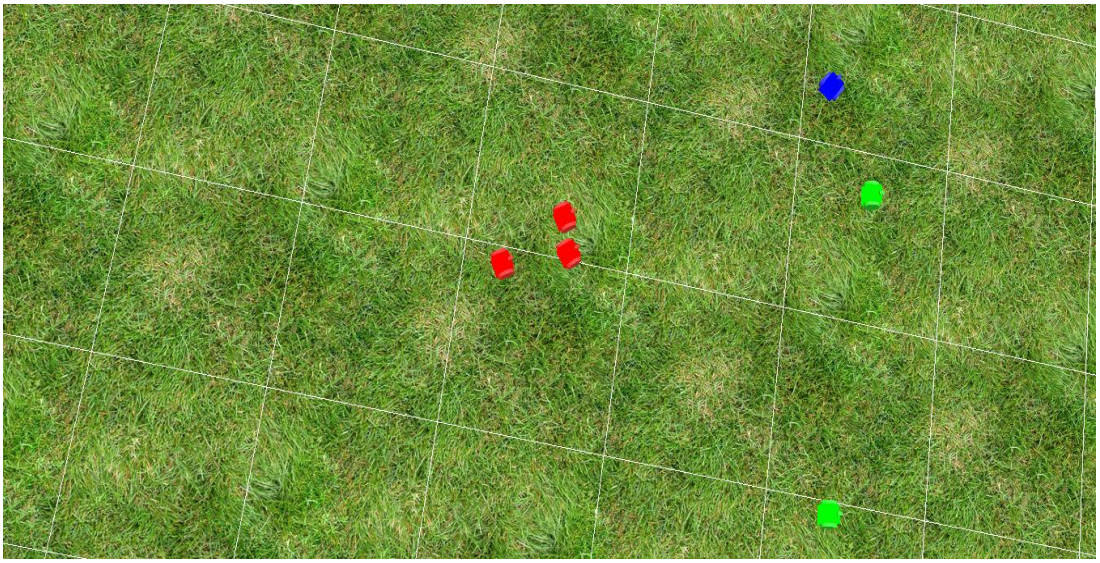
Στην εικόνα 4.12 φαίνεται ότι κάποιοι από τους "λύκους" έχουν εντοπίσει "θήραμα" οπότε θα επιτεθούν σε αυτό όσο πιο άμεσα μπορούν. Ένας μόνο "λύκος", αυτός που είδε πρώτος το "θήραμα", και βρίσκεται εγγύτερα σε αυτό, θα καταφέρει στο τέλος να "φάει" το "θήραμα". Η αναζήτηση θα συνεχιστεί εφ ίσοις όροις έως ότου όλα τα "θηράματα" καταναλωθούν. Η συμπεριφορά του "θηράματος" δεν θα αναλυθεί περαιτέρω καθώς παραμένει πανομοιότυπη με αυτή στο οργανωμένο, ιεραρχικό "κυνήγι". Η κίνηση και η συμπεριφορά των ρομπότ-"λύκων" στην συγκεκριμένη περίπτωση είναι όμοια με τη συμπεριφορά του alpha που αναλύθηκε στην προηγούμενη ενότητα. Στην οργάνωση αυτή κάθε "λύκος" είναι "alpha" προσπαθώντας να μεγιστοποιήσει τις δικές του πιθανότητες για αιχμαλωσία του "θηράματος".

Εικόνα 4.12: Στιγμιότυπο στο οποίο η αναζήτηση "θηράματος" έχει αποδώσει καρπούς για κάποιους από τους "λύκους".

4.5. Ο ρόλος του Supervisor στην προσομοίωση

Η χρήση του Supervisor αποτελεί αναπόσπαστο κομμάτι της προσομοίωσης καθώς διευκολύνει σημαντικά την εξαγωγή συμπερασμάτων και την λήψη μετρήσεων για διάφορες παραμέτρους του κυνηγιού.

Πιο συγκεκριμένα, ο Supervisor, όπως φαίνεται και στην εικόνα 4.13, αναλαμβάνει να εξαφανίσει το "θήραμα" όταν κάποιος λύκος το πιάσει ενώ όταν όλα τα "θηράματα" αιχμαλωτιστούν (στην συγκεκριμένη περίπτωση και τα τρία) ο supervisor ξεκινάει την προσομοίωση από την αρχή. Αν με την πάροδο 5 λεπτών κυνηγιού οι "λύκοι" αποτύχουν να αιχμαλωτίσουν ένα ή περισσότερα "θηράματα" τότε ο supervisor πάλι αναλαμβάνει να επανεκκινήσει την προσομοίωση, διευκολύνοντας έτσι την πρόοδο του κυνηγιού και την εξαγωγή στατιστικών. Τα στατιστικά κάθε προσομοίωσης, συγκεκριμένα οι χρόνοι που υπέπεσε σε αιχμαλωσία κάθε "θήραμα" -0 sec αν το "θήραμα" δεν αιχμαλωτίστηκε- αποτυπώνονται σε ένα φύλλο excel. Οι ενέργειες της εγγραφής και διαχείρισης των στατιστικών προκύπτουν πάλι από τον προγραμματισμό του Supervisor.



Εικόνα 4.13: Σε συνέχεια του στιγμιότυπου της εικόνας 4.9 , ο supervisor έχει εξαφανίσει το θήραμα

5. Ο ρόλος του ελεγκτή (controller)

Στο κεφάλαιο αυτό θα παρουσιάσουμε το προγραμματιστικό κομμάτι της εργασίας όπου θα αναλυθούν οι αλγόριθμοι που υλοποιήθηκαν με τη βοήθεια συναρτήσεων και πρωτοκόλλων του προγράμματος προσομοίωσης. Θα παρουσιάσουμε με τη σειρά που εμφανίζονται στον ελεγκτή μας (και όχι με τη σειρά που εκτελούνται) όλες τις συναρτήσεις του ελεγκτή.

Στις συναρτήσεις alpha, beta και prey, τις συναρτήσεις που ελέγχουν όλη τη συμπεριφορά του alpha, beta και prey αντίστοιχα και αξιοποιούν το μεγαλύτερο κομμάτι των συναρτήσεων που κατασκευάστηκαν , θα χρησιμοποιήσουμε τις "Μηχανές Πεπερασμένων Καταστάσεων ή Finite State Machines" σαν μαθηματικό μοντέλο για να περιγράψουμε την ακολουθία των καταστάσεων και τις μεταβάσεις που πραγματοποιούνται από κατάσταση σε κατάσταση. Με τα Διαγράμματα (Μετάβασης) Καταστάσεων των σχημάτων 4.1,4.2,4.3 θα αναπαραστήσουμε το παραπάνω μοντέλο.

Στο παράρτημα Α παρουσιάζεται ο κώδικας που εμφανίζεται στον ελεγκτή.

5.1. Η συνάρτηση `doWheels`

Η συνάρτηση `doWheels` είναι αυτή που διαχειρίζεται την ταχύτητα των τροχών και την διεύθυνση που θα πάρει το ρομπότ, η οποία λόγω της διαφορικής κίνησης προκύπτει μόνο από κατάλληλο χειρισμό των ταχυτήτων.

5.2. Η συνάρτηση `doTurn`

Η συνάρτηση `doTurn` καλείται όταν απαιτείται το ρομπότ να στρίψει ή να αλλάξει κατεύθυνση επιτόπου.

5.3. Ο αλγόριθμος της όρασης (vision)

Ο αλγόριθμος της όρασης βασίζεται στο μοντέλο χρώματος RGB όπου όλα τα χρώματα προέρχονται από συνδυασμό των τριών χρωμάτων (red,green,blue). Ο αλγόριθμος χρησιμοποιεί σαν "βάση" μια δομή (struct) όπου και με τη βοήθεια συναρτήσεων-πρωτοκόλλων του Webots καταφέρνουμε να "συνθέσουμε" το χρώμα οποιουδήποτε αντικειμένου στο χώρο. Με χρήση απλής τριγωνομετρίας και γνωρίζοντας το πεδίο θέασης της κάμερας(field of view) υπολογίζουμε και τη θέση του αντικειμένου από την κάμερα και άρα και από το ρομπότ που φέρει την κάμερα.

5.4. Η συνάρτηση `setUpEmitterAndReceiver`

Η συνάρτηση setUpEmitterAndReceiver όπως δείχνει και το όνομα της είναι αυτή η οποία διαχειρίζεται τα πρωτόκολλα των Emitter και Receiver. Η μόνη της λειτουργία είναι να θέσει στην διάθεση μας τους Emitter και Receiver και να ρυθμίσει τις κατάλληλες διόδους επικοινωνίας μεταξύ πομπού και δέκτη (βλ. κανάλι).

5.5. Η συνάρτηση `setUpTouchSensor`

Η συνάρτηση `setUpTouchSensor` όπως δείχνει και το όνομα της είναι αυτή η οποία διαχειρίζεται τα πρωτόκολλα του TouchSensor. Ο TouchSensor είναι τύπου "bumper" και επιστρέφει μια τιμή boolean 0 ή 1 (0 αν δεν έχει ανιχνευτεί σύγκρουση και 1 αν ανιχνευτεί σύγκρουση).

5.6. Η συνάρτηση `sendPosition`

Η συνάρτηση `sendPosition` όπως δείχνει και το όνομα είναι αυτή η οποία διαχειρίζεται το μήνυμα το οποίο θα σταλεί με τον emitter καθώς και τον τύπο του μηνύματος.

5.7. Η εύρεση της σχετικής θέσης και της απόστασης του πομπού (emitter) ως προς τον δέκτη (receiver)

Μέσα στη δομή emitterpos επιστρέφεται η σχετική θέση του emitter ως προς τον receiver σαν διάνυσμα [x y z] από την συνάρτηση του Webots `wb_receiver_get_emitter_direction(communication).` Επιστρέφεται ακόμα μέσα στη δομή η απόσταση του πομπού από το δέκτη με τη συνάρτηση `wb_receiver_get_signal_strength (communication).`

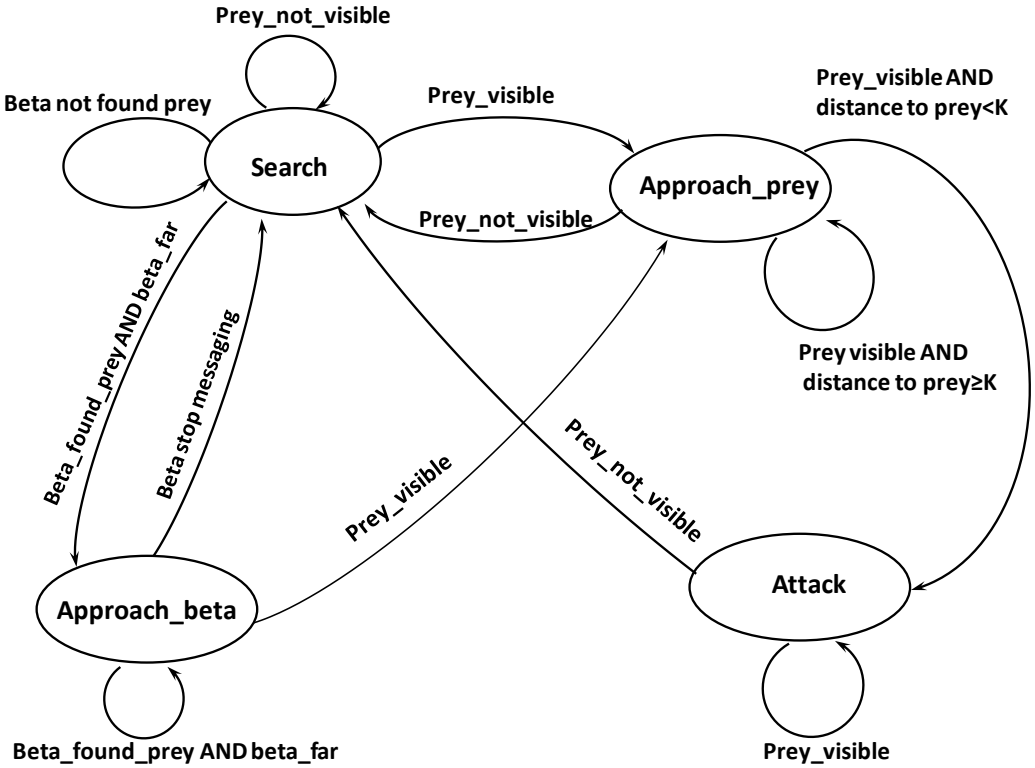
5.8. Η συνάρτηση `avoidOtherWolves`

Αξιοποιώντας τα ευρήματα του αλγορίθμου vision, η συνάρτηση `avoidOtherWolves` εξασφαλίζει σε μεγάλο βαθμό το ότι στην ομάδα των ρομπότ-"λύκων" αλλά και στων "θηραμάτων" δεν θα συγκρουστεί το ένα με το άλλο. Εάν συγκρουστούν ενεργοποιείται ο αισθητήρας αφής, οποίος ανιχνεύει ενδεχόμενη σύγκρουση στο πλαϊνό και στο μπροστινό μέρος του ρομπότ και έτσι καθίσταται πιο εύκολο το "ξεκόλλημα" των συγκρουσθέντων ρομπότ. Η σίγουρη αποφυγή της σύγκρουσης σε κάποιο time step της προσομοίωσης είναι αδύνατη, βέβαια, χωρίς τα ρομπότ να γνωρίζουν τη θέση τους στο χώρο. Το τελευταίο πρόβλημα ανάγεται σε πρόβλημα localization/mapping/navigation εξαιρετικά δυσεπίλυτου και χρονοβόρου για τα πλαίσια της μελέτης της συνεργατικής συμπεριφοράς αγέλης λύκων καθώς κυνηγάνε.

5.9. Η συνάρτηση `Alpha`

Η συνάρτηση Alpha είναι η συνάρτηση που καθορίζει την συμπεριφορά του alpha. Η κίνηση του περιγράφεται βασικά από 4 καταστάσεις, οι οποίες αναπαριστώνται με “Μηχανές Πεπερασμένων Καταστάσεων ή Finite State Machines” .

Ρομπότ-λύκος alpha. Το σχήμα 4.1 παρουσιάζει την κυνηγετική συμπεριφορά του ρομπότ άλφα η οποία καθορίζεται από τέσσερις καταστάσεις.



Σχήμα 4.1: Διάγραμμα (Μετάβασης) Καταστάσεων για τον alpha

- **Αναζήτηση(Search):** Σε αυτό το στάδιο ο alpha εξερευνά το περιβάλλον αναζητώντας ένα θήραμα να "φάει". Η ταχύτητα και η κατεύθυνση με την οποία κινείται το ρομπότ είναι τυχαία. Η αναζήτηση όπως έχουμε περιγράψει σε παραπάνω κεφάλαιο στηρίζεται στη κάμερα (βλ. αλγόριθμο vision) και στην εντοπισμό των αντίστοιχων χρωμάτων (μπλε για τον alpha, πράσινο για το "θήραμα" και κόκκινο για τον beta). Μόλις εντοπίσει το θήραμα η συνθήκη prey_visible (το θήραμα είναι ορατό) ενεργοποιείται υποδεικνύοντας την αλλαγή στην επόμενη κατάσταση, το Approach_Prey. Αν αντί για τον alpha, ο beta δει πρώτος το "θήραμα" και βρίσκεται αρκετά μακριά του για να αποσπαστεί από την αγέλη και να το κυνηγήσει μόνος του, τότε ειδοποιεί τον alpha, ο οποίος θα γυρίσει να κοιτάξει προς το μέρος του beta που έκανε τον εντοπισμό και θα τον πλησιάσει περνώντας έτσι στην κατάσταση Approach_beta.

- **Προσέγγιση του beta(Approach_beta)**: Στην κατάσταση αυτή ο alpha προσεγγίζει τον beta που έχει δει το θήραμα και κινείται προς το μέρος του με σχετικά μεγάλη ταχύτητα. Το νόημα αυτής της συμπεριφοράς είναι, με τους διαθέσιμους αισθητήρες και συσκευές, να μπορέσει ο alpha να "δει αυτό που βλέπει ο beta", δηλαδή το "θήραμα". Εδώ μπορούν να συμβούν δύο μεταβάσεις: μία στην κατάσταση του Approach_Prey στην περίπτωση που ο alpha δει το θήραμα και μία στην κατάσταση της Αναζήτησης (Search) στην περίπτωση που ο beta χάσει το "θήραμα". Αν

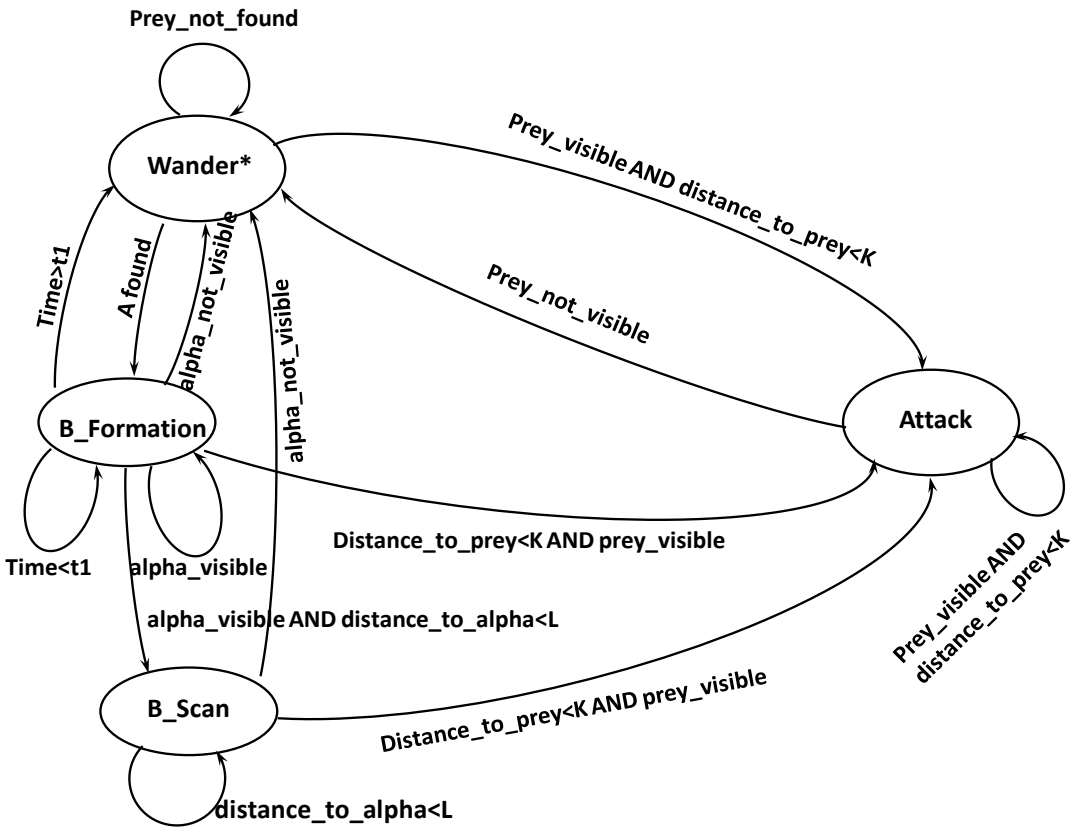
ο beta συνεχίζει να βλέπει το "θήραμα" τότε παραμένουμε στην κατάσταση που ο alpha προσεγγίζει τον beta.

- **Προσέγγιση του "θηράματος"(Approach Prey):** Στην κατάσταση αυτή ο alpha έχει δει το θήραμα, έχει εστιάσει σε αυτό μέσω της κάμερας και κινείται προς το μέρος του με μεγαλύτερη ταχύτητα από αυτή με την οποία κάνει Αναζήτηση. Εδώ μπορούν να συμβούν δύο μεταβάσεις: μία προς το στάδιο της Αναζήτησης πάλι (κατάσταση Search), στην περίπτωση που το θήραμα καταφέρει να απομακρυνθεί ή ο alpha το χάσει κατά την αναζήτηση (prey_not_visible). Η άλλη μετάβαση προκύπτει όταν το θήραμα εντοπίζεται σε κοντινή απόσταση <K, όπου και μεταβαίνουμε στην κατάσταση Attack.

- **Επίθεση (attack):** Σε αυτό το στάδιο ο alpha πλησιάζει το θήραμα μέχρι να το "πιάσει". Είναι στραμμένο συνεχώς προς το μέρος του και η ταχύτητα του alpha όταν μπει στην κατάσταση της Επίθεσης είναι διπλάσια από αυτήν του Approach_Prey. Εάν ο alpha χάσει την οπτική επαφή με το "θήραμα" , πηγαίνει πίσω στην κατάσταση της Αναζήτησης το οποίο ενεργοποιείται από την prey_not_visible.

5.10. Η συνάρτηση Beta

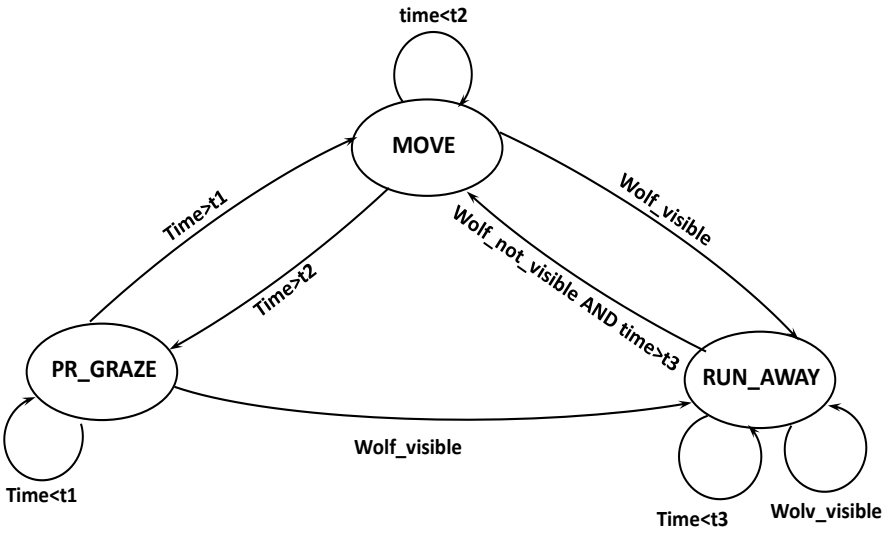
Ρομπότ-λύκος beta. Το σχήμα 4.2 παρουσιάζει την κυνηγετική συμπεριφορά του ρομπότ beta η οποία καθορίζεται από πέντε στάδια.



Σχήμα 4.2: Διάγραμμα (Μετάβασης) Καταστάσεων για τον beta

- **<u>Περιπλάνηση(Wander)</u>:** Ο beta ερευνά το περιβάλλον αναζητώντας τον αρχηγό της ομάδας ή ένα "θήραμα". Η εξερεύνηση του περιβάλλοντος στηρίζεται και σε αυτήν την περίπτωση στην κάμερα. Εάν ο beta βρει τον αρχηγό (Afound) μεταβαίνουμε στην κατάσταση Σχηματισμού (B_ Formation) . Εάν βρει "θήραμα" και αυτό βρίσκεται κοντά του (Prey_visible AND distance_to_prey<K) μεταβαίνουμε στην κατάσταση της Επίθεσης(Attack).
- **<u>Σχηματισμός(B_Formation)</u>**: Όσο ο beta βλέπει τον alpha, μένει κοντά στον αρχηγό. Εάν η οπτική επαφή με τον αρχηγό χαθεί , δηλαδή ισχύει το alpha_not_visible ο beta γυρνάει στην κατάσταση της Περιπλάνησης. Εάν ο beta βρίσκεται κοντά στον alpha αλλά ελάχιστα πιο μακριά από την προκαθορισμένη ακτίνα σχηματισμού, μπορεί να μεταβεί στην κατάσταση B_Scan. Ο beta θα παραμείνει στην κατάσταση B_Formation το μέγιστο πέντε δευτερόλεπτα εάν καμία άλλη μετάβαση δεν θα πυροδοτηθεί. Μετά την πάροδο των πέντε δευτερολέπτων θα επανέλθει στην αρχική κατάσταση της Περιπλάνησης.
- **<u>Σάρωση του χώρου(B_Scan)</u>:** Η κατάσταση αυτή είναι κατάσταση αναζήτησης αλλά επιτόπου, χωρίς μετακίνηση. Έχει το νόημα της αναζήτησης "θηράματος" ακόμη και μέσα στα όρια της αγέλης και όχι την προσκόλληση στην ανεύρεση του alpha. Σε αυτό το στάδιο μπορούν να συμβούν δύο μεταβάσεις: μια μετάβαση προκύπτει όταν ο beta εντοπίσει ένα "θήραμα" σε κοντινή απόσταση (Distance_to_prey<K AND prey_visible) η οποία οδηγεί στην κατάσταση της Επίθεσης. Μια άλλη μετάβαση συμβαίνει στην περίπτωση που ο beta χάσει τα ίχνη του alpha (alpha_not_visible), και οδηγεί στην κατάσταση της Περιπλάνησης.
- **<u>Επίθεση(Attack)</u>:** Ο beta θεωρεί τον εαυτό του αρκετά κοντά στο θήραμα αν και ακόμη λαμβάνει υπόψιν τη σχετική θέση του ίδιου και του αρχηγού. Ωστόσο, σε αυτό το στάδιο η κίνηση για την προσέγγιση και το πιάσιμο του "θηράματος" έχει μεγαλύτερη προτεραιότητα από το να ακολουθήσει τον αρχηγό. Παρ' όλα αυτά το θήραμα μπορεί να ξεφύγει επειδή είναι πιο γρήγορο. Εάν συμβεί αυτό τότε ισχύει το Prey_not_visible (το θήραμα είναι μακριά), όπως φαίνεται και στο σχήμα 4.2 και ο beta επιστρέφει στο στάδιο Περιπλάνησης.

5.11. Η συνάρτηση `Prey`



Σχήμα 4.3: Διάγραμμα (Μετάβασης) Καταστάσεων για το ''θήραμα''

- **Βόσκηση (PR_GRAZE):** Το "θήραμα" περιπλανάται στο χώρο με μικρή ταχύτητα και τυχαία κατεύθυνση-η οποία όμως αλλάζει σχετικά λίγο-προσομοιώνοντας έτσι την βόσκηση ενός τυπικού θηράματος του λύκου, όπως ο τάρανδος, ο βίσωνας κλπ. Το "θήραμα" παραμένει σ αυτήν την κατάσταση για πέντε δευτερόλεπτα ενώ με την πάροδο των πέντε δευτερολέπτων -αν δεν υπάρξει άλλη μετάβαση- μεταβαίνουμε στην κατάσταση "Μετατόπιση (MOVE) ".
- **Μετατόπιση (MOVE):** Στην κατάσταση αυτή το "θήραμα" κινείται πιο επιδέξια στο χώρο δηλαδή με μεγαλύτερη ταχύτητα και αλλάζοντας πιο γρήγορα την κατεύθυνση του- προσομοιώνοντας έτσι την μετατόπιση ενός τυπικού θηράματος από μια περιοχή βόσκησης σε μια άλλη. Το "θήραμα" παραμένει σ αυτή την κατάσταση δέκα δευτερόλεπτα και όταν αυτά παρέλθουν επανέρχεται στην αρχική κατάσταση της "βόσκησης ".
- **Αποφυγή(RUN_AWAY):** Αν στο οπτικό πεδίο του "θηράματος " (βλ. Field of view της κάμερας) βρεθεί κάποιος "λύκος" τότε ενεργοποιείται η συμπεριφορά αποφυγής σε όποια άλλη κατάσταση και να βρισκόμαστε και το "θήραμα" απομακρύνεται από τον κυνηγό του, δίδοντας μεγάλη και αντίθετη, από αυτή που είχε τη στιγμή που είδε τους "λύκους", ταχύτητα στους τροχούς. Η κατάσταση αυτή ισχύει για όσο το "θήραμα" "βλέπει" κάποιο "λύκο". Αν οι κυνηγοί εξαφανιστούν από το οπτικό του πεδίο τότε η κατάσταση παύει να ισχύει μετά από χρόνο είκοσι-πέντε δευτερολέπτων όπου η μετάβαση που πραγματοποιείται είναι στην κατάσταση "Μετατόπιση".

Σημαντικό να αναφερθεί είναι ότι η ταχύτητα στην κατάσταση της Αποφυγής είναι μεγαλύτερη από την κατάσταση με την οποία επιτίθενται ο alpha και ο beta. Αυτό συμβαίνει για να προσομοιώσουμε την ταχύτητα κίνησης των θηραμάτων στην πραγματικότητα η οποία είναι μεγαλύτερη σε πολλές περιπτώσεις από αυτή των λύκων.

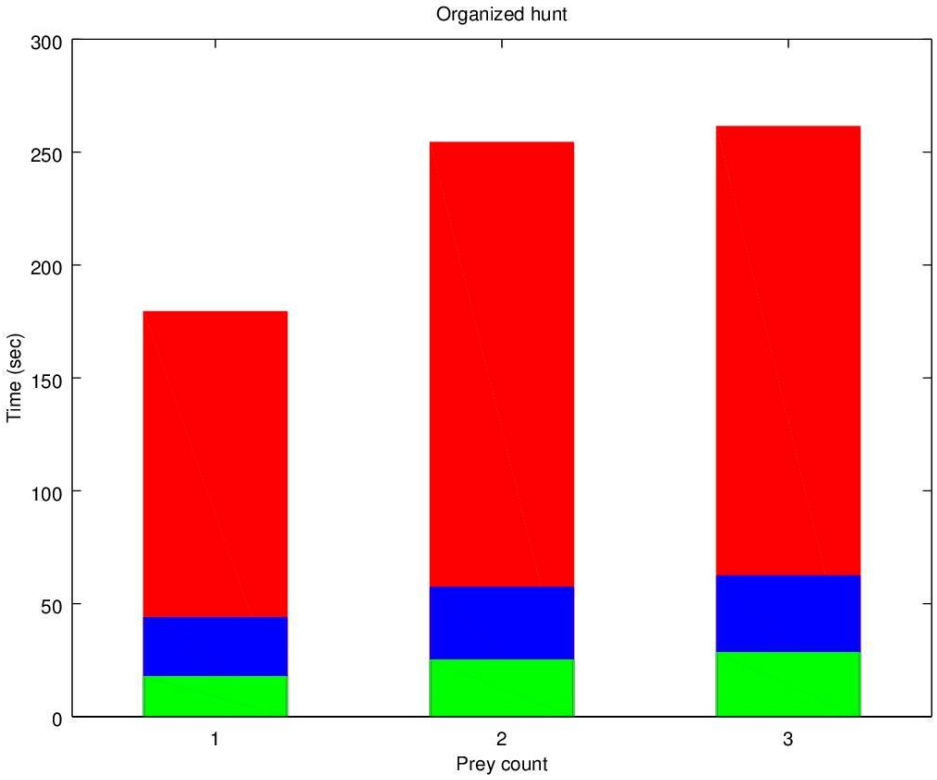
5.12. Η Συνάρτηση `main`

Όλα τα προγράμματα της C πρέπει να έχουν μια συνάρτηση με όνομα main(). Από την συνάρτηση αυτή ξεκινάει η εκτέλεση του προγράμματος. Μετά από την main() ακολουθεί το αριστερό άγκιστρο. Αυτό σηματοδοτεί την αρχή των εντολών που αποτελούν τον κορμό της συνάρτησης. Η εκτέλεση του προγράμματος ολοκληρώνεται με τον τυπικό τρόπο, όταν συναντήσει το δεξιό άγκιστρο που κλείνει τον κορμό εντολών της main().

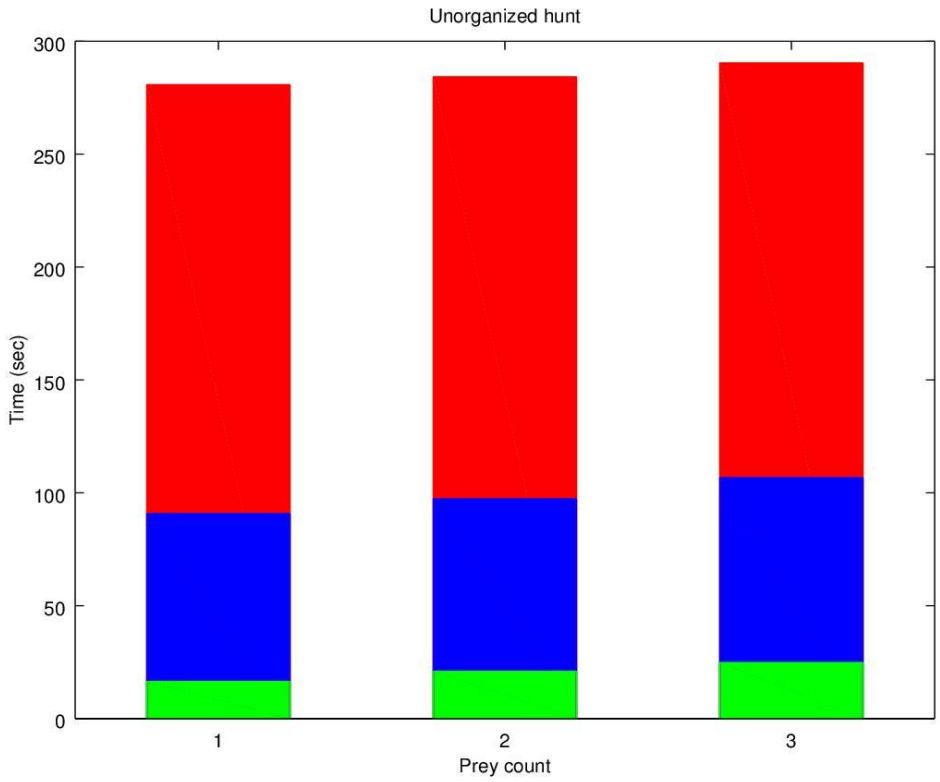
6. Πειραματικά Αποτελέσματα

Για καθένα από τα σενάρια οργανωμένου και ανοργάνωτου κυνηγιού, όπου και στις δύο περιπτώσεις οι συνθήκες είναι όμοιες (ίδια αφετηρία ρομπότ, ίδια διάταξη και κατεύθυνση ρομπότ στο χώρο κ.λ.π.), διενεργήθηκαν 200 κυνήγια/προσομοιώσεις. Τα πειραματικά αποτελέσματα που προέκυψαν σχετίζονται με το χρόνο στον οποίο αιχμαλωτίζεται καθένα από τα τρία θηράματα (βλ. παράρτημα Β΄). Έπειτα, η ανάλυση των αποτελεσμάτων στηρίζεται στο στατιστικό κριτήριο confidence interval (διάστημα εμπιστοσύνης) και από αυτήν προκύπτει ο μέγιστος παρατηρούμενος, μέσοι χρόνοι και ελάχιστος παρατηρούμενος χρόνος αιχμαλώτισης των θηραμάτων. Οι χρόνοι αυτοί παρουσιάζονται στα σχήματα 6.1 και 6.2 για τις περιπτώσεις του οργανωμένου και ανοργάνωτου κυνηγιού αντίστοιχα. Φαίνεται η υπεροχή του μοντέλου που στηρίζεται στην κατανομή ρόλων (alpha,beta) έναντι του ανοργάνωτου, ανιεραρχικού.

Πιο συγκεκριμένα στα παρακάτω ιστογράμματα (σχήμα 6.1 και 6.2) παρουσιάζονται τα αποτελέσματα κάθε κυνηγιού ενώ στον πίνακα 6.1 φαίνονται συγκεντρωτικά όλοι οι χρόνοι που έκαναν για να αιχμαλωτίσουν κάθε ένα από τα τρία "θηράματα".



Σχήμα 6.1: Χρόνος αιχμαλωσίας των τριών θηραμάτων για το Σενάριο του Οργανωμένου Κυνηγιού (Organized Hunt). Κόκκινο χρώμα: Μέγιστος χρόνος, t_max. Μπλε χρώμα: Μέσος χρόνος t_mean. Πράσινο χρώμα: Ελάχιστος χρόνος t_min.



Σχήμα 6.2: Χρόνος αιχμαλωσίας των τριών θηραμάτων για το Σενάριο του Ανοργάνωτου Κυνηγιού (Unorganized Hunt). Κόκκινο χρώμα: Μέγιστος χρόνος, t_max. Μπλε χρώμα: Μέσος χρόνος t_mean. Πράσινο χρώμα: Ελάχιστος χρόνος t_min.

Πίνακας 6.1: Συγκεντρωτικοί χρόνοι αιχμαλωσίας των τριών θηραμάτων για το Σενάριο του Οργανωμένου και του Ανοργάνωτου Κυνηγιού (Organized and Unorganized Hunt)

Ανοργάνωτο κυνήγι				Οργανωμένο κυνήγι			
	Θήραμα 1	**Θήραμα 2**	**Θήραμα 3**		**Θήραμα 1**	**Θήραμα 2**	**Θήραμα 3**
t_ max	280.77	284.22	290.37	**t_ max**	179.33	254.34	261.31
t_ mean	90.783	97.324	106.774	**t_ mean**	43.842	57.246	62.390
t_ min	16.448	20.928	24.896	**t_ min**	17.664	25.088	28.288

7. Συμπεράσματα

Η διπλωματική εργασία αφορά την ανάπτυξη ενός μοντέλου εμπνευσμένου από τα ζώα και τη συμπεριφορά τους με σκοπό να εξερευνήσουμε έννοιες όπως σχηματισμός ομάδας και συνεργασία στη Ρομποτική και να αποφανθούμε αν πολύπλοκα βιολογικά μοντέλα μπορούν να χρησιμεύσουν σαν έμπνευση σε ρομποτικές εφαρμογές καταδίωξης ή απόδρασης. Για την επίτευξη των παραπάνω στόχων εκτός από τη δημιουργία του συμπεριφοριστικού μοντέλου και τη σχεδίαση των διάφορων στρατηγικών έπρεπε να έρθουμε σε επαφή με θέματα υλικού των ρομπότ, που έφεραν κανονικά αισθητήρες, μηχανισμούς κίνησης, ελεγκτές κλπ. Το

πολύ-ρομποτικό σύστημα που δημιουργήθηκε πέρασε από πολλά στάδια πειραματισμών ενώ δοκιμάστηκαν πολύ διαφορετικοί τρόποι μοντελοποίησης της βιομιμητικής συμπεριφοράς, για να φτάσει στην τελική του μορφή την οποία παρουσιάσαμε σε αυτή την εργασία.

Γενικά, το μοντέλο από μόνο του είναι πολύ περιορισμένο από πολλές πλευρές καθώς η συμπεριφορά του λύκου μπορεί να γίνει αρκετά πολύπλοκη έτσι ώστε να μην μπορεί να προσομοιωθεί από ρομπότ. Για παράδειγμα, τα ρομπότ στην περίπτωση μας μπορούν να χρησιμοποιήσουν μόνο την όραση μέσω κάμερας (που προφανώς υστερεί σε σχέση με την όραση των ζώων). Το γεγονός αυτό τα περιορίζει αρκετά σε σχέση με τους λύκους που έχουν το πλεονέκτημα της όσφρησης και της ακοής για τον εντοπισμό του θηράματος.

Η λειτουργικότητα και η αξιοπιστία του συστήματος εξακριβώθηκε πολλές φορές στο περιβάλλον της προσομοίωσης ενώ οι πειραματικές δοκιμές και η συνέπεια των αποτελεσμάτων επιβεβαίωσαν τις υποθέσεις μας ότι το οργανωμένο, ιεραρχικό κυνήγι θα ήταν πιο γρήγορο και αποτελεσματικό από το ανοργάνωτο. Στην εποχή της ακραιφνούς ανάπτυξης αυτόνομων ρομποτικών συστημάτων με έμφαση στην ομαδικότητα και την κατανομή εργασιών η συνεργατική συμπεριφορά ενός τόσο "επιτυχημένου" βιολογικού μοντέλου σαν το κυνηγετικό μοντέλο των λύκων δεν μπορεί παρά να έχει αμέτρητες εφαρμογές στην καθημερινή, και όχι, ζωή. Εφαρμογές όπως :
Εξερεύνηση και καταγραφή εξωτερικών χώρων: ένα σμήνος/ ομάδα αυτόνομων ρομποτικών πρακτόρων "διανέμεται" σε ένα περιβάλλον και καταγράφει χαρτογραφεί το χώρο. Αποστολή του μπορεί να είναι ο εντοπισμός μια νάρκης σε μια εμπόλεμη περιοχή ή απλά η χαρτογράφηση ενός δυσπρόσιτου μέρους.
Εντοπισμός/διάσωση θυμάτων σε φυσική καταστροφή: ένα σμήνος/ομάδα αυτόνομων ρομποτικών πρακτόρων αναζητεί θύματα στα συντρίμμια ενός σεισμού ή μιας πλημμύρας, μιας πυρκαγιάς ή μιας πυρηνικής έκρηξης, εργασία εξαιρετικά δύσκολη για το ανθρώπινο σώμα.
Ιατρική: διεκπεραίωση επεμβάσεων που απαιτείται τεράστια ακρίβεια και λεπτοί χειρισμοί. Το ανθρώπινο χέρι ακόμα δεν έχει καταφέρει αποτελεσματικά επεμβάσεις όπως ενδοσκόπηση οπού αυτόνομα νάνο-ρομπότ μπορούν να εισέλθουν στην κυκλοφορία του αίματος και να εκτελέσουν την επέμβαση με μεγαλύτερη αποτελεσματικότητα.
Swarm Robotics: η ιδέα της ύπαρξης και του συντονισμού μεγάλων ομάδων ρομπότ εξελίσσεται ραγδαία και ο κλάδος του Swarm Robotics ασχολείται με αυτό ακριβώς το αντικείμενο. Έχει εμπνευστεί από την ομαδική συμπεριφορά ζώων που συγκροτούνται σε σμήνη και μέσω της συνεργασίας διασφαλίζουν την επιβίωση τους και εκτελούν πολλές διαφορετικές εργασίες τη μέρα (βλ. μυρμήγκια, έντομα κλπ.)

8. Βιβλιογραφία

1. Robin R. Murphy, Introduction to AI ROBOTICS , The MIT Press , 2000
2. Δ.Μ. Εμίρης ,Δ.Ε. Κουλουριώτης , ΡΟΜΠΟΤΙΚΗ ,3η έκδοση, ΑΘΗΝΑ 2013
3. Peter Menzel and Faith D'Aluisio , Robo Sapiens: Evolution of a new species, The MIT Press,2000
4. Webots Reference Manual and User Guide, [Online], Available at https://www.cyberbotics.com/
5. Wolf's Hunting,[Online], Available at https://livingwithwolves.org/ , Non Profit Organisation
6. Douglas W. Smith, Daniel R. Stahler, and Debra S. Guernsey, Matthew Metz, Abigail Nelson, Erin Albers, Richard McIntyre, Yellowstone Wolf Project, Annual Report, 2006
7. Daniel R. MacNulty, Douglas W. Smith, L. David Mech, John A. Vucetich, and Craig Packera, 2011,Nonlinear effects of group size on the success of wolves hunting elk
8. Daniel R. MacNulty, Aimee Tallian, Daniel R. Stahler, Douglas W. Smith,2014, Influence of Group Size on the Success of Wolves Hunting Bison
9. Daniel MacNulty, L. David Mech, Douglas Smith,2007, A Proposed Ethogram of Large-Carnivore Predatory Behavior, Exemplified by the Wolf
10. Alfredo Weitzenfeld, *Senior Member, IEEE*, Alberto Vallesa and Horacio Flores,2006, A Biologically-Inspired Wolf Pack Multiple Robot Hunting Model
11. John D. Madden, Ronald C. Arkin, *Fellow, IEEE*, and Daniel R. MacNulty, 2010, Multi-robot System Based on Model of Wolf Hunting Behavior to Emulate Wolf and Elk Interactions
12. Barbara Webb , 2000, What does robotics offer animal behavior ?
13. Ronald C. Arkin ,Behavior based Robotics, 1998 , Massachusetts Institute of Technology
14. Πιπερίδης Σάββας, Συνεργατική συμπεριφορά υποβρυχίων αυτόνομων ρομποτικών σκαφών, 2013, Διδακτορική Διατριβή
15. Πιπερίδης Σάββας, Τσουρβελούδης Νίκος, 2014, Biomimetic behaviour based underwater control, Πλήρης Δημοσίευση σε Συνέδριο

ΠΑΡΑΡΤΗΜΑ Α΄

Συνάρτηση doTurn

```
void doTurn(float direction)
{
  double M=wb_differential_wheels_get_max_speed();
  wb_differential_wheels_set_speed(direction*M,-direction*M);

}
```

Συνάρτηση doWheels

```
void  doWheels( float speed, float direction) //Direction>0 = right
  {
   float left,right;
   if(direction>0)
     right=speed-speed*direction;
   else
     right=speed;
   if(direction<0)
     left=speed+speed*direction;
   else
     left=speed;

   double M=wb_differential_wheels_get_max_speed();
   wb_differential_wheels_set_speed(left*M,right*M);
  }
```

Ο αλγόριθμος της όρασης (vision)

```
struct visionobject cameraSearch(int r, int g, int b)
{
  struct visionobject ret;
  WbDeviceTag camera;
  int width, height;
  const unsigned char *image;
  int i, j;
```

```
  /* Get the camera device, enable it, and store its width and height */
camera = wb_robot_get_device("camera");
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
image = wb_camera_get_image(camera);

int mi=0,mj=0;
float mdist;
mdist=2e5;
for (i = 0; i < width; i++) {
  for (j=0; j<height; j++){
          int rr, gg,bb;
          rr=wb_camera_image_get_red(image, width, i, j);
          gg=wb_camera_image_get_green(image, width, i, j);
          bb=wb_camera_image_get_blue(image, width, i, j);

          float dist=sqrt((rr-r)*(rr-r)+(gg-g)*(gg-g)+(bb-b)*(bb-b));
          if(dist<mdist)
          {
            mi=i;
            mj=j;
            mdist=dist;
          }

  }
}
ret.x=(mi*1.0)/width;
ret.y=(mj*1.0)/height;
double vfov=wb_camera_get_fov(camera)*height/width;

double lastdist=mdist;
for (j=mj;j<height;j++)
{
          int rr, gg,bb;
          rr=wb_camera_image_get_red(image, width, mi, j);
          gg=wb_camera_image_get_green(image, width, mi, j);
          bb=wb_camera_image_get_blue(image, width, mi, j);
          float dist=sqrt((rr-r)*(rr-r)+(gg-g)*(gg-g)+(bb-b)*(bb-b));

          if(fabs(dist-lastdist)>60)
          {
            j--;
            break;
          }

          lastdist=dist;

}
double phi=vfov*(((j*1.0)/height)*2.0-1.0)/2;
double d=0.12/atan(phi);

if(j==height)  d=0;
//printf("phi:%f %f %d %f\n",phi,d,j,mdist);
ret.d=d;



ret.fidelity=mdist;
return ret;
```

```
};
```

Συνάρτηση setUpEmitterAndReceiver

```
void setUpEmitterAndReceiver(unsigned transmit,unsigned receive)
{
  WbDeviceTag communication;
   communication = wb_robot_get_device("receiver");
   wb_receiver_enable(communication, TIME_STEP);
   wb_receiver_set_channel(communication,receive);
   communication = wb_robot_get_device("emitter");
   wb_emitter_set_channel(communication,transmit);

}
```

Συνάρτηση sendPosition

```
void sendPosition()
{
  WbDeviceTag communication;
   communication = wb_robot_get_device("emitter");
   const char *message = "Hello !";
   wb_emitter_send(communication, message, strlen(message) + 1);


}
```

Η εύρεση της σχετικής θέσης και της απόστασης του πομπού (emitter) ως προς τον δέκτη(receiver)

```
struct emitterpos
{
  double dirx,dirz;
  double distance;
};
struct emitterpos getEmmitterDirectionAndDistance()
{
   WbDeviceTag communication;
   struct emitterpos returnvalue;
   returnvalue.distance=-1;
   returnvalue.dirx=-1;
   returnvalue.dirz=-1;
```

```
   communication = wb_robot_get_device("receiver");

     while (wb_receiver_get_queue_length(communication) > 0) {

        /* read current packet's data */
        //const char *buffer = wb_receiver_get_data(communication);
        double                                                *directionvector=
wb_receiver_get_emitter_direction(communication);

//printf("%f,%f,%f\n",directionvector[0],directionvector[1],directionvector
[2]);
        returnvalue.dirx=-directionvector[0];                          //
        returnvalue.dirz=directionvector[2];
        double
inverseradiussquared=wb_receiver_get_signal_strength(communication);     //=
1/r^2
        returnvalue.distance=sqrt(1/inverseradiussquared);

        /* fetch next packet */
        wb_receiver_next_packet(communication);

      }
      return returnvalue;
}
```

Συνάρτηση avoidOtherWolves

```
bool avoidOtherWolves(float minimum_distance)
{

     {
       //printf("RED\n");
       struct visionobject r=cameraSearch(REDROBOT);


       int foundTarget=r.fidelity<FIDELITYTHRESHOLD;
       if(foundTarget==1)
       {
         if(r.d<minimum_distance)
         {
             float direction=r.x-0.5;
             doWheels(-0.3,-direction);
             //printf("OBSTACLE------------------------------\n");
             return true;
         }

       }

     }

     {
       //printf("BLUE\n");
       struct visionobject r=cameraSearch(BLUEROBOT);
```

```
        int foundTarget=r.fidelity<FIDELITYTHRESHOLD;
        if(foundTarget==1)
        {
          if(r.d<minimum_distance)
          {
              float direction=r.x-0.5;
              doWheels(-0.3,-direction);
              return true;
          }

        }

      }

      {

         WbDeviceTag b;
         b = wb_robot_get_device("bumper");
         int value=wb_touch_sensor_get_value(b);
//         printf("%d\n",value);
         if(value>0.5)
         {

            doWheels(-0.1,0);
            printf("bumped each other\n");
            return true;
         }

      }
      return false;


}
```

Συνάρτηση setUpTouchSensor

```
void setUpTouchSensor()
{
 WbDeviceTag b;
 b = wb_robot_get_device("bumper");
 wb_touch_sensor_enable(b,TIME_STEP);


}
```

Συνάρτηση Alpha

```
enum                              UnorganizedStates                              {
SEARCH,APPROACH_PREY,APPROACH_BETA,ATTACK,CAPTURE};
```

```
void Alpha()
{
   int channel;
   enum UnorganizedStates  state=SEARCH;

  {
    WbDeviceTag camera;
    camera = wb_robot_get_device("camera");
    wb_camera_enable(camera, TIME_STEP);
  }

setUpEmitterAndReceiver(COMMUNICATION_CHANNEL_ALPHA,COMMUNICATION_CHANNEL_B
ETA);
  setUpTouchSensor();

  while (wb_robot_step(TIME_STEP) != -1)
  {
     struct visionobject r=cameraSearch(GREENROBOT);

     struct emitterpos betainformation=getEmmitterDirectionAndDistance();

     //  printf("BETASENDSSOMETHING\n");
     //printf("Green object found at : %f,%f %f\n",r.x,r.y,r.fidelity);

     int foundTarget=r.fidelity<FIDELITYTHRESHOLD;


     if(avoidOtherWolves(OBSTACLE_AVOID_DISTANCE)==true) continue;

     //printf("%d\n",state);
     if(state==SEARCH)
     {
       float speed = (rand()*1.0)/RAND_MAX;
       float direction = (rand()*1.0)/RAND_MAX;
       speed=speed*0.6;
       direction=direction*2.0-1.0;
       direction=direction/fabs(direction);
       doWheels(speed,direction/1.0);      // strivw deksia!
       if(foundTarget==1)
         state=APPROACH_PREY;
       else
if(betainformation.distance>0&&betainformation.distance>ALPHA_FORMATION_RAD
IUS)
         state=APPROACH_BETA;
       printf("Search\n");




     }

else if(state==APPROACH_PREY)
     {
     //printf("Approach %d\n",foundTarget);
         if(foundTarget==0)
         {
             state=SEARCH;

         }
         else if(foundTarget==1 )
         {
```

```
            if(r.d<ALPHA_PREY_NEAR)
             state=ATTACK;
            else

            {
              float direction=r.x-0.5;
               doWheels(0.2,direction);

            }

        }
    }

    else if(state==APPROACH_BETA)
    {
    printf("FOLLOW BETA\n");
        if(foundTarget==1)
            state=APPROACH_PREY;

        else if(betainformation.distance<0)
        {
             printf("nodata");
          state=SEARCH;

        }
        else
if(betainformation.distance>0&&betainformation.distance<ALPHA_FORMATION_RAD
IUS)
        {
                 printf("have data----------------------------------------
\n");
          state=SEARCH;
        }
        else if(betainformation.distance>0)
        {
         printf("FOLLOW BETA--------------------\n");
         float direction=betainformation.dirx*0.5;
         //printf("%f\n",direction);
         if(betainformation.dirz<0)
         {
            if(betainformation.dirx>=0)
                doTurn(0.5);
            else
               doTurn(-0.5);
         }
         else
           doWheels(0.4,direction);
        }

    }
    else   if(state==ATTACK)
    {
        // printf("attack %d\n",foundTarget);
        if(foundTarget==0)
        {
           state=SEARCH;
        }
        else
        {
           float direction=r.x-0.5;
           doWheels(0.5,direction);
```

```
        }
    }

  }
}
```

Συνάρτηση Beta

```
enum BetaStates {B_WANDER,B_FORMATION,B_SCAN,B_ATTACK,B_EAT};
unsigned BetaStatesTimer[4]= {1000/TIME_STEP,5000/TIME_STEP,0,0};
void Beta()
 {
    //int message_printed = 0; /* used to avoid printing continuously the
communication state */

    enum BetaStates  state=B_WANDER;
    unsigned stateTimer =0;

  {
    WbDeviceTag camera;
    camera = wb_robot_get_device("camera");
    wb_camera_enable(camera, TIME_STEP);
  }


setUpEmitterAndReceiver(COMMUNICATION_CHANNEL_BETA,COMMUNICATION_CHANNEL_AL
PHA);
  setUpTouchSensor();
  while (wb_robot_step(TIME_STEP) != -1)
  {

     stateTimer++;
     //printf("-------------------------------------------------\n");
     if(avoidOtherWolves(OBSTACLE_AVOID_DISTANCE)==true) continue;

     struct visionobject r=cameraSearch(GREENROBOT);

     int foundTarget=r.fidelity<FIDELITYTHRESHOLD;
     if(foundTarget==1)
     {

          sendPosition();
          //printf("Beta:send\n");
          if(r.d<BETA_PREY_NEAR)
          {
            //printf("near %f\n",r.d);
            state=B_ATTACK;
            stateTimer=0;
          }

     }
```

```
      if(state==B_WANDER)
      {


        if(stateTimer>BetaStatesTimer[0])
           stateTimer=0;
        if(stateTimer==0)
        {

           float direction = (rand()*1.0)/RAND_MAX;
           float speed=  (rand()*1.0)/RAND_MAX;
           speed=speed*0.1;
           direction=direction*2.0-1.0;

           if( ((rand()*1.0)/RAND_MAX)>0.5)
             doTurn(direction*0.5);
           else
             doWheels(speed,direction);

        }


         {
           struct visionobject r=cameraSearch(BLUEROBOT);
           int foundTarget=r.fidelity<FIDELITYTHRESHOLD;
           if(foundTarget==1 )
           {
              state=B_FORMATION;
              stateTimer=0;


           }

        }

      }
      else if(state==B_FORMATION)
      {

           struct visionobject r=cameraSearch(BLUEROBOT);
           int foundTarget=r.fidelity<FIDELITYTHRESHOLD;

           float direction=r.x-0.5;
           double dradius=r.d-ALPHA_FORMATION_RADIUS;


           if(foundTarget==0||stateTimer>BetaStatesTimer[1])
           {

              state=B_WANDER;
              stateTimer=0;

           }
           else if(dradius<ALPHA_FORMATION_RADIUS/10.0)
          {
              state=B_SCAN;
              stateTimer=0;
          }
//           printf("%f\n",dradius);
          else
          {
```

```
            dradius=          dradius*0.25;
            float speed=dradius;
            if(fabs(dradius)>1)
                speed=dradius/fabs(dradius);
            if(speed<0)
               direction=-direction;
            doWheels(speed,direction);
          }
      }
      else if(state==B_SCAN)
      {

          struct visionobject r=cameraSearch(BLUEROBOT);
           int foundTarget=r.fidelity<FIDELITYTHRESHOLD;
            if(foundTarget==0)
           {

              state=B_WANDER;
              stateTimer=0;

           }
           else
           {

            float         direction         =         (rand()*1.0)/RAND_MAX;
direction=direction*2.0-1.0;
            doTurn(direction*0.5);
           //stateTimer=0;
           }


      }
      else if(state==B_ATTACK)
      {
         if(foundTarget==0)
         {
            state=B_WANDER;
            stateTimer=0;

         }
         else
         {


            float direction=r.x-0.5;
            doWheels(0.4,direction);
         }


      }

  }
}
```

Συνάρτηση Prey

```
enum PreyStates { PR_GRAZE, PR_MOVE, PR_RUN_AWAY};
unsigned                                                PreyStatesTimes[3]=
{5000/TIME_STEP,10000/TIME_STEP,25000/TIME_STEP};
void Prey()
{

   enum PreyStates  state=PR_GRAZE;
   unsigned stateTimer=0;

  {
    WbDeviceTag camera;
    camera = wb_robot_get_device("camera");
    wb_camera_enable(camera, TIME_STEP);
  }
  setUpTouchSensor();

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


     stateTimer=stateTimer+1;

     if( avoidOtherWolves(PREY_AVOID_DIStANCE)==true)
     {
         state=PR_RUN_AWAY;
         stateTimer=0;
     }


     if(state==PR_RUN_AWAY )
     {

        float x = (rand()*1.0)/RAND_MAX;
        float y = (rand()*1.0)/RAND_MAX;
        y=(y*2.0-1.0);             // Giana to kanw apo -1 ews 1(aritera h
deksia)
        x=x*0.2+0.8;
        x=-x;
         printf("Run away %f,%f\n",x,y);
        //wb_differential_wheels_set_speed(x,y);
         doWheels(x,y);
         if(stateTimer>PreyStatesTimes[2])
         {
            state=PR_MOVE;
            stateTimer=0;
         }

     }
     else if(state==PR_GRAZE)
     {
         float x = (rand()*1.0)/RAND_MAX;
        float y = (rand()*1.0)/RAND_MAX;
```

```
        y=(y*2.0-1.0);            // Giana to kanw apo -1 ews 1(ariστera h
deksia)
        x=x*0.05;
       // printf("Random %f,%f\n",x,y);
        //wb_differential_wheels_set_speed(x,y);
         doWheels(x,y);
         if(stateTimer>PreyStatesTimes[0])
         {
            state=PR_MOVE;
            stateTimer=0;
         }

     }
     else if(state==PR_MOVE)
     {

         float x = (rand()*1.0)/RAND_MAX;
         float y = (rand()*1.0)/RAND_MAX;
         y=(y*2.0-1.0);
          x=x*0.2;
         // printf("Random %f,%f\n",x,y);
         //wb_differential_wheels_set_speed(x,y);
         doWheels(x,y);

         if(stateTimer>PreyStatesTimes[1])
         {
            state=PR_GRAZE;
            stateTimer=0;
         }
     }


  }


}
```

Supervisor

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <webots/robot.h>
#include <webots/supervisor.h>
#include <webots/emitter.h>

#define TIME_STEP 64

#define MAX_TIME (5*60)
#define ITERATIONS 1
```

```
void writeToFile(double *times,int size)
{
     FILE * fdata=fopen("C:/Users/george/Documents/results.csv","a");
     unsigned i;
  for( i=0;i<size-1;i++)
  {
    fprintf(fdata,"%f,",times[i]);

  }
  fprintf(fdata,"%f\n",times[size-1]);
  fflush(fdata);
  fclose(fdata);


}
int main() {



    WbNodeRef root, node;
    WbFieldRef children,field;

      WbFieldRef robot_translation_i,robot_translation_j;
      WbFieldRef robot_controllerArgs_i,robot_controllerArgs_j;


  const double *gravity;
    double location[3] = {0.5, 0.3, 0.5};
  int n,i,j;


  WbNodeRef *robotNodes=NULL;
  unsigned robotcount=0;
  unsigned preycount=0;
  double simulationTime = 0.0;   // elapsed simulation time

  unsigned iterationcount=0;

  wb_robot_init();







  root = wb_supervisor_node_get_root();
  children = wb_supervisor_node_get_field(root,"children");
  n = wb_supervisor_field_get_count(children);
  for(i=0; i<n; i++) {
    node = wb_supervisor_field_get_mf_node(children,i);

if(strcmp(wb_supervisor_node_get_type_name(node),"DifferentialWheels")==0)
          robotcount++;
  }
  printf("Found %d robots \n",robotcount);

  robotNodes = (WbNodeRef *) malloc(sizeof(WbNodeRef)*robotcount);
  //============================
```

```
  //Save robot nodes to array
  children = wb_supervisor_node_get_field(root,"children");
  n = wb_supervisor_field_get_count(children);
  j=0;
  for(i=0; i<n; i++) {
    node = wb_supervisor_field_get_mf_node(children,i);

if(strcmp(wb_supervisor_node_get_type_name(node),"DifferentialWheels")==0)
    {
      robotNodes[j]=node;
      robot_controllerArgs_j=
wb_supervisor_node_get_field(robotNodes[j],"controllerArgs");
      const                              char                              *
args_j=wb_supervisor_field_get_sf_string(robot_controllerArgs_j);
      if(strcmp(args_j,"Prey")==0)
            preycount++;


      j++;
    }

  }

  printf("Found %d preys \n",preycount);

  double *preyEatTimes = (double  *)malloc(preycount*sizeof(double));

  for(i=0;i<preycount;i++)
     preyEatTimes[i]=0;
  n=0;//N BELOW IS THE NUMBER OF EATEN PREYS


  while(wb_robot_step(TIME_STEP)!=-1) {
    /* At each step, the position of the robot is get and sent through the
emitter
     */
      simulationTime += (double)TIME_STEP / 1000.0;

      if(simulationTime>=MAX_TIME)
      {
        writeToFile(preyEatTimes,n+preycount);
         wb_supervisor_simulation_revert();
       simulationTime=0;
       iterationcount++;
      }

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

         robot_translation_i=
wb_supervisor_node_get_field(robotNodes[i],"translation");

robot_controllerArgs_i=wb_supervisor_node_get_field(robotNodes[i],"controll
erArgs");
         const                              char                              *
args_i=wb_supervisor_field_get_sf_string(robot_controllerArgs_i);
         const             double             *roboti_coords             =
wb_supervisor_field_get_sf_vec3f(robot_translation_i);
         //printf("i controller args:%s\n",args_i);

         if(strcmp(args_i,"Prey")!=0)
```

```
            continue;
        for(j=0;j<robotcount;j++)
        {

            robot_translation_j=
wb_supervisor_node_get_field(robotNodes[j],"translation");

robot_controllerArgs_j=wb_supervisor_node_get_field(robotNodes[j],"controll
erArgs");
            const char                                                    *
args_j=wb_supervisor_field_get_sf_string(robot_controllerArgs_j);
            const   double   *robotj_coords                               =
wb_supervisor_field_get_sf_vec3f(robot_translation_j);
            // printf("j controller args:%s\n",args_j);
            if(strcmp(args_j,"Prey")==0)
              continue;

            double dx,dy,dz;
            dx=roboti_coords[0]-robotj_coords[0];
            dy=roboti_coords[1]-robotj_coords[1];
            dz=roboti_coords[2]-robotj_coords[2];

            if(sqrt(dx*dx+dy*dy+dz*dz)<0.2)
            {

//printf("OMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOMOMNOM\n");
                double  newroboti_coords[3];    ///  roboti_coords  is  read
only, copy to new variable!
                newroboti_coords[0]=roboti_coords[0];
                newroboti_coords[1]=roboti_coords[1];
                newroboti_coords[2]=roboti_coords[2];
                newroboti_coords[1]=-100;

wb_supervisor_field_set_sf_vec3f(robot_translation_i,newroboti_coords);

                preycount--;
                preyEatTimes[n]=simulationTime;
                n++;
                if(preycount<=0)
                {
                   writeToFile(preyEatTimes,n);

                    wb_supervisor_simulation_revert();
                    simulationTime=0;
                    iterationcount++;
                    printf("SIMULATION REVERT %d\n",iterationcount);


                 }
                 else

                printf("Eat \n");



            }


        }
```

```
    }

  }
  wb_robot_cleanup();

  return 0;
}
```

Συνάρτηση main

```
int main(int argc,char **argv)
{
   srand(time(NULL));
   wb_robot_init();
   printf("|%s|\n",argv[1]);
   if(argc>1)
   {
     if(strcmp(argv[1],"Prey")==0) Prey();
    else if(strcmp(argv[1],"Alpha")==0) Alpha();
    else if(strcmp(argv[1],"Beta")==0) Beta();
    //else if(strcmp(argv[1],"BetaWolfSearch")==0) BetaWolfSearch();

   }


// cleanup the Webots API

  wb_robot_cleanup();
  return 0; //EXdoWheelsSearchIT_SUCCESS

}
```

ΠΑΡΑΡΤΗΜΑ Β΄

Πίνακας Β.1 : Χρόνοι οργανωμένου- ιεραρχικού κυνηγιού (alpha, beta).

κυνήγια/ αριθμός προσομοιώσεων	prey 1	prey 2	prey 3
1	28.928	35.52	38.72
2	72.32	78.656	80.832
3	91.776	99.84	100.8
4	30.72	37.056	40.32
5	38.464	45.632	48.448
6	68.48	74.496	77.12
7	34.368	40.896	43.968
8	22.016	31.552	35.904
9	47.424	53.76	56.192
10	27.328	34.304	37.248
11	60.16	66.24	69.376
12	55.68	66.624	70.464
13	36.288	42.56	45.568
14	31.424	44.352	48.384
15	29.376	36.224	39.552
16	28.544	35.52	38.72
17	30.336	37.44	40.512
18	33.216	43.84	48.128
19	77.056	82.24	86.464
20	32.896	39.552	42.688
21	36.544	43.968	47.04
22	44.16	52.032	55.232
23	39.936	46.656	49.472
24	42.304	47.552	51.392
25	29.632	36.288	39.488
26	48.704	56	59.136
27	26.688	38.08	41.792
28	106.56	112.576	114.88
29	26.944	45.12	49.024
30	37.248	47.744	51.904
31	31.104	37.184	40.192
32	32.768	46.848	51.264
33	34.368	40.96	43.968
34	36.736	48.64	52.864

35	49.984	229.376	231.68
36	48.768	55.68	58.752
37	27.904	57.152	60.032
38	33.152	40.32	43.584
39	86.08	91.264	94.592
40	23.36	30.784	33.664
41	33.024	40.256	43.52
42	29.184	35.776	38.912
43	31.936	41.728	44.928
44	45.824	52.16	55.296
45	83.264	88.704	94.528
46	139.264	145.664	148.736
47	29.056	35.904	39.04
48	36.608	43.648	46.784
49	36.608	69.76	71.488
50	49.472	55.68	58.56
51	29.056	35.392	38.4
52	27.904	39.232	42.624
53	34.432	41.728	44.992
54	29.696	77.184	80.512
55	31.04	43.008	46.912
56	36.224	43.008	46.08
57	69.248	75.456	78.208
58	55.04	63.232	66.304
59	42.752	52.672	56.192
60	40.896	46.848	49.408
61	40.192	54.08	57.216
62	31.808	41.088	41.856
63	89.984	239.872	0
64	49.6	55.872	58.752
65	82.816	100.224	104.832
66	33.664	40.064	43.776
67	25.792	34.88	112.32
68	30.016	36.928	40.128
69	30.912	37.824	41.024
70	41.28	50.112	257.472
71	27.2	33.28	36.352
72	35.52	48.832	53.312
73	51.328	57.792	60.608
74	30.464	37.12	40.192
75	33.408	40.704	43.84
76	27.456	33.6	36.928
77	27.968	34.432	37.44

78	116.544	127.296	237.504
79	55.744	61.44	65.984
80	47.424	54.016	57.088
81	37.44	167.936	170.368
82	35.392	140.288	143.36
83	55.552	61.76	64.448
84	48.832	55.232	58.432
85	38.016	45.76	48.32
86	38.016	44.8	47.68
87	28.864	34.816	37.824
88	115.84	124.416	255.68
89	38.912	46.016	48.832
90	48.832	57.984	61.12
91	26.624	33.536	36.864
92	60.352	62.08	65.728
93	48.32	55.232	58.496
94	30.336	36.864	39.936
95	30.208	37.056	40.32
96	52.928	62.464	67.264
97	0	0	0
98	25.472	32.704	35.52
99	41.216	47.424	50.112
100	31.488	38.4	41.6
101	29.952	37.44	40.704
102	17.728	25.088	28.288
103	58.432	64.384	67.264
104	31.488	39.232	42.944
105	38.912	45.12	47.744
106	42.176	50.944	54.016
107	32.192	38.72	41.792
108	48.64	57.6	58.88
109	124.672	133.632	0
110	80.896	88.576	92.096
111	41.024	54.784	59.072
112	40	55.68	60.032
113	37.76	44.16	47.296
114	41.472	48.448	51.264
115	28.48	34.368	37.952
116	46.656	86.016	89.152
117	42.112	48.768	51.712
118	33.728	41.216	44.352
119	26.816	40.064	44.864
120	179.328	187.072	237.184

121	38.336	61.312	64.384
122	48.256	54.016	56.896
123	17.664	28.224	32.512
124	24.384	31.744	34.56
125	49.664	254.336	258.816
126	26.688	37.888	41.984
127	37.376	43.904	46.592
128	29.568	36.288	39.36
129	62.72	67.968	72.512
130	24.832	32.064	34.944
131	31.36	37.76	40.832
132	21.568	28.8	31.808
133	42.112	51.008	101.888
134	127.296	136.192	0
135	35.84	64.448	67.776
136	18.048	27.584	32.192
137	59.648	68.032	69.568
138	28.416	35.328	39.616
139	27.968	34.816	37.888
140	30.72	37.248	39.68
141	25.28	32.384	35.264
142	31.808	37.824	40.896
143	46.464	52.8	55.808
144	28.864	35.328	38.4
145	38.272	48.512	49.472
146	36.096	45.44	49.664
147	34.816	39.552	40.128
148	23.36	44.864	48.32
149	41.152	47.04	49.6
150	136.896	142.208	145.344
151	24.896	30.848	34.24
152	37.504	43.904	46.784
153	54.016	131.712	135.168
154	25.92	33.088	36.032
155	40.96	47.296	49.984
156	19.392	30.336	30.592
157	43.264	49.856	53.12
158	28.928	35.584	38.848
159	71.04	77.568	80.384
160	38.592	48.256	52.736
161	106.176	114.432	261.312
162	23.104	30.464	33.28
163	62.784	75.008	78.912

164	41.344	55.552	59.136
165	27.904	43.904	47.168
166	33.536	40.832	43.904
167	34.496	41.856	45.12
168	27.712	34.88	37.952
169	43.584	52.672	153.984
170	52.544	64.064	68.352
171	23.808	32.832	37.376
172	21.632	89.216	180.544
173	29.824	36.736	39.936
174	48.768	55.296	58.432
175	46.656	82.24	85.824
176	35.456	46.272	47.424
177	33.472	39.872	43.008
178	34.432	41.6	44.672
179	42.368	50.88	54.336
180	45.824	57.152	61.44
181	23.04	32.192	37.376
182	31.36	40.704	118.656
183	158.976	164.224	0
184	42.048	48.64	51.392
185	31.552	38.912	42.176
186	85.376	92.864	97.344
187	46.272	52.16	54.912
188	22.4	29.824	32.768
189	31.552	38.848	41.92
190	38.08	45.376	48.192
191	68.416	74.24	76.992
192	39.808	44.8	181.888
193	47.296	53.952	57.152
194	29.312	36.032	39.232
195	30.976	88.64	91.584
196	26.816	33.344	36.224
197	52.992	62.976	64.768
198	30.848	48.512	52.16
199	49.152	58.432	0

Πίνακας Β.2 : Χρόνοι ανοργάνωτου κυνηγιού

κυνήγια/ αριθμός προσομοιώσεων	prey 1	prey 2	prey 3
1	67.52	74.56	77.824
2	116.672	123.968	193.472

3	0	0	0
4	88.768	93.568	95.872
5	46.656	55.552	106.048
6	54.912	61.44	62.464
7	145.408	154.688	156.864
8	95.744	99.968	102.464
9	231.616	235.328	237.824
10	21.312	30.336	180.224
11	99.584	108.16	109.312
12	42.176	46.016	48.896
13	26.112	33.344	37.504
14	49.6	58.112	59.52
15	74.432	78.848	83.776
16	93.696	99.072	102.912
17	30.784	37.76	41.152
18	68.864	75.904	77.248
19	79.424	88.576	93.056
20	23.488	30.336	31.488
21	62.4	66.304	73.28
22	168.128	171.264	174.144
23	209.408	212.544	215.168
24	197.44	203.776	0
25	120.896	127.168	0
26	64	71.68	147.84
27	273.152	276.096	278.848
28	36.224	42.368	43.328
29	17.28	20.928	24.896
30	196.928	200.448	202.88
31	68.608	74.688	75.584
32	23.936	31.104	32.64
33	0	0	0
34	198.72	202.112	204.928
35	63.616	69.952	70.272
36	80.192	87.104	87.936
37	268.992	275.264	0
38	87.168	92.736	93.696
39	250.432	254.336	257.024
40	166.976	169.92	172.608
41	75.392	82.816	86.4
42	151.488	154.496	157.248
43	33.6	41.92	45.888
44	115.776	122.112	223.168
45	231.936	236.352	238.72

46	60.544	65.536	69.248
47	194.944	198.08	200.64
48	20.096	30.4	34.048
49	0	0	0
50	223.616	227.392	229.888
51	60.928	67.008	139.136
52	242.176	248.448	0
53	0	0	0
54	38.784	44.928	45.952
55	194.368	200.384	203.52
56	63.872	68.544	71.936
57	93.568	98.944	100.096
58	70.4	75.712	79.104
59	22.656	32.064	33.024
60	142.016	146.88	149.632
61	80.704	84.16	87.04
62	42.112	49.216	167.488
63	213.056	219.456	219.968
64	66.752	71.616	75.072
65	185.472	189.12	191.808
66	79.744	0	0
67	119.616	124.544	126.016
68	79.744	85.44	86.4
69	32.704	46.208	51.968
70	75.904	81.664	219.52
71	60.544	66.176	66.944
72	60.672	64.704	68.352
73	41.024	46.208	48.768
74	157.888	161.408	164.16
75	36.864	42.624	43.712
76	103.104	110.336	119.168
77	100.928	109.568	111.104
78	98.816	105.472	245.056
79	144.96	152.192	226.56
80	112.256	124.864	129.024
81	45.632	50.432	54.08
82	16.448	127.936	131.136
83	44.928	55.232	59.008
84	37.312	55.232	58.752
85	86.464	90.048	96.384
86	72.064	77.504	81.6
87	87.104	93.376	93.568
88	202.304	205.632	208.576

89	173.824	177.6	180.288
90	172.992	179.52	0
91	61.824	66.688	69.056
92	144.448	150.144	151.36
93	21.888	31.68	47.424
94	86.912	92.544	94.272
95	46.336	53.504	54.848
96	103.68	109.76	290.368
97	20.416	26.176	27.904
98	133.376	208.768	0
99	211.712	217.792	218.112
100	180.224	183.616	185.856
101	85.312	91.328	91.84
102	30.08	37.376	41.344
103	87.04	96.384	105.408
104	102.08	107.648	110.912
105	59.2	64.128	65.664
106	89.92	95.68	96.448
107	268.736	272.192	274.816
108	150.848	154.304	156.736
109	32.128	42.368	139.776
110	79.616	88.32	91.008
111	170.496	178.56	180.16
112	32.128	40	43.392
113	60.544	64.384	66.304
114	22.848	28.992	34.24
115	33.472	38.72	40.32
116	79.872	86.336	86.464
117	27.008	43.136	46.4
118	29.376	39.424	43.584
119	84.16	97.408	100.8
120	259.008	264.896	273.92
121	106.304	113.408	115.008
122	115.712	119.36	122.048
123	95.872	101.504	102.848
124	34.88	40.384	41.984
125	89.728	96.384	150.592
126	48.96	54.208	57.6
127	39.296	45.632	46.08
128	194.88	200.448	201.92
129	38.272	46.848	50.368
130	32.96	44.032	44.096
131	93.76	102.592	286.72

132	152.896	158.336	160.448
133	162.752	169.024	172.288
134	73.408	80.064	289.536
135	0	0	0
136	17.344	24.704	25.6
137	45.568	0	0
138	109.632	117.248	246.4
139	242.56	256.256	258.752
140	280.768	284.224	287.04
141	108.864	115.648	123.648
142	58.56	65.408	69.056
143	28.352	38.272	55.936
144	88.448	100.672	247.68
145	56	62.784	70.016
146	204.096	209.792	0
147	96.576	102.144	104.64
148	0	0	0
149	33.408	41.792	48.96
150	88.064	94.144	94.464
151	77.696	83.008	86.592
152	63.104	70.656	71.616
153	0	0	0
154	36.288	68.096	118.912
155	29.44	36.8	40.64
156	174.72	178.944	181.44
157	39.808	66.688	71.104
158	0	0	0
159	59.136	65.024	68.672
160	94.656	102.848	103.36
161	0	0	0
162	63.936	72.128	76.096
163	138.752	143.872	145.472
164	72.064	82.944	86.784
165	29.696	39.36	49.536
166	0	0	0
167	29.76	36.032	36.032
168	73.792	79.296	81.216
169	178.624	184.96	284.672
170	81.28	86.464	90.048
171	72.512	78.208	81.6
172	63.168	67.84	71.68
173	92.416	95.424	98.176
174	82.56	88.448	89.6

175	65.664	71.232	74.816
176	162.112	165.632	168.32
177	201.28	207.808	208.064
178	25.728	35.072	38.976
179	70.336	76.352	77.12
180	64.64	70.272	73.856
181	67.072	74.368	124.992
182	227.2	233.28	236.864
183	93.12	98.112	99.712
184	57.536	61.632	63.936
185	28.928	38.72	42.752
186	177.216	183.744	267.776
187	111.552	116.992	118.272
188	134.08	139.776	140.48
189	34.56	41.664	45.504
190	50.112	56.32	59.968
191	27.84	36.16	42.24
192	91.136	98.048	182.08
193	245.632	249.152	272.832
194	38.08	44.608	44.928
195	104.128	110.208	283.328
196	89.408	95.168	98.944
197	138.688	142.272	144.96
198	27.264	37.952	140.8
199	97.28	103.104	104.128