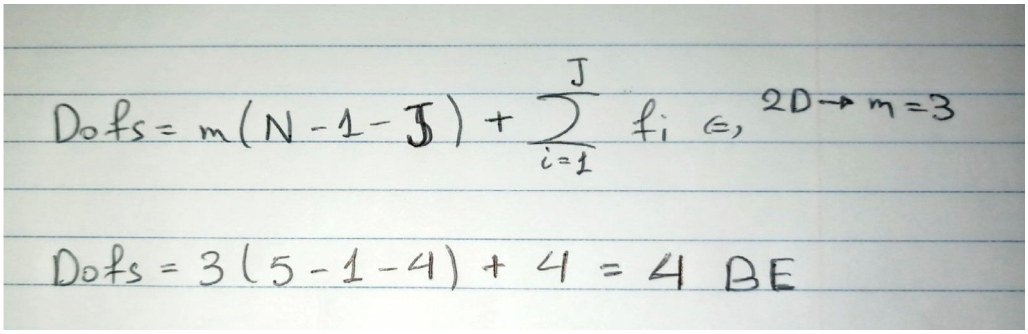
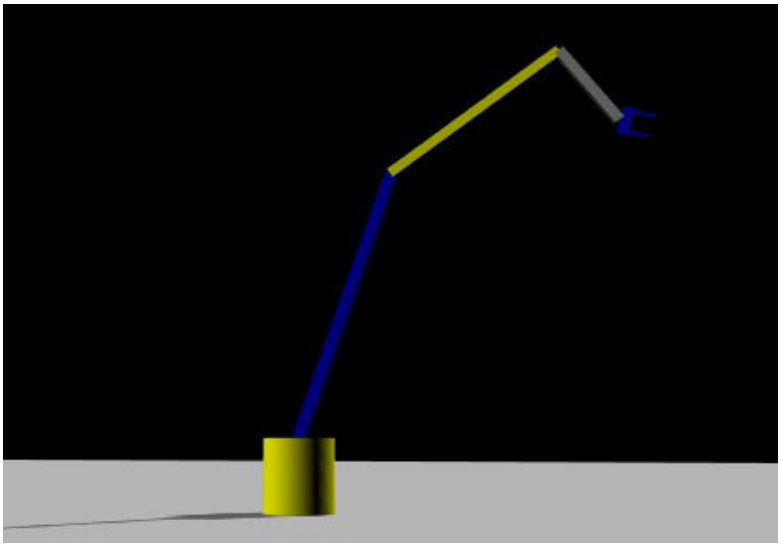
Τεχνολογίες Ευφυών Συστημάτων και Ρομποτική 2020-21
1η Ατομική Εργασία

Μπουλαφέντης Χρήστος ΑΜ:1059612 Δ’ Έτος

1) 2Δ Ρομποτικός Βραχίονας:

- Αρχικά αποφάσισα να δημιουργήσω έναν βραχίονα ο οποίος να αποτελείται από 5 ξεχωριστά μέλη και 4 αρθρώσεις που θα έχουν την παρακάτω μορφή.
  (Θεωρώ την δαγκάνα ως ένα μέλος που αποτελείται απο τρία κομμάτια και για αυτό δεν μετράω τις αρθρώσεις που περιέχει ως ξεχωριστές.)

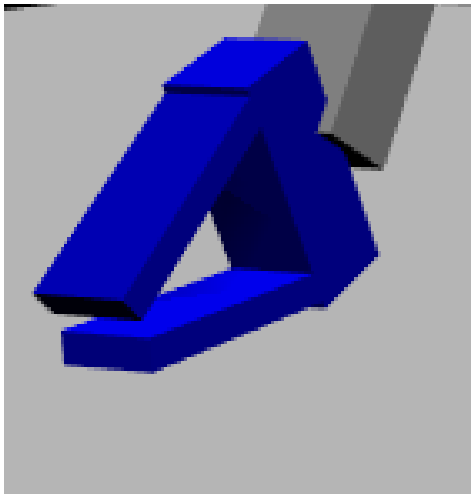




- Τα μεγέθη των μελών του παραπάνω βραχίονα έχουν τα μεγέθη:
  L0=10cm, L1=40cm, L2=30cm, L3=13,5cm και τέλος, η δαγκάνα μαζί με τα δύο grips, L4=5,5cm
  Θεωρώ πως τα μεγέθη αυτά των μελών είναι δυνατά για να προσφέρουν στον βραχίονα την αναγκαία ευελιξία, ώστε να φτάνει εύκολα σε όλα τα σημεία και να είναι πλήρως λειτουργικός ανεξάρτητα από το αν ο στόχος βρίσκεται κοντά ή μακριά από την βάση.

- Όλες οι αρθρώσεις του βραχίονα θα είναι τύπου “revolute”, ώστε να επιτρέπεται σε κάθε μια από αυτές να κινείται στον άξονα yy’.

- Δεν κατάφερα να δημιουργήσω Gripper 1ος βαθμού ελευθερίας. Επομένως ο Gripper αποτελείται από 3 links και 2 joints, δηλαδή έχει 2 βαθμούς ελευθερίας. Εχει την ακόλουθη μορφή:



- Ακολουθεί κώδικας που υπολογίζει την θέση των μελών του βραχίονα. Συγκεκριμένα ο κώδικας που ακολουθεί εκτυπώνει την θέση του 5ου μέλους, δηλαδή της βάσης της δαγκάνας:

```
L1 = 0.1
L2 = 0.4
L3 = 0.3
L4 = 0.135
L5 = 0.015

joints = robot.positions()

tf_12 = dartpy.math.Isometry3()
tf_12.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[0], 0.]))
tf_12.set_translation([0., 0., L1])

tf_23 = dartpy.math.Isometry3()
tf_23.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[1], 0.]))
tf_23.set_translation([0., 0., L2])

tf_34 = dartpy.math.Isometry3()
tf_34.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[2], 0.]))
tf_34.set_translation([0., 0., L3])

tf_45 = dartpy.math.Isometry3()
tf_45.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[3], 0.]))
tf_45.set_translation([0., 0., L4])
```

```
tf_56 = dartpy.math.Isometry3()
tf_56.set_rotation(dartpy.math.eulerZYXToMatrix([0., 0., 0.]))
tf_56.set_translation([0., 0., L5])

tf = tf_12.multiply(tf_23).multiply(tf_34).multiply(tf_45).multiply(tf_56)
print(tf)
```

2) 3Δ Ρομποτικός Βραχίονας:

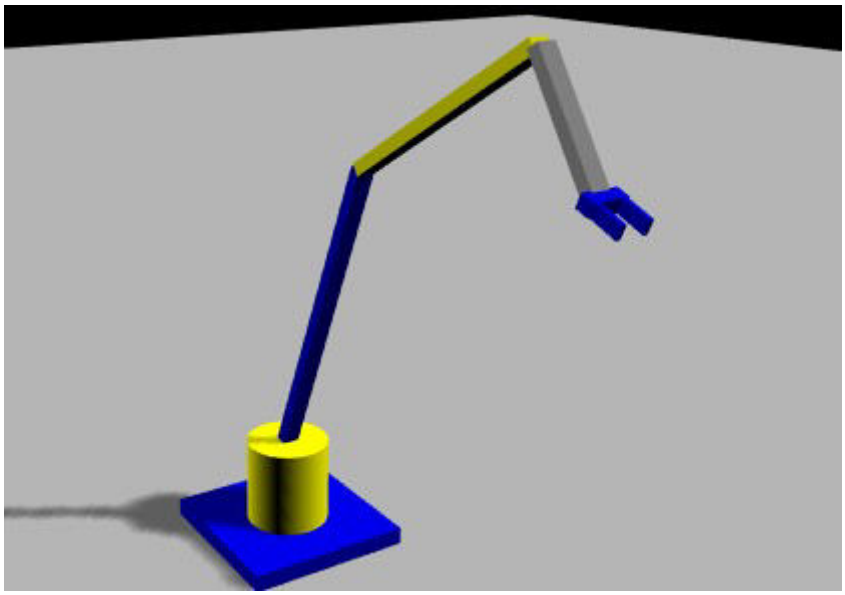
- Για να καταστεί δυνατή η κίνηση του παραπάνω βραχίονα στον χώρο(3 Διαστάσεις), θα προσθέσω ένα link στην βάση του κυλίνδρου που έχω και θα προσθέσω μεταξύ τους μία άρθρωση τύπου “revolute”, στην οποία θα επιτρέπεται η περιστροφική κίνηση γύρω από τον άξονα zz’.

- Ομοίως με πριν, ο κώδικας που ακολουθεί υπολογίζει την θέση των μελών του βραχίονα στον 3διάστατο χώρο. Σε αντίθεση με τον δισδιάστατο χώρο, η άρθρωση joint[0] περιστρέφεται γύρω από τον άξονα zz’.

```
L0 = 0.02
L1 = 0.1
L2 = 0.4
L3 = 0.3
L4 = 0.135
L5 = 0.015

joints = robot.positions()

tf_01 = dartpy.math.Isometry3()
tf_01.set_rotation(dartpy.math.eulerZYXToMatrix([joints[0], 0., 0.]))
tf_01.set_translation([0., 0., L0])

tf_12 = dartpy.math.Isometry3()
tf_12.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[1], 0.]))
tf_12.set_translation([0., 0., L1])

tf_23 = dartpy.math.Isometry3()
tf_23.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[2], 0.]))
tf_23.set_translation([0., 0., L2])

tf_34 = dartpy.math.Isometry3()
tf_34.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[3], 0.]))
tf_34.set_translation([0., 0., L3])

tf_45 = dartpy.math.Isometry3()
tf_45.set_rotation(dartpy.math.eulerZYXToMatrix([0., joints[4], 0.]))
tf_45.set_translation([0., 0., L4])

tf_56 = dartpy.math.Isometry3()
```

```
tf_56.set_rotation(dartpy.math.eulerZYXToMatrix([0., 0., 0.]))
tf_56.set_translation([0., 0., L5])

tf = tf_01.multiply(tf_12).multiply(tf_23).multiply(tf_34).multiply(tf_45).multiply(tf_56)
print(tf)
```

Η τελική μορφή του βραχίονα 3 διαστάσεων είναι η εξής:



3) URDF αρχεία και robot dart:

Περιγραφή urdf κώδικα:

- Αρχικά ορίζω το όνομα του ρομπότ και τα χρώματα με τα οποία αργότερα θα χρωματίσω τα μέλη του ρομπότ.
- Στην συνέχεια δημιουργώ τα links του ρομποτ και τα joints που θα τα συνδέουν. Το κάθε πεδίο link αποτελείται από το visual, collision και internal, που αφορούν την δημιουργία σχήματος, τα πεδία σύσκρουσης με άλλα links και το κέντρο βάρους μεταξύ άλλων. Ενώ τα joints αποτελούνται από parent, child, limit, origin, axis και dynamics, που αφορούν το parent και child μέλος της άρθρωσης, την θέση του και την κατεύθυνση κινησης.

Περιγραφή python κώδικα:

- Αρχικά κάνω τα απαραίτητα import και ορίζω τις παραμέτρους του simulation.
- Έπειτα, θέτω τα γραφικά και την οπτική γωνία της κάμερας.
- Εισάγω το ρομπότ, ορίζω το είδος των αρθρώσεων, τις γωνίες τους και εισάγω δάπεδο στην προσομοίωση.
- Ακολουθεί ο παραπάνω κώδικας υπολογισμού θέσης των μελών του βραχίονα για 2 και 3 διαστάσεις αντίστοιχα και ο ίδιος υπολογισμός, χρησιμοποιώντας την εντολή “print(robot.body_pose("end"))” της robot_dart.
- Τέλος πραγματοποιείται το “τρέξιμο” της προσομοίωσης.

Τα αποτελέσματα του κώδικα που παρουσίασα παραπάνω,όπως και της εντολής “print(robot.body_pose("end"))” της robot_dart για τον υπολογισμό της θέσης των μελών του βραχίονα είναι ίδια. Το αποτέλεσμα αυτό είναι αναμενόμενο, καθώς επιτελούν την ίδια διαδικασία.

Από το link “base_link” έχω βάλει ως σχόλιο το πεδίο “collision”, διότι αλλιώς εμφάνιζε επαναλαμβανόμενα το ακόλουθο Error και δυσκόλευε την ανάγνωση των θέσεων.