Διεθνές Πανεπιστήμιο της Ελλάδος

Μεταπτυχιακό Πρόγραμμα Σπουδών στη Ρομποτική

Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών καιΤηλεπικοινωνιών

Διπλωματική Εργασία

Προσομοιωτής ρομποτικού βραχίονα με χρήση της μηχανής παιχνιδιών Unity: Μελέτη και υλοποίηση

Εργασία που υποβλήθηκε στο Πρόγραμμα Μεταπτυχιακών Σπουδών στη Ρομποτική,

του Διεθνούς Πανεπιστημίουτης Ελλάδος, για τη μερική εκπλήρωση υποχρεώσεων

για το Δίπλωμα Ειδίκευσης στη Ρομποτική.

Εκπονήτρια : Ειρήνη Θεοδώρα Κούκα, Α.Μ :45

Επιβλέπων Καθηγητής : Αθανάσιος Νικολαΐδης

Σέρρες, 2021

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν με την πολύτιμη βοήθεια τους στην ολοκλήρωση αυτής της διπλωματικής εργασίας.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας, κύριο Αθανάσιο Νικολαΐδη, καθηγητή του Π.Μ.Σ στη Ρομποτική του Διεθνούς Πανεπιστημίου Ελλάδος, για την εμπιστοσύνη που μου έδειξε, τη βοήθεια που μου προσέφερε καθ' όλη τη διάρκεια εκπόνησής της, την καθοδήγηση και τις πολύτιμες συμβουλές.

Επίσης, ευχαριστώ όλους τους διδάσκοντες του Μεταπτυχιακού Προγράμματος για τις γνώσεις που μου προσέφεραν και τη συμβολή τους στην επιστημονική και τεχνολογική μου συγκρότηση κατά τη διάρκεια των σπουδών μου.

Ακόμη, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους συμφοιτητές μου, για τις στιγμές που μοιραστήκαμε, την υποστήριξη και την ψυχική δύναμη που μου προσέφεραν αυτά τα χρόνια.

Τέλος, ένα μεγάλο ευχαριστώ στους γονείς μου, για την οικονομική τους υποστήριξη αλλά και τη δύναμη για να συνεχίζω και να εκπληρώνω τους στόχους μου.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως θέμα τη δημιουργία και την προσομοίωση ενός μηχανικού ρομποτικού βραχίονα τεσσάρων (4) βαθμών ελευθερίας στο περιβάλλον της μηχανής παιχνιδιών Unity. Στο εισαγωγικό μέρος, παρουσιάζεται το αντικείμενο της επιστήμης της ρομποτικής και η συμβολή της στην ανάπτυξη και αυτοματοποίηση του βιομηχανικού κόσμου, ενώ παράλληλα, εξηγείται το πρόβλημα και ο σκοπός της εργασίας. Έπειτα, αναφέρονται οι τελευταίες εξελίξεις (StateoftheArt) στον τομέα της ρομποτικής και των ρομποτικών βραχιόνων. Πραγματοποιείται, επίσης, μια εισαγωγή στη μηχανή παιχνιδιώνUnity.

Στη συνέχεια, περιγράφονται αναλυτικά τα είδη και οι λειτουργίες των ρομποτικών βραχιόνων καταλήγοντας, έτσι, στη μεθοδολογία και την επεξήγηση του κινηματικού προβλήματος. Το τελευταίο κεφάλαιο του θεωρητικού μέρους της εργασίας αφορά την ανάλυση και επίλυση του κινηματικού προβλήματος (ευθύ και αντίστροφο) για τον μηχανικό βραχίονα.

Στο πρακτικό μέρος της εργασίας παρουσιάζεται η σχεδίαση του ρομποτικού βραχίονα στο περιβάλλον της Unityκαι ο κώδικας προσομοίωσης σε γλώσσα C#.

Abstract

The present dissertation aims at creating and simulating a mechanical robotic arm of four (4) degrees of freedom. In the introductory part, the field of robotics and its contribution to the development and automation of the industrial world is presented while, at the same time, the problem and the purpose of the work are explained. Next, the State of the Art in the field of robotics and robotic arms is reported, as well as an introduction to the Unity programming language.

Furthermore, the types and functions of robotic arms are described in detail, thus concluding with the methodology and explanation of the kinematic problem. The last chapter of the theoretical part of the work concerns the analysis and solution of the kinematic problem (forward and inverse) for the mechanical arm.

The practical part of the work presents the design of the robotic arm in Unity and the simulation code in C#.

Περιεχόμενα

1. Εισαγωγή

Εικόνα 1: Ρομποτικός βραχίονας

1.1 Τι είναι ρομποτική

Ρομποτική είναι ο σύγχρονος επιστημονικός κλάδος ο οποίος ασχολείται με τη σύλληψη, τη σχεδίαση, την κατασκευή, τη θεωρία και τις εφαρμογές των ρομπότ. Σύμφωνα με το Ινστιτούτο Ρομποτικήςτης Αμερικής, τα ρομπότ είναι αναπρογραμματιζόμενοι και πολυλειτουργικοί χωρικοί μηχανισμοί, σχεδιασμένοι να μετακινούν υλικά, αντικείμενα, τεμάχια, εργαλεία ή εξειδικευμένες συσκευές, με κατάλληλες προγραμματισμένες κινήσεις που στοχεύουν στη βελτίωση της απόδοσης μιας σειράς εργασιών. Η χρήση τους αποσκοπεί στην αντικατάσταση του ανθρώπου στην εκτέλεση εργασιών που αφορά τόσο το φυσικό επίπεδο, όσο και το επίπεδο λήψης απόφασης.

Το κύριο πλεονέκτημα των ρομπότ είναι η ευελιξία τους. Μπορούν να προσαρμοστούν σε διάφορα προϊόντα στην ίδια γραμμή παραγωγής, όπως απαιτούν οι αλλαγές της αγοράς, αλλά και να επαναπρογραμματιστούν, έτσι ώστε να είναι κατάλληλα για μικρές ή μεγάλες μεταβολές στη γραμμή παραγωγής.

Η εξέλιξη των ρομπότ έχει περάσει από πολλά στάδια. Τα ρομπότ της πρώτης γενιάς δεν είχαν την ικανότητα υπολογισμού και αίσθησης, σε αντίθεση με τα ρομπότ της δεύτερης γενιάς, τα οποία διαθέτουν περιορισμένη υπολογιστική ισχύ, γλώσσες προγραμματισμού υψηλού επιπέδου και

αισθητήρες ανατροφοδότησης. Τα ρομπότ της τρίτης γενιάς διαθέτουν νοημοσύνη, δηλαδή είναι ικανά να παίρνουν αποφάσεις κατά τη διάρκεια εκτέλεσης της εργασίας τους. Τις ικανότητες αυτές τις αποκτούν μέσω τεχνικών της τεχνητής νοημοσύνης σε συνδυασμό με εξελιγμένες μορφές αισθητήρων αφής, δύναμης, απόστασης, όρασης. Τα βιομηχανικά ρομπότ είναι εξελιγμένα συστήματα αυτοματισμού, που χρησιμοποιούν ηλεκτρονικό υπολογιστή προκειμένου να επιτευχθεί ο έλεγχός τους. Σήμερα οι υπολογιστές αποτελούν ένα αναπόσπαστο τμήμα του βιομηχανικού αυτοματισμού. Κατευθύνουν γραμμές παραγωγής και ελέγχουν συστήματα κατασκευής (όπως εργαλειομηχανές, συγκολλητές, κοπτικές διατάξεις LASER κ.α.) Τα νέα ρομπότ εκτελούν πληθώρα εργασιών στα βιομηχανικά συστήματα και γενικά συμμετέχουν στον πλήρη αυτοματισμό των εργοστασίων.

Η έρευνα στην περιοχή της ρομποτικής εκτείνεται σε τρεις κυρίως κατευθύνσεις. Η πρώτη αφορά την εφαρμογή ή την ανάπτυξη τεχνικών ελέγχου για τη βελτίωση της απόδοσης των ρομπότ. Η δεύτερη αφορά την εφαρμογή και την ανάπτυξη λογισμικού για τη διαχείριση των εργασιών των ρομπότ. Τέλος,η τρίτη σχετίζεται με τη σχεδίαση υλικού υπολογιστών για την εκτέλεση του λογισμικού και την καλύτερη επικοινωνία με τους αισθητήρες και τους ενεργοποιητές των ρομπότ. Είναι γεγονός ότι η ρομποτική ωφελείται από τις εξελίξεις σε αρκετούς κλάδους, όπως είναι η ηλεκτρολογία, η μηχανολογία και τα μαθηματικά.

Τα ρομπότ αποτελούνται από δύο υποσυστήματα προκειμένου να εκτελέσουν την εργασία τους. Αυτά είναι το μηχανολογικό και το υποσύστημα της αίσθησης. Το μηχανολογικό υποσύστημα, που αποτελείται από τη βάση του ρομπότ, τις αρθρώσεις, τους συνδέσμους και το εργαλείο τελικής δράσης, επιτρέπει στο ρομπότ να εκτελέσει τη λειτουργία του σε συνδυασμό με το υποσύστημα της αίσθησης. Το υποσύστημα της αίσθησης συλλέγει πληροφορίες από τους αισθητήρες ή άλλα όργανα μέτρησης, ελέγχει την κατάσταση του ρομπότ, δέχεται και επεξεργάζεται τις εντολές που θα του δώσει ο χρήστης ή οι αισθητήρες και τα όργανα μέτρησης, τις επεξεργάζεται και τις μετατρέπει σε ισχύ για τους κινητήρες, οι οποίοι θα εκτελέσουν στο τέλος την εντολή.

1.2 Σκοπός της Εργασίας

Η παρούσα διπλωματική εργασία έχει ως σκοπό τη μοντελοποίηση και την ανάλυση ενός μηχανικού ρομποτικού βραχίονα τεσσάρων (4) βαθμών ελευθερίας με άκρο – αρπάγη και την προσομοίωσή τους στο γραφικό περιβάλλον της Unity. Η εργασία βασίζεται σε δύο μέρη, το θεωρητικό και το πρακτικό. Στο θεωρητικό μέρος γίνεται η περιγραφή του προβλήματος, η εξήγηση της μεθοδολογίας που ακολουθείται και η θεωρητική ανάλυση του ορθού και αντίστροφου κινηματικού προβλήματος. Στο πρακτικό μέρος, ο ρομποτικός βραχίονας προσομοιώνεται στημηχανή παιχνιδιώνUnityκαι εξηγείται η αλγοριθμική προσέγγιση και ο κώδικας που αναπτύχθηκε.

Αναλυτικότερα, στο δεύτερο κεφάλαιο της εργασίας πραγματοποιείται μια ιστορική αναδρομή της επιστήμης της ρομποτικής, ενώ παράλληλα αναφέρονται γεγονότα σταθμοί που συντέλεσαν στην ανάπτυξη της επιστήμης αυτής. Στη συνέχεια, αναφέρονται εφαρμογές στις οποίες βρίσκει ευρεία χρήση η ρομποτική επιστήμη καταδεικνύοντας τη σημαντικότητά της στο σύγχρονο βιομηχανικό κόσμο. Τέλος, παρουσιάζεται η μηχανή παιχνιδιώνUnityμαζί με διάφορες βασικές έννοιες και λειτουργίες της.

Στο τρίτο κεφάλαιο αναλύεται η δομή και η λειτουργία των ρομποτικών βραχιόνων που αποτελούν το κύριο αντικείμενο ενασχόλησης της παρούσας εργασίας. Παρουσιάζονται τα είδη των αρθρώσεων και των ρομποτικών βραχιόνων και επεξηγούνται βασικές έννοιες όπως οι βαθμοί ελευθερίας και κινητικότητας, ο χώρος εργασίας και το άκρο – αρπάγη.

Στο τέταρτο κεφάλαιο πραγματοποιείται μια αναλυτική και θεωρητική επεξήγηση του ορθού και αντίστροφου κινηματικού προβλήματος του ρομποτικού βραχίονα με ευρέως διαδεδομένες τεχνικές ανάλυσης.

Στο πέμπτο κεφάλαιο, που αφορά το θεωρητικό μέρος της εργασίας, πραγματοποιείται επίλυση του ορθού και κινηματικού προβλήματος για βραχίονα τεσσάρων βαθμών ελευθερίας. Μάλιστα, για το ευθύ κινηματικό πρόβλημα, παρουσιάζονται 2 διαφορετικοί τύποι επίλυσής του, οι οποίοι στο τέλος συγκρίνονται και αποδεικνύεται πως στο τέλος καταλήγουμε στις ίδιες κινηματικές εξισώσεις.

Στα δύο τελευταία κεφάλαια παρουσιάζονται η προσομοίωση και ο κώδικας που αναπτύχθηκε για τον ρομποτικό βραχίονα τεσσάρων βαθμών ελευθερίας στη Unity.

2. Οι τελευταίες εξελίξεις (stateoftheart)

2.1 Ιστορική αναδρομή και εξελικτική πορεία

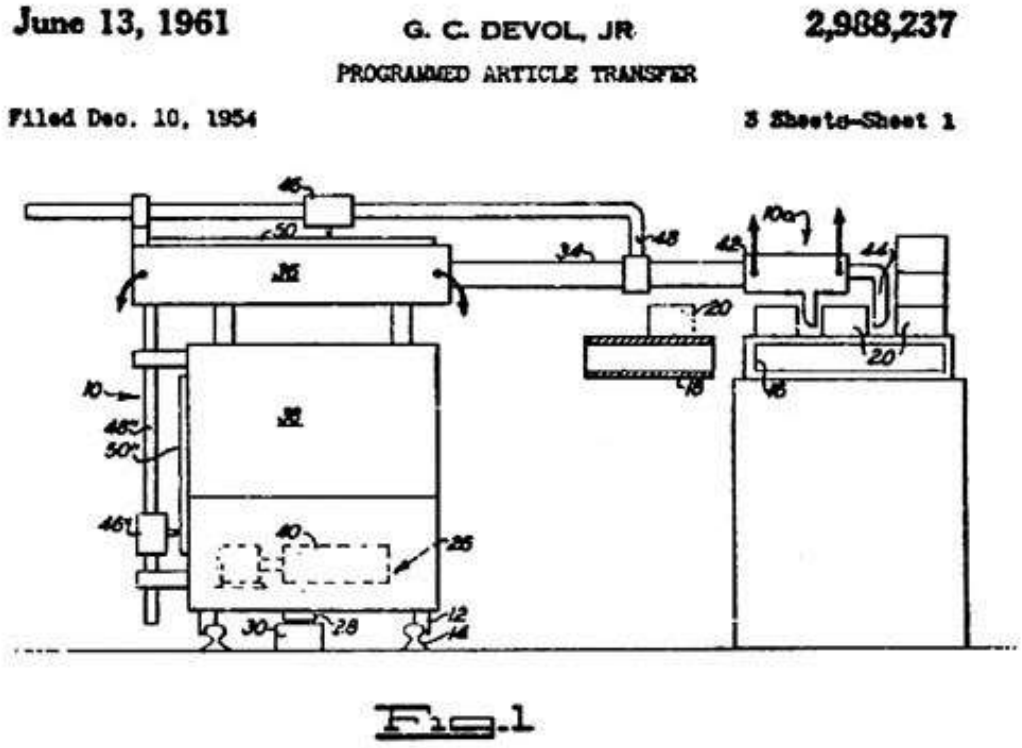
Ο όρος ρομπότ θεωρείται πως έκανε πρώτη φορά την εμφάνισή του στην αρχαία Ελλάδα. Συγκεκριμένα, ο Όμηρος, στην Ιλιάδα, αναφέρει πως ο Ήφαιστος είχε κατασκευάσει ένα μηχανισμό που του επέτρεπε να ανεβαίνει στο εργαστήριό του στον Όλυμπο. Βέβαια, ο αρχαιότερος γνωστός αυτοματισμός που κατασκευάστηκε, είναι ο Μηχανισμός των Αντικυθήρων, ο οποίος χρονολογείται μεταξύ 150-100 π.Χ. και λειτουργούσε ως αναλογικός μηχανικός υπολογιστής και όργανο αστρονομικών παρατηρήσεων.

Εικόνα 2: Ο μηχανισμός των Αντικυθήρων [1]

Αν και υπάρχουν εκατοντάδες ακόμα αναφορές για αυτοματισμούς και ρομποτικές κατασκευές, τόσο τα ανθρωπόμορφα ρομπότ, όσο και οι ρομποτικοί βραχίονες άργησαν πολύ να δημιουργηθούν. Συγκεκριμένα, το πρώτο ανθρωποειδές εφευρέθηκε το 1928 από τον W. H. Richards ενώ, μετά από 3 δεκαετίες περίπου, το 1954, ο George Charles Devol Jr. κατασκεύασε τον πρώτο ρομποτικό βραχίονα με το όνομα “Unimate”. Μάλιστα, το 1960 αγοράστηκε από την General Motors και το 1961 εγκαταστάθηκε στα εργοστάσια δημιουργώντας, έτσι, τα θεμέλια της σύγχρονης ρομποτικής βιομηχανίας.

Η εφεύρεση αποτελεί ορόσημο για την επιστήμη της ρομποτικής και, κυρίως, στον τρόπο λειτουργίας των εργοστασίων. Ο τρόπος λειτουργίας των ρομπότ αυτών ήταν με υδραυλικούς ενεργοποιητές.Τα αποτελέσματα που προέκυπταν κατά τη διάρκεια της εκμάθησης αποθηκεύονταν ώστε στη συνέχεια να χρησιμοποιούνται ξανά για την επανάληψη της λειτουργίας τους. Προς τα τέλη της δεκαετίας του ‘60 έκανε την εμφάνισή του ο πιο γνωστός, ίσως, στις μέρες μας ρομποτικός βραχίονας, Stanford Arm, δημιουργός του οποίου ήταν ο Victor Scheinman.

Πρόκειται για έναν ρομποτικό βραχίονα με 6 βαθμούς ελευθερίας και αποτέλεσε έναν από τους πρώτους που ελέγχονταν αποκλειστικά μέσα από υπολογιστές.


June 13, 1961 G. C. DEVOL, JR 2,988,237

PROGRAMMED ARTICLE TRANSFER

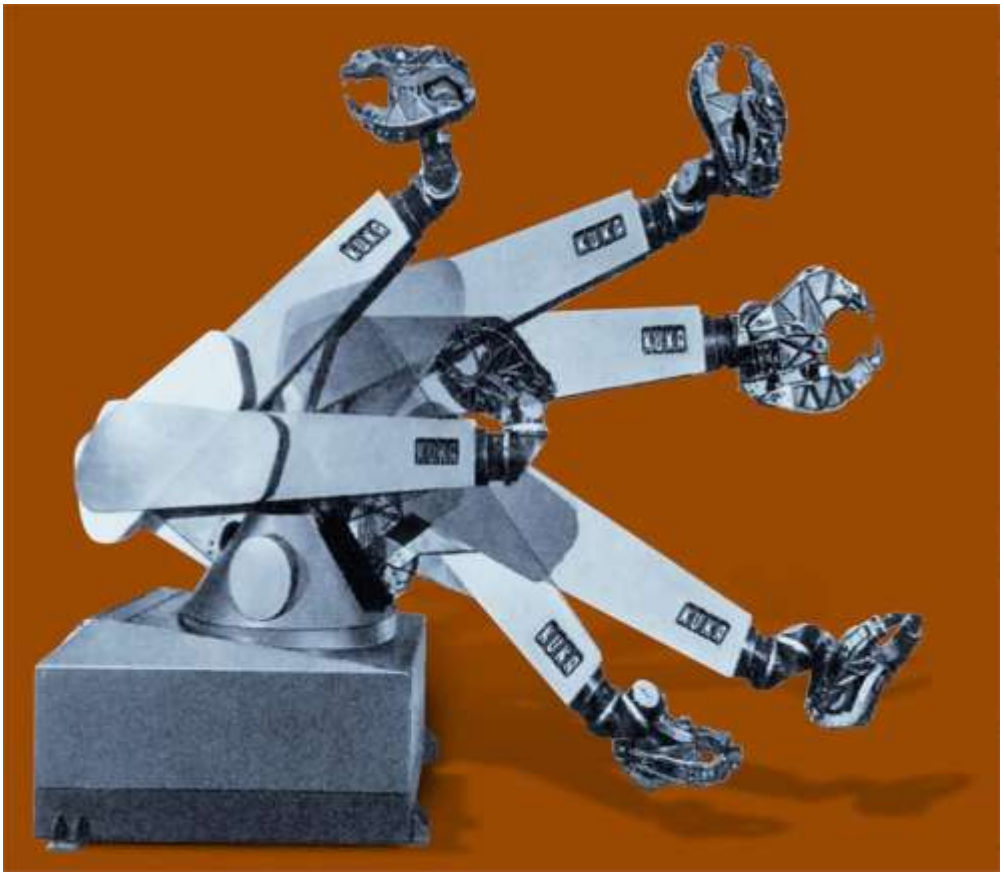
Filed Dec. 10, 1954 3 Sheets-Sheet 1

Fig. 1

Εικόνα 3: UnimatePatent [2]

Εικόνα 4: Stanfordmanipulator [3]

Η μεγάλη άνθιση των βιομηχανικών ρομπότ ήρθε το 1973, όταν ξεκίνησε η ευρύτερη εμπορική τους διάθεση, αλλά και η χρονιά που κατασκευάστηκε από την εταιρία KUKA Robotics το FAMULUS που ήταν ένα από τα πρώτα αρθρωτά ρομπότ με6 ηλεκτρομηχανικούς άξονες.



Εικόνα 5: KUKA FAMULUS [4]

2.2 Σύγχρονες εφαρμογές

Στη σύγχρονη εποχή, οι ανάγκες σε ποικίλες εφαρμογές για ακρίβεια, ασφάλεια, δύναμη, ευελιξία μεταξύ των εφαρμογών και η ταχύτερη λειτουργία είναι απαραίτητες, και στις περισσότερες περιπτώσεις η ικανότητα του ανθρώπου δεν επαρκεί. Για τον λόγο αυτόν, η ρομποτική όχι μόνο αναδύεται αλλά και κυριαρχεί. Έτσι, τα ρομπότ αντικαθιστούν σε αρκετές εφαρμογές τον άνθρωπο, καθώς έχουν τη δυνατότητα, εκτός από τα παραπάνω, να αυξήσουν μια γραμμή παραγωγής, να βελτιώσουν την ποιότητα των παραγόμενων προϊόντων, να μειώσουν το κόστος αλλά και να τον αντικαταστήσουν σε αρκετές εφαρμογές που υπάρχει αυξημένος κίνδυνος τραυματισμού του. Γι'αυτό και χρησιμοποιούνται σε μεγάλο βαθμό πετυχαίνοντας συνεχή ανάπτυξη μέσω τηςέρευνας.

Κύριοι τομείς στους οποίους χρησιμοποιούνται τα ρομπότ είναι η βιομηχανία και ο στρατός. Τα τελευταία χρόνια υπάρχει μια ραγδαία ανάπτυξη των ρομπότ και στην ιατρική. Στη βιομηχανία χρησιμοποιούνται σε εφαρμογές όπως η μεταφορά υλικών, η ταξινόμηση αποθηκών, η συναρμολόγηση συσκευών και μηχανισμών, η συγκόλληση μεταλλικών κατασκευών και ηλεκτρικών στοιχείων, οι εργασίες σε επικίνδυνους και ανθυγιεινούς χώρους κ.α.

Σε στρατιωτικές εφαρμογές χρησιμοποιούνται στην αφόπλιση εκρηκτικών μηχανισμών, στην πλοήγηση, στη μεταφορά αγαθών ή στα οπλικά συστήματα. Τις τελευταίες δεκαετίες, η ρομποτική αναπτύσσεται στην ιατρική σε χειρουργικές επεμβάσεις ή σε προσομοίωση αυτών, σε διαγνώσεις ασθενειών, αλλά και σε μηχανήματα αποκατάστασης ανθρώπινων μελών. Τα μηχανήματα αποκατάστασης, όπως οι ρομποτικοί βραχίονες, μπορούν να χρησιμοποιηθούν πλέον από ανθρώπους μιας και οι κινήσεις γίνονται ολοένα πιο ρεαλιστικές και ανθρώπινες.

2.3 Είδη και κατηγορίες ρομπότ

Στη σημερινή εποχή, τα ρομπότ χρησιμοποιούνται σεένα μεγάλο εύρος απόεφαρμογές και τομείς ανάπτυξης με αποτέλεσμα να υπάρχουν διάφορα είδη ρομπότ τα οποία προσαρμόζονται ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Τα κυριότερα είδη ρομπότ είναι τα εξής:

- Ρομπότ σταθερής βάσης: Το είδος των ρομπότ αυτών αποτελείται από μια βάση η οποία είναι σταθερή στο χώρο εργασίας των ρομπότ και πάνω σε αυτήν είναι τοποθετημένοι σύνδεσμοι, αρθρώσεις και το εργαλείο τελικής δράσης. Σε αυτήν την κατηγορία ανήκουν οι ρομποτικοί βραχίονες, όπως αυτός που θα παρουσιασθεί και θα αναλυθεί στην παρούσα εργασία.

Εικόνα 6: Ρομπότ με μηχανικά πόδια για τη μεταφορά υλικών

- Κινούμενα ρομπότ: Την κατηγορία αυτή αποτελούν ρομπότ τα οποία μπορούν να μετακινηθούν στο χώρο είτε με τροχούς, είτε με έλικες, είτε με προπέλες, είτε με μηχανικά πόδια κ.α. Αυτό έχει ως αποτέλεσμα να υπάρχουν υποκατηγορίες κινούμενων ρομπότ ανάλογα με τον τρόπο κίνησής τους.
- Έντροχα ρομπότ: Η κίνησή τους γίνεται με τη χρήση τροχών.
- Βαδίζοντα ρομπότ: Για την κίνησή τους χρησιμοποιούνται μηχανικά πόδια διαφόρων τύπων.
- Εναέρια ρομπότ: Σε αυτήν την κατηγορία συναντάμε μη επανδρωμένα αεροπλάνα ή άλλα ιπτάμενα ρομπότ όπως drone.
- Υποβρύχια ρομπότ: Τα ρομπότ αυτά είναι κατάλληλα για υποθαλάσσιες λειτουργίες και κινούνται με τη χρήση προπέλας.

Όλα τα είδη ρομπότ μπορούν να εκτελέσουν μια λειτουργία ή μια κίνηση, είτε μέσω ενός χρήστη, είτε αυτοματοποιημένα με τη χρήση αισθητήρων και πομπών.

Εικόνα 7: Υποβρύχιο ρομπότ

2.4 Η μηχανή παιχνιδιώνUnity

Η Unity υποστηρίζει scripting σε C# και υπάρχουν δύο βασικοί τρόποι δημιουργίας των σεναρίων C# στη Unity: αντικειμενοστραφής σχεδιασμός, που είναι η παραδοσιακή και ευρύτερα χρησιμοποιούμενη προσέγγιση, και σχεδιασμός προσανατολισμένος στα δεδομένα, ο οποίος είναι δυνατός με τη Unity, για συγκεκριμένες περιπτώσεις, μέσω της νέας πολυνηματικήςυψηλής απόδοσης Data-Oriented Technology Stack (DOTS).

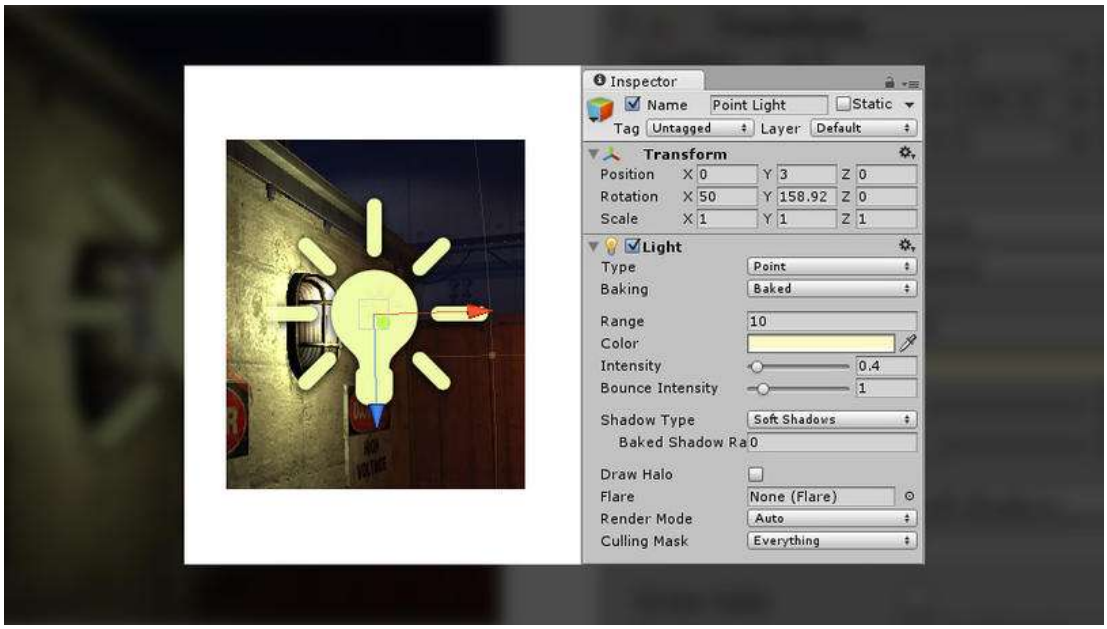
Η Unity υποστηρίζει τη C#, μια βιομηχανική γλώσσα με ορισμένες ομοιότητες με την Java ή τη C++. Σε σύγκριση με τη C++, η C# είναι μια "διαχειριζόμενη γλώσσα", που σημαίνει ότι κάνει αυτόματα τη διαχείριση μνήμης για τον προγραμματιστή: εκχώρηση-απενεργοποίηση μνήμης, κάλυψη διαρροών μνήμης και ούτω καθεξής. Γενικά, η C# είναι προτιμότερη από τη C++ για δημιουργία παιχνιδιών και διαδραστικών εφαρμογών.

2.4.1 Αντικείμενα παιχνιδιών (Gameobjects) και συστατικά (components) της Unity

Όλο το παιχνίδι και η αλληλεπίδραση που αναπτύσσεται στη Unity βασίζεται σε τρία βασικά δομικά στοιχεία: GameObjects, Components και Variables - Μεταβλητές. Κάθε αντικείμενο σε ένα παιχνίδι είναι ένα GameObject: χαρακτήρες, φώτα, ειδικά εφέ, στηρίγματα κ.α.

Τα GameObjects δεν μπορούν να κάνουν τίποτα από μόνα τους. Για να γίνει πραγματικά κάτι, πρέπει να δοθούν ιδιότητες GameObject, οι οποίες δίνονται προσθέτοντας στοιχεία. Τα στοιχεία καθορίζουν και ελέγχουν τη συμπεριφορά των GameObjects στα οποία είναι συνδεδεμένα. Ένα απλό παράδειγμα θα ήταν η δημιουργία ενός φωτός (Εικόνα 8), το οποίο περιλαμβάνει την προσάρτηση ενός Φωτιστικού Συστήματος σε ένα GameObject ή προσθέτοντας ένα άκαμπτο στοιχείο σώματος σε ένα αντικείμενο για να το κάνει να πέσει.

Τα Componenets έχουν οποιονδήποτε αριθμό ιδιοτήτων επεξεργασίας ή μεταβλητών που μπορούν να τροποποιηθούν μέσω του παραθύρου Inspector στο πρόγραμμα επεξεργασίας Unity ή/και μέσω script. Στο παραπάνω παράδειγμα, ορισμένες ιδιότητες του φωτός είναι το εύρος, το χρώμα και η ένταση.



Εικόνα 8: Προσάρτηση ενός Φωτιστικού Συστήματος σε ένα GameObject

2.4.2 Scripting components στη Unity

Τα ενσωματωμένα components της Unity είναι πολύ ευπροσάρμοστα, ωστόσο ο προγραμματιστής πρέπει να υπερβεί ό,τι μπορούν να παρέχουν για να εφαρμόσει τη δική του λογική. Για να τοκάνει αυτό, χρησιμοποιεί σενάρια για να εφαρμόσει τη δική του λογική και συμπεριφορά παιχνιδιού και στη συνέχεια να προσθέσει αυτά τα σενάρια ως Components σταGameObjects. Κάθε σενάριο συνδέεται με τις εσωτερικές λειτουργίες τηςUnity εφαρμόζοντας μια κλάση που προέρχεται από την ενσωματωμένη κλάσηπου ονομάζεται MonoBehaviour. Τα στοιχεία του σεναρίου θα επιτρέψουν στον χρήστη να κάνειδιάφορα πράγματα όπως ενεργοποίηση συμβάντων παιχνιδιού, έλεγχος για συγκρούσεις, εφαρμογή φυσικής, ανταπόκριση στην είσοδο του χρήστη και πολλά άλλα.

2.4.3 Οφέλη του σχεδιασμού με βάση τα δεδομένα με το DOTS

Εικόνα 9: Σχεδιασμός με χρήση DOTS

Η παραδοσιακή ιδέα του GameObject-Component συνεχίζει να λειτουργεί καλά επειδή είναι κατανοητό τόσο για προγραμματιστές όσο και για μη προγραμματιστές, καθώς και εύκολο στη δημιουργία διαισθητικών διεπαφών χρήστη. Εάν προστεθεί ένα Rigidbody Component σε ένα GameObject, θα αρχίσει να πέφτει. Αν προστεθεί ένα Light Component σε ένα GameObject θα εκπέμπει φως και ούτω καθεξής. Ωστόσο, το σύστημα Component γράφτηκε σε ένα αντικειμενοστραφές πλαίσιο και δημιουργεί προκλήσεις για τους προγραμματιστές όσον αφορά τη διαχείριση της προσωρινής μνήμης και της μνήμης σε συνεχώς εξελισσόμενο υλικό.

Τα συστατικά και τα GameObjects είναι "βαριά C++" αντικείμενα. Όλα τα GameObjects έχουν ένα όνομα. Τα συστατικά τους είναι περιτυλίγματα C# πάνω από τα στοιχεία C++. Αυτό τους καθιστά εύκολο να εργαστούν με αυτά, ωστόσο, μπορεί να έχει κόστος για την απόδοση, επειδή καταλήγουν ενδεχομένως να αποθηκεύονται με μη δομημένο τρόπο. Αυτό το αντικείμενο C# θα μπορούσε να είναι οπουδήποτε στη μνήμη. Το αντικείμενο C++ μπορεί,επίσης, να είναι οπουδήποτε στη μνήμη. Τα πράγματα δεν ομαδοποιούνται μαζί σε συνεχόμενη μνήμη. Κάθε φορά που οτιδήποτε φορτώνεται στην

CPU για επεξεργασία, όλα πρέπει να ληφθούν από πολλές τοποθεσίες. Μπορεί να γίνει αργό και αναποτελεσματικό και ως εκ τούτου, απαιτεί πολλούς τρόπους βελτιστοποίησης.

Για να αντιμετωπίσουμε αυτά τα προβλήματα απόδοσης, ανακατασκευάζουμε το βασικό θεμέλιο της Unity με την υψηλής απόδοσης, πολυνηματική τεχνολογία Data-Oriented Technology Stack ή DOTS.

Το DOTS επιτρέπει στο παιχνίδι να χρησιμοποιεί πλήρως τους πιο πρόσφατους επεξεργαστές πολλαπλών πυρήνων αποτελεσματικά. Αποτελείται από:

- Το C# JobSystem για την αποτελεσματική εκτέλεση πολλαπλών νημάτων κώδικα.
- Το EntityComponentSystem (ECS) για τη σύνταξη κώδικα υψηλής απόδοσης από προεπιλογή.
- Το BurstCompiler για την παραγωγή εξαιρετικά βελτιστοποιημένου εγγενούς κώδικα.

Στο DOTS, το ECS είναι το νέο σύστημα Component. Δηλαδή ό,τι πραγματοποιείται με ένα GameObject με τον παραδοσιακό αντικειμενοστραφή τρόπο, πραγματοποιείται με μια οντότητα σε αυτό το νέο σύστημα. Τα στοιχεία ονομάζονται ακόμα έτσι. Η κρίσιμη διαφορά είναι στη διάταξη δεδομένων.

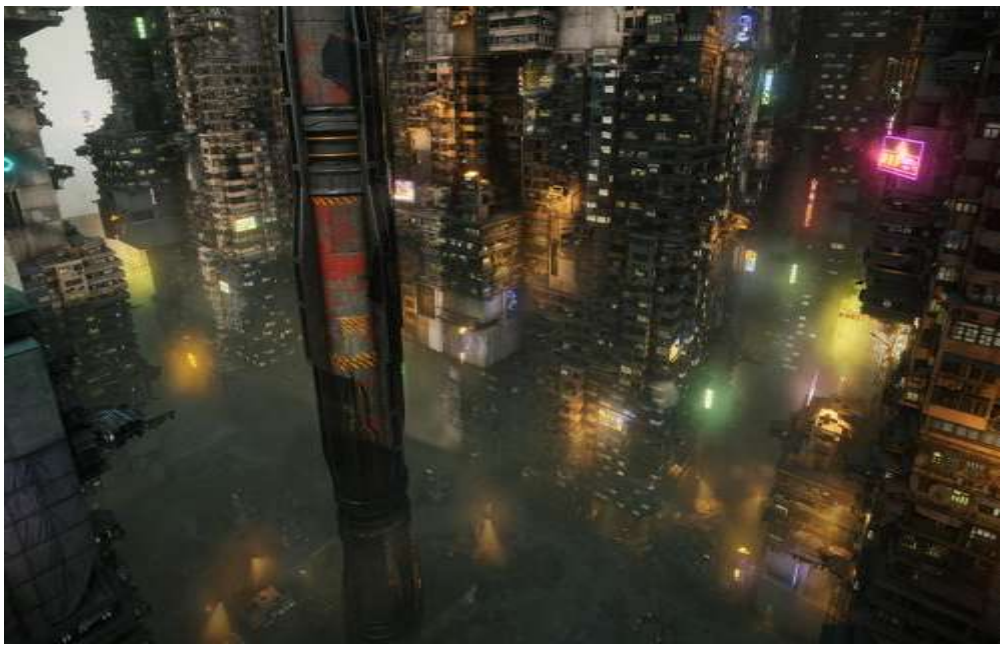
2.4.4 Χρησιμοποιώντας το μοντέρνο hardware

Εκτός από τον καλύτερο τρόπο προσέγγισης του προγραμματισμού παιχνιδιών για λόγους σχεδίασης, η χρήση του ECS είναι ιδανική για να αξιοποιηθούν τοC# JobSystem της Unity και το BurstCompiler, επιτρέποντασταν πλήρη αξιοποίηση του σύγχρονουhardware.

Τα συστήματα πολλαπλών νημάτων της DOTS επιτρέπουν τη δημιουργία παιχνιδιών που εκτελούνται σε μια ποικιλία υλικού και,κατά συνέπεια, τη δημιουργία πλουσιότερων κόσμων παιχνιδιών με περισσότερα στοιχεία και πολύπλοκες προσομοιώσεις. Ο εκτελέσιμοςκώδικας με τη σειρά του συμβάλλει στο βέλτιστο θερμικό έλεγχο και τη διάρκεια ζωής της μπαταρίας στις κινητές

συσκευές των παικτών. Μεταβαίνοντας από αντικειμενοστραφή σε σχεδίαση προσανατολισμένη σε δεδομένα, μπορεί να γίνεται ευκολότερη η επαναχρησιμοποίηση κώδικα και ηεργασία πάνω σε άλλους.

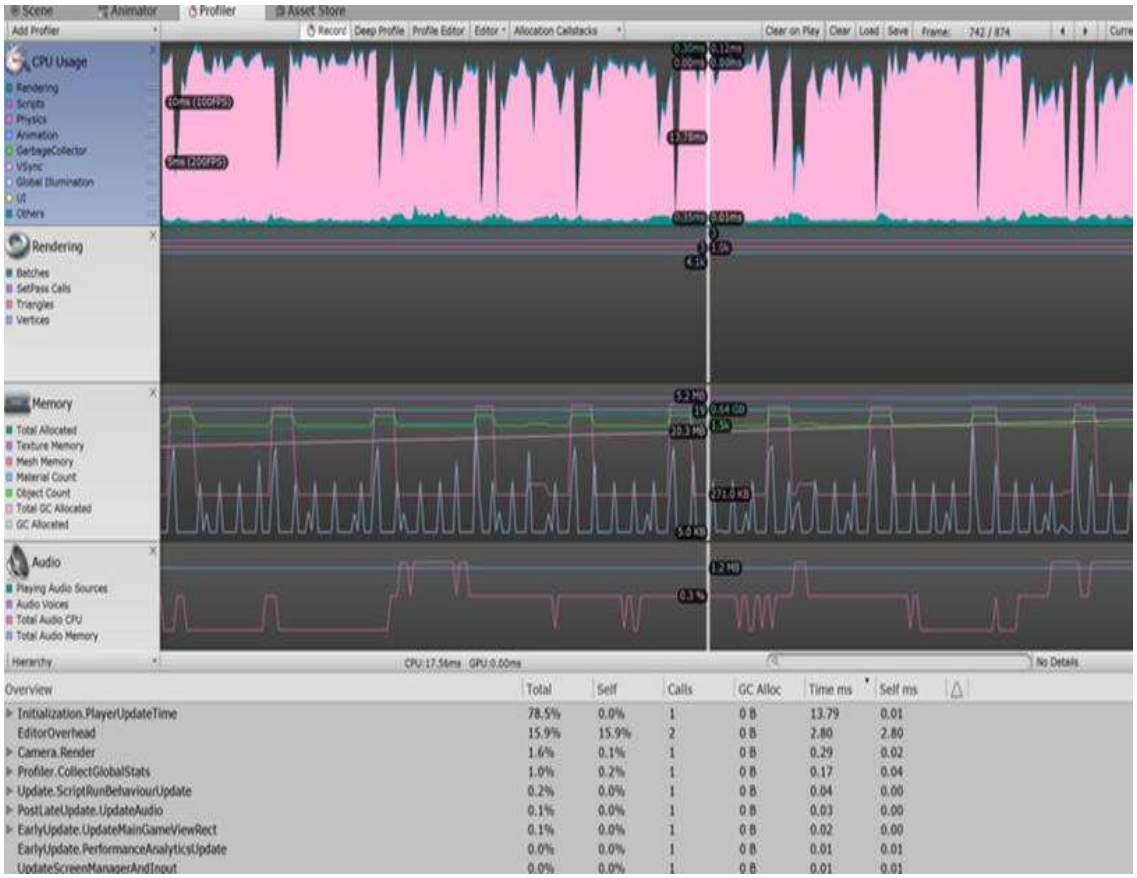
Καθώς μέρος της τεχνολογίας του DOTS βρίσκεται στην προεπισκόπηση, συνιστάται οι προγραμματιστές να τη χρησιμοποιούν για να λύσουν μια συγκεκριμένη πρόκληση απόδοσης στα έργα τους, σε αντίθεση με την κατασκευή ολόκληρων έργων σε αυτό.



Εικόνα 10: Περιβάλλον παιχνιδιού σε γλώσσα Unity

2.4.5 Debugging στη Unity

Το tweaking και ο εντοπισμός σφαλμάτων είναι αποτελεσματικά στο Unity, επειδή όλες οι μεταβλητές του παιχνιδιού εμφανίζονται σωστά καθώς παίζουν οι προγραμματιστές, οπότε τα πράγματα μπορούν να αλλάξουν χωρίς καθυστέρηση. Το παιχνίδι μπορεί να τεθεί σε παύση ανά πάσα στιγμή ή ο χρήστης μπορεί να εκτελεί βήμα-βήμα τον κώδικα κάθε φορά.



Εικόνα 11: Περιβάλλον debuggingστη Unity

3 Ρομποτικοί βραχίονες

Εικόνα 12: Ρομποτικός βραχίονας βιομηχανικού τύπου

Το αντικείμενο μελέτης της παρούσας διπλωματικής είναι ένας ρομποτικός βραχίονας, επομένως κρίνεται σημαντικό, πριν ξεκινήσει σε βάθος η ανάλυση όλων των στοιχείων της, να αναφερθεί ο ορισμός και τα βασικά χαρακτηριστικά αυτού.

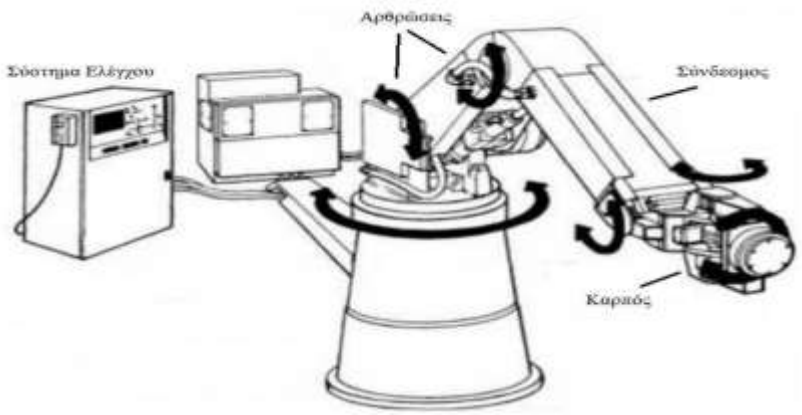
3.1 Δομικά χαρακτηριστικά

Οι ρομποτικοί βραχίονες αποτελούνται από τη βάση, τις αρθρώσεις, τους συνδέσμους και το τελικό εργαλείο δράσης. Η βάση είναι στερεωμένη στον χώρο εργασίας του ρομποτικού βραχίονα και πάνω σε αυτή συνδέονται διαδοχικά οι αρθρώσεις και οι σύνδεσμοι με κατάληξη το τελικό εργαλείο δράσης. Οι σύνδεσμοι είναι ο «σκελετός» των ρομπότ και είναι στερεά σώματα. Οι αρθρώσεις είναι οι μηχανισμοί κίνησης μεταξύ των συνδέσμων. Τέλος, το εργαλείο τελικής δράσης είναι το εργαλείο με

το οποίο το ρομπότ εκτελεί τη δοσμένη εργασία. Ανάλογα με την εργασία που εκτελεί ένας ρομποτικός βραχίονας, το εργαλείο τελικής δράσης μπορεί να προσαρμοστεί σε αυτή (π.χ. μπορεί να είναι εργαλείο ηλεκτροσυγκόλλησης, κατσαβίδι ή το συνηθέστερο να είναι μια αρπάγη). Η αρπάγη μπορεί και αυτή με τη σειρά της να έχει διάφορες μορφές όπως για παράδειγμα βεντούζα, δαγκάνα κ.ά. Η αρίθμηση των αρθρώσεων και των συνδέσμων γίνεται από τη βάση του βραχίονα προς το εργαλείο τελικής δράσης, όπως φαίνεται στο παρακάτω σχήμα.

Ο κάθε βραχίονας αποτελείται από τρία μέρη: α)τη βάση, β) τον κορμό και γ) τον καρπό του. Όπως προαναφέρθηκε, η βάση είναι το στήριγμα του βραχίονα όπου πάνω σε αυτή βρίσκεται ο κορμός του, ο οποίος αποτελείται από διαδοχικούς συνδέσμους και αρθρώσεις. Το εργαλείο τελικής δράσης κινείται με μια ομάδα αρθρώσεων που ονομάζεται «καρπός». Η κάθε άρθρωση αντιστοιχεί σε έναν βαθμό ελευθερίας, οπότε ένας βραχίονας ο οποίος έχει n αριθμό αρθρώσεων θα έχει και n βαθμούς ελευθερίας.

Ένας ρομποτικός βραχίονας αποτελείται ακόμα από ενεργοποιητές «κινητήρες», αισθητήρες, σύστημα επικοινωνίας «εγκέφαλο», που συνήθως είναι ηλεκτρονικός υπολογιστής, και σύστημα αυτόματου ελέγχου (Σ.Α.Ε.). Με τη χρήση των παραπάνω, ο χρήστης μπορεί να προγραμματίσει το ρομπότ ώστε αυτό να εκτελέσει μια σειρά κινήσεων.



Εικόνα 13: Διάταξη ρομποτικού βραχίονα

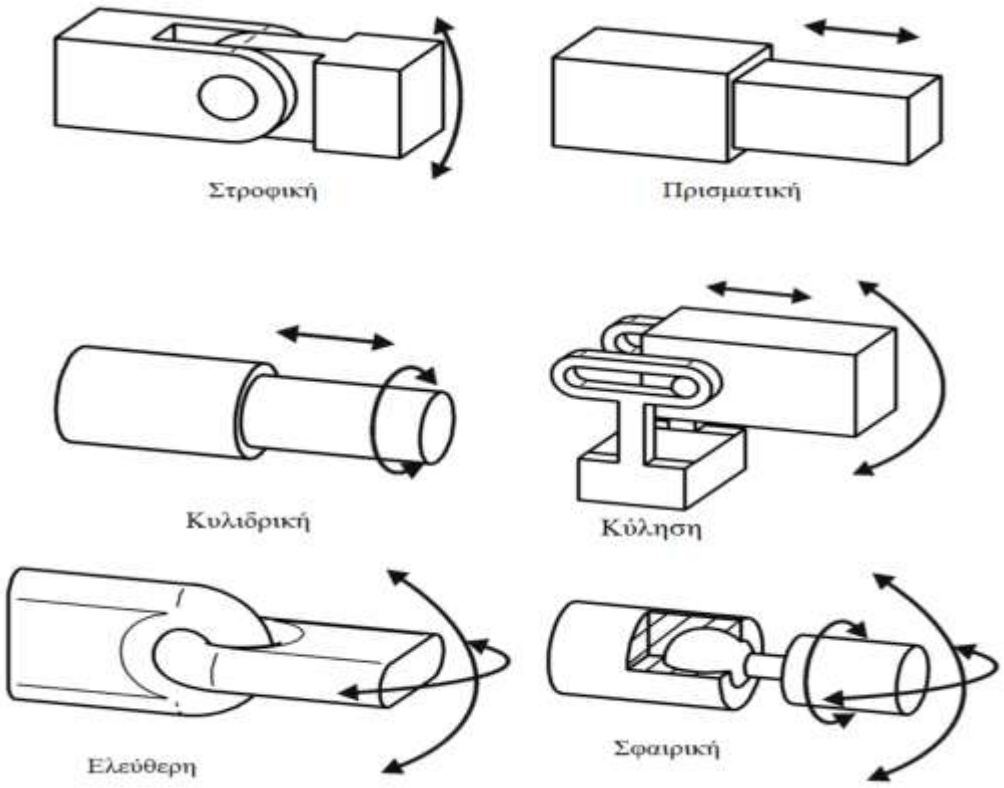
3.1.1 Είδη αρθρώσεων

Αρθρώσεις είναι οι διατάξεις οι οποίες συνδέουν δύο συνδέσμους μεταξύ τους. Με τη βοήθεια των ενεργοποιητών κινούν τους συνδέσμους και κατά συνέπεια τον βραχίονα. Στον χώρο των τριών διαστάσεων (μήκος, πλάτος και ύψος) οι κινήσεις που μπορούν να πραγματοποιηθούν μεταξύ δύο

στερεών σωμάτων εκ των οποίων το ένα θεωρείται ακίνητο, είναι τρεις μεταφορικές κινήσεις στις διευθύνσεις των αξόνων του καρτεσιανού συστήματος και τρεις περιστροφικές κινήσεις.

Ένας βραχίονας με τη χρήση δύο αρθρώσεων μπορεί να κινηθεί μεταξύ δύο αξόνων X,Y (μήκος και πλάτος). Με τρεις αρθρώσεις μπορεί να κινηθεί στο χώρο μεταξύ τριών αξόνων X,Y,Z (μήκος, πλάτος και ύψος).

Οι αρθρώσεις ενός βραχίονα χωρίζονται σε δύο κατηγορίες; τις στροφικές και τις πρισματικές (τηλεσκοπικές). Υπάρχουν και άλλα είδη αρθρώσεων όπως είναι οι κυλινδρικές, οι ελεύθερες, οι σφαιρικές και οι αρθρώσεις κύλισης, οι οποίες προκύπτουν από τη σύνδεση των δύο βασικών αρθρώσεων. Τα σχήματα που ακολουθούν αναπαριστούν τα είδη των αρθρώσεων.



Εικόνα 14: Είδη ρομποτικών αρθρώσεων

3.1.2Βαθμοί ελευθερίας και κινητικότητας

Ως βαθμός ελευθερίας ορίζεται ο αριθμός των ανεξάρτητων μεταβλητών θέσεων ενός ρομποτικού βραχίονα. Σε κάθε διάταξη, ο αριθμός αυτός πρέπει να είναι γνωστός και ορισμένος προκειμένου να γίνεται εφικτός ο προσδιορισμός των θέσεων των τμημάτων που την αποτελούν.

Σε ένα ρομπότ, εκτός από τους βαθμούς ελευθερίας, υπάρχουν και οι βαθμοί κινητικότητας. Το πλήθος των βαθμών κινητικότητας είναι ίσο με το πλήθος των αρθρώσεων του ρομπότ, ενώ οι βαθμοί ελευθερίας προκύπτουν από την εκάστοτε εργασία την οποία εκτελεί το ρομπότ. Επομένως, σε μερικές διεργασίες, το πλήθος των βαθμών ελευθερίας μπορεί να είναι διαφορετικό από το πλήθος των βαθμών κινητικότητας. Υπάρχουν, ωστόσο, και διεργασίες όπου το πλήθος των βαθμών είναι ίσο μεταξύ τους. Για παράδειγμα, στην απλή περίπτωση που θέλουμε να μετακινήσουμε και να τοποθετήσουμε ένα αντικείμενο στο χώρο χρειαζόμαστε έξι βαθμούς ελευθερίας. Από αυτούς οι τρείς χρειάζονται για την μετακίνηση του αντικειμένου και οι άλλοι τρείς για τον προσανατολισμό του. Σε αυτή την περίπτωση, ένας βραχίονας ο οποίος αποτελείται από έξι αρθρώσεις και κατά συνέπεια ίσους βαθμούς κινητικότητας, μπορεί να πραγματοποιήσει αυτήν τη λειτουργία.

3.1.3 Ενεργοποιητές – Κινητήρες

Οι ενεργοποιητές είναι συσκευές που ενεργοποιούν την κίνηση των τμημάτων του βραχίονα. Δέχονται σαν είσοδο ένα ηλεκτρικό σήμα από έναν ηλεκτρονικό υπολογιστή προκειμένου να ξεκινήσει η λειτουργία τους. Τέτοιες συσκευές είναι τα υδραυλικά συστήματα και οι κινητήρες συνεχούς ρεύματος, οι βηματικοί κινητήρες και οι servo κινητήρες. Η κάθε κατηγορία των κινητήρων έχει πλεονεκτήματα και μειονεκτήματα, γι’αυτό σε κάθε διεργασία χρησιμοποιείται ο κατάλληλος κινητήρας.

Οι κινητήρες συνεχούς ή εναλλασσόμενου ρεύματος με γωνία στρέψης τις 360º, έχουν μεγάλη ροπή. Ένα μεγάλο μειονέκτημα είναι ότι δεν υπάρχει έλεγχος της θέσης του άξονά τους και γι’αυτό χρειάζονται εξωτερικό κύκλωμα οδήγησης. Οι βηματικοί κινητήρες έχουν και αυτοί γωνία στρέψης τις 360º, οπότε έχουν μικρή ροπή. Όπως και στη προηγούμενη κατηγορία κινητήρων, έτσι και οι βηματικοί, δεν έχουν κάποιο κύκλωμα προκειμένου να ελεγχθεί η θέση τους και κατά συνέπεια χρειάζονται και αυτοί εξωτερικό κύκλωμα οδήγησης. Οι κινητήρες servoέχουν γωνία περιστροφής τις 180º. Έχουν μεγάλη ροπή και διαθέτουν ενσωματωμένο κύκλωμα οδήγησης ώστε να μπορεί να γίνει έλεγχος της θέσης τους. Αυτό γίνεται με τη χρήση ενός ποτενσιόμετρου, το οποίο, μαζί με το κύκλωμα, είναι ενσωματωμένα στον σερβοκινητήρα.

Τέλος, τα υδραυλικά συστήματα είναι έμβολα που με τη χρήση υγρού, κυρίως λάδι, κινούν την άρθρωση κατάλληλα. Λόγω της υδραυλικής πίεσης, η ροπή τους είναι μεγαλύτερη από αυτή των προηγούμενων ενεργοποιητών-κινητήρων.

3.1.4 Χώρος εργασίας

Ως χώρος εργασίας ορίζεται ο γεωμετρικός τόπος των σημείων του χώρου τα οποία μπορεί να τα προσεγγίσει το εργαλείο τελικής δράσης του ρομποτικού βραχίονα. Ανάλογα με το είδος του ρομποτικού βραχίονα, ο χώρος μπορεί να είναι ακίνητος ως προς αυτόν ή όχι.

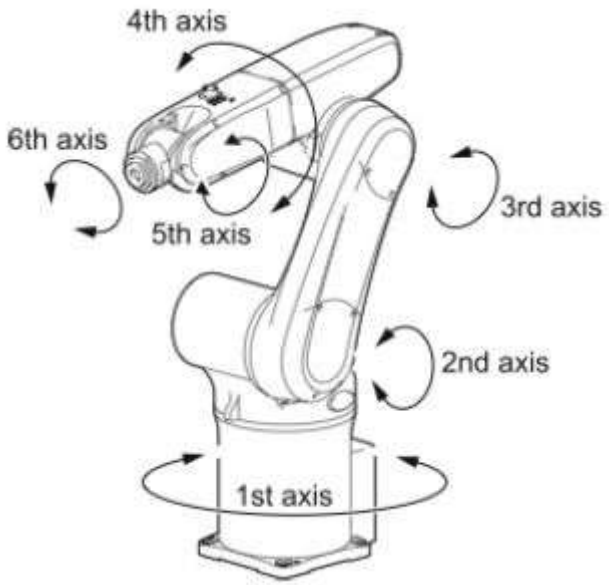
3.1.5 Ελεγκτής

Σε κάθε βραχίονα υπάρχει μια μονάδα ελεγκτή η οποία λαμβάνει δεδομένα του ρομποτικού βραχίονα μέσω του ηλεκτρονικού υπολογιστή. Τα δεδομένα ή μεταβλητές του ρομποτικού βραχίονα είναι η θέση, η ταχύτητα και η γωνία μετατόπισης των αρθρώσεων αλλά και του εργαλείου τελικής δράσης. Ο ελεγκτής επεξεργάζεται αυτές τις μεταβλητές προκειμένου η λειτουργία και οι κινήσεις του βραχίονα να είναι ιδανικές για τη λειτουργία του.

3.1.6 Είδη ρομποτικών βραχιόνων

Αρθρωτοί

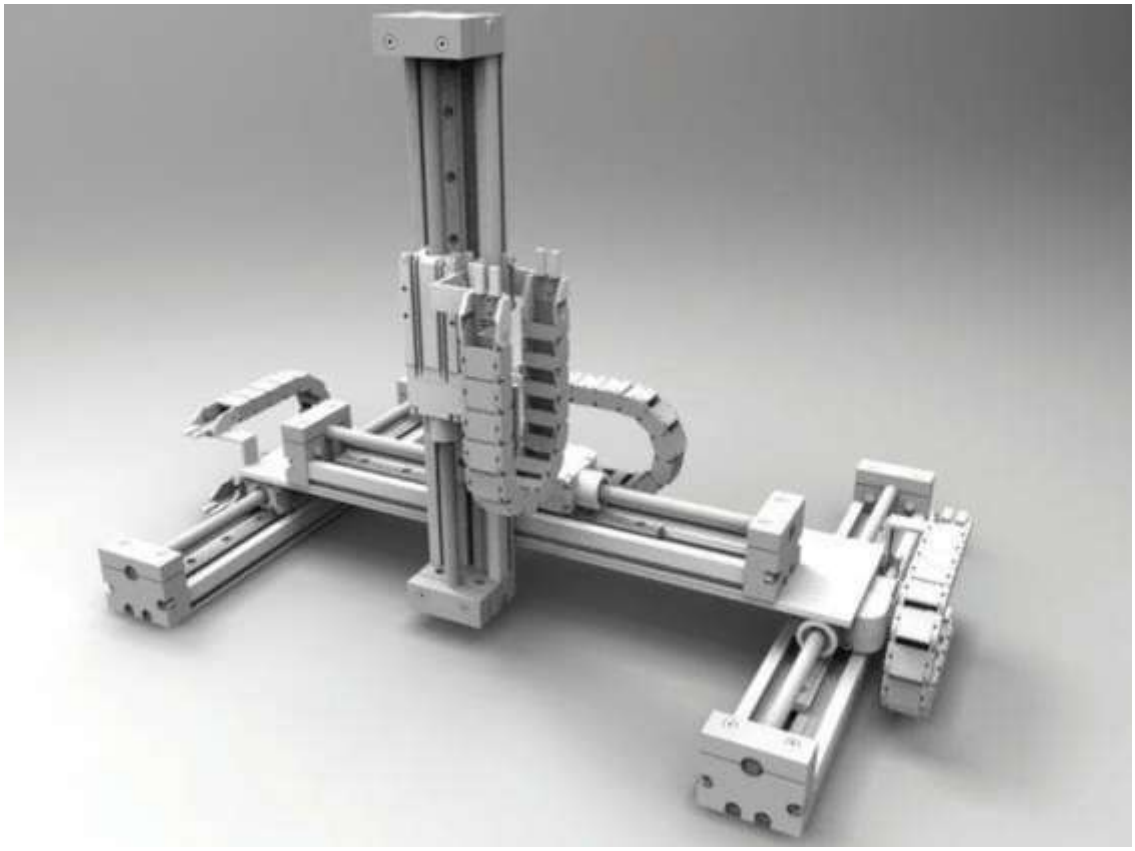
Αυτός ο σχεδιασμός περιλαμβάνει περιστροφικές αρθρώσεις και έχει εύρος από δύο δομές αρθρώσεων έως,συνήθως, τις δέκα. Ο βραχίονας είναι συνδεδεμένος σε μια σταθερή βάση που μπορεί να συστρέφεται. Οι σύνδεσμοι είναι συνδεδεμένοι με περιστροφικές αρθρώσεις. Με κάθε άρθρωση προστίθεται ένας επιπλέον βαθμός ελευθερίας ή αυξάνεται το εύρος των κινήσεων.



Εικόνα 15: IndustrialArticulatedRobot [6]

Ευθύγραμμοι

Η κατηγορία αυτών των βραχιόνων αποτελείται από τρεις γραμμικούς συνδέσμους που χρησιμοποιούν το καρτεσιανό σύστημα συντεταγμένων (X,Y,Z). Σε μερικές περιπτώσεις, στην άκρη τους συνδέεται ένας περιστροφικός σύνδεσμος για την προσθήκη ενός ακόμα βαθμού ελευθερίας.



Εικόνα 16: Καρτεσιανό ρομπότ [6]

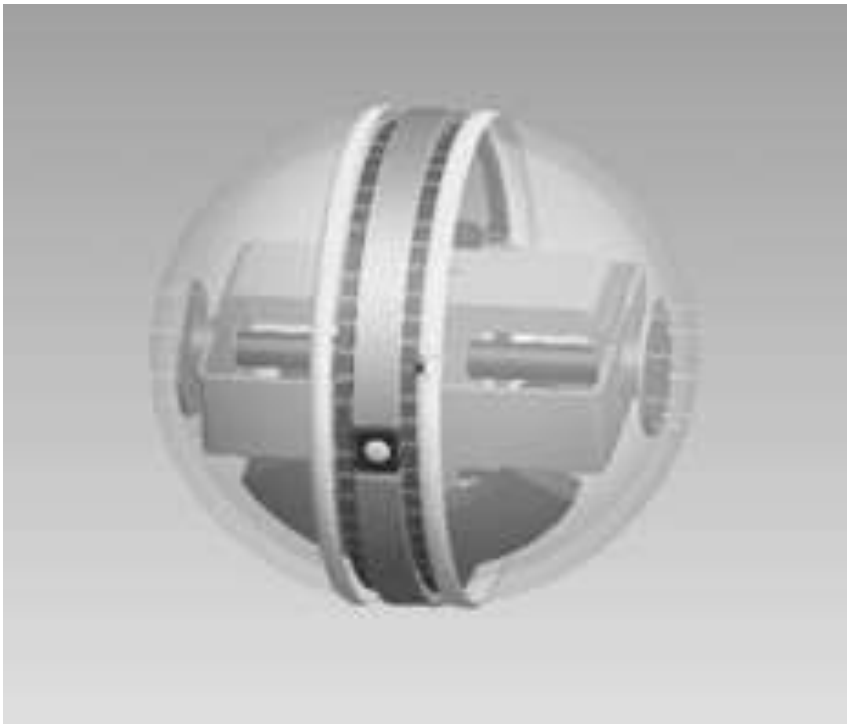
Κυλινδρικοί

Οι κυλινδρικοίβραχίονες, έχουν τουλάχιστον μία περιστροφική άρθρωση στη βάση και τουλάχιστον μία πρισματική άρθρωση για την ένωση των συνδέσμων. Η περιστροφική κίνηση γίνεται κατά μήκος του άξονα του συνδέσμου, ενώ η πρισματική κινείται γραμμικά.

Εικόνα 17: Κυλινδρικό ρομπότ [7]

Σφαιρικοί

Οι σφαιρικοί βραχίονες, όπως υποδηλώνει και το όνομα, έχουν δυνατότητα κίνησης σε ένα σφαιρικό πεδίο στον χώρο. Οι άξονες τους είναι συνδεδεμένοι σε μία σταθερή βάση, ενώ ο συνδυασμός μίας γραμμικής άρθρωσης και δύο περιστροφικών αρθρώσεων, τους επιτρέπει να συστρέφονται σε ένα πολικό σύστημα συντεταγμένων.



Εικόνα 18: Σφαιρικό ρομπότ [8]

Δελτοειδείς

Οι βραχίονες αυτοί είναι φτιαγμένοιαπό παράλληλες αρθρώσεις συνδεδεμένες σε μία κοινή βάση. Όλοι οι σύνδεσμοι κινούν ένα μονό άκρο του βραχίονα, το οποίο είναι ικανό να έχει πολύ υψηλή ακρίβεια. Κυρίως χρησιμοποιείται σε βιομηχανίες ηλεκτρονικών, φαρμακευτικές εταιρίες και σε ορισμένες περιπτώσεις σε εγκαταστάσεις επεξεργασίας τροφίμων.

Εικόνα 19: Delta robot arm [9]

2.4.6 SCARA

Αυτός ο τύπος βραχίονα είναι κυρίως για χρήση σε εφαρμογές συναρμολόγησης. Έχει κυκλικό σχεδιασμό και διαθέτει δύο παράλληλες αρθρώσεις που το βοηθούν να κινείται σε ένα συγκεκριμένο επίπεδο, προκαθορισμένο κάθε φορά.

Εικόνα 20: SCARA arm[10]

4 Κινηματική θεωρία

4.1 Εισαγωγή στην κινηματική θεωρία

Η Κινητική ή Κινηματική (Kinematics, από το ελληνικό κινεῖν) είναι κλάδος της μηχανικής που περιγράφει την κίνηση των σωμάτων αδιαφορώντας για τη μάζα τους ή τις αιτίες, δυνάμεις, που προκαλούν την κίνησή τους. Επίκεντρο του ενδιαφέροντος της κινηματικής είναι η θέση και όλα τα παράγωγα των μεταβλητών της (όπως η ταχύτητα, η επιτάχυνση).

Η κινηματική ανάλυση ασχολείταιμε την επίλυση δύο προβλημάτων, του ευθέος και του αντίστροφου κινηματικού προβλήματος. Στην περίπτωση της εύρεσης της θέσης και του προσανατολισμού του άκρου του βραχίονα ως προς τη βάση του, όταν είναι γνωστέςοι θέσεις κάθε άρθρωσης, συνίσταται το ευθύ κινηματικό πρόβλημα, ενώ το αντίστροφο κινηματικό πρόβλημα στην εύρεση της θέσης και του προσανατολισμού της κάθε άρθρωσης, σε σχέση με το εργαλείο τελικής δράσης. Ενώ η λύση του ορθού κινηματικού προβλήματος είναι μοναδική, η λύση του αντίστροφου κινηματικού προβλήματος δεν είναι, διότι η κινηματική εξίσωση περιέχει τριγωνομετρικές συναρτήσεις.

Απαραίτητη προϋπόθεση για τη μετακίνηση του άκρου του βραχίονα από μία θέση σε μία άλλη είναι η γνώση των γωνιών των αρθρώσεων που αντιστοιχούν σε αυτήν. Επιλύοντας το αντίστροφο κινηματικό πρόβλημα, μπορούν να υπολογιστούν οι γωνίες αυτές. Τέλος, η γεωμετρία του βραχίονα είναι αυτή που καθορίζει τον βαθμό δυσκολίας επίλυσης των προβλημάτων αυτών.

4.2 Κινηματική αλυσίδα

Για να μελετηθεί η κινηματική του εκάστοτε ρομπότ, πρέπει πρώτα να αναλυθούν τα επιμέρους κινηματικά προβλήματα των συνδέσμων και των αρθρώσεων του ρομπότ. Κινηματική αλυσίδα είναιοι διαδοχικοίσύνδεσμοι και οι αρθρώσεις του ρομπότ που αντιστοιχούν σε μια αλληλουχία κινηματικών προβλημάτων. Η μελέτη αυτή στηρίζεται στην εξής παραδοχή: Κάθε άρθρωση έχει έναν μόνο βαθμό ελευθερίας που συμβολίζεται με τη μεταβλητήq_n,όπου n είναι το σύνολο των αρθρώσεων ενός ρομπότ. Αν και οι σφαιρικές αρθρώσεις αποτελούνται από δύο στροφικές, δεν αναλύονται σε δύο αλλά σε έναν βαθμό ελευθερίας ακολουθώντας την παραπάνω παραδοχή.

Για να μελετηθεί σωστά η κινηματική ενός ρομποτικού βραχίονα, μελετάται ξεχωριστά η διάταξη μεταξύ των αρθρώσεων και των συνδέσμων καθώς και το σύστημα συντεταγμένων του. Η μελέτη αυτή γίνεται εκτελώντας τα εξής βήματα:

- Η βάση του βραχίονα είναι ο σύνδεσμος 0
- Ο σύνδεσμος μεταξύ της βάσης και της πρώτης άρθρωσης είναι ο σύνδεσμος 1
- Η άρθρωση μεταξύ των 0 και 1 συνδέσμων καλείται άρθρωση 0.
- Γενικότερα, η άρθρωση μεταξύ του συνδέσμου i-1 και ieίναι η άρθρωση i (όπου i= 1,2,3….,n-1).
- Τέλος, ως τελευταίο σύνδεσμο θεωρούμε το εργαλείο τελικής δράσης σαν σύνδεσμο υπ' αριθμόν n.

Για την επιλογή του συστήματος συντεταγμένων ακολουθείται η εξής λογική:

Ο κάθε σύνδεσμος αντιστοιχίζεται σε ένα σύστημα συντεταγμένων $\{O_i\ ,\ x_i\ ,\ y_i\ ,\ z_i\}$.

Οι γενικευμένες συντεταγμένες ενός τυχαίου σημείου Α σε χώρο με σημεία a_iκαι a_{i-1} ως προς τα συστήματα συντεταγμένων $\{O_i\ ,\ x_i\ ,\ y_i\ ,\ z_i\}$ και $\{O_{i-1}\ ,\ x_{i-1}\ ,\ y_{i-1}\ ,\ z_{i-1}\}$ συνδέονται με την ακόλουθη σχέση:

$$a_{i-1} = H_{i-1}^{i}\ (q_i)a_i,$$

όπου $H_{i-1}^{i}\ (q_i)$ είναι ο ομογενής μετασχηματισμός που συνδέει τα δύο αυτά συστήματα και q_iη γωνία μετατόπισης της i άρθρωσης. Η γωνία μετατόπισης επηρεάζει τον παραπάνω μετασχηματισμό με τελική μορφή την παρακάτω:

$$H_{i-1}^{i}\ (q_i) = \begin{bmatrix} R_{i-1}^{i}\ (q_i) & d_{i-1}^{i}\ (q_i) \\ 0_{1x3} & 1 \end{bmatrix}$$

Όπου $d_{i-1}^{i}\ (q_i)$ πίνακας μετατόπισης, ο οποίος περιλαμβάνει τις συντεταγμένες με αρχή το σημείο O_i και τελικό σημείο το O_{i-1} του συστήματος i-1. Και ο R_{i-1}^{i} είναι πίνακας στροφής του συστήματος i ως προς το i-1 σύστημα.

Η σύνδεση δύο συστημάτων συντεταγμένων i και j του σημείου Α από το σύστημα i ως προς το σύστημα j γίνεται με τον μετασχηματισμό H_i^j και δίνεται από τις σχέσεις:

Για i < j , $H_i^j = H_i^{i+1}, H_{i+1}^{i+2} \ldots\ldots H_{j-1}^j$

Για i > j , $H_i^j = (H_i^j)^{-1}$

Για i = j, $H_i^j = I_4$, όπου I_4 μοναδιαίος πίνακας 4 x 4

Στην περίπτωση που ισχύει η συνθήκη i<j:

$$H_i^j = \begin{bmatrix} R_i^j & d_i^j \\ 0_{1x3} & 1 \end{bmatrix}$$

4.3 Ευθύ κινηματικό πρόβλημα

Όπως προαναφέρθηκε, ένας βραχίονας αποτελείται από συνδέσμους και αρθρώσεις, οι οποίοι είναι τοποθετημένοι από τη βάση του μέχρι το εργαλείο τελικής δράσης. Προκειμένου να εφαρμοστεί το ευθύ κινηματικό πρόβλημα, πρέπει να καθοριστεί το κατάλληλο σύστημα συντεταγμένων για κάθε άρθρωση του ρομποτικού βραχίονα. Στη συνέχεια, πρέπει να προσδιοριστεί η τελική θέση αλλά και ο προσανατολισμός του εργαλείου τελικής δράσης με δεδομένες τις τιμές γωνιών των αρθρώσεων. Για την επίλυση του προβλήματος αυτού, υπάρχει η μέθοδοςDenavit–Hartenberg,η οποία χρησιμοποιεί τέσσερις παραμέτρους. Αυτές είναι: το μήκος, η στρέψη, το περιθώριο και η γωνία μεταξύ των αξόνων.

4.3.1 Μέθοδος Denavit–Hartenberg

Με τη μέθοδο Denavit–Hatenberg μελετάται το ευθύ κινηματικό πρόβλημα ενός ρομποτικού βραχίονα ώστε να προσδιοριστεί το εργαλείο τελικής δράσης ως προς το χώρο εργασίας του, με δεδομένες τις μεταβλητές των αρθρώσεων του βραχίονα. Για την επίλυση του προβλήματος με τη χρήση αυτής της μεθόδου, τοποθετούνται δεξιόστροφα ορθοκανονικά συστήματα συντεταγμένων μεταξύ των συνδέσμων του βραχίονα, θεωρώντας τον χώρο εργασίας σταθερό και αμετάβλητο.

Αρχικά καθορίζονται τα συστήματα συντεταγμένων $\{O_n , x_n , y_n , z_n\}$ ξεκινώντας από τη βάση του βραχίονα,που θεωρείταιτο σημείο 0,έως το εργαλείο τελικής δράσης, το οποίο είναι το n σημείο του

βραχίονα. Στη συνέχεια, καθορίζεται κατάλληλα το σύστημα συντεταγμένων κάθε άρθρωσης σύμφωνα με την παρακάτω διαδικασία:

- Ο άξονας z πρέπει να συμπίπτει με τον άξονα κίνησης της άρθρωσης.
- Ο άξονας x τοποθετείται έτσι ώστε να είναι κάθετος με τον άξονα z της τρέχουσας αλλά και της προηγούμενης άρθρωσης.
- Η προέκταση του άξονα x πρέπει να τέμνει την προέκταση του άξονα z της προηγούμενης άρθρωσης.
- Ο άξονας y τοποθετείται στο τέλος ώστε το σύστημα να υπακούει στον κανόνα του δεξιού χεριού.
- Μετά την ολοκλήρωση της παραπάνω διαδικασίας για κάθε άρθρωση, προσδιορίζονται οι παράμετροι της μεθόδου D-H, οι οποίες είναι:
    - Το μήκος a_n μεταξύ των αξόνων.
    - Η γωνία στρέψης b_n.
    - Το περιθώριο d_n.
    - Και η γωνία μετατόπισης θ_i μεταξύ των σημείων n και n+1.

Πλέον, μπορούν να υπολογιστούν οι πίνακες R_0^n ομογενούς μετασχηματισμού από το αρχικό σύστημα συντεταγμένων $\{O_0\ ,\ x_0\ ,\ y_0\ ,\ z_0\}$ μέχρι το $\{O_n\ ,\ x_n\ ,\ y_n\ ,\ z_n\}$. επίσης,υπολογίζεται ο πίνακαςd_o^n για κάθε σύστημα συντεταγμένων μεταξύ των σημείων 0 και n. Έτσι, μπορεί να υπολογιστεί ο μετασχηματισμός του συστήματος συντεταγμένων που ορίζεται με τον πίνακα H_0^n. Για να συνδυαστούν οι μετασχηματισμοί από το σημείο 0 μέχρι το n, πραγματοποιείται πολλαπλασιασμός πινάκων.

$H_0^n = H_0^1\ H_0^2 \ldots .\ H_{n-2}^{n-1} H_{n-1}^n$.

Ορίζεται,επίσης, ο πίνακαςS_A, ο οποίος περιέχει τις γενικευμένες συντεταγμένες του σημείου A ως προς το σύστημα $\{0_n\ ,\ x_n\ ,\ y_n\ ,\ z_n\}$. Τέλος, οι συντεταγμένες του σημείου A προκύπτουν από τη σχέση:

$$P_0^A = H_0^n\ *\ S_A$$

Με τη χρήση του παραπάνω τύπου γίνονται γνωστές οι συντεταγμένες των ενδιάμεσων σημείων μεταξύ του σημείου 0 και A.

4.4 Αντίστροφο κινηματικό πρόβλημα

Σε αυτήν την περίπτωση, γνωρίζοντας τη θέση και τον προσανατολισμό του εργαλείου τελικής δράσης, ζητείται να προσδιοριστούν οι μεταβλητές των γωνιών των αρθρώσεων του βραχίονα. Οι εξισώσεις που χρησιμοποιούνται είναι μη γραμμικές και οι μορφές τους είναι πολυωνυμικές και τριγωνομετρικές. Αυτό έχει ως αποτέλεσμα να μην είναι εφικτή η αναλυτική επίλυση του προβλήματος.

Αρχικά, πραγματοποιείται η εύρεση των πινάκωνH_0^n του συστήματοςσυντεταγμένων, όπως ακριβώς και στη μέθοδο Denavit-Hartenberg. Τα μήκη των συνδέσμων, αλλά και οι γωνίες στρέψης των αρθρώσεων, θεωρούνται άγνωστα και αντικαθίστανται με τις μεταβλητές X,Y,Z και φ.

X,Y,Z είναι τα μήκη των συνδέσμων ως προς τον κάθε άξονα και φ η γωνία μετατόπισης των αρθρώσεων.

Από τον πίνακα H_0^n υπολογίζονται οι μη γραμμικές εξισώσεις με σκοπό την εύρεση της θέσης της n άρθρωσης, η οποία είναι μεταξύ του εργαλείου τελικής δράσης και του τελευταίου συνδέσμου της διάταξης του βραχίονα.

Για να βρεθεί η κίνηση μια άρθρωσης,χρησιμοποιείται η συνάρτηση δύο ορισμάτων του τόξου εφαπτομένης της.

$$\theta_n = \text{Atan2}(\text{sinn},\text{cosn})$$

Το πρόσημο του ημιτόνου καθορίζει τη διεύθυνση που θα έχει ο βραχίονας. Έτσι,είναι δυνατό να γίνει επιλογή της λύσης προκειμένου ο βραχίονας να κινηθεί κατάλληλα προς το επιθυμητό σημείο. Προκειμένου να υπολογιστεί η γωνία, θα υπολογιστεί το ημίτονο και το συνημίτονο για να εκχωρηθούν ως ορίσματα στη συνάρτηση του τόξου εφαπτομένης. Με τον τρόπο αυτό, εξασφαλίζονται όλες οι πιθανές λύσεις.

Γνωρίζοντας την n γωνία, είναι εφικτό να υπολογιστεί η n-1 με τη χρήση των ίδιων μη γραμμικών εξισώσεων. Η διαδικασία αυτή επαναλαμβάνεται έως ότου μείνει να προσδιοριστεί η τελευταία άρθρωση. Ο υπολογισμός της θα γίνει με το άθροισμα των γωνιών και των ορισμάτων του τόξου εφαπτομένης της γωνίας φ που ορίστηκε στην αρχή.

$$(\theta_0 + \theta_1 + ... + \theta_{n-1} + \theta_n) = \text{Atan2}(\sin\varphi, \cos\varphi)$$

Αν από τις γραμμικές εξισώσεις οι μεταβλητές X,Y,Z είναι 0, τότε οι γωνίες μπορεί να έχουν οποιαδήποτε τιμή και να είναι αόριστες. Τέλος, οι τιμές που θα δοθούν στις γωνίες είναι υποθετικές προκειμένου να βρεθούν οι πιθανές πορείες όλων των αρθρώσεων.

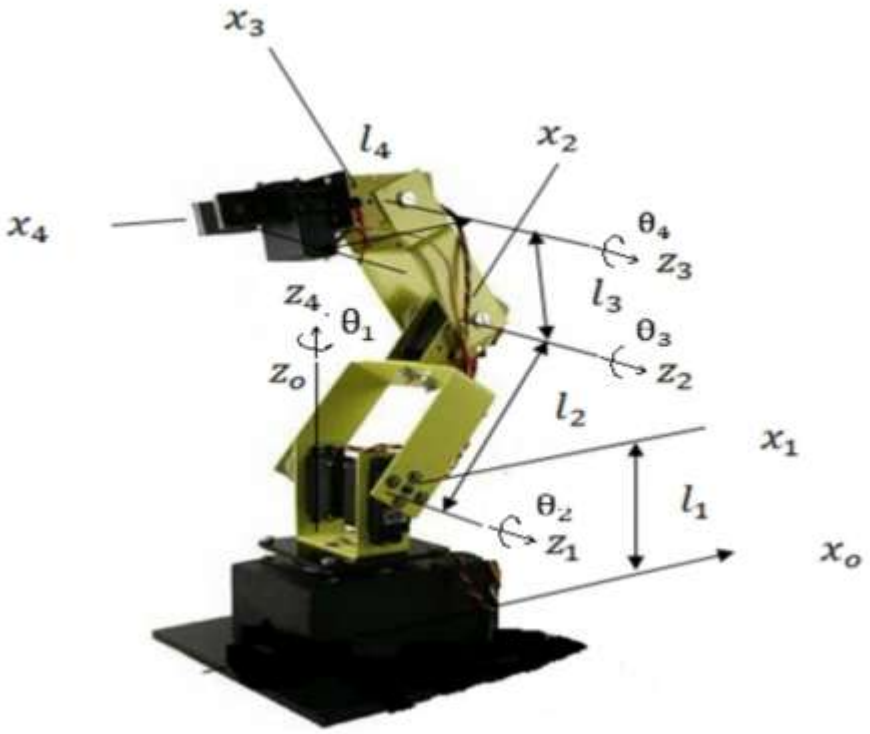
5 Ρομποτικός βραχίονας 4 βαθμών ελευθερίας – ανάλυση

5.1 DH ανάλυση

Η μέθοδοςDenavit–Hartenberg χρησιμοποιείται συχνά κατά την κινηματική ανάλυση ενός ρομποτικού βραχίονα. Βασίζεται στην προσαρμογή ενός συστήματος συντεταγμένων σε κάθε άρθρωση του βραχίονα, καθορίζοντας τις 4 παραμέτρους, γνωστές και ως παράμετροι DH, για κάθε σύνδεσμο και κατασκευάζοντας τον σχετικό πίνακα. Έτσι, στο τέλος της διαδικασίας εξάγεται ένας πίνακας μετασχηματισμού. Ο σημαντικότερος στόχος είναι να γίνει έλεγχος τόσο για την θέση,όσοκαι για τον προσανατολισμό του ελεύθερου άκρου ή της αρπάγης στην άκρη του βραχίονα στο χώρο εργασίας.

Για την επίτευξη του στόχου αυτού, εξάγεται πρώτα η σχέση μεταξύ των μεταβλητών των αρθρώσεων και της θέσης και προσανατολισμού της αρπάγης, χρησιμοποιώντας τις παραμέτρους DH. Ως παράδειγμα,χρησιμοποιείται ο βραχίοναςRA-02A της εταιρίας ImagesSIInc.

Εικόνα 21: Ο ρομποτικός βραχίονας



Εικόνα 22: DH παράμετροι

Διάφορες περιστροφικές αλλά και γραμμικές κινήσεις πραγματοποιούνται από τον συγκεκριμένο βραχίονα. Η κάθε κίνηση πραγματοποιείται από τον εκάστοτε σερβοκινητήρα.

5.2 Ευθύ κινηματικό πρόβλημα

Όπως φαίνεται στην παραπάνω εικόνα, ένα σύστημα συντεταγμένων έχει οριστεί για κάθε σύνδεσμο με σκοπό τον υπολογισμό των γειτονικών frames χρησιμοποιώντας τον κατάλληλο τύπο κίνησης.

Frame (i)	a_i	b_i	d_i	θ_i
1	0	90	l_1	θ_1
2	l_2	0	0	θ_2
3	l_3	0	0	θ_3
4	l_4	0	0	θ_4

Με εφαρμογή των παραμέτρων Denavit–Hartenberg για τις συντεταγμένες των αρθρώσεων, ο πίνακας DH κατασκευάζεται όπως παραπάνω. Τα μήκη των συνδέσμων είναι:

$$l_1 = 11.5\ cm, l_2 = \ 12\ cm, l_3 = \ l_4 = 9\ cm$$

5.2.1 Εμπρόσθιες μήτρες μετασχηματισμού

Ο πίνακας μετασχηματισμού από το σύστημαB_i έως το B_{i-1},για την καθιερωμένη μέθοδο DH δίνεται από:

$$T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & \alpha_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & \alpha_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} (1)$$

Στη συνέχεια, οι πίνακες μετασχηματισμού για i = 1, 2, 3, 4 μπορούν να εξαχθούν εύκολα. Επομένως, ολόκληρος ο μετασχηματισμός T_4^0 βρίσκεται με πολλαπλασιασμό των πινάκων ως εξής:

$$T_4^0 = \ T_1^0 T_2^1 T_3^2 T_4^3 \ (2)$$

Αφού γίνει η εύρεση του πίνακαT_4^0 , μπορούν να βρεθούν οι παγκόσμιες συντεταγμένες της αρπάγης. Το άκρο του βραχίονα είναι η ρίζα του συστήματοςB_4, δηλαδή στο $[0\ 0\ 0\ 1]^T$. Οπότε, η θέση του παγκόσμιου συστήματος γίνεται:

$r_P^0 = \ T_4^0 r_P^4 = \ T_4^0 [0\ 0\ 0\ 1]^T = \ [r_{14}\ r_{24}\ r_{34}\ 1]^T = \ [dx\ dy\ dz\ 1]^T (3)$

Χρησιμοποιώντας τριγωνομετρικές εξισώσεις προκύπτουν οι παρακάτω τύποι:

dx = cθ_1 [l_2cθ_2 + l_3 c (θ_2 + θ_3)] + l_3 $c\theta_1$ c (θ_2 + θ_3 + θ_4) (4)

dy = sθ_1 [l_2cθ_2 + l_3 c (θ_2 + θ_3)] + l_4 $s\theta_1$ c (θ_2 + θ_3 + θ_4) (5)

dz = [l_1 + $l_2 s\theta_2$ + l_3 s (θ_2 + θ_3)] + l_4 s(θ_2 + θ_3 + θ_4), (6)

όπου dx, dy, dz ονομάζονται παγκόσμιες συντεταγμένες της αρπαγής – άκρου.

Επιπροσθέτως, ο προσανατολισμός του άκρου ορίζεται ως

$$\varphi = \theta_2 + \theta_3 + \theta_4 \quad (7)$$

Οι παράμετροι DH εξαρτώνται από τη μορφολογία και τη δομή του κάθε ρομπότ. Για διαφορετικούς βραχίονες, οι κινηματικές εξισώσεις αλλάζουν. Επιπροσθέτως, οι κινηματικές εξισώσεις του βραχίονα που βασίζονταιστη μέθοδοDH μπορούν να οδηγήσουν σε ιδιαιτερότητες, κάνοντας τις εξισώσεις δύσκολες στην επίλυση ή καθιστώντας τις άλυτες πολλές φορές. Έτσι, όταν δύο άξονες δύο αρθρώσεων είναι παράλληλοιμεταξύ τους, δεν έχουμε καλό ορισμό. Στην περίπτωση αυτή, η μέθοδος DH έχει ένα singularitypoint, όπου μια μικρή αλλαγή στις συντεταγμένες των παράλληλων αρθρώσεων μπορούν να δημιουργήσουν μια τεράστια δυσμορφία στην αναπαράσταση των DH συντεταγμένων της σχετικής θέσης. Στη συνέχεια, παρουσιάζεται μία εναλλακτική μέθοδος επίλυσης του ευθέως κινηματικού προβλήματος.

5.2.2 Γινόμενο των εκθετικών

Εκτός από τη μέθοδο DH, μία άλλη μέθοδος είναι το επονομαζόμενογινόμενο των εκθετικών. Θεωρεί ότι η κίνηση κάθε άρθρωσης προκαλείται από μια στροφή σε σχέση με τον άξονα της άρθρωσηςκαι,έτσι, μπορεί να προκύψει μια πιο γεωμετρική αναπαράσταση των κινηματικών εξισώσεων. Συμβολίζεται ως ξ η στροφή και η εμπρόσθια κινηματική δίνεται από:

$$g_{st}(\theta) = e^{\xi 1 \theta \iota} e^{\xi 2 \theta 2} \dots e^{\xi n \theta n} g_{st}(0) \qquad (8)$$

Η παραπάνω εξίσωση καλείται γινόμενο των εκθετικών για την εμπρόσθια κινηματική του ρομπότ, όπου $g_{st}(\theta)$ είναι η τελική ρύθμιση του ρομπότ και ο $e^{\xi n \theta n}$ ένας πίνακας εκθετικών ορισμένος ως:

$$e^{\xi n \theta n} = \begin{bmatrix} e^{\theta n \omega n}\left(I - e^{\theta n \omega n}\right)(\omega_n x\, v_n) + \theta_n \omega_n \omega_n^T \theta_n \\ 1 \end{bmatrix} \qquad (9)$$

Για μια πρισματική άρθρωση η στροφή ξ δίνεται από τον τύπο:

$$\xi_i = \begin{bmatrix} v_i \\ 0 \end{bmatrix}$$

Ενώ για μία περιστροφική άρθρωση δίνεται από τον τύπο:

$$\xi_i = \begin{bmatrix} -\omega_i \; x \; q_i \\ \omega_i \end{bmatrix}$$

Όπου το ω_i είναι ένα μοναδιαίο διάνυσμα στην κατεύθυνση του άξονα στροφής, το q_i ένα οποιοδήποτε σημείο αυτού του άξονα και το v_i ένα μοναδιαίο διάνυσμα με κατεύθυνση αυτή της μεταφοράς. Στην περίπτωση αυτή, η στροφή ξ των διαφόρων συνδέσμων του ρομπότ δίνεται από:

$$\xi_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \xi_2 = \begin{bmatrix} L_1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \xi_3 = \begin{bmatrix} L_1 \\ 0 \\ -L_2 \\ 0 \\ -1 \\ 0 \end{bmatrix} \xi_4 = \begin{bmatrix} L_1 \\ 0 \\ -(L_2 + L_3) \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

Επιπλέον, το $g_{st}\ (0)$ αποτελεί μια αρχική ρύθμιση του ρομπότ και δίνεται από:

$$\mathrm{g_{st}}\ (0) = \begin{bmatrix} 1 & 0 & 0 & \mathrm{l}_2 + \mathrm{l}_3 + \mathrm{l}_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{l}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ο χάρτης της εμπρόσθιας κινηματικής του βραχίονα έχει τη μορφή:

$$g_{st}\ (\theta) = e^{\xi 1 \theta \iota} e^{\xi 2 \theta 2} e^{\xi 3 \theta 3} e^{\xi 4 \theta 4} g_{st}\ (0) \begin{bmatrix} R(\theta) & p(\theta) \\ 0 & 1 \end{bmatrix} \qquad (10)$$

Όπου:

$$R(\theta) = \begin{bmatrix} cos(\theta_2 + \theta_3 + \theta_4)cos\theta_1 & -sin\theta_1 & -sin(\theta_2 + \theta_3 + \theta_4)cos\theta_1 \\ cos(\theta_2 + \theta_3 + \theta_4)sin\theta_1 & cos\theta_1 & -sin(\theta_2 + \theta_3 + \theta_4)sin\theta_1 \\ sin(\theta_2 + \theta_3 + \theta_4) & 0 & cos(\theta_2 + \theta_3 + \theta_4) \end{bmatrix} \quad (11)$$

$$p(\theta) = \begin{bmatrix} cos\theta_1(l_3\ (\cos(\theta_2 + \theta_3) + l_2\ cos\theta_2 + l_4\ cos(\theta_2 + \theta_3 + \theta_4)) \\ sin\theta_1(l_3\ (\cos(\theta_2 + \theta_3) + l_2\ cos\theta_2 + l_4\ cos(\theta_2 + \theta_3 + \theta_4)) \\ l_1 + l_3\ (\cos(\theta_2 + \theta_3) + l_2\ sin\theta_2 + l_4\ sin(\theta_2 + \theta_3 + \theta_4) \end{bmatrix} \quad (12)$$

Είναι προφανές πως η εξίσωση (12) είναι η ίδια με τις (4)–(6). Επομένως, το κινηματικό μοντέλο που εξάγεταιμε τη μέθοδο DH είναι το ίδιο με αυτό του γινομένου των εκθετικών.

5.3 Αντίστροφο κινηματικό πρόβλημα – ανάλυση

Σύμφωνα με τα παραπάνω, υπάρχουν ως δεδομένα 4 μη γραμμικές εξισώσεις με 4 αγνώστους. Για να λυθούν αλγεβρικά αυτές οιεξισώσεις, γνωστές και ως αντίστροφη κινηματική, πρέπει να προσδιοριστούν οι μεταβλητές των αρθρώσεων $\theta_1, \theta_2,\ \theta_3, \theta_4$ για μια δεδομένη θέση του άκρου –

αρπαγής [dx, dy, dz] και προσανατολισμού φ. Στις εξισώσεις (4)–(7) εφαρμόζεται παραγώγιση, ύψωση στο τετράγωνο, πρόσθεση, και εφαρμογή τριγωνομετρικών εξισώσεων. Τα αποτελέσματα είναι:

$$\theta_1 = tan^{-1}(\frac{dy}{dx}) \quad (13)$$

$$\theta_2 = tan^{-1}\left(c \pm \sqrt{r^2 - c^2}\right) - tan^{-1}(a, b) \quad (14)$$

$$\theta_3 = cos^{-1}\left(\frac{A^2 + B^2 + C^2 - l_2^2 - l_3^2}{2l_2 l_3}\right) \quad (15)$$

Όπου:

- α = $l_3 sin\theta_3$
- b = $l_2 + l_3 cos\theta_3$
- c = dz - $l_1 - l_4 sin\varphi$
- r = $\sqrt{a^2 + b^2}$

Επομένως:

A = dx - $l_4 c\theta_1 c\varphi$

B = dy - $l_4 s\theta_1 c\varphi$

C = dz - $l_1 - l_4 s\varphi$

Τέλος, γίνεται η εύρεση τηςθ_4 $από$ $τον$ $προσανατολισμό$ φ $του$ $άκρου - αρπάγης$.

$$\theta_4 = \varphi - \theta_2 - \theta_3 \qquad (16)$$

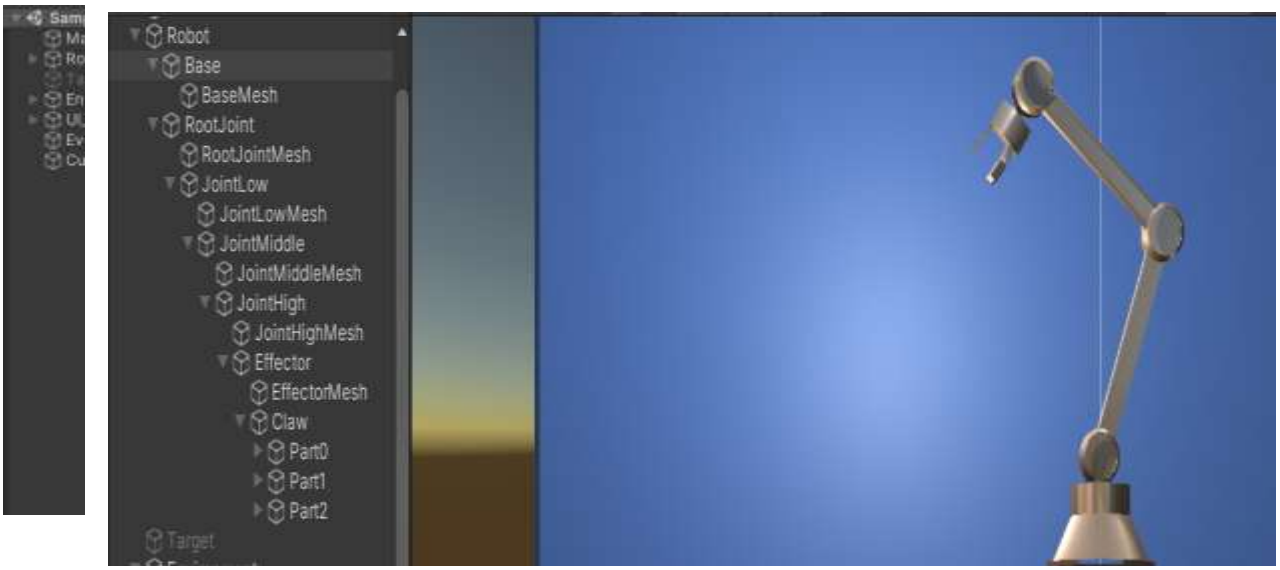
6 Σχεδίαση και Προσομοίωση του Ρομποτικού Βραχίονα

6.1 Το σχέδιο του Ρομποτικού Βραχίονα και το περιβάλλον της Unity

Το περιβάλλον αποτελείται από τα παρακάτω GameObjects(αντικείμενα) :

- Την κάμερα
- Robot- Το μοντέλο του ρομποτικού βραχίονα

- Target - Μια σφαίρα με τη βοήθεια της οποίας επιλύεται το αντίστροφο κινηματικό πρόβλημα
- Environment – Ο χώρος εργασίας του βραχίονα
- UI_Canvas– Το παράθυρο του μενού από όπου ο χρήστης θα ελέγχει την κίνηση του βραχίονα
- Cube – Το αντικείμενο το οποίο θα πρέπει να μετακινεί



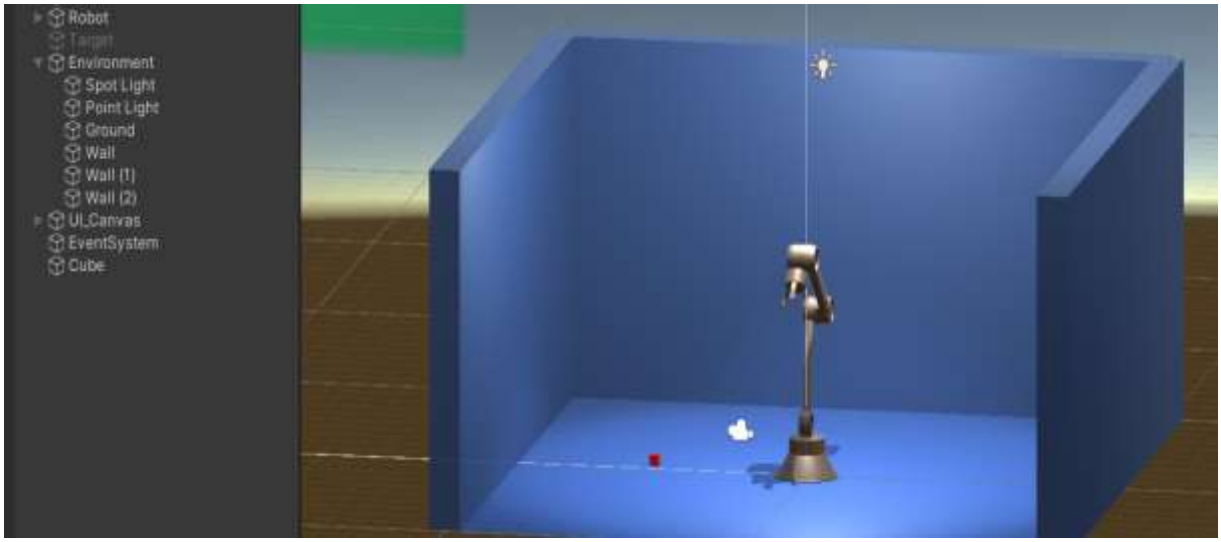
6.1.1 Το μοντέλο του Ρομποτικού Βραχίονα

Τομοντέλο του βραχίονα αποτελείται από :

- Μία βάση (Base), σταθερή στο έδαφος.
- Την πρώτη άρθρωση (RootJoint). Η άρθρωση αυτή μπορεί να περιστρέφεται γύρω από τον άξονα των y.
- Τη δεύτερη άρθρωση (JointLow), η οποία περιστρέφεται ως προς τον άξονα των x.
- Την τρίτη άρθρωση (JointMiddle), η οποία περιστρέφεται ως προς τον άξονα των x.
- Την τέταρτη άρθρωση (Joint High), η οποία περιστρέφεται ως προς τον άξονα των x.
- Την κυλινδρική βάση της αρπάγης (Effector).
- Τοσύστημα των τριών μελών της αρπάγης (Claw).
- Τα τρία μέλη της αρπάγης (Part0, Part1, Part2).

Τα meshes είναι πλέγματα γύρω από τα σχήματα. Πιο συγκεκριμένα, αποτελούνται από κορυφές και ακμές και περιγράφουν το σχήμα ενός 3D GameObject (Αντικειμένου).

6.1.2 Ο χώρος εργασίας του Ρομποτικού Βραχίονα



Ο χώρος εργασίας του ρομποτικού βραχίονα αποτελείται από :

- SpotLight–Το φως διαχέεται από μια καθορισμένη θέση αλλά και έχει συγκεκριμένο εύρος. Περιορίζεται σε μια γωνία, με αποτέλεσμα ο φωτισμός να έχει σχήμα κώνου. Το φως μειώνεται σταδιακά από την κορυφή του κώνου προς τις άκρες. Φωτίζει μόνο το μέρος της σκηνής όπου βρίσκεται το μοντέλο του ρομποτικού βραχίονα.
- PointLight–Η πηγή του είναι ένα σημείο του χώρου το οποίο στέλνει φως προς όλες τις κατευθύνσεις. Η ένταση μειώνεται όσο η απόσταση από το φως αυξάνεται.
- Ground – Το έδαφος. Είναι μία επίπεδη επιφάνεια η οποία διαχωρίζει το χώρο σε δύο μισά.
- Walls – Τρεις κύβοι οι οποίοι διαμορφώνουν τον χώρο στον οποίο βρίσκεται ο βραχίονας και ο κύβος.

6.1.3 Σχεδίαση διεπαφής χρήστη (UserInterface)

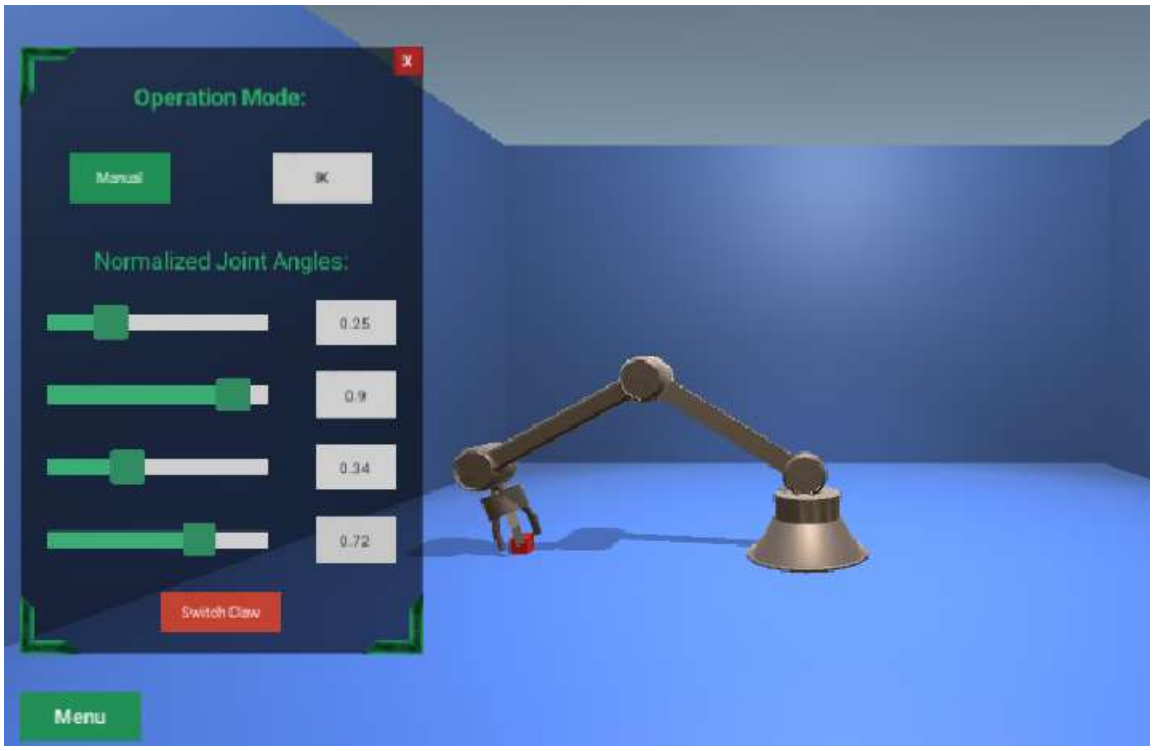
Για τη δημιουργία του UserInterface, η Unityδιαθέτει μια εργαλειοθήκη που επιτρέπει τη σχεδίαση διεπαφών χρήστη για παιχνίδια και εφαρμογές. Δίνει την δυνατότητα προσθήκης πάνελ, κουμπιών, μπάρας κύλισης, πεδίων εισαγωγής τιμών κ.α.

6.2 Περιγραφή λειτουργίας – Προσομοίωση

Ο χρήστης, πατώντας το κουμπί “Menu”, μπορεί να εμφανίσει το πάνελ για το μενού. Στη συνέχεια, μπορεί να επιλέξει το κουμπί “Manual” ή “IK” για να κινήσει τον βραχίονα με το ευθύ κινηματικό ή με το αντίστροφο κινηματικό, αντίστοιχα.

Εάν επιλέξει το κουμπί του ευθέως κινηματικού, εμφανίζονται τέσσερις μπάρες κύλισης και τέσσερα πεδία εισαγωγής τιμών μέσω των οποίων μπορεί να ορίσει τις τιμές των αρθρώσεων του ρομποτικού βραχίονα. Η πρώτη μπάρα κύλισης ή το πρώτο πεδίο εισαγωγής καθορίζουν τη γωνία της πρώτης άρθρωσης. Η δεύτερη μπάρα κύλισης ή το δεύτερο πεδίο εισαγωγής καθορίζουν τη γωνία της δεύτερης άρθρωσης κ.ο.κ.

Αφού έχει βρει την επιθυμητή θέση του ρομποτικού βραχίονα προκειμένου να πιάσει τον κύβο, μπορεί να πατήσει το κουμπί “SwitchClaw” για να κλείσει η αρπάγη.

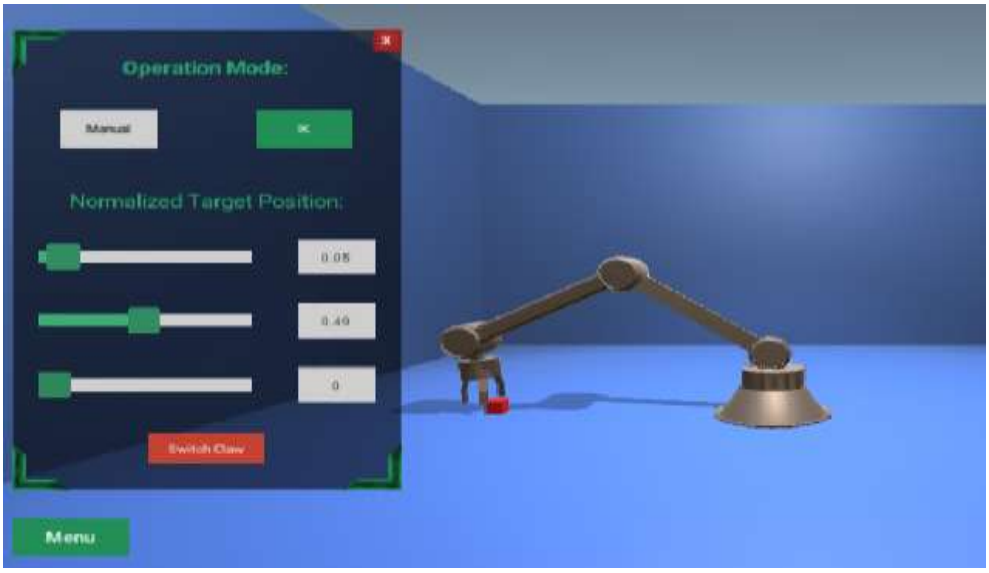

Operation Mode:
Manual
IK
Normalized Joint Angles:
0.25
0.9
0.34
0.72
Switch Claw
Menu

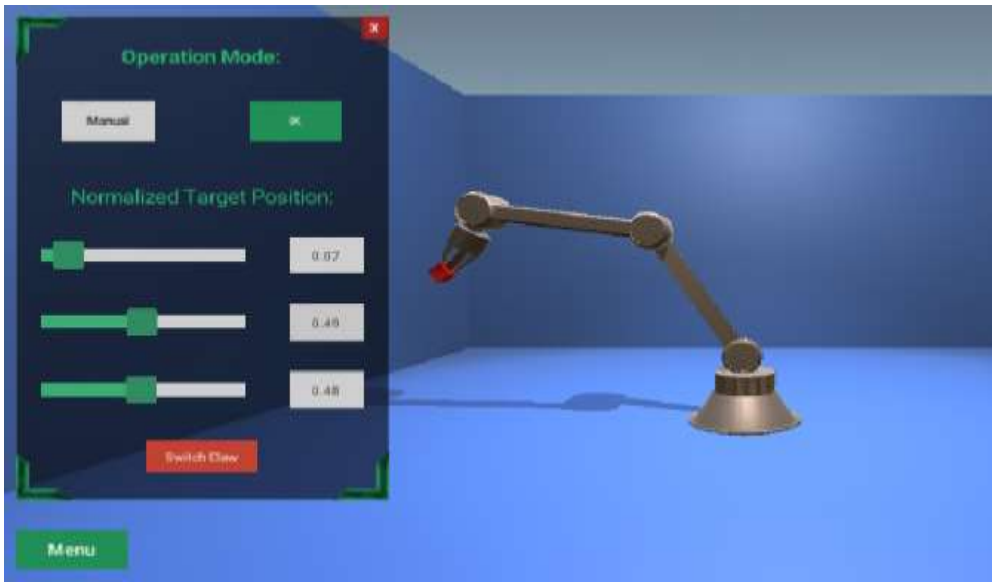

Operation Mode:
Manual
IK
Normalized Joint Angles:
0.25
0.69
0.34
0.72
Switch Claw
Menu

Εάν επιλέξει το κουμπί για το αντίστροφο κινηματικό, θα μπορεί να μετακινεί μία κόκκινη σφαίρα στο σημείο στο οποίο επιθυμεί να φτάσει ο βραχίονας. Ο βραχίονας ακολουθεί τη σφαίρα.

Με την επιλογή της κίνησης με αντίστροφο κινηματικό, εμφανίζονται τρείς μπάρες κύλισης και τρία πεδία εισαγωγής τιμών, αντίστοιχα. Κάθε μπάρα κύλισης ή κάθε πεδίο εισαγωγής αντιπροσωπεύει την κίνηση της σφαίρας. Η πρώτη μπάρα κύλισης ή το πρώτο πεδίο εισαγωγής αντιπροσωπεύει την κίνηση της σφαίρας κατά μήκος του άξονα x. Η δεύτερη μπάρα κύλισης ή το δεύτερο πεδίο εισαγωγής αντιπροσωπεύει την κίνηση της σφαίρας κατά μήκος του άξονα z. Ητρίτη μπάρα κύλισης ή το τρίτο πεδίο εισαγωγής αντιπροσωπεύει την κίνηση της σφαίρας κατά μήκος του άξονα y.

Θα πρέπει να τοποθετήσει την σφαίρα πολύ κοντά στον κύβο. Με αυτό τον τρόπο, ο βραχίονας θα πλησιάσει τον κύβο και τότε θα πρέπει να πατήσει το κουμπί “SwitchClaw” για να κλείσει η αρπάγη.





7 Ο κώδικας

7.1 Κώδικας Δημιουργίας του Μενού

Ακολουθεί ο κώδικας του Μενού “UIController”. Σε αυτόν ενεργοποιείται/απενεργοποιείται το πάνελ του μενού, ενεργοποιούνται/απενεργοποιούνται οι μπάρες κύλισης, τα πεδία εισαγωγής τιμών, οι περιγραφές του μενού και τα κουμπιά. Επιπλέον, μετατρέπονται οι τιμές των πεδίων εισαγωγής τιμών σε αντίστοιχες τιμές στις μπάρες κύλισης και το αντίστροφο. Τέλος, καλούνται οι κώδικες για το ευθύ ή το αντίστροφο κινηματικό, ανάλογα με το ποιο κουμπί είναι ενεργοποιημένο.

```
public class UIController : MonoBehaviour
{
    public Color activeTabColor;
    public Color disabledTabColor;
    [Range(1, 5)]
    public int inputFieldLimit = 4;

    public RobotControlerManual manualScript;
    public RobotControllerIK ikScript;

    public GameObject popupPanel;

    public Button manualButton;
    public Button ikButton;


    public GameObject description;
    public Slider slider0;
    public Slider slider1;
    public Slider slider2;
    public Slider slider3;
    public InputField inputField0;
    public InputField inputField1;
    public InputField inputField2;
    public InputField inputField3;
    public GameObject grabButton;

    Unity Message | 0 references
    private void Start()
    {
        popupPanel.SetActive(false);
    }
```

```
public void SwitchPopupPanel()
{
    popupPanel.SetActive(!popupPanel.activeInHierarchy);
}

0 references
public void ModeButtonPressed(Button button)
{
    ModeButtonsClearColor();
    DeactivateScripts();

    button.image.color = activeTabColor;
    button.transform.GetChild(0).GetComponent<Text>().color = Color.white;

    if (button == manualButton)
    {
        ManualModeActivateUI();
        Slider[] s = { slider0, slider1, slider2, slider3 };
        manualScript.Activate(s);
    }
    else if (button == ikButton)
    {
        IkModeActivateUI();
        Slider[] s = { slider0, slider1, slider2 };
        ikScript.Activate(s);
    }

}

//Slider values to input field values
7 references
public void SliderValueChanged(Slider slider)
{
    if (slider == slider0)
    {
        inputField0.text = slider.value.ToString();
    }
    else if (slider == slider1)
    {
        inputField1.text = slider.value.ToString();
    }
    else if (slider == slider2)
    {
        inputField2.text = slider.value.ToString();
    }
    else if (slider == slider3)
    {
        inputField3.text = slider.value.ToString();
    }
}
```

```
//Input field values to slider values
0 references
public void InputFieldValueChanged(InputField field)
{
    if (field == inputField0)
    {
        field.text = Mathf.Clamp01(float.Parse(field.text)).ToString();
        slider0.value = float.Parse(field.text);
    }
    else if (field == inputField1)
    {
        field.text = Mathf.Clamp01(float.Parse(field.text)).ToString();
        slider1.value = float.Parse(field.text);
    }
    else if (field == inputField2)
    {
        field.text = Mathf.Clamp01(float.Parse(field.text)).ToString();
        slider2.value = float.Parse(field.text);
    }
    else if (field == inputField3)
    {
        field.text = Mathf.Clamp01(float.Parse(field.text)).ToString();
        slider3.value = float.Parse(field.text);
    }
}

public void InputFieldLimitLength(InputField field)
{
    if (field.text.Length <= inputFieldLimit)
    {
        return;
    }

    field.text = field.text.Substring(0, inputFieldLimit); //displays the first 4 characters
}

1 reference
private void ModeButtonsClearColor()
{
    manualButton.image.color = disabledTabColor;
    ikButton.image.color = disabledTabColor;


    manualButton.transform.GetChild(0).GetComponent<Text>().color = Color.black;
    ikButton.transform.GetChild(0).GetComponent<Text>().color = Color.black;

}
```

```
//activate sliders,sliders values to input fields and grab Button
1 reference
private void ManualModeActivateUI()
{
    ClearContentUI();

    SetDescriptionText("Normalized Joint Angles:");
    description.SetActive(true);

    slider0.transform.parent.gameObject.SetActive(true);
    slider1.transform.parent.gameObject.SetActive(true);
    slider2.transform.parent.gameObject.SetActive(true);
    slider3.transform.parent.gameObject.SetActive(true);
    SliderValueChanged(slider0);
    SliderValueChanged(slider1);
    SliderValueChanged(slider2);
    SliderValueChanged(slider3);

    grabButton.SetActive(true);
}

1 reference
private void IkModeActivateUI()
{
    ClearContentUI();

    SetDescriptionText("Normalized Target Position:");
    description.SetActive(true);

    slider0.transform.parent.gameObject.SetActive(true);
    slider1.transform.parent.gameObject.SetActive(true);
    slider2.transform.parent.gameObject.SetActive(true);
    SliderValueChanged(slider0);
    SliderValueChanged(slider1);
    SliderValueChanged(slider2);

    grabButton.SetActive(true);
}
```

```
private void ClearContentUI()
{
    description.SetActive(false);

    slider0.transform.parent.gameObject.SetActive(false);
    slider1.transform.parent.gameObject.SetActive(false);
    slider2.transform.parent.gameObject.SetActive(false);
    slider3.transform.parent.gameObject.SetActive(false);

    grabButton.SetActive(false);
}

1 reference
private void DeactivateScripts()
{
    manualScript.Deactivate();
    ikScript.Deactivate();
}

2 references
private void SetDescriptionText(string s)
{
    description.transform.GetChild(0).GetComponent<Text>().text = s;
}
```

7.2 Κώδικας Ευθέως Κινηματικού Προβλήματος

Οκώδικας του ευθέως κινηματικού προβλήματος “RobotControlerManual” καταχωρεί τις γωνίες των αρθρώσεων που υπάρχουν κατά την έναρξη λειτουργίας της εφαρμογής στα αντίστοιχα πεδία εισαγωγής τιμών ή στις μπάρες κύλισης του μενού. Στη συνέχεια, ανανεώνει τις μοίρες των γωνιών των αρθρώσεων σύμφωνα με τις τιμές που ορίζει ο χρήστης.

Οι μοίρες των γωνιών Eulertων αρθρώσεων προκειμένου να γίνεται η εύρεση της νέας θέσης του βραχίονα κάθε φορά, πρέπει να μετατρέπονται σε τετραδόνια (Quaternions) για να μη χάνεται κάποιος προσανατολισμός. Το πρόβλημα απώλειας κάποιου προσανατολισμού είναι γνωστό ως “GimbalLock”.

Πρόκειται για την απώλεια ενός βαθμού ελευθερίας σε έναν τρισδιάστατο μηχανισμό και συμβαίνει όταν οι δύο από τους τρείς άξονες περιστροφής οδηγούνται σε μία παράλληλη διαμόρφωση, <<κλειδώνοντας>> ολόκληρο το σύστημα σε περιστροφή δύο διαστάσεων.

Ακολουθεί ο κώδικας.

```
public class RobotControlerManual : MonoBehaviour
{
    public RobotJoint[] joints;

    private bool _activated;
    private Slider[] _sliders;


    Unity Message | 0 references
    void Start()
    {
        _sliders = new Slider[4];

    }

    Unity Message | 0 references
    void Update()
    {
        if (!_activated)
        {
            return;
        }

        for (int i = 0; i < joints.Length; i++)
        {
            joints[i].SetRotationAngle(SliderValueToDegrees(i));

        }

    }
```

```
//Activate and initialize the sliders values
1 reference
public void Activate(Slider[] s)
{
    for (int i = 0; i < _sliders.Length; i++)
    {
        _sliders[i] = s[i];
    }

    for (int i = 0; i < _sliders.Length; i++)
    {
        _sliders[i].value = DegreesToSliderValue(i);

    }

    _activated = true;

}

//Deactivate the manual
1 reference
public void Deactivate()
{
    _activated = false;
}
```

```
//Return the degree depending on the slider value.
1 reference
private float SliderValueToDegrees(int index)
{
    if (index == 0)
    {
        return _sliders[index].value * 360.0f;
    }

    return (_sliders[index].value * (joints[index].MaxAngle - joints[index].MinAngle)) + joints[index].MinAngle;
}

//Returns a value for sliders depending on the degree of Joint
1 reference
private float DegreesToSliderValue(int index)
{
    //Debug.Log(joints[index].GetRotationAngle());
    if (index == 0)
    {

        return joints[index].GetRotationAngle();
    }

    return (joints[index].GetRotationAngle() - joints[index].MinAngle) / (joints[index].MaxAngle - joints[index].MinAngle);

}
```

Οισυναρτήσεις “GetRotationAngle”, “SetRotationAngle”,οιοποίες καλούνται στον κώδικα του ευθέος αλλά και του αντίστροφου κινηματικού ανήκουν στην κλάση “RobotJoint”. Ο κώδικάς του δίνεται παρακάτω.

```
public enum RotationAxis
{
    XPos,
    XNeg,
    YPos,
    YNeg,
    ZPos,
    ZNeg
}

Unity Script | 2 references
public class RobotJoint : MonoBehaviour
{
    [SerializeField]
    private RotationAxis rotationAxis = default;
    [SerializeField]
    private float minAngle = default;
    [SerializeField]
    private float maxAngle = default;

    5 references
    public float MinAngle
    {
        get { return minAngle; }
        private set {}
    }
    3 references
    public float MaxAngle
    {
        get { return maxAngle; }
        private set {}
    }

public Vector3 AxisVector
{
    get { return _axisVector; }
    private set {}
}
1 reference
public Vector3 RestPos
{
    get { return _restPos; }
    private set {}

}

private Vector3 _axisVector;
private Vector3 _restPos;


Unity Message | 0 references
void Start()
{
    _restPos = this.transform.localPosition;
    _axisVector = RotationAxisToVector(rotationAxis);
}
```

```
//Retun the angle value
3 references
public float GetRotationAngle()
{
    float ret;
    Vector3 localEuler = QuatToEuler(this.transform.localRotation);

    switch (rotationAxis)
    {
        case RotationAxis.XPos:
            ret = localEuler.x;
            break;
        case RotationAxis.XNeg:
            ret = -localEuler.x;
            break;
        case RotationAxis.YPos:
            ret = localEuler.y;
            break;
        case RotationAxis.YNeg:
            ret = -localEuler.y;
            break;
        case RotationAxis.ZPos:
            ret = localEuler.z;
            break;
        case RotationAxis.ZNeg:
            ret = -localEuler.z;
            break;
        default:
            ret = 0.0f;
            break;

    }

    return ret;

}

public Vector3 QuatToEuler(Quaternion q)
{
    Vector3 angles;


    float singularityTest = q.x * q.y + q.z * q.w;
    if (singularityTest > 0.499f)
    {
        angles.x = 0.0f;
        angles.y = 2 * Mathf.Atan2(q.x, q.w);
        angles.z = Mathf.PI / 2.0f;
    }
    else if (singularityTest < -0.499f)
    {
        angles.x = 0.0f;
        angles.y = -2 * Mathf.Atan2(q.x, q.w);
        angles.z = -Mathf.PI / 2.0f;
    }
    else
    {
        angles.x = Mathf.Atan2(2.0f * (q.x * q.w - q.y * q.z), 1.0f - 2.0f * (q.x * q.x - q.z * q.z));
        angles.y = Mathf.Atan2(2.0f * (q.y * q.w - q.x * q.z), 1.0f - 2.0f * (q.y * q.y - q.z * q.z));
        angles.z = Mathf.Asin(2.0f * singularityTest);
    }

    angles *= 180.0f / Mathf.PI;

    return angles;
}
```

```
//Set the angle depending on the angle
2 references
public void SetRotationAngle(float angle)
{
    float angleRad = angle * Mathf.PI / 180.0f;
    float quatW = Mathf.Cos(angleRad / 2.0f);
    Vector3 quatVector = _axisVector * Mathf.Sin(angleRad / 2.0f);

    Quaternion newRotation = new Quaternion(quatVector.x, quatVector.y, quatVector.z, quatW);

    this.transform.localRotation = newRotation;
}


Vector3 RotationAxisToVector(RotationAxis axis)
{
    Vector3 ret;


    switch (axis)
    {
        case RotationAxis.XPos:
            ret = Vector3.right;
            break;
        case RotationAxis.XNeg:
            ret = Vector3.left;
            break;
        case RotationAxis.YPos:
            ret = Vector3.up;
            break;
        case RotationAxis.YNeg:
            ret = Vector3.down;
            break;
        case RotationAxis.ZPos:
            ret = Vector3.forward;
            break;
        case RotationAxis.ZNeg:
            ret = Vector3.back;
            break;
        default:
            ret = Vector3.zero;
            break;

    }

    return ret;

}
```

7.3 Κώδικας Αντίστροφου Κινηματικού Προβλήματος

Ο κώδικας του αντίστροφου κινηματικού προβλήματος “RobotControllerIK”, καταχωρεί τις θέσεις τησσφαίρας που υπάρχουν κατά την έναρξη λειτουργίας της εφαρμογής στα αντίστοιχα πεδία εισαγωγής τιμών ή στις μπάρες κύλισης του μενού. Στη συνέχεια, ανανεώνει τις θέσεις της σφαίρας σύμφωνα με τις τιμές που ορίζει ο χρήστης.

Ο βραχίονας ακολουθεί τη σφαίρα με την εφαρμογή του αλγορίθμου κατάβασης κλίσης “gradientdescent”.Πρόκειται για έναν αλγόριθμο επαναληπτικής βελτιστοποίησης για την εύρεση τοπικού ελαχίστου μιας συνάρτησης. Με αυτόν, πραγματοποιούνται επαναλαμβανόμενα βήματα προς την αντίθετη κατεύθυνση της κλίσης, δηλαδή προς την κατεύθυνση της πιο απότομης πλαγιάς. Το μέγεθος των βημάτων καθορίζεται από τον συντελεστή “learningrate”- βήμα κατάβασης.

Ο αλγόριθμος περιγράφεται από την ακόλουθη εξίσωση :$b = a - \gamma * \nabla f(a, b)$

- Το a αντιπροσωπεύει τη γωνία της αρχικής θέσης της άρθρωσης.
- Το b αντιπροσωπεύει τη γωνία της νέας θέσης της άρθρωσης, αφού έχει αυξηθεί κατά μία σταθερά.
- Το γ αντιπροσωπεύει το βήμα κατάβασης.

Τέλος, γίνεται έλεγχος της απόστασης του ρομποτικού βραχίονα από την σφαίρα. Αν είναι αρκετά κοντά, ο βραχίονας σταματάει και οι αρθρώσεις παίρνουν την τελική τους μορφή.

Ακολουθεί ο κώδικας.

```
public class RobotControllerIK : MonoBehaviour
{
    public Transform target;

    public RobotJoint[] joints;
    public float samplingDistance;
    [Range(0.01f, 1.0f)]
    public float distanceThreshold;
    public float learningRate;

    private bool _activated;
    private Slider[] _sliders;

    Unity Message | 0 references
    void Start()
    {
        _sliders = new Slider[3];
    }

    Unity Message | 0 references
    void Update()
    {
        if (!_activated)
        {
            return;
        }
```

```
    Vector3 targetPosition = new Vector3(-SliderValueToCoord(0), SliderValueToCoord(2), SliderValueToCoord(1));

    target.position = targetPosition;

    InverseKinematics(target.position);
}

1 reference
public void Activate(Slider[] s)
{
    target.gameObject.SetActive(true);

    for (int i = 0; i < _sliders.Length; i++)
    {
        _sliders[i] = s[i];
    }

    Vector3 sliderValues = CoordsToSliderValues(target.position);
    _sliders[0].value = sliderValues.x;
    _sliders[1].value = sliderValues.z;
    _sliders[2].value = sliderValues.y;
    _activated = true;
}

1 reference
public void Deactivate()
{
    _activated = false;
    target.gameObject.SetActive(false);
}

  //Find the distance between target and current position
  3 references
  public float DistanceToTarget(Vector3 target, float[] systemAngles)
  {
      Vector3 currentPosition = ForwardKinematics(systemAngles);
      return Vector3.Distance(currentPosition, target);

  }

  1 reference
  public float PartialGradient(Vector3 target, float[] angles, int i)
  {
      float angle = angles[i];

      float f_x = DistanceToTarget(target, angles);
      angles[i] += samplingDistance;
      float f_xdx = DistanceToTarget(target, angles);
      float gradient = (f_xdx - f_x) / samplingDistance;

      angles[i] = angle;
      return gradient;

  }
```

```
public void InverseKinematics(Vector3 target)
{
    float[] angles = GetJointAngles();


    for (int i = (joints.Length - 1); i >= 0; i--)
    {
        if (DistanceToTarget(target, angles) < distanceThreshold)
        {
            SetJointAngles(angles);

            return;
        }

        float gradient = PartialGradient(target, angles, i);

        angles[i] -= learningRate * gradient;
      angles[i] = Mathf.Clamp(angles[i], joints[i].MinAngle, joints[i].MaxAngle);
    }

    SetJointAngles(angles);

}
```

```
public Vector3 ForwardKinematics(float[] angles)
{
    Vector3 prevPoint = joints[0].transform.position;
    Vector3 nextPoint = Vector3.zero;
    Vector3 rotationAxis = joints[0].AxisVector;
    Quaternion rotation = Quaternion.identity;

    for (int i = 1; i < joints.Length; i++)
    {
        rotationAxis = joints[i - 1].AxisVector;

        rotation *= Quaternion.AngleAxis(angles[i - 1], rotationAxis);

        nextPoint = prevPoint + rotation * joints[i].RestPos;

        prevPoint = nextPoint;
    }

    return prevPoint;
}
```

```
2 references
void SetJointAngles(float[] angles)
{
    for (int i = 0; i < joints.Length; i++)
    {
        joints[i].SetRotationAngle(angles[i]);

    }
}
```

```
//Get the angles of the joints
1 reference
float[] GetJointAngles()
{
    float[] angles = new float[joints.Length];

    for (int i = 0; i < joints.Length; i++)
    {
        angles[i] = joints[i].GetRotationAngle();
    }

    return angles;
}

//Change the coords depending of slider values
3 references
private float SliderValueToCoord(int index)
{
    if (index == 2)
    {
        return _sliders[index].value * 6.0f + 1.0f;

    }

    return _sliders[index].value * 12.0f - 6.0f;

}

//Change the values of slider depending on coords
1 reference
private Vector3 CoordsToSliderValues(Vector3 coords)
{
    Vector3 ret;

     ret.x = -(coords.x + 6.0f) / 12.0f;
     ret.y = (coords.y - 1.0f) / 6.0f;
     ret.z = (coords.z + 6.0f) / 12.0f;


    return ret;
}
```

7.4 Κώδικας Εργαλείου Τελικής Δράσης

Στον κώδικα γίνονται έλεγχοι για την κατάσταση της αρπάγης. Ελέγχεται, δηλαδή, αν είναι ανοιχτή ή κλειστή. Με αυτόν τον τρόπο κλείνει ή ανοίγει, αντίστοιχα, με το πάτημα του κουμπιού από το μενού. Η αρπάγη κλείνει ή ανοίγει με τη συνάρτηση “Rotate”.

Επιπλέον, γίνονται έλεγχοι συγκρούσεων. Ο κύβος φέρει την ετικέτα “PickUpObject”. Όταν πραγματοποιείται σύγκρουση με τον κύβο, ο κύβος μετατρέπεται σε αντικείμενο με μορφή “Kinematic”, παύει, δηλαδή, να υπακούει στους νόμους της φυσικής και μετακινείται μαζί με την αρπάγη όσο αυτή τον κρατάει με τη δημιουργία μιας σχέσης «Πατέρα-Παιδιού» μεταξύ τους.

Ακολουθεί ο κώδικας. Βρίσκεται στις κλάσεις “ClawController”, “ClawPart”.

```
public class ClawController : MonoBehaviour
{
    public Transform clawTransform;
    public ClawPart[] parts;
    [Range(1.0f, 100.0f)]
    public float grabSpeed = 12.0f;

    private GameObject _grabbedObject;
    private bool _shouldOpen;
    private bool _shouldClose;
    private bool _isOpen;
    private bool _isClose;

    Unity Message | 0 references
    void Start()
    {
        _isOpen = true;
        _isClose = false;
        _shouldOpen = false;
        _shouldClose = false;
    }

    Unity Message | 0 references
    void FixedUpdate()
    {
        if (_shouldOpen)
        {
            OpenClaw();
        }
        else if (_shouldClose)
        {
            CloseClaw();
        }
    }
```

```
public void SwitchGrab()
{
    if (_isOpen)
    {
        _shouldOpen = false;
        _shouldClose = true;
        _isOpen = false;
    }
    else if (_isClose)
    {
        _shouldOpen = true;
        _shouldClose = false;
        _isClose = false;
    }
    else
    {
        _shouldOpen = !_shouldOpen;
        _shouldClose = !_shouldClose;
    }
}

private void CloseClaw()
{
    bool holdingGrip = true;
    GameObject collidingObject = parts[0].CollidingObject;

    foreach(ClawPart p in parts)
    {
        holdingGrip &= p.IsColliding;

        if (p.CollidingObject != collidingObject)
        {
            collidingObject = null;
        }
    }

    if (holdingGrip)
    {
        _isClose = true;

        if (collidingObject != null)
        {
            _grabbedObject = collidingObject;
            _grabbedObject.GetComponent<Rigidbody>().isKinematic = true;
            _grabbedObject.transform.parent = clawTransform;
        }

        return;
    }

    _grabbedObject = null;

    foreach (ClawPart p in parts)
    {
        p.RotateInwards(grabSpeed * Time.fixedDeltaTime);
    }
}
```

```
private void OpenClaw()
{
    bool inRest = true;

    if (_grabbedObject != null)
    {
        _grabbedObject.transform.parent = null;
        _grabbedObject.GetComponent<Rigidbody>().isKinematic = false;
        _grabbedObject = null;
    }

    foreach (ClawPart p in parts)
    {
        if (p.IsInRestRotation())
        {
            continue;
        }

        inRest = false;

        p.RotateOutwards(grabSpeed * Time.fixedDeltaTime);
    }

    if (inRest)
    {
        _isOpen = true;
    }
}
```

```
public class ClawPart : MonoBehaviour
{
    private bool _isColliding;
    private Quaternion _restRotation;
    private bool _canHoldObject;

    1 reference
    public bool IsColliding
    {
        get { return _isColliding; }
        private set {}
    }

    7 references
    public GameObject CollidingObject
    {
        get;
        private set;

    }

    Unity Message | 0 references
    void Start()
    {
        CollidingObject = null;
        _canHoldObject = true;
        _isColliding = false;
        _restRotation = this.transform.localRotation;
    }
```

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject == this.gameObject)
    {
        return;
    }

    if (collision.gameObject.GetComponent<ClawPart>() != null)
    {
        _isColliding = true;
        _canHoldObject = false;
        CollidingObject = null;
    }

    if (collision.gameObject.CompareTag("PickupObject"))
    {
        _isColliding = true;
    }

}
```

Unity Message | 0 references

```
void OnCollisionStay(Collision collision)
{
    if (collision.gameObject.CompareTag("PickupObject") && _canHoldObject)
    {
        CollidingObject = collision.gameObject;
    }

}
```

```
void OnCollisionExit(Collision collision)
{
    if (collision.gameObject == this.gameObject)
    {
        return;
    }

    if (collision.gameObject.GetComponent<ClawPart>() != null)
    {
        _isColliding = false;
        _canHoldObject = true;
    }

    if (collision.gameObject.CompareTag("PickupObject"))
    {
        if (CollidingObject != null)
        {
            _isColliding = false;
        }
        CollidingObject = null;
    }
}
```

1 reference

```
public void RotateInwards(float speed)
{
    this.transform.Rotate(Vector3.down * speed, Space.Self);
}
```

1 reference

```
public void RotateOutwards(float speed)
{
    this.transform.Rotate(Vector3.up * speed, Space.Self);
}

public bool IsInRestRotation()
{
    return Quaternion.Angle(this.transform.localRotation, _restRotation) < 0.1f ? true : false;
}
}
```

Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

[1]«AntikytheraMechanism,» [Online]. Available: https://en.wikipedia.org/wiki/Antikythera_mechanism.

[2]«Cartesian Robot,» [Online]. Available: https://gr.pinterest.com/pin/862720872340891346/.

[3]«Cylindrical Robot,» [Online]. Available: https://www.researchgate.net/publication/336702692_Data_analysis_and_force_control_of_a_6-DoF_industrial_Robot.

[4]«Delta Robot Arm,» [Online]. Available: https://www.researchgate.net/publication/326571609_The_Phantom_Delta_Robot_A_New_Device_for_Parallel_Robot_Investigations.

[5]«FAMULUS by KUKA Robotics,» [Online]. Available: https://cz.pinterest.com/pin/726135139895086730/.

[6]«Polar Robot Arm,» [Online]. Available: https://www.researchgate.net/publication/336676027_robot_arm_project.

[7]«Stanford Arm,» [Online]. Available: https://en.wikipedia.org/wiki/Victor_Scheinman.

[8]A. Aristidou and J. Lasenby. "Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver," 2009.

[9]Barinka, Lukas, and Roman Berka. ‘Inverse kinematics-basic methods.’ Web.< http://www. cescg. org/CESCG- 2002/LBarinka/paper. pdf (2002).”

[10]J. G. C. Devol, Programmed article transfer, 1954.

[11]J. Kaur and V. K. Banga, "Simulation of Robotic Arm having three link Manipulator," International Journal of Research in Engineering and Technology (IJRET), vol. 1, no. 2, March, 2012, ISSN: 2277- 4378, 2012.

[12]K. E. Clothier and Y. Shang, “A Geometric Approach for Robotic Arm Kinematics with Hardware Design, Electrical Design, and Implementation,” J. Robot., 2010.

[13]M. Leba, E. Pop, P. Vamvu, C. Corbu et C. Covaciu, «Modeling and simulation for cinematic and dynamic regime of an industrial articulated robot,» chez CSECS'09 Proceedings of the 8th WSEAS International Conference on Circuits, systems, electronics, control & signal processing, 2009.

[14]R. N. Jazar, Theory of Applied Robotics. Boston, MA: Springer US, 2010.

[15]Sagris, Dimitrios & Mitsi, Sevasti & Bouzakis, K.-D & Mansour, G.. (2011). Spatial RRR robot manipulator optimum geometric design by means of a hybrid algorithm Spatial RRR Robot Manipulator Optimum Geometric Design by Means of a Hybrid Algorithm. Romanian Review Precision Mechanics, Optics and Mechatronics.

[16]Scripting in Unity for experienced programmers [online]: https://unity.com/how-to/programming-unity

[17]The top industrial robotics arms from the GrabCAD community [online]: https://blog.grabcad.com/blog/2019/10/15/industrial-robotic-arms-from-the-grabcad-community/

[18]X. F. Ge and J. T. Jin, "The algorithm of redundant robotic kinematics based on exponential product," Appl. Mech. Mater., vol. 58-60, pp. 1902-1907, Jun. 2011.

[19]X. Xiao, Y. Li, and H. Tang, "Department of electromechanical engineering, university of Macau, Macao SAR, China," IEEE International Conference on Information and Automation (ICIA), 2013, pp. 1177-1182.

[20]Δ.Καλλιγερόπουλος, Σ.Βασιλειάδου ΑΕΙ Πειραιά Τ.Τ. Ιστορία της Τεχνολογίας & των Αυτομάτων, Σύγχρονη Εκδοτική, Αθήνα 2005

[21]Φ.Ν. Κουμπούλης, Β.Γ. Μέρτζιος Εισαγωγή στη Ρομποτική Εκδόσεις Παπασωτηρίου Αθήνα 2002

[22] "Gimbal Lock" [online] : https://en.wikipedia.org/wiki/Gimbal_lock

[23] "Gradient Descent [online] : https://en.wikipedia.org/wiki/Gradient_descent