Πανεπιστήμιο Πατρών

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

3Δ Υπολογιστική Γεωμετρία και Όραση

UAV Collision Detection and Path Planning

Τσάμπρας Κωνσταντίνος

up1083865

Περιεχόμενα

Περίληψη

Η εργασία αυτή έχει ως σκοπό την απεικόνηση ενός αριθμού UAVs στον τρισδιάστατο χώρο, την δημιουργία περιβαλλόντων όγκων για τα αυτά, τον εντοπισμό συγκρούσεων σε κάθε στιγμή, αλλά και την μοντελοποίηση της κίνησης των UAVs, τον εντοπισμό συγκρούσεων σε χρονικά διαστήματα, και τέλος, την δημιουργία πρωτοκόλλων κίνησης των UAVs με στόχο την αποφυγή συγκρούσεων και την επίτευξη ενός στόχου (προσγείωση/απογείωση).

Προσέγγιση

Για την επίτευξη των παραπάνω γίνεται χρήση της γλώσσας Python, της βιβλιοθήκης Open3D, καθώς και πληθώρας άλλων βοηθητικών βιβλιοθηκών.

Η μοντελοποίηση του περιβάλλοντος, των UAVs και των περιβαλλόντων όγκων γίνεται με την χρήση αντικειμενοστραφούς προγραμματισμού, με κλάσεις όπως:

- Airspace (Ο εναέριος χώρος, υπεύθυνος για την δημιουργία του περιβάλλοντος, των UAVs, τον ορισμό των πρωτοκόλλων και την αποφυγή συγκρούσεων)
- LandingPad (Ο χώρος προσγείωσης, δεν έχει ιδιαίτερες λειτουργίες)
- UAV (Η κλάση που υλοποιεί τα UAVs, υπεύθυνη για την δημιουργία του κάθε UAV, την εμφάνιση και μετακίνησή του, την δημιουργία περιβάλλοντων όγκων, τον εντοπισμό στατικών/χρονικών συγκρούσεων με άλλα UAVs)

Καθώς και με έναν αριθμό βοηθητικών κλάσεων για την περιγραφή περιβαλλόντων όγκων και, γενικότερα, σχημάτων στον τρισδιάστατο χώρο:

- Point3D, Line3D, Cuboid3D, Sphere3D, Mesh3D (Χρησιμοποιούνται για την εμφάνιση απλών έως και σύνθετων αντικειμένων στον τρισδιάστατο χώρο)
- Triangle3D (Μια κλάση που κληρονομεί από την Mesh3D και διευκολύνει τον χειρισμό απλών τριγώνων στον χώρο)
- ConvexPolygon3D (Επίσης μια κλάση που κληρονομεί από την Mesh3D με σκοπό την αναπαράσταση κυρτών πολυγώνων στον τρισδιάστατο χώρο)
- Polyhedron3D (Μια χρήσιμη κλάση για αναπαράσταση περιβαλλόντων όγκων -όπως το kdop- που αποτελείται από έναν αριθμό κυρτών πολυγώνων)

- AabbNode (Μια κλάση η οποία δημιουργεί ένα δέντρο με σκοπό την εύρεση συγκρούσεων, ακολουθώντας την λογική του «διαίρει και βασίλευε»)
- Kdop (Η κλάση η οποία αναπαριστά το Kdop ενός UAV, κληρονομεί από την κλάση Polyhedron3D)

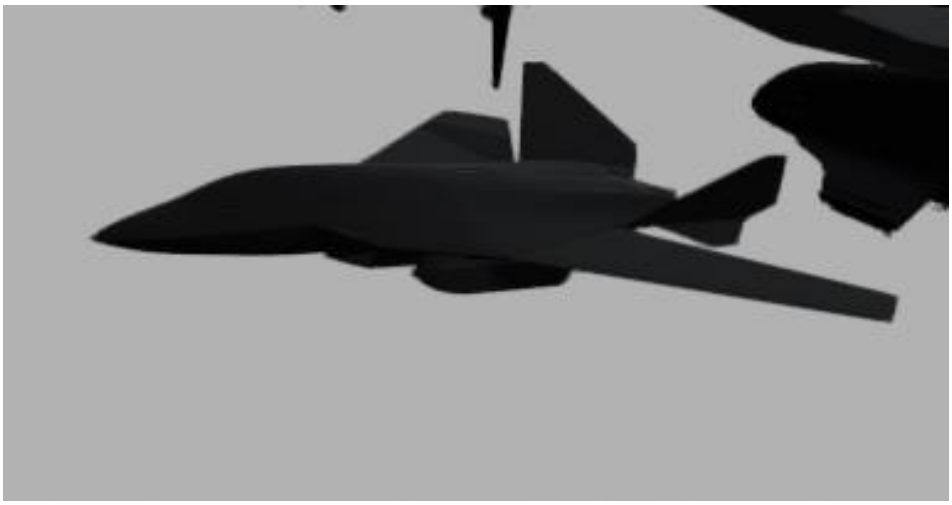
Ταυτόχρονα με την επεξήγηση της επίλυσης του κάθε ερωτήματος, οι κύριες μέθοδοι και λειτουργίες των παραπάνω κλάσεων θα αναφέρονται με περισσότερη λεπτομέρεια.

Μοντέλα

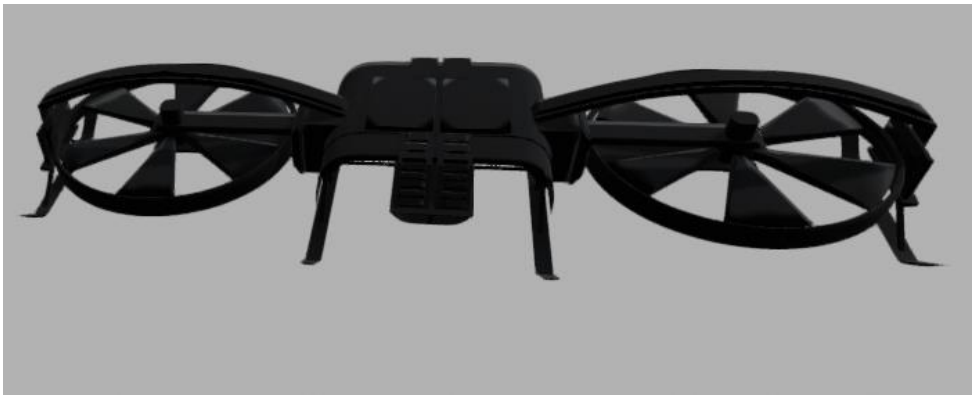
Παρακάτω παρατίθενται τα μοντέλα που χρησιμοποιήθηκαν:

1. v22_osprey (43.000 τρίγωνα):

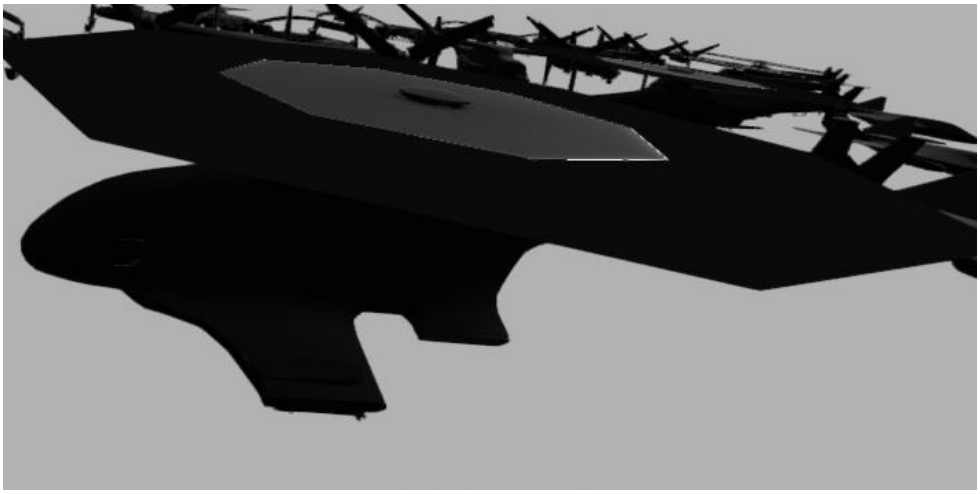
2. F52 (62.000 τρίγωνα):



3. twin_copter (47.000 τρίγωνα):



4. Helicopter (4.400 τρίγωνα):



5. quadcopter_scifi (200.000 σημεία και 100.000 τρίγωνα, δεν χρησιμοποιείται από default, γιατί η βιβλιοθήκη παίρνει πολύ χρόνο για να το μετακινήσει, αλλά, εκτός από το χρονικό αυτό θέμα, όλα δουλεύουν και με αυτό το μοντέλο):

Ερωτήματα

Ερώτημα 1. (Απεικόνιση)

«Επιλέξτε κάποια 3D μοντέλα drone και οπτικοποιήστε τα σε τυχαίες θέσεις. Επίσης μοντελοποιήστε μία επιφάνεια προσγείωσης απογείωσης με NxN θέσεις drones»

Για το ερώτημα αυτό, καθώς και για τα επόμενα, συλλέχθηκαν διάφορα μοντέλα UAVs, και γενικότερα ιπτάμενων οχημάτων. Αυτά είναι τα:
- v22_osprey
- twin_copter
- F52
- Helicopter

Με αριθμό vertices από 4.000 έως 50.000. Επιπλέον υπάρχει και το μοντέλο quadcopter_scifi το οποίο έχει 200.000 vertices, αλλά δεν χρησιμοποιείται στις περισσότερες περιπτώσεις λόγω της καθυστέρησης που προκαλεί ένα τόσο μεγάλο αντικείμενο όταν μετακινείται στην σκηνή της open3d (ειδικά στα ερωτήματα Z και έπειτα). Παρόλα αυτά μπορεί να χρησιμοποιηθεί κάνοντας uncomment το όνομά του στον ορισμό των μοντέλων.

Το πρώτο ερώτημα δεν είναι σύνθετο, οπότε θα αναφερθούν οι μέθοδοι της κλάσης Airspace υπεύθυνες για την δημιουργία του περιβάλλοντος.

Οι μέθοδοι αυτοί είναι:

1. create_uavs(): Δημιουργεί στοιχισμένα NxN UAVs πάνω στην επιφάνεια προσγείωσης/απογείωσης. Χρήσιμη μέθοδος για την επόπτευση των περιβαλλόντων όγκων των μοντέλων.

2. create_colliding_uavs(): Δημιουργεί 3 UAVs τα οποία συγκρούονται. Χρήσιμη μέθοδος για οπτικοποίηση των μεθόδων ανίχνευσης σύγκρουσης.

3. create_random _uavs(): Δημιουργεί UAVs NxN με τυχαία θέση και τυχαίο προσανατολισμό πάνω σε ένα επίπεδο. Χρησιμοποιείται για την ανίχνευση συγκρούσεων σε μια τυχαία τοποθέτηση UAVs.

4. create_random_uavs_non_colliding(): Δημιουργεί τυχαία NxN UAVs πάνω σε ένα επίπεδο, με κριτήριο να μην συγκρούονται. Χρησιμοποιείται για την αρχικοποίηση καταστάσεων όπου θέλουμε να επιδείξουμε το πρωτόκολλο αποφυγής συγκρούσεων.

5. create_time_colliding_uavs(): Δημιουργεί 2 UAVs με θέσεις και ταχύτητες τέτοιες ώστε να συγκρούονται στο επόμενο χρονικό διάστημα. Χρήσιμο για την δοκιμή ανίχνευσης συγκρούσεων σε ένα χρονικό διάστημα.

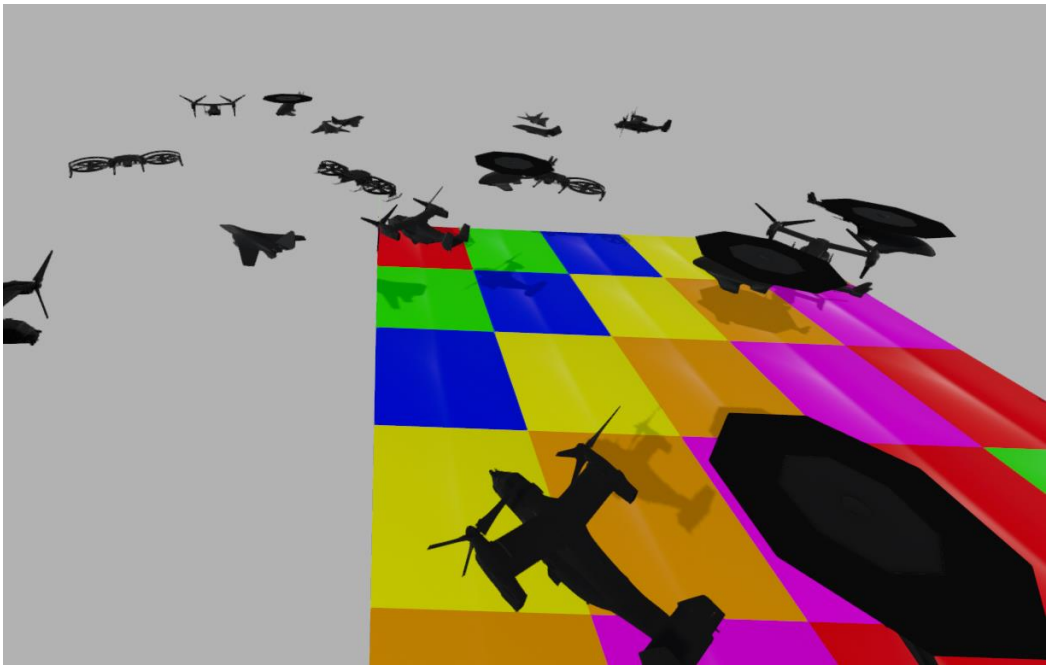
6. create_taking_off_uavs(): Δημιουργεί NxN UAVs πάνω στην επιφάνεια προσγείωσης με τυχαίο προσανατολισμό. Αυτή η αρχικοποίηση χρησιμοποιείται όταν ορίζουμε το πρωτόκολλο target_beacon (Ερώτημα Η (8). επιλογή 1.). Παράλληλα, δημιουργεί και NxN στόχους πάνω στο dome, έναν για κάθε UAV.

7. create_landing_uavs (): Δημιουργεί NxN UAVs πάνω σε σφαίρα (dome) με την δοθείσα ακτίνα γύρω από την περιοχή προσγείωσης/ απογείωσης.

Τα UAVs βρίσκονται στο πάνω ημισφαίριο, και όχι πολύ κοντά στο έδαφος. Αυτή η αρχικοποίηση χρησιμοποιείται όταν ορίσουμε το πρωτόκολλο landing (Ερώτημα Η (8). επιλογή 2.).

8. create_landing_uavs_time(): Δημιουργεί UAVs εντός του dome με τυχαία θέση και προσανατολισμό. Χρησιμοποιείται όταν θέσουμε το πρωτόκολλο landing_time (Ερώτημα Η (8) επιλογή 3.). Τα UAVs δεν δημιουργούνται όλα αρχικά (όπως στις προηγούμενες μεθόδους): ένας αριθμός δημιουργείται στην εκκίνηση της προσομοίωσης και στην συνέχεια, σε κάθε frame γίνεται μια «κλήρωση» για το αν θα δημιουργηθεί ένα νέο UAV. Η πιθανότητες αυτής της κλήρωσης ορίζονται στο όρισμα flow (flow = 1 σημαίνει ότι σε κάθε frame θα προστίθεται ένα νέο UAV στον εναέριο χώρο). Η δημιουργία του UAV γίνεται με την create_new_landing_uav().

Παρακάτω φαίνεται η εκτέλεση της μεθόδου create_random_uavs() για το ερώτημα 1:



Εικόνα 1. Airspace.create_random_uavs()

Ερώτημα 2. (Περιβάλλοντες όγκοι)

«Υπολογίστε το κυρτό περίβλημα, AABB, και k-DOP»

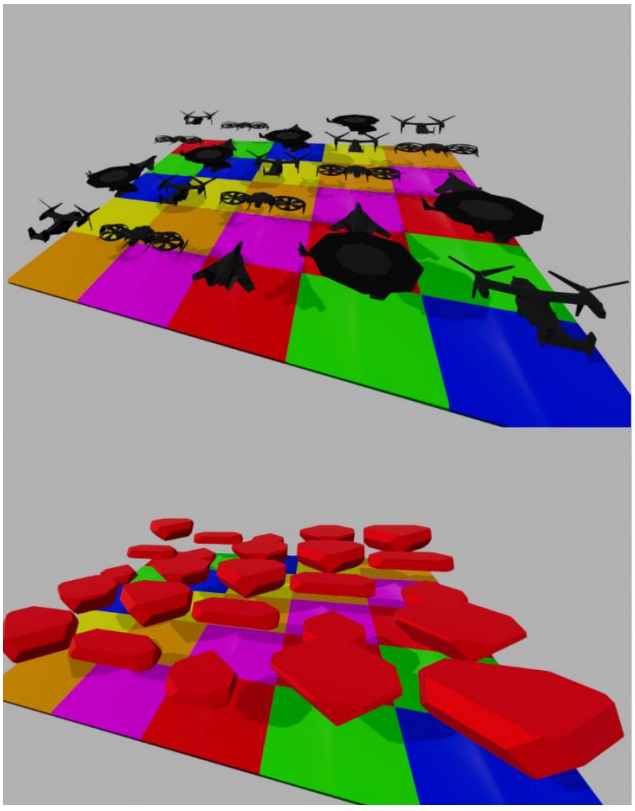
Αρχικά τα UAVs τοποθετούνται στοιχισμένα στις θέσεις απογείωσης. Υπάρχουν οι εξής δυνατότητες για το κάθε UAV που ενεργοποιούνται με την χρήση των παρακάτω πλήκτρων:

- K. Εμφάνιση kdop (το k ορίζεται σύμφωνα με την μεταβλητή kdop_number, έχουν υλοποιηθεί οι περιπτώσεις για k=6 και k=14, default k = 14).
- C. Εμφάνιση κυρτού περιβλήματος.
- U. Εμφάνιση μοναδιαίας σφαίρας γύρω από το UAV.
- B. Εμφάνιση axis aligned bounding box.

(Ο «περιβάλλων όγκος» Aabb node έχει υλοποιηθεί αλλά δεν υπάρχει κάποιος καλός τρόπος για να οπτικοποιηθεί)

Ακολουθούν παραδείγματα για τον κάθε όγκο, καθώς και η διαδικασία υπολογισμού του:

Kdop:



Εικόνα 2. Δημιουργία 14Dops

Η περίπτωση των 6Dops είναι απλή καθώς αφορά την ουσιαστικά την εύρεση του Aabb που θα εξεταστεί παρακάτω.

Η περίπτωση του 14Dop είναι αρκετά πιο σύνθετη.

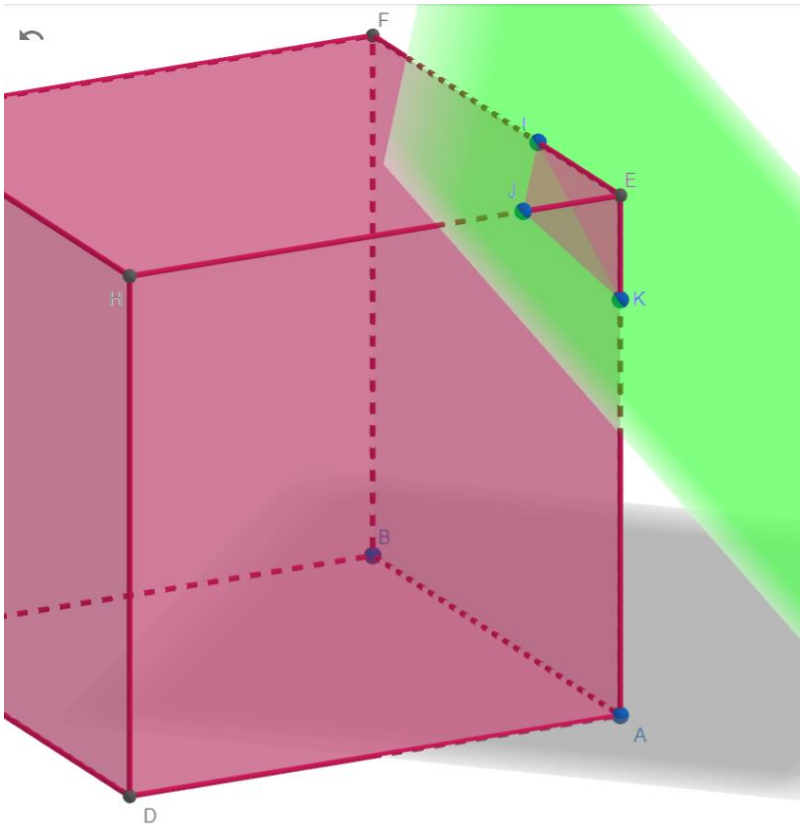
Η μέθοδος που υλοποιήθηκε έχει πολύ καλά αποτελέσματα σε UAV με φυσιολογικές αναλογίες (όχι πολύ επίπεδο ή λεπτά).

Η δημιουργία του 14Dop γίνεται στην μέθοδο της κλάσης Kdop create_14dop(). Ακολουθεί την λογική της «αφαίρεσης» των γωνιών ενός Aabb. Τα βήματα που ακολουθούνται περιγράφονται παρακάτω:

a. Ορισμός χρήσιμων συναρτήσεων:
   - find_intersection(plane1, plane2, plane2). Βρίσκει το σημείο τομής των 3 επιπέδων.
   - check_if_neighbor(corner_dir, face_dir). Ελέγχει εάν δύο κατευθύνσεις (γωνιακή και πλευράς) «συνορεύουν». Για παράδειγμα η πλευρά της οποία η κατεύθυνση είναι η [1, 0, 0], είναι γείτονας της γωνίας [1, 1, 1].
   - plane_equation_from_point_normal(point, normal). Εξάγει την εξίσωση του επιπέδου. Δοθέντος ενός σημείου του και του normal του επιπέδου.

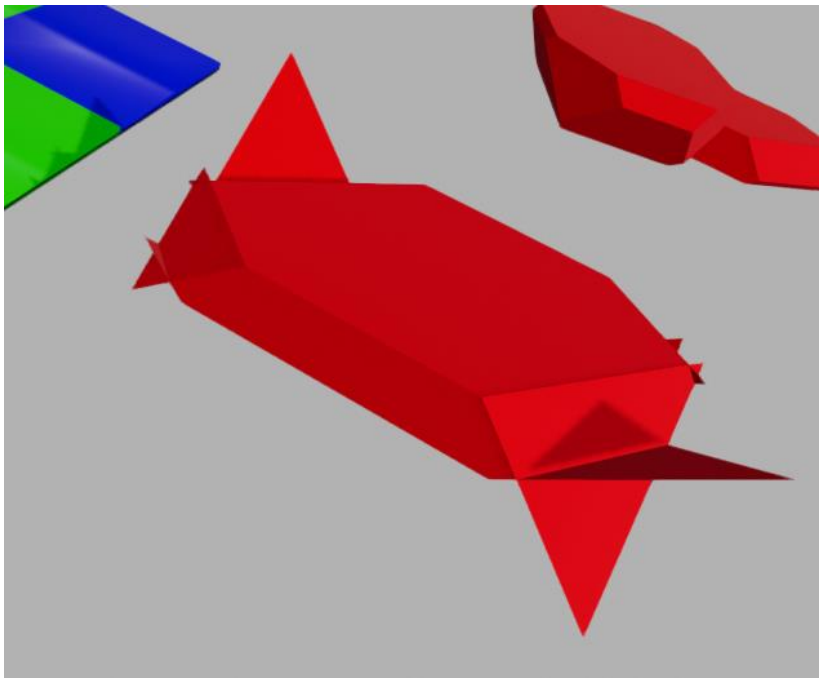
b. Εύρεση ενός vertex για κάθε κατεύθυνση, του οποίου η προβολή πάνω στην κατεύθυνση είναι η μέγιστη για αυτήν την κατεύθυνση. Δηλαδή βρίσκουμε το σημείο που ορίζει το επίπεδο που περιορίζει το kdop προς κάθε κατεύθυνση (ένα επίπεδο για κάθε κατεύθυνση).

c. Στην συνέχεια, για κάθε κατεύθυνση γωνίας (πχ [1,-1,1]) βρίσκουμε την εξίσωση του επιπέδου της και τις γειτονικές της κατευθύνσεις που αντιστοιχούν σε πλευρές. Για τις δυάδα γειτονικών κατευθύνσεων βρίσκουμε το σημείο τομής του επιπέδου της γωνίας με τα δύο επίπεδα τωνγειτονικών αυτών κατευθύνσεων. Για παράδειγμα στο παρκάτω σχήμα, έχουμε το επίπεδο της γωνίας (πράσινο) και τα δύο γιετονικά επίπεδα πλευρών (EHA, EFH) η τομή των οποίων μας δίνει το J. Ομοίως οι άλλοι συνδυασμοί μας δίνουν τα I,K, τα τρία αυτά σημεία μας δίνουν ένα τρίγωνο για κάθε γωνία.



Εικόνα 3. Εύρεση σημείων του 14Dop

d. Η παραπάνω διαδικασία δημιουργεί τρίγωνα, τα οποία συχνά έχουν σημεία τα οποία δεν ανήκουν στο 14Dop (Βλ. Εικόνα 4.) Οπότε το επόμενο στάδιο είναι να αφαιρεθούν τα σημεία τα οποία δεν βρίσκονται στο εσωτερικό όλων των τριγώνων. Τα υπόλοιπα σημεία των τριγώνων θεωρούνται valid_points.



Εικόνα 4. Μη έγκυρα σημεία τριγώνων (εκτός 14Dop)

e. Μία ακόμη περίπτωση είναι δύο τρίγωνα να τέμνονται (όπως στην εικόνα 4.) και η τομή τους να είναι απαραίτητη για τον ορισμό του 14Dop. Οπότε το επόμενο βήμα είναι η εύρεση των τομών των τριγώνων και η προσθήκη των άκρων των ευθυγράμμων τμημάτων σαν έγκυρα σημεία του 14Dop.

f. Έχοντας πλέον την λίστα με όλα τα σημεία τα οποία ορίζουν το 14Dop, επόμενο βήμα είναι να δημιουργήσουμε τις πλευρές του. Για τον σκοπό αυτό χρησιμοποιούμε τις εξισώσεις των επιπέδων που αντιστοιχούν σε κάθε κατεύθυνση και κατηγοριοποιούμε το κάθε σημείο ανάλογα με τα επίπεδα που ανήκει.

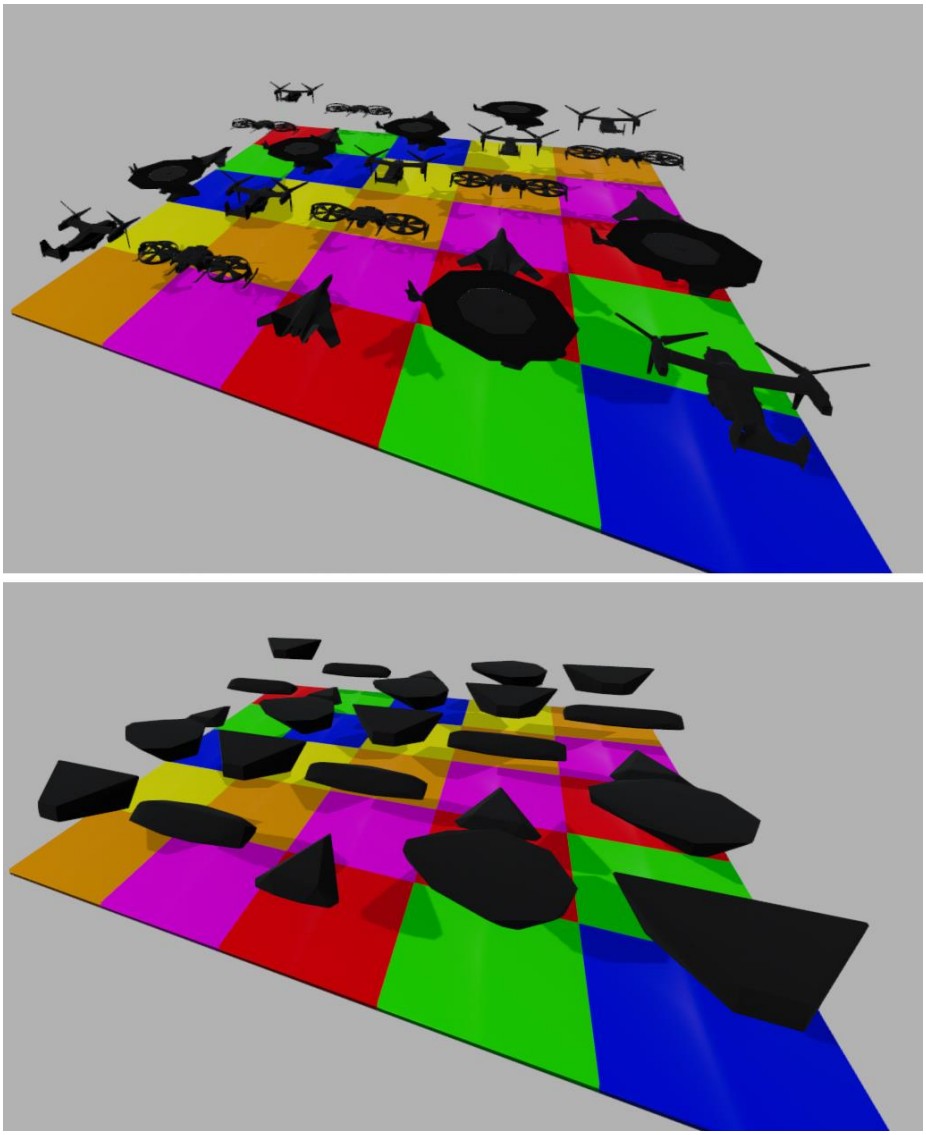
g. Έτσι έχουμε ένα σύνολο σημείων για κάθε πλευρά του 14Dop. Δημιουργούμε ένα στιγμιότυπο της κλάσης ConvexPolygon3D δίνοντας ως όρισμα το σύνολο των σημείων. Η Κλάση αυτή τα ταξινομεί με ωρολογιακή φορά και δημιουργεί τρίγωνα που αναπαριστούν το πολύγωνο αυτό στον τρισδιάστατο χώρο.

h. Τέλος, δίνουμε το σύνολο των πολυγώνων αυτών σαν όρισμα στον κονστράκτορα της Polyhedron3D η οποία αναπαριστά τελικά το 14Dop.

Σε αυτό το σημείο σημειώνεται ότι επειδή η δημιουργία ενός 14Dop με την παραπάνω μεθοδολογία απαιτεί χρόνο της τάξεως των δεκάτων του δευτερολέπτου (ειδικά για UAVs με αριθμό σημείων >= 40.000), έχει υλοποιηθεί μέθοδος σύμφωνα με την οποία, αν έχει δημιουργηθεί 14Dop για την ίδια κλάση αεροσκάφους, αντί να δημιουργηθεί από την αρχή καινούριο, αντιγράφεται το παλιό (κάνοντας βεβαίως τους απαραίτητους μετασχηματισμούς θέσεως και περιστροφής ώστε να «ταιριάξει» στο νέο UAV). Η διαδικασία αντιγραφής απαιτεί μόνο εκατοστά του δευτερολέπτου.

Για την εμφάνιση και απόκρυψη των Kdops των UAVs είναι υπεύθυνες οι μέθοδοι create_kdop, remove_kdop.

Convex Hull

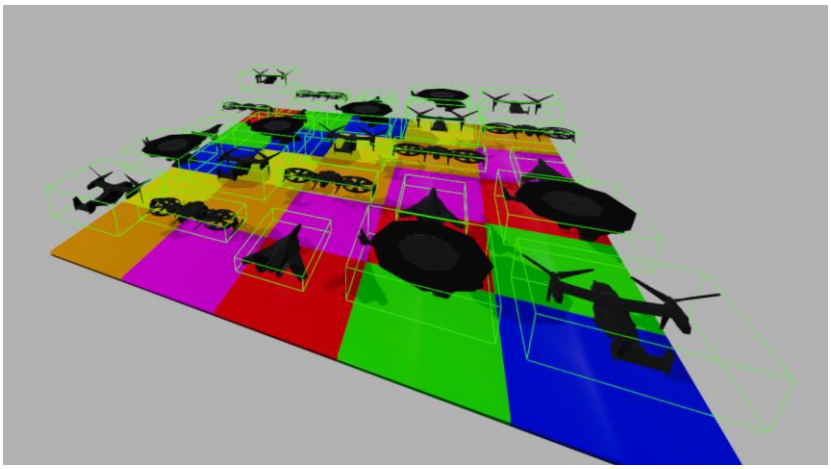


Εικόνα 5. Convex Hulls

Για την δημιουργία του κυρτού περιβλήματος χρησιμοποιήθηκε η μέθοδος TriangleMesh.compute_convex_hull() η οποία υπολογίζει το convex hull σε μορφή Triangle mesh με μεγάλη ταχύτητα.

Για την εμφάνιση και απόκρυψη των Convex Hulls των UAVs είναι υπεύθυνες οι μέθοδοι create_convex_hull, remove_convex_hull.

Aabb

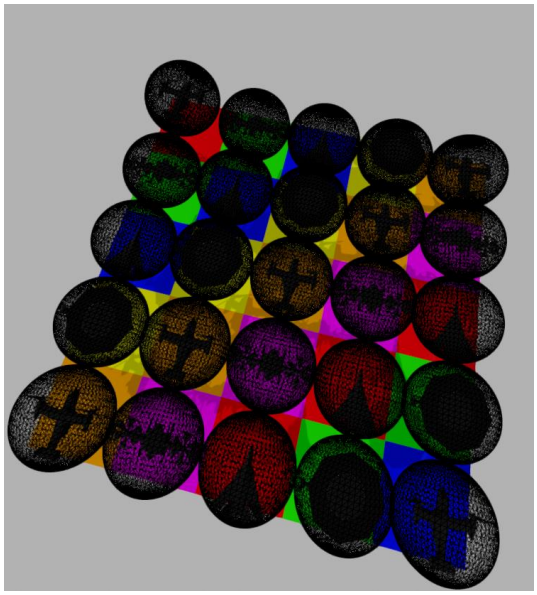


Εικόνα 6. Axis Aligned Bounding Boxes

Η δημιουργία του Aabb είναι εξαιρετικά απλή, αφού το Aabb απλά ορίζεται από τις ελάχιστες και τις μέγιστες τιμές της κάθε συνιστώσας των συντεταγμένων των σημείων του UAV. Για την απεικόνισή του χρησιμοποιήθηκε η κλάση Cuboid3D.

Υπεύθυνες για την εμφάνιση/απόκρυψη των Aabb είναι οι μέθοδοι create_aabb, remove_aabb.

Unit Sphere



Εικόνα 7. Εμφάνιση Unit spheres

Απλή είναι και η υλοποίηση των μοναδιαίων σφαιρών γύρω από τα UAVs με τις μεθόδους create_unit_sphere, remove_unit_sphere.

Ερώτημα 3. (Ανίχνευση συγκρούσεων βάσει περιβάλλοντων όγκων)

«Υλοποιήστε και εκτελέστε αλγορίθμους ανίχνευσης σύγκρουσης Α βάσει μόνο των περιβαλλόντων όγκων»

Σημειώνεται ότι σε όλα τα ερωτήματα από εδώ και πέρα, μπορούμε να δείξουμε τις στιγμιαίες συγκρούσεις κάθε στιγμή με το πλήκτρο L (αν και στα ερωτήματα που ζητείται, οι συγκρούσεις εμφανίζονται στην αρχή της εκτέλεσης από μόνες τους, αλλά αν κινήσουμε το αεροσκάφος στο οποίο έχουμε έλεγχο και θέλουμε να δούμε τις νέες συγκρούσεις, πρέπει να πατήσουμε το L).

Υπεύθυνη για τον εντοπισμό στιγμιαίων συγκρούσεων μεταξύ UAVs είναι η μέθοδος collides() της κλάσης UAV.

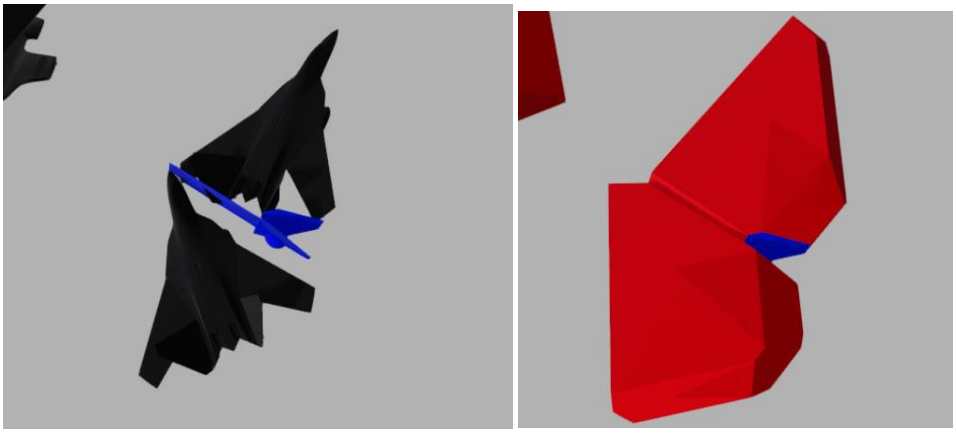
Μέσα σε αυτήν ορίζεται (ανάλογα με το ερώτημα, ή γενικά την επιθυμία του προγραμματιστή) μια ιεραρχία περιβαλλόντων όγκων/συναρτήσεων εύρεσης συγκρούσεων.

Η λογική της ιεραρχίας είναι ότι θα δοκιμαστούν με σειρά μία-μία οι (συντηρητικές) μέθοδοι που ορίζονται μέσα σε αυτήν, μέχρι κάποια από αυτές να υποδείξει ότι δεν υπάρχει σύγκρουση.

Αν μία από τις μεθόδους υποδείξει έλλειψη σύγκρουσης με το άλλο UAV, τότε η αναζήτηση σταματάει η αναζήτηση και η μέθοδος επιστρέφει False.

Η μέθοδος σύμφωνα με την οποία εμφανίζεται στην σκηνή η σύγκρουση είναι η τελευταία μέθοδος εντός της ιεραρχίας.

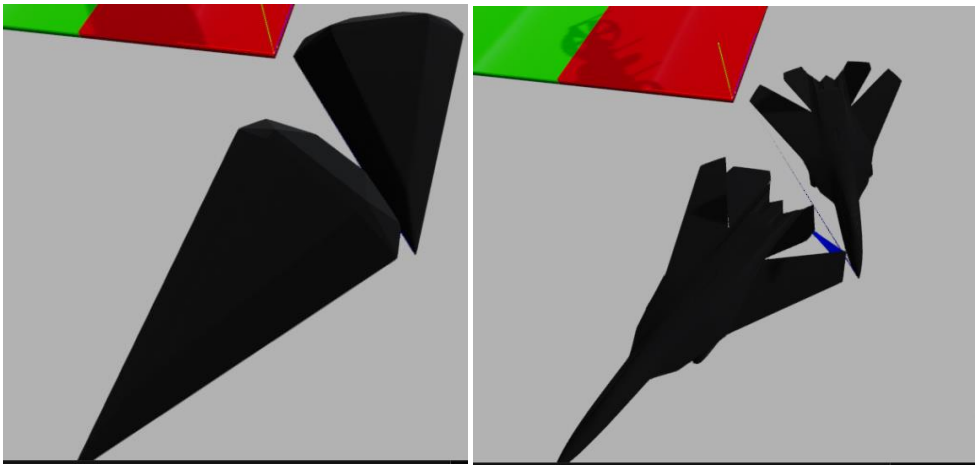
Παρακάτω υπάρχει το παράδειγμα εκτέλεσης του τρίτου ερωτήματος, όπου η ορισμένη ιεραρχία είναι η aabb->convex_hull->kdop:



Εικόνα 8. Απεικόνιση σύγκρουσης Kdop

Βλέπουμε ότι επειδή ο έλεγχος σύγκρουσης Kdop ήταν ο τελευταίος στην ιεραρχία, η απεικόνιση της σύγκρουσης γίνεται με την τα στοιχεία της μεθόδου αυτής. Στην συγκεκριμένη περίπτωση χρωματίζονται με μπλε οι πλευρές των 14Dops οι οποίες συγκρούονται

Παρακάτω έχουμε παράδειγμα με την ιεραρχία aabb ->kdop->convex_hull:



Εικόνα 9. Απεικόνιση σύγκρουσης Convex Hull

Βλέπουμε ότι το αποτέλεσμα απεικόνισης της σύγκρουσης των κυρτών περιβλημάτων είναι η εμφάνιση ενός ζευγαριού συγκρουόμενων τριγώνων με μπλε χρώμα.

Σε αυτό το σημείο σημειώνεται ότι οι μέθοδοι που αναφέρθηκαν μέχρι τώρα και απαιτούν εύρεση σύγκρουσης mesh με mesh (convex hull και kdop) έχουν υλοποιηθεί στο πρόγραμμα και πραγματοποιούν την παραπάνω επίδειξη σύγκρουσης.

Παράλληλα όμως, έχουν υλοποιηθεί και μέθοδοι που χρησιμοποιούν μια εξωτερική βιβλιοθήκη για εύρεση συγκρούσεων mesh με mesh, την trimesh, η οποία δεν μας δείχνει την ακριβή σύγκρουση (όπως οι παραπάνω

υλοποιημένες), αλλά είναι πολύ ταχύτερη λόγω της χαμηλότερης υλοποίησής της (δεν χρειάζεται να συγκρίνει Triangle3D αντικείμενα κάθε φορά).

Οπότε για περιπτώσεις όπου μας ενδιαφέρει περισσότερο η ταχύτητα εύρεσης σύγκρουσης από ότι η οπτικοποίησή της, (πχ στις προσομοιώσεις παρακάτω) στις ιεραρχίες θα χρησιμοποιούνται κυρίως οι μέθοδοι που εκμεταλλεύονται την προαναφερθείσα βιβλιοθήκη.

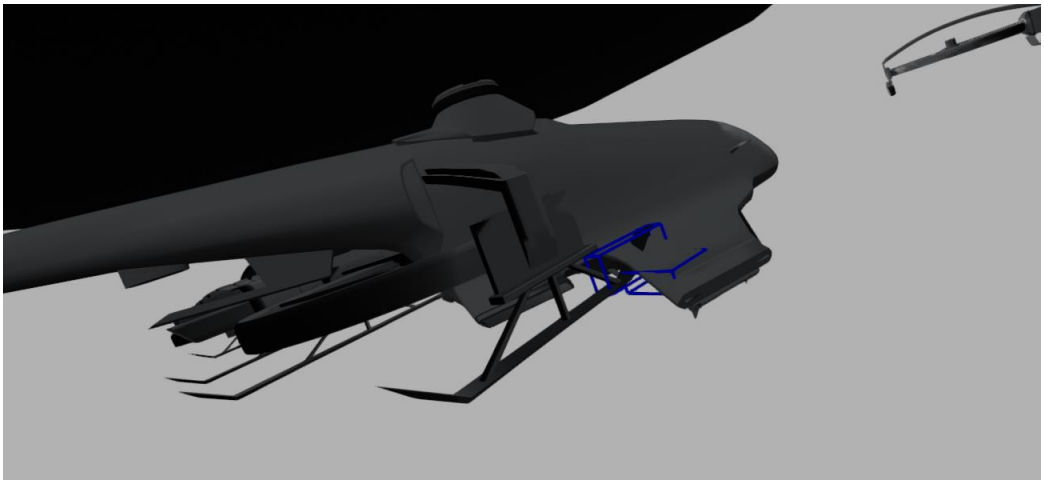
Ερώτημα 4. (Βελτίωση ανίχνευσης συγκρούσεων)

«Βελτιώστε λίγο την ακρίβεια της μεθόδου B και υλοποιήστε μία απόλυτα ακριβή μέθοδο C.»

Εκτός από την λογική της ιεραρχίας περιβάλλοντων όγκων, η βελτίωση της μεθόδου έγκειται και στην υλοποίηση του «περιβάλλοντως όγκου» του AabbNode.

Η λογική της μεθόδου αυτής έγκειται στην διαίρεση του καθενός από τα δύο UAV σε 8 «τεταρτημόρια» και την εύρεση συγκρούσεων μεταξύ των 8 «παιδιών». Η διαδικασία αυτή εύκολα γίνεται αναδρομική και στην περίπτωση του προγράμματος υλοποιείται μέχρι βάθος 4. Αν κάποιο «παιδί» δεν συγκρούεται με το συγκρινόμενο παιδί του άλλου UAV, η αναδρομική διαδικασία σταματάει για εκείνον τον κλάδο, σε αντίθετη περίπτωση συνεχίζει μέχρι κάποιος απόγονος να συγκρούεται με τον συγκρινόμενο απόγονο του άλλου UAV, και να μην έχει δικούς του απογόνους (έχει φτάσει μέγιστο βάθος). Τότε δεχόμαστε ότι υπάρχει σύγκρουση.

Η οπτικοποίηση της σύγκρουσης αυτής είναι η εμφάνιση δύο αντικειμένων της κλάσης Cuboid3D που αντιπροσωπεύουν τους δύο αυτούς απογόνους των δύο UAv, που συγκρούονται (Βλ. Εικόνα 9). Ακολουθεί παράδειγμα όπου η ιεραρχία είναι aabb->kdop->aabb_node:

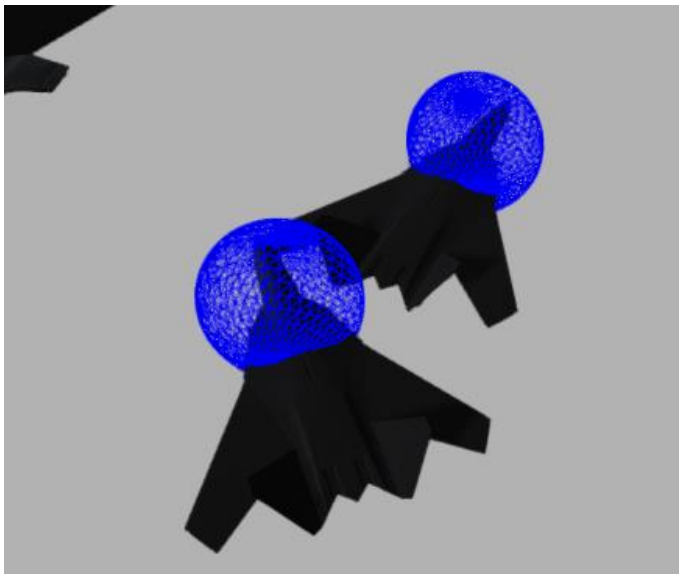


Εικόνα 10. Απεικόνιση σύγκρουσης AAbbNode

Τέλος έχουμε την απόλυτα ακριβή μέθοδο σύγκρισης των δύο mesh η οποία τοποθετείται στο τέλος της ιεραρχίας.

Ο έλεγχος σύγκρισης δύο mesh των 50.000 σημείων και αντίστοιχου αριθμού τριγώνων είναι εξαιρετικά αργός (όπως αναφέρθηκε παραπάνω). Έτσι δεν συστήνεται να χρησιμοποιείται η προσωπικά υλοποιημένη μέθοδος collides_mesh(), αλλά η μέθοδος collides_mesh_trimesh() που χρησιμοποιεί την προαναφερθείσα βιβλιοθήκη για την εύρεση συγκρούσεων μεταξύ δύο mesh.

Η μέθοδος αυτή δεν μας ξεχωρίζει τα τρίγωνα που συγκρούονται οπότε απλά οπτικοποιούμε τα δύο UAVs τα οποία συγκρούονται με 2 μπλε σφαίρες:

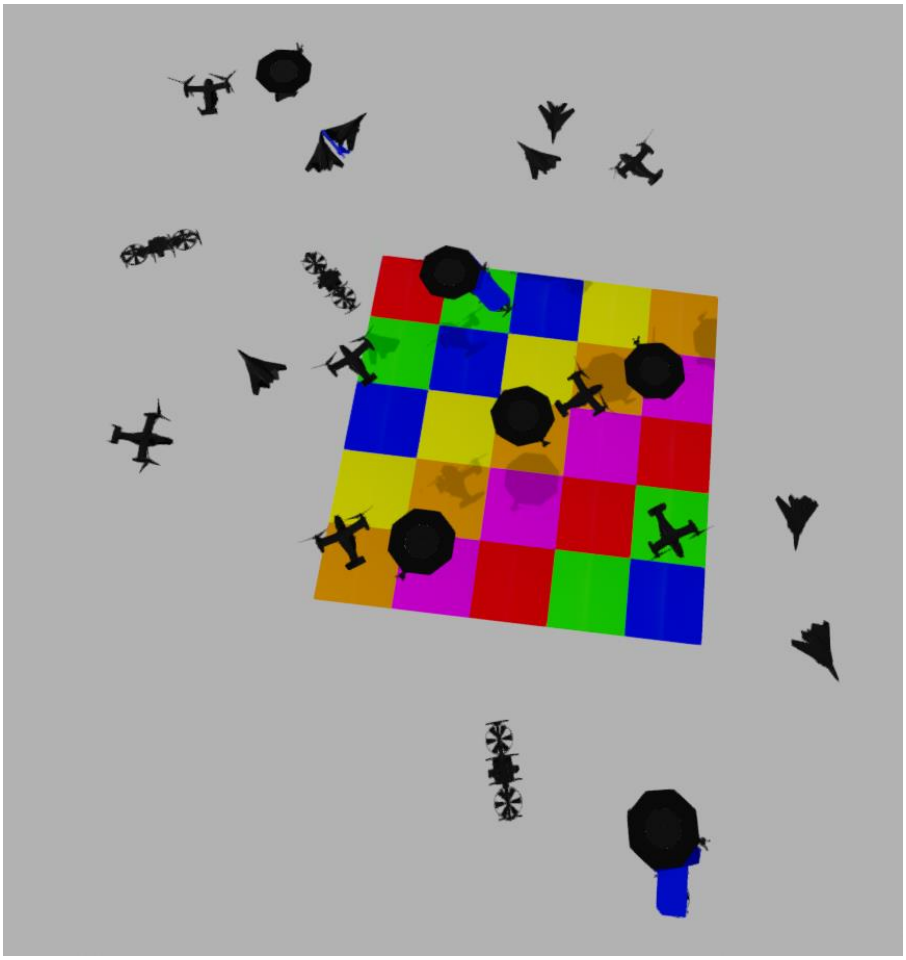


Εικόνα 11. Σύγκρουση με χρήση σύγκρισης mesh (trimesh)

Ερώτημα 5. (Απεικόνιση σγκρούσεων)

«Αρχικοποιήστε τυχαία τις θέσεις και τον προσανατολισμό των drones και απεικονίστε τις συγκρούσεις».

Στην εκτέλεση του καθενός από τα δύο προηγούμενα ερωτήματα γίνεται αυτό, οπότε για την εκτέλεση του ερωτήματος αυτού απλά θα επιλεχθεί η παρακάτω ιεραρχία:
Aabb->aabb_node->kdop_trimesh->mesh_trimesh->kdop
(η ύπαρξη του ελέγχου kdop στο τέλος είναι απλά για την καλύτερη οπτικοποίηση της σύγκρουσης).



Εικόνα 12. Εκτέλεση ερωτήματος 5.

Ερώτημα 6. (Απεικόνιση συγκρούσεων σε χρονικά διαστήματα)

«Υποθέστε ότι η συχνότητα επεξεργασίας είναι dt. Χρησιμοποιήστε την ταχύτητα κάθε οχήματος ώστε να ανανεώσετε το περίβλημά του και να μπορείτε να κάνετε έλεγχο σύγκρουσης για το διάστημα dt. Απεικονίστε τις συγκρούσεις».

Στο πρόβλημα εύρεσης συγκρούσεων τώρα εισάγεται ο χρόνος.

Μια απλή προσέγγιση θα ήταν να ελέγξουμε αν υπάρχει σύγκρουση στον χρόνο t_0 και αν υπάρχει σύγκρουση στον χρόνο t_0 + dt. Όμως αυτή η προσέγγιση δεν θα μας εγγυάται ότι στο ενδιάμεσο διάστημα τα UAVs δεν συγκρούονται.

Μια άλλη προσέγγιση θα ήταν να κάνουμε μεγαλύτερο αριθμό ελέγχου συγκρούσεων (πχ 10) στο δίαστημα dt. Αυτή η προσέγγιση είναι πιο ακριβής, αλλά και πιο κοστοβόρα (αυξάνει x10 τον αριθμό ελέγχων).

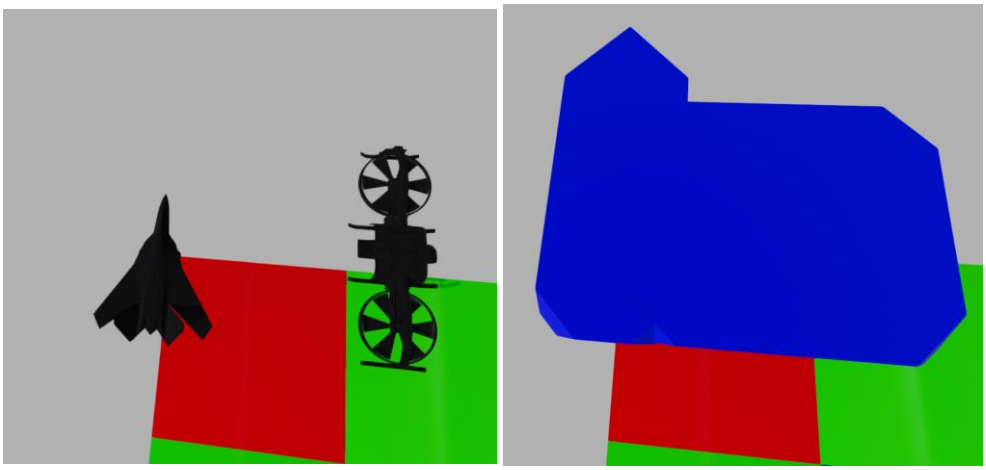
Έτσι αποφασίστηκε να χρησιμοποιηθεί μια διαφορετική μέθοδος για να μην μειώσουμε την ακρίβεια και να μην αυξήσουμε (σημαντικά) το κόστος.

Η μέθοδος αυτή περιλαμβάνει την δημιουργία του «συνεχούς περιβάλλοντως όγκου» (αυτοσχέδιος όρος), και πιο συγκεκριμένα του συνεχούς Kdop. Δηλαδή την δημιουργία ενός περιβάλλοντος όγκου ο οποίος περιβάλλει το όλα τα Kdops που θα είχε το UAV μέσα στο διάστημα dt.

Καθώς το UAV δεν αλλάζει ταχύτητα εντός του διαστήματος dt, η δημιουργία αυτού του του όγκου δεν απαιτεί κάτι παραπάνω από την δημιουργία του convex_hull του αρχικού (την στιγμή t_0) και του τελικού (την στιγμή t_0+dt) Kdop.

Σημειώνεται ότι θα μπορούσαμε να χρησιμοποιήσουμε το «συνεχές mesh», δηλαδή το convex hull του αρχικού και του τελικού mesh, αλλά αυτό θα αύξανε κατά πολύ τον χρόνο υπολογισμού, οπότε επιλέχθηκε το Kdop ως πιο συντηρητικός και ταχύτερος έλεγχος.

Παρακάτω βλέπουμε ένα παράδειγμα ελέγχου σύγκρισης στον χρόνο με αυτήν την μέθοδο (με μπλε εμφανίζονται οι όγκοι όταν εντοπίζεται σύγκρουση):



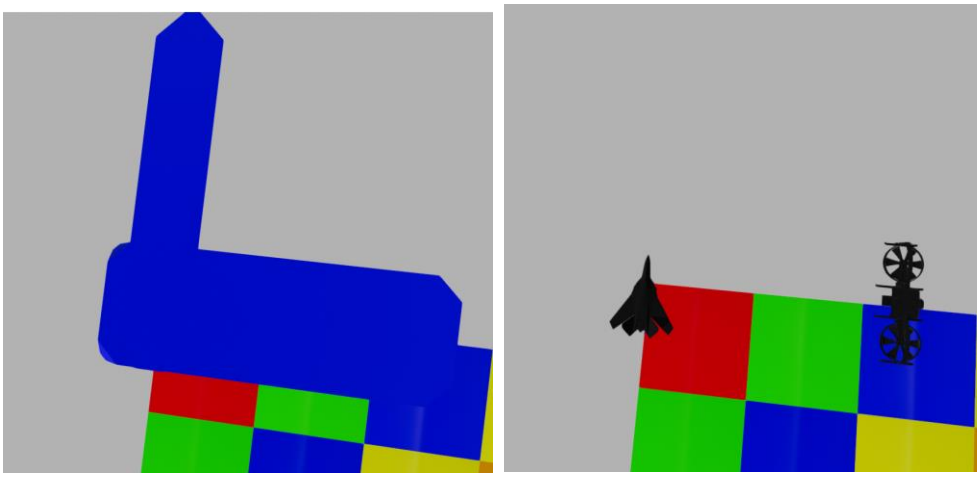
Εικόνα 13. Εύρεση σύγκρουσης με χρήση συνεχούς Kdop

Βέβαια η μέθοδος, αν και ικανοποιητική για όλες τις πρακτικές περιπτώσεις, δεν είναι εντελώς επιστημονικά ορθή, καθώς με την τωρινή της μορφή, οποιαδήποτε επαφή μεταξύ των δύο όγκων θα μας σηματοδοτούσε σύγκρουση μεταξύ των δύο UAVs σε κάποια χρονική στιγμή.
Όμως αν το «τέλος» του όγκου Α τέμνει την «αρχή» του όγκου Β, καταλαβαίνουμε ότι δεν θα υπάρξει σύγκρουση σε καμία χρονική στιγμή.

Άρα μένει να προσθέσουμε την περίπτωση περαιτέρω ελέγχων σε διαφορετικές στιγμές εντός του διαστήματος dt. Η μέθοδος που θα χρησιμοποιηθεί, ο αριθμός των βημάτων, αλλά και γενικότερα η μέθοδος του συνεχούς περιβάλλοντως όγκου είναι υλοποιημένα στην μέθοδο **collides_dt()** της κλάσης UAV.
Μπορούμε να επιλέξουμε οποιαδήποτε από τις προαναφερθείσες μεθόδους ως δευτερεύουσα μέθοδο ελέγχου, αλλά στον κώδικα έχει επιλεχθεί η μέθοδος collides_aabb() για λόγους ταχύτητας της προσομοίωσης.

Έτσι πετύχαμε στην γενική περίπτωση να γίνεται έλεγχος μεταξύ μόνο δύο περιβάλλοντων όγκων, και αν βρεθεί σύγκρουση, να γίνεται περαιτέρω έλεγχος.

Στο παρακάτω (ακραίο) παράδειγμα βλέπουμε την διαφορά που επιφέρει η χρήση της επιπλέον μεθόδου. Στην αριστερή περίπτωση δεν χρησιμοποιείται η επιπλέον μέθοδος και εντοπίζεται σύγκρουση (ενώ τα UAVs δεν θα συγκρούονταν), ενώ στην δεξιά με χρήση μόνο 2 ενδιάμεσων ελέγχων δεν εντοπίζεται σύγκρουση.

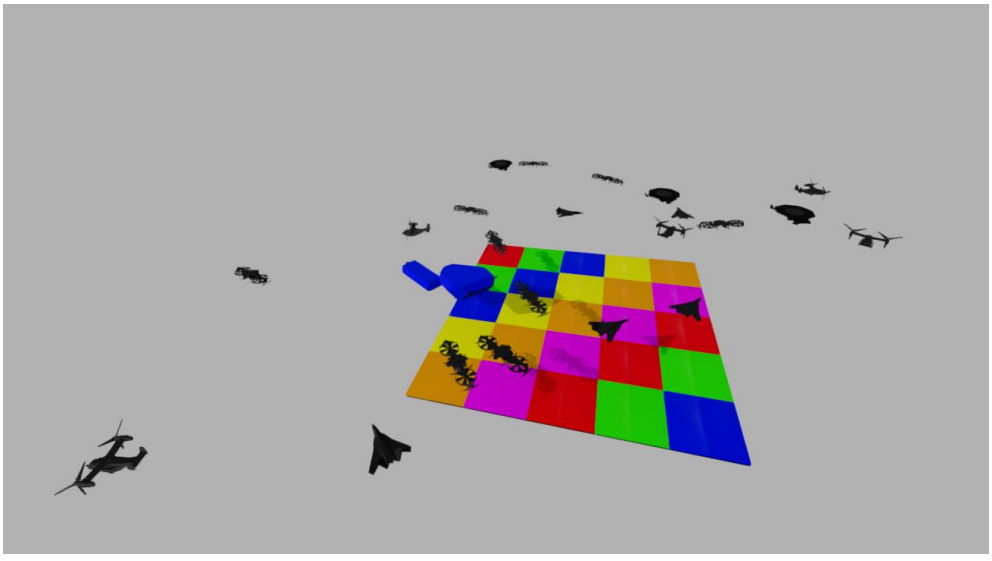


Εικόνα 14. Αριστερά, απλός περιβάλλων όγκος, δεξιά επιπλέον έλεγχος

Βέβαια στην περίπτωση των προσομοιώσεων στα παρακάτω ερωτήματα, δεν έχουμε αρκετά μεγάλα dt ώστε να έχουμε τόσο ακραίες περιπτώσεις, παρόλα αυτά η μέθοδος παραμένει ως έχει για χάριν της επιστημονικής ορθότητας

Η εκτέλεση του ερωτήματος 6 περιλαμβάνει την τυχαία τοποθέτηση μη συγκρουόμενων UAVs στον χώρο και την κίνηση τους με πρωτόκολλο αποφυγής. Οι συγκρούσεις τους (που αποφεύγονται) εμφανίζονται με το μπλε συνεχές Kdop όπως παραπάνω. Το πρωτόκολλο αποφυγής θα αναφερθεί με παραπάνω λεπτομέρεια στο επόμενο ερώτημα.

Σημειώνεται πως φαίνεται τα UAVs να βρίσκονται εκτός του συνεχούς Kdop την στιγμή της σύγκρουσης, αυτό γίνεται γιατί αποφεύγουν την σύγκρουση στο ίδιο frame που εντοπίζεται από το συνεχές kdop.



Εικόνα 15. Παράδειγμα εκτέλεσης του ερωτήματος 6.

Ερώτημα 7. (Πρωτόκολλο αποφυγής)

«Ανανεώστε την ταχύτητα κάποιων οχημάτων ώστε να μην υπάρχει σύγκρουση».

Σημειώνεται ότι στα ερωτήματα που αφορούν κίνηση, η κίνηση της κάμερας δεν ανταποκρίνεται άμεσα (πχ όταν κινούνται τα αντικείμενα στην σκηνή). Οπότε προτείνεται, όταν είναι επιθυμητό να κινηθεί η κάμερα, να χρησιμοποιείται το πλήκτρο P (που θέτει σε παύση την προσομοίωση) και να κινείται ελεύθερα η κάμερα. Όταν είναι επιθυμητή η συνέχεια της προσομοίωσης, ξαναπιέζεται το P.

Η κλάση Airspace, μέχρι στιγμής (εκτός του ερωτήματος 6) ήταν υπεύθυνη μόνο για την δημιουργία του περιβάλλοντος και την εύρεση συγκρούσεων.

Στην συνέχεια, όμως θα χρειαστεί να λαμβάνει αποφάσεις και για τα UAVs, για να τα κατευθύνει ώστε να αποφεύγουν συγκρούσεις και να κινούνται προς τον στόχο τους.

Για τον σκοπό αυτό έχουν υλοποιηθεί οι μέθοδοι find_collisions_dt, protocol_avoidance, protocol_landing, protocol_take_target_beacon.

Στο ερώτημα αυτό χρησιμοποιείται μόνο η protocol_avoidance, η οποία με την σειρά της καλεί την find_collisions_dt.

Η find_collisions_dt ελέγχει όλα τα ζευγάρια UAV καλώντας τις αντίστοιχες collides_dt μεθόδους τους και επιστρέφει ένα λεξικό με κλειδιά τον δείκτη του κάθε UAV, και ως value ένα set με τους δείκτες των UAVs με τα οποία συγκρούεται.

Η protocol_avoidance αποτελείται από τρία μέρη:

Μέρος 1^{ο}: Εύρεση συγκρούσεων. Καλεί την find_collisions_dt και αποθηκεύει τα UAVs με τα οποία το κάθε αεροσκάφος συγκρούεται.

Μέρος 2^{ο}: Εύρεση της βέλτιστης κατεύθυνσης. Με βάση τις συγκρούσεις του κάθε αεροπλάνου αποφασίζει την βέλτιστη κατεύθυνση (μέσω της find_best_direction) προς την οποία πρέπει να κινηθεί, σε αυτήν την περίπτωση, η βέλτιστη αυτή κατεύθυνση είναι το αντίθετο του κανονικοποιημένου αθροίσματος των διανυσμάτων απόστασης προς τα UAVs με τα οποία συγκρούεται.

Μέρος 3ο: Επανέλεγχος για συγκρούσεις. Η παραπάνω στρατηγική δεν μας εγγυάται ότι με την νέα κατεύθυνση που επιλέχθηκε δεν θα συγκρούεται με τα ίδια (ή και με άλλα) UAVs, οπότε κάθε UAV,μόνο αν δεν συγκρούεται με κανένα άλλο (με βάση τις νέες ταχύτητες), μπορεί να κινηθεί.

Τέλος, αναφέρεται ότι το κάθε frame καλείται ανά (τουλάχιστον) dt δευτερόλεπτα (πραγματικού χρόνου) όπως φαίνεται στην μέθοδο on_idle της κλάσης Airspace. Βέβαια λόγω των καθυστερήσεων στον εντοπισμό συγκρούσεων, στην μετακίνηση αεροσκαφών, κ.α., η μόνη εγγύηση είναι ότι το κάθε frame διαρκεί πάνω από dt δευτερόλεπτα πραγματικού χρόνου.
Το dt μπορεί να οριστεί από τον διάλογο στο τερματικό (όπου ορίζονται και άλλοι παράμετροι) αλλά συνίσταται να είναι κοντά στο 0.15.

Ερώτημα 8. (Πρωτόκολλα επίτευξης στόχου)

«Ορίστε πρωτόκολλο προσγείωσης-απογείωσης ώστε να μην υπάρχει σύγκρουση. Το πλήθος των drones να είναι παραμετρικό όπως και η χρονική τους εμφάνιση και ταχύτητα τυχαία..»

Όπως αναφέρθηκε, για την περίπτωση αυτή θα χρησιμοποιηθούν τρία πρωτόκολλα, ένα για την απογείωση, ένα για την προσγείωση και ένα για την ταυτόχρονη απογείωση και προσγείωση UAVs.

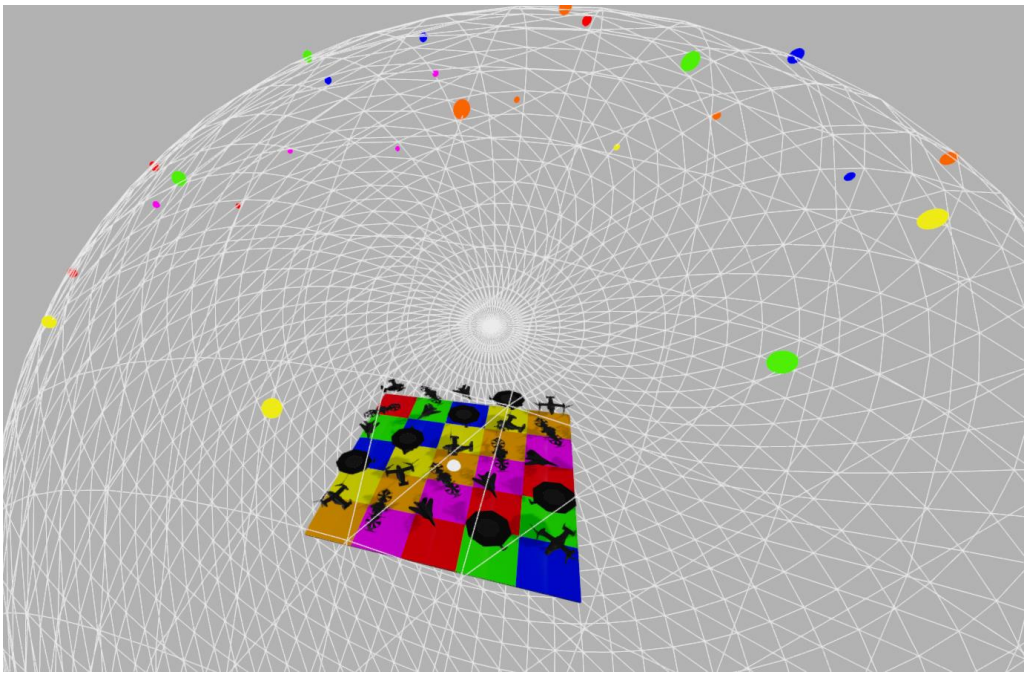
Ακόμη το ερώτημα θα χωριστεί σε 3 υποερωτήματα. Ένα για την απογείωση (με στόχους πάνω σε ένα dome), την προσγείωση (από το dome στα landing_spots) και την προσγείωση/απογείωση τυχαία εμφανιζόμενων UAVs εντός του dome.

Take off (8.1)

Σε αυτή την υλοποίηση (η αρχικοποίηση της οποία γίνεται με την μέθοδο create_taking_off_uavs() της κλάσης Airspace) τα NxN UAVs ξεκινούν από τις θέσεις απογείωσης και όλα μαζί απογειώνονται με σκοπό να φτάσουν τυχαία τοποθετημένα beacons πάνω στο dome με δοθείσα ακτίνα.

Για τον σκοπό αυτό έχει δημιουργηθεί η μέθοδος protocol_target_beacon, η οποία αποτελεί μια τροποποιημένη μέθοδο της protocol_avoidance, αφού σε αυτήν την περίπτωση τα UAVs δεν αποφεύγουν απλά τα κοντινά τους αεροσκάφη, αλλά ταυτόχρονα κινούνται προς έναν στόχο.

Έτσι η ειδοποιός διαφορά στην υλοποίηση αυτού του πρωτοκόλλου είναι ότι η συνάρτηση find_best_direction αντί απλά να δίνει το διάνυσμα αποφυγής (δηλαδή το αντίθετο του κανονικοποιημένου αθροίσματος διανυσμάτων απόστασης προς τα συγκρουόμενα UAVs) για κάθε UAV, προσθέτει σε αυτό το κανονικοποιημένο διάνυσμα κατεύθυνσης προς τον στόχο, που σε αυτήν την περίπτωση είναι το beacon που αντιστοιχεί στο UAV.



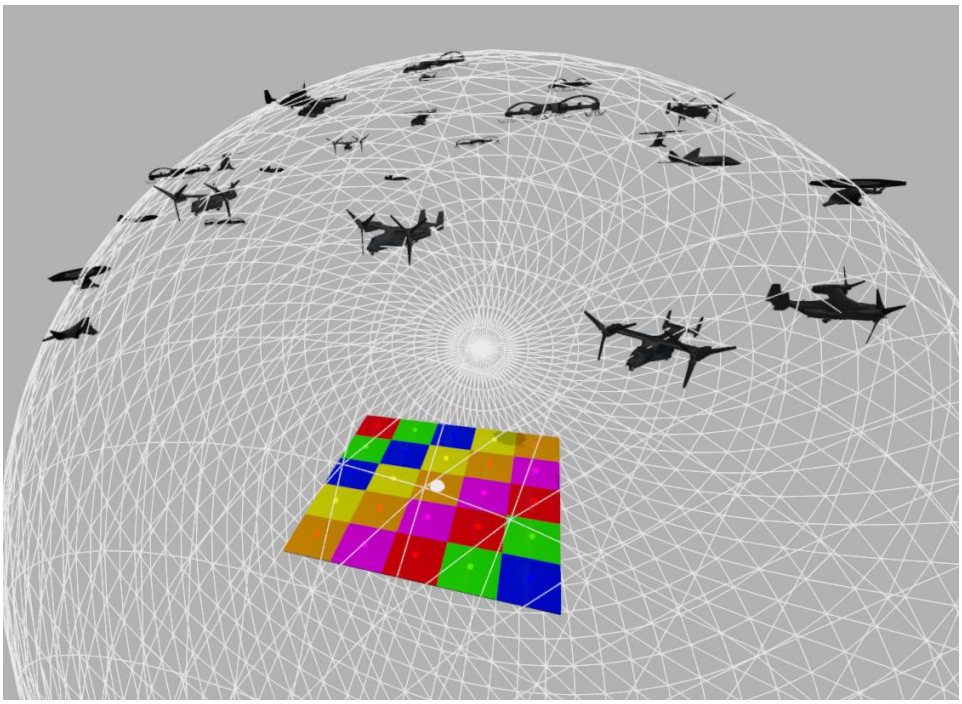
Εικόνα 16. Παράδειγμα αρχικοποίησης ερωτήματος 8.1

Landing (8.2)

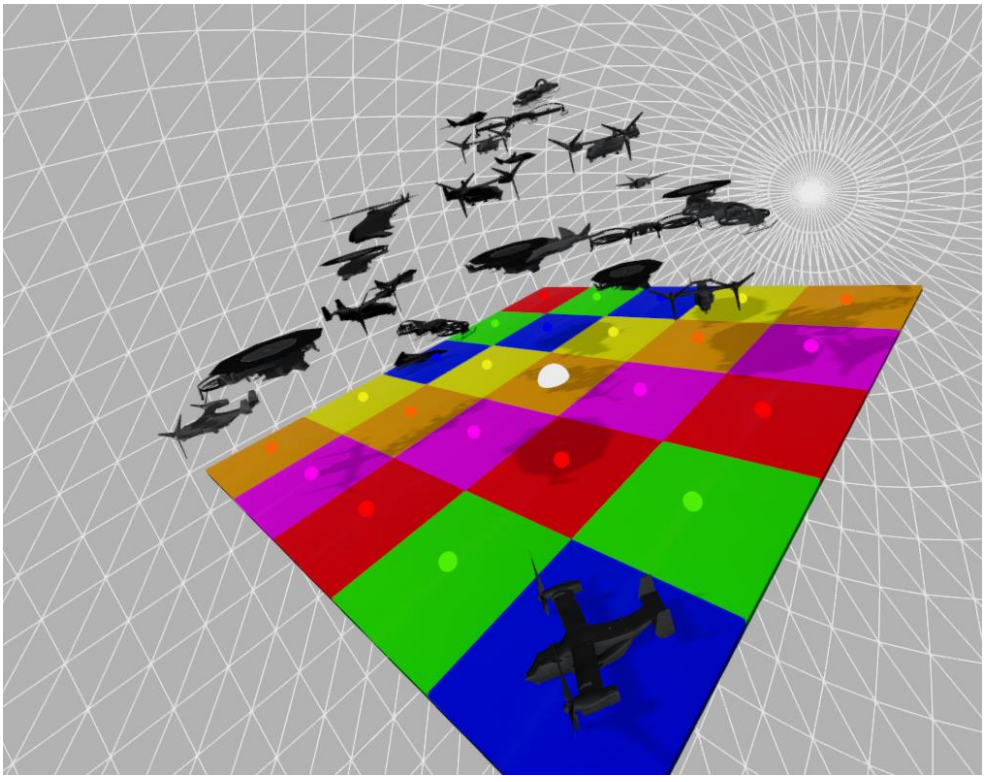
Σε αυτήν την υλοποίηση χρησιμοποιείται η αρχικοποίηση/μέθοδος create_landing_uavs() όπου τα UAVs και τα beacons του προηγούμενου ερωτήματος αλλάζουν θέσεις. Δηλαδή τα NxN UAVs ξεκινάνε από τον θόλο/dome και έχουν ως στόχους τα landing_spots στην επιφάνεια προσγείωσης.

Η υλοποίηση αυτή χρησιμοποιεί το πρωτόκολλο/μέθοδο protocol_landing το οποίο είναι ίδιο με το protocol_target_beacon, με την μόνη διαφορά να έγκειται ότι ο στόχος δεν είναι πλέον beacon, αλλά landing_spot.

Όπως βλέπουμε παρακάτω, η συγχρονισμένη «επιθυμία» των UAVs να κινηθούν προς την ίδια περιοχή δημιουργεί μεγάλο συνωστισμό πάνω από την επιφάνεια προσγείωσης, χωρίς όμως να παρατηρούνται συγκρούσεις!



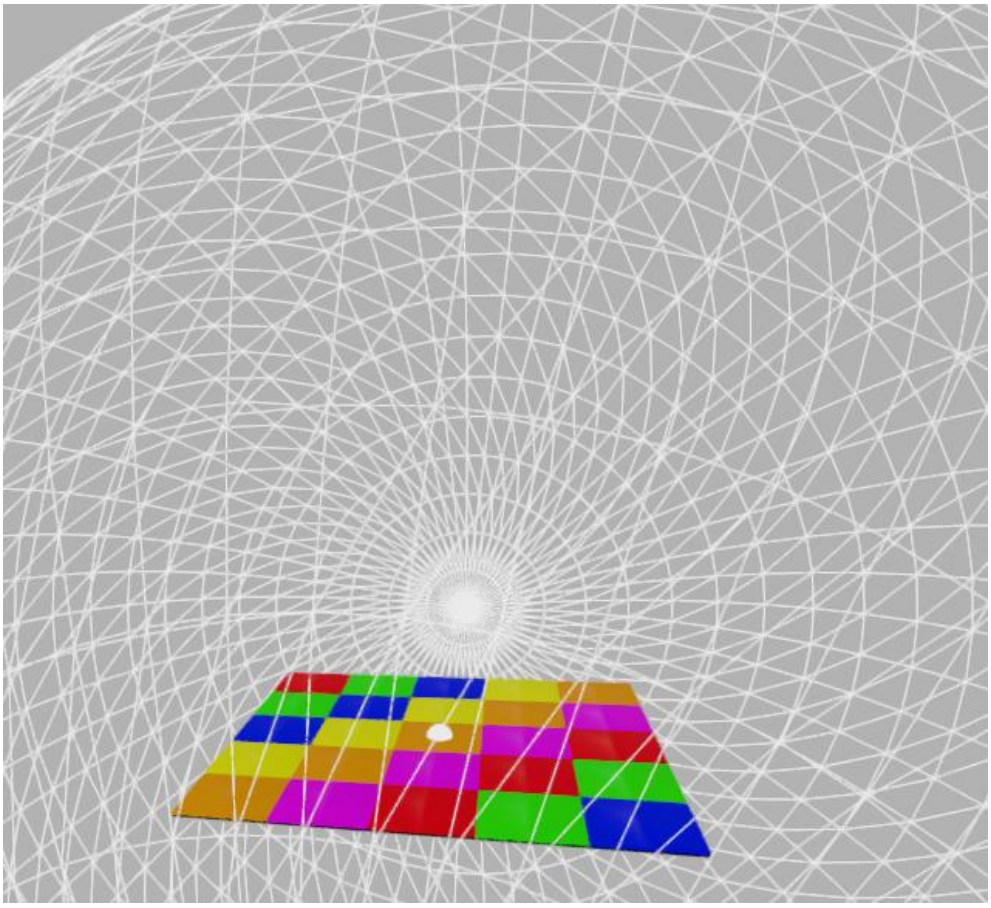
Εικόνα 17. Αρχικοποίηση ερωτήματος 8.2



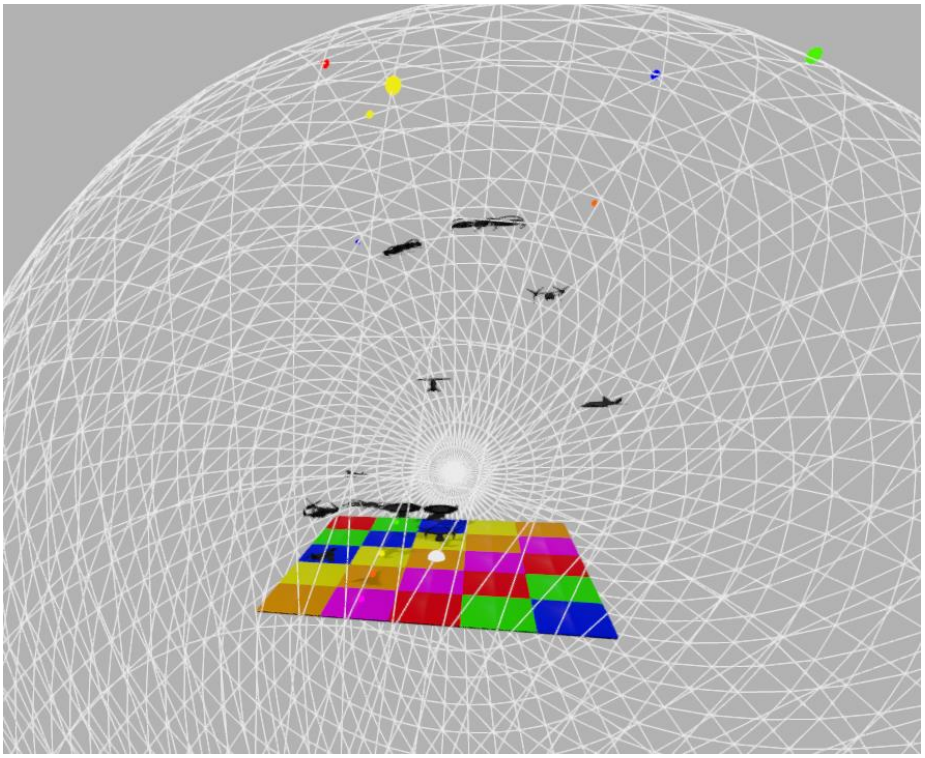
Εικόνα 18. Ενδιάμεση κατάσταση ερωτήματος 8.2

Landing_take_off (8.3)

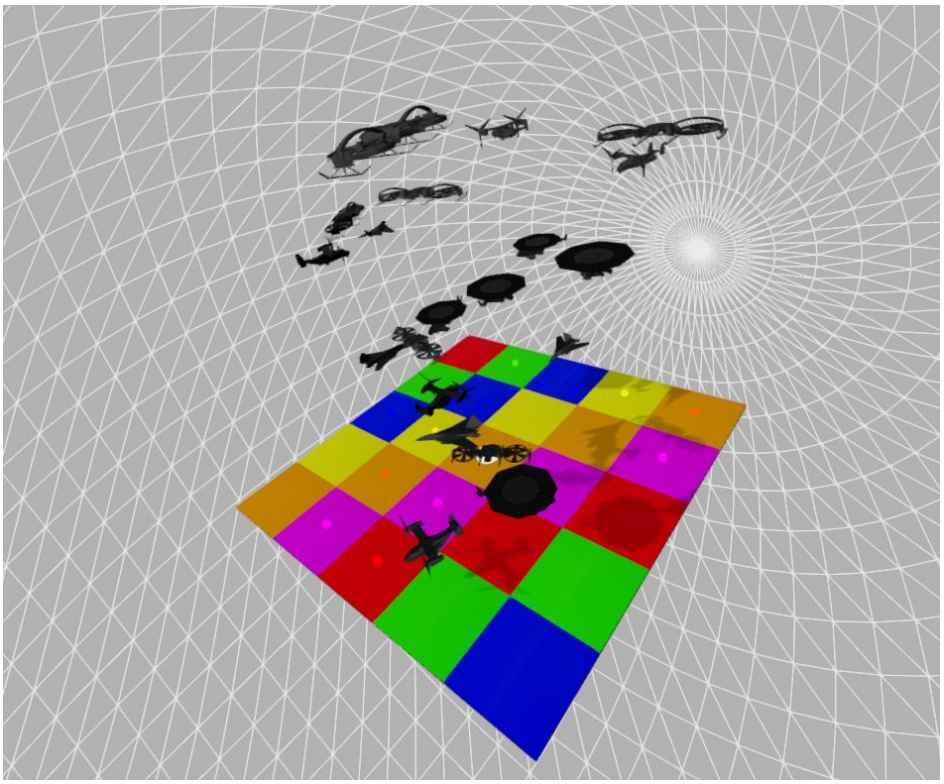
Η υλοποίηση 8.3 είναι παρόμοια με τις δύο προηγούμενες, με την κύρια διαφορά να είναι ότι δεν έχουν όλα τα αεροσκάφη τον ίδιο τύπο προορισμού, μερικά από αυτά εμφανίζονται στις θέσεις προσγείωσης και έχουν στόχο τα beacons, ενώ τα υπόλοιπα εμφανίζονται στον αέρα και έχουν ως στόχο τις θέσεις προσγείωσης. Η λογική εύρεσης της καλύτερης διεύθυνσης κίνησης είναι ακριβώς η ίδια με πριν, μόνο που τώρα εξετάζεται ο στόχος του συγκεκριμένου αεροπλάνου είτε είναι landing_spot είτε είναι beacon.
Ακόμη, η δημιουργία των αεροσκαφών δεν γίνεται στο πρώτο frame όπως μέχρι τώρα, αλλά κατά τη διάρκεια της προσομοίωσης. Σε κάθε frame υπάρχει μια πιθανότητα (flow, η οποία ορίζεται στο τερματικό κατά την διάρκεια εισαγωγής παραμέτρων) να εμφανιστεί ένα νέο UAV, η πιθανότητα αυτό να απογειώνεται είναι 50%.



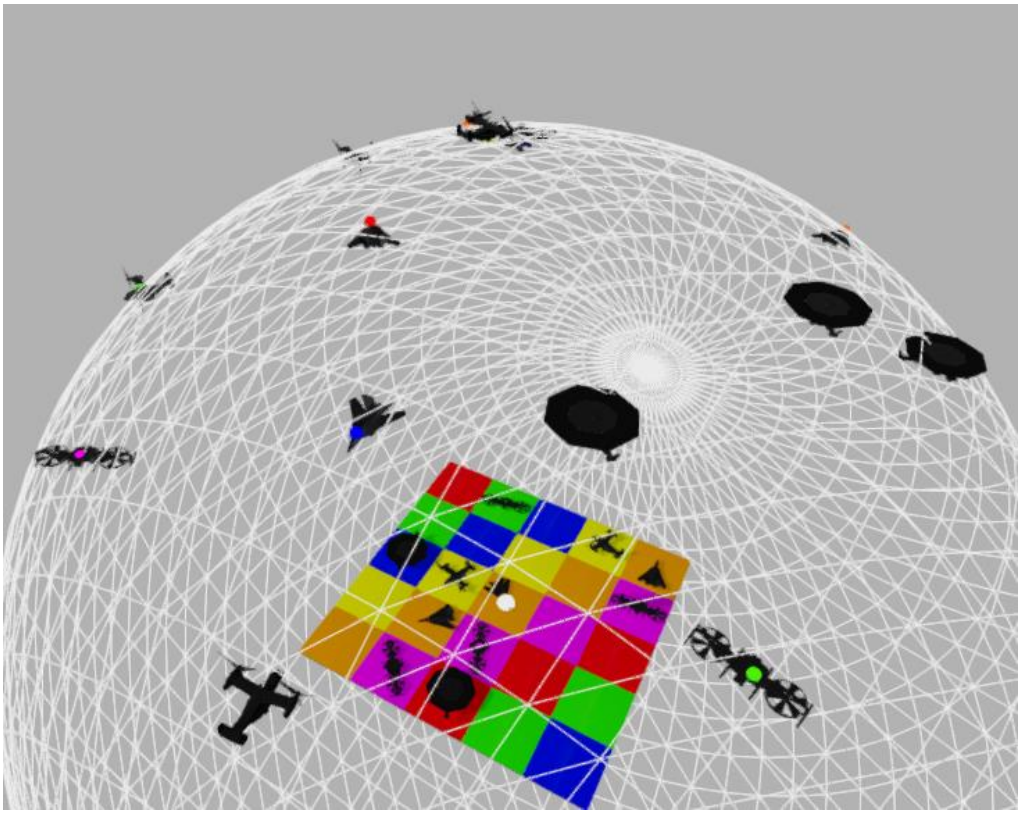
Εικόνα 19. Αρχικοποίηση του ερωτήματος 8.3



Εικόνα 20. Ενδιάμεση κατάσταση του ερωτήματος 8.3



Εικόνα 21. Πιο προχωρημένη κατάσταση του 8.3



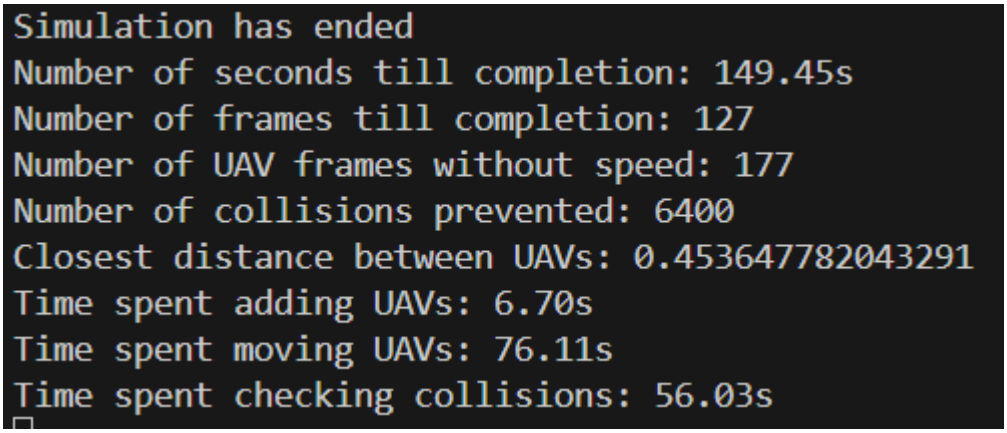
Εικόνα 22. Τελική κατάσταση του 8.3

Ερώτημα 9. (Στατιστικά)

«Απεικονίστε την προσομοίωση και ενδιαφέροντα στατιστικά στοιχεία».

Σε όλες τις προσομοιώσεις που έχουν τελική κατάσταση (δηλαδή αυτές του ερωτήματος 8), συλλέγονται μερικά στατιστικά τα οποία εκτυπώνονται μετά το πέρας της προσομοίωσης (το οποίο ελέγχεται με την μέθοδο simulation_has_ended() της κλάσης Airspace).

Ένα παράδειγμα τέτοιων στατιστικών μετά την εκτέλεση του ερωτήματος 8.2 είναι το παρακάτω:



Εικόνα 23. Στατιστικά μετά την εκτέλεση του 8.2

Όπου:

1. Number of seconds till completion: Πραγματικός χρόνος εκτέλεσης της προσομοίωσης

2. Number of frames till completion: Αριθμός frames μέχρι την ολοκλήρωση της προσομοίωσης

3. Number of UAV frames without speed: Το άθροισμα των frames για τα οποία ένα UAV είχε μηδενική ταχύτητα (Αν π.χ. σε ένα frame 2 UAVs είχαν μηδενική ταχύτητα, τότε ο αριθμός αυτός θα ήταν 2)

4. Number of collisions prevented: Ο αριθμός των συγκρούσεων που εντοπίστηκαν και αποφεύχθηκαν συνολικά στην διάρκεια της προσομοίωσης.

5. Closest distance between UAVs: Μικρότερη απόσταση μεταξύ κέντρων UAVs

6. Time spent adding UAVs: Χρόνος που δαπανήθηκε για να προστεθούν τα μοντέλα στην σκηνή (στην αρχή της προσομοίωσης)

7. Time spent moving UAVs: Χρόνος που δαπανήθηκε στην κίνηση των UAVs (αυτός είναι περιορισμός της open3D, αφού -όπως είναι λογικό- η μετακίνηση αντικειμένων των 50.000 σημείων σε κάθε frame απαιτεί χρόνο)

8. Time spent checking collisions: Ο χρόνος που δαπανήθηκε για την εύρεση και αποφυγή συγκρούσεων. Αξιοσημείωτο το γεγονός ότι είναι μικρότερος του χρόνου μετακίνησης των αντικειμένων.

Οδηγίες εγκατάστασης

Ακολουθούν οδηγίες για την εγκατάσταση και την εκτέλεση του προγράμματος:

1. Δημιουργία κλώνου του repository ή κατέβασμα του αρχείου zip και εξαγωγή αυτού.
2. (Προαιρετικό) Δημιουργία νέου εικονικού περιβάλλοντος conda/venv
3. Ικανοποίηση των απαιτήσεων που βρίσκονται στο αρχείο requirements.txt με την εντολή *pip install -r requirements.txt*
4. Εκτέλεση του αρχείου *main.py*

(Αναλυτικές οδηγίες βρίσκονται και στο αρχείο *README.md* του project)

Χρήση

Ακολουθούν αναλυτικές οδηγίες για την χρήση του προγράμματος.

1. Μετά την εκτέλεση του αρχείου main.py, θα ξεκινήση ένας διάλογος στο τερματικό όπου μας ζητείται να ορίζουμε αρχικά ποιο ερώτημα θέλουμε να εκτελεστεί (1-8) και άλλες παραμέτρους όπως
   a. Τον συνολικό αριθμό Drones (πρέπει να είναι τετράγωνο πχ 25)
   b. Την παράμετρο dt (ερωτήματα 6+) (συνίσταται τιμή 0.15)
   c. Την παράμετρο flow (ερώτημα 8.3) (συνίσταται μεγαλύτερη του 0.5)

2. Στην συνέχεια θα δημιουργηθεί το περιβάλλον του εναέριου χώρου όπου ανάλογα με το ερώτημα θα παρουσιάζει την κατάλληλη συμπεριφορά όπως αναφέρθηκε παραπάνω.
3. Σε ερώτημα έχουμε την δυνατότητα αλληλεπίδρασης με την προσομοίωση με τα εξής:
   a. K (εμφάνιση/απόκρυψη KDOP για όλα τα αεροσκάφη)
   b. C (εμφάνιση/απόκρυψη Convex Hull για όλα τα αεροσκάφη)
   c. U (εμφάνιση/απόκρυψη Unit Sphere για όλα τα αεροσκάφη)
   d. B (εμφάνιση/απόκρυψη AABB για όλα τα αεροσκάφη)

e. L (εντοπισμός και εμφάνιση στιγμιαίων συγκρούσεων στον εναέριο χώρο)
f. P (Pause/unpause για τα ερωτήματα που αφορούν προσομοιώσεις με χρόνο).
Κάθε προσομοίωση ξεκινά από κατάσταση Paused.
Για την κίνηση της κάμερας στην διάρκεια προσομοίωσης συνίσταται η πρότερη χρήση του Pause για την ομαλή λειτουργία της κάμερας.
g. T (Εμφάνιση/απόκρυψη των συγκρούσεων των continuous kdops οι οποίες αποφεύγονται).
h. Σε κάθε ερώτημα έχουμε την δυνατότητα χειρισμού ενός αεροσκάφους. Το αεροσκάφος αυτό είναι το πρώτο v22_osprey που φορτώθηκε (όνομα: v22_osprey_0). Οι δυνατές λειτουργίες του είναι οι εξής:
- Arrow keys (για κίνηση εμπρός, πίσω, δεξιά, αριστερά σύμφωνα με τους άξονες)
- R (για ωρολογιακή περιστροφή κατά π/4)
- Space (για ανύψωση)
- Backspace (για κατάβαση)

4. Μετά το πέρας της προσομοίωσης (για τα ερωτήματα 8.x) εμφανίζονται στο τερματικό στατιστικά όπως αναφέρθηκαν στο ερώτημα 9.

Στον παρακάτω σύνδεσμο βρίσκεται ένα βίντεο εκτέλεσης του ερωτήματος 8.3 με τις default παραμέτρους:
https://youtu.be/xLuo_WNb07s

Βιβλιογραφία

1. Open3D: Qian-Yi Zhou and Jaesik Park and Vladlen Koltun (2018).
   A Modern Library for 3D Data processing

2. Numpy: Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020).
   Array programming with Numpy.

3. Trimesh: Dawson-Haggerty, S. (2019).
   A python library, in this case used to find mesh collisions.

4. Vvrpywork: Εργαστήριο 3D, Τμήμα ΗΜΤΥ.
   Μία βιβλιοθήκη-wrapper για πολλές από τις λειτουργίες της Open3D.