3Δ ΥΠΟΛΟΓΙΣΤΙΚΗ ΓΕΩΜΕΤΡΙΑ ΚΑΙ ΟΡΑΣΗ

UAV COLLISION DETECTION AND PATH PLANNING

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΕΠΤΕΜΒΡΙΟΣ 2024

ΖΕΥΓΟΥΛΑ ΝΕΚΤΑΡΙΑ

ΑΜ: 1089133

ΤΕΤΑΡΤΟ ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ

Contents

ΠΕΡΙΓΡΑΦΗ ΘΕΜΑΤΟΣ

Στην εργασία αυτή γίνεται επεξεργασία και απεικόνιση διάφορων διεργασιών για τρισδιάστατα μοντέλα στον χώρο. Συγκεκριμένα, με την βιβλιοθήκη Open3D απεικονίζονται τα μοντέλα και χαρακτηριστικά αυτών (π.χ. κυρτό περίβλημα), εκτελούνται κινήσεις και γίνεται έλεγχος για τυχόν συγκρούσεις μεταξύ των μοντέλων. Ο χρήστης της εφαρμογής (μέσω του πληκτρολογίου) εκτελεί τα διάφορα ζητούμενα-tasks.

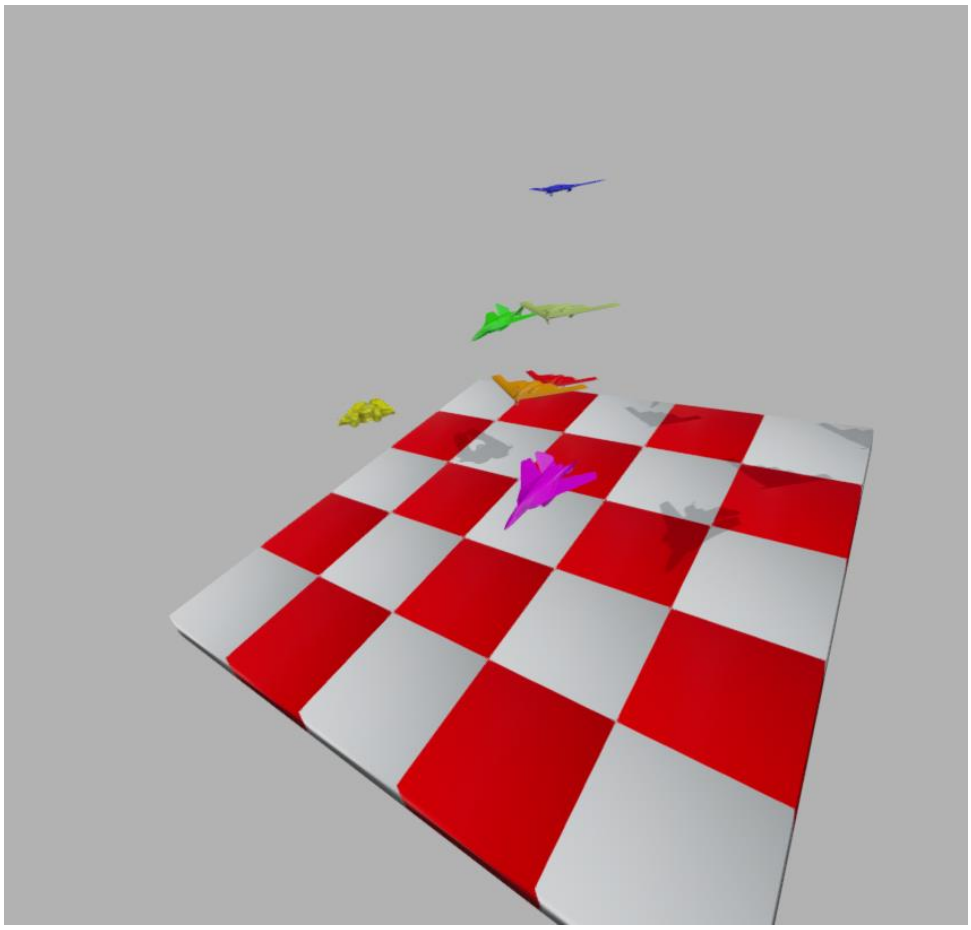
TASK-1

Επιλέξτε κάποια 3D μοντέλα drone και οπτικοποιήστε τα σε τυχαίες θέσεις. Επίσης μοντελοποιήστε μία επιφάνεια προσγείωσης απογείωσης με NxN θέσεις drones. Στη συνέχεια, για κάθε μοντέλο:

Τα βήματα που εκτελέστηκαν για την υλοποίηση του task αυτού, είναι τα εξής:

- ADD AND RANDOMIZE MESH POSITION:
    1. Αναζήτηση στο διαδίκτυο για κατάλληλα μοντέλα uav.Σημειώνεται, ότι πολλά μοντέλα δεν εμφανίζονταν σωστά, οπότε ο αριθμός των uavs που χρησιμοποιήθηκαν ήταν περιορισμένος.
    2. Αποθήκευση αρχείων .obj σε έναν φάκελο resources.
    3. Ό χρήστης πατώντας το πλήκτρο “U”, μπορεί να εμφανίσει (self.n – αριθμός uavs που θέλουμε να απεικονιστούν) uavs στον χώρο.
    4. Με το πάτημα του πλήκτρου αυτού καλείται η συνάρτηση addUavs με παράμετρο self.n
    5. Ανοίγονται με την Mesh3D n uav αρχεία.
    6. Γίνεται κανονικοποίηση του μεγέθους του uav με την unit_sphere_normalization. Δηλαδή τα uavs περιβάλλονται από την μοναδιάια σφαίρα. Αυτό γίνεται όταν οι συντεταγμένες όλων των σημείων διαιρούνται με την μέγιστη τιμής της νόρμας των σημείων.
    7. Με την randomize_mesh_position, επιστρέφεται το mesh με μία τυχαία θέση στον χώρο (μπορούμε να ορίσουμε τα όρια του χώρου αυτού). Για το ερώτημα αυτό, η περιστροφή του κάθε mesh είναι ίδια και σταθερή.
    8. Με την addShape, τοποθετούνται στο scene όλα τα uavs.
    9. Σημειώνεται ότι έχει οριστεί λεξικό self.uavs = {} το οποίο έχει για κλειδί το όνομα του κάθε uav (π.χ. “uav3”) και ως τιμή το αντίστοιχο αντικείμενο-mesh.
    10. Σημειώνεται ότι αν πατηθεί ξανά το “u”, διαγράφονται τα uavs από το scene αλλά και από το λεξικό.

- PLATFORM CREATIION
    1. Για την δημιουργία πλατφόρμας, θα απεικονίσουμε ένα σύνολο NxN κυβοειδών με ίδιο width, το ένα δίπλα στο άλλο .
    2. Με το πάτημα πλήκτρου “P”, καλείται η συνάρτηση createPlatform με παράμετρο τον αριθμό των uavs (n).
    3. Εδώ διατρέχεται ένα διπλό for-loop, όπου θα δημιουργήσει το NxN επίπεδο. Το N για λόγους απεικόνισης ισούται με τον αριθμό των uavs. Ωστόσο, όταν απεικονίζονται περισσότερα από 5 uavs, τότε η πλατφόρμα παραμένει 5x5.
    4. Για τον ορισμό του κάθε κυβοειδούς (Cuboid3D) χρειάζονται δύο σημεία τα οποία είναι τα άκρα της διαγώνιου του κυβοειδούς.
    5. Για την απεικόνιση του grid αυτού, εναλλάσσονται τα χρώματα (άσπρο-κόκκινο)
    6. Τα κυβοειδή αυτά προστίθενται στο scene και στο λεξικό plane = {}
    7. Σημειώνεται ότι όταν πατηθεί ξανά το “P” διαγράφονται τα coboids από το scene αλλά και από το λεξικό



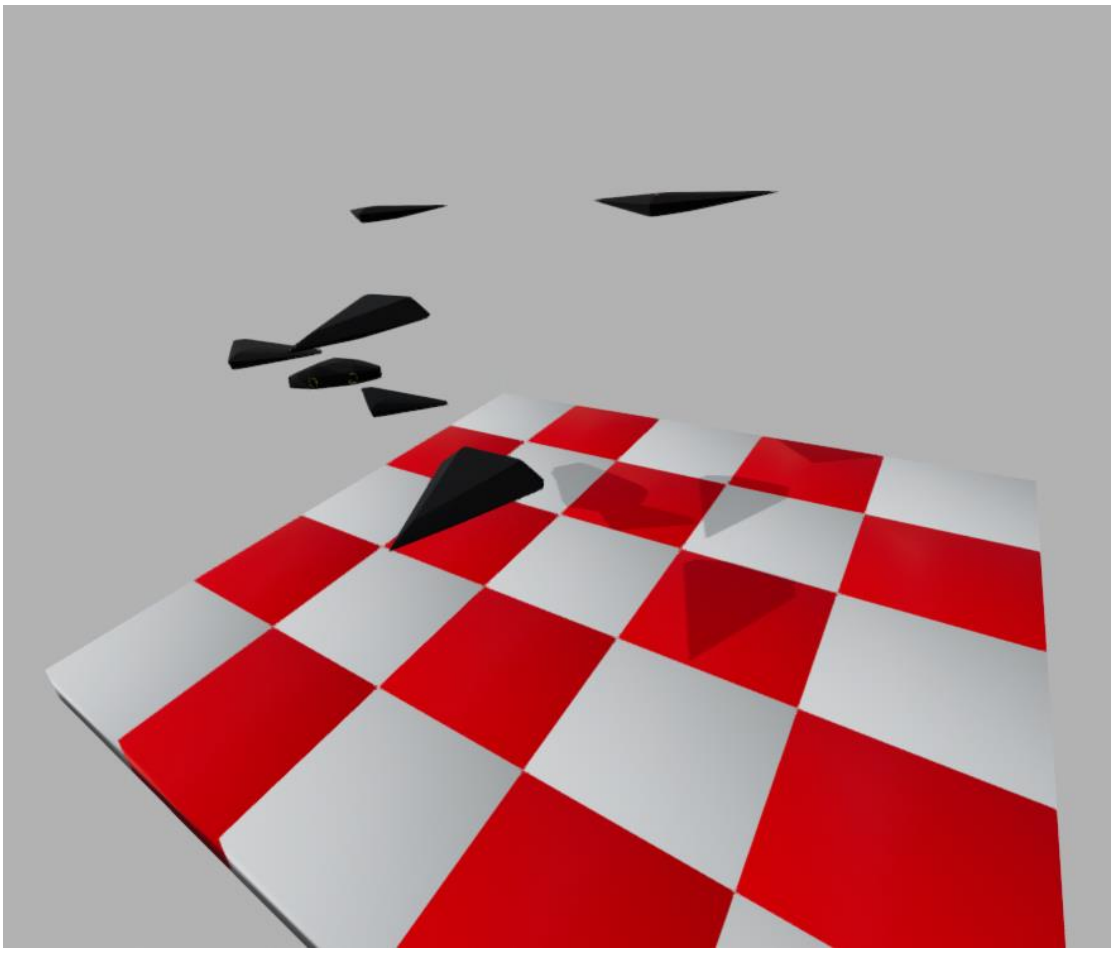
Εικόνα 1 Task 1 application

TASK-2

Υπολογίστε το κυρτό περίβλημα, AABB, και k-DOP

- ΚΥΡΤΟ ΠΕΡΙΒΛΗΜΑ – CONVEX HULL

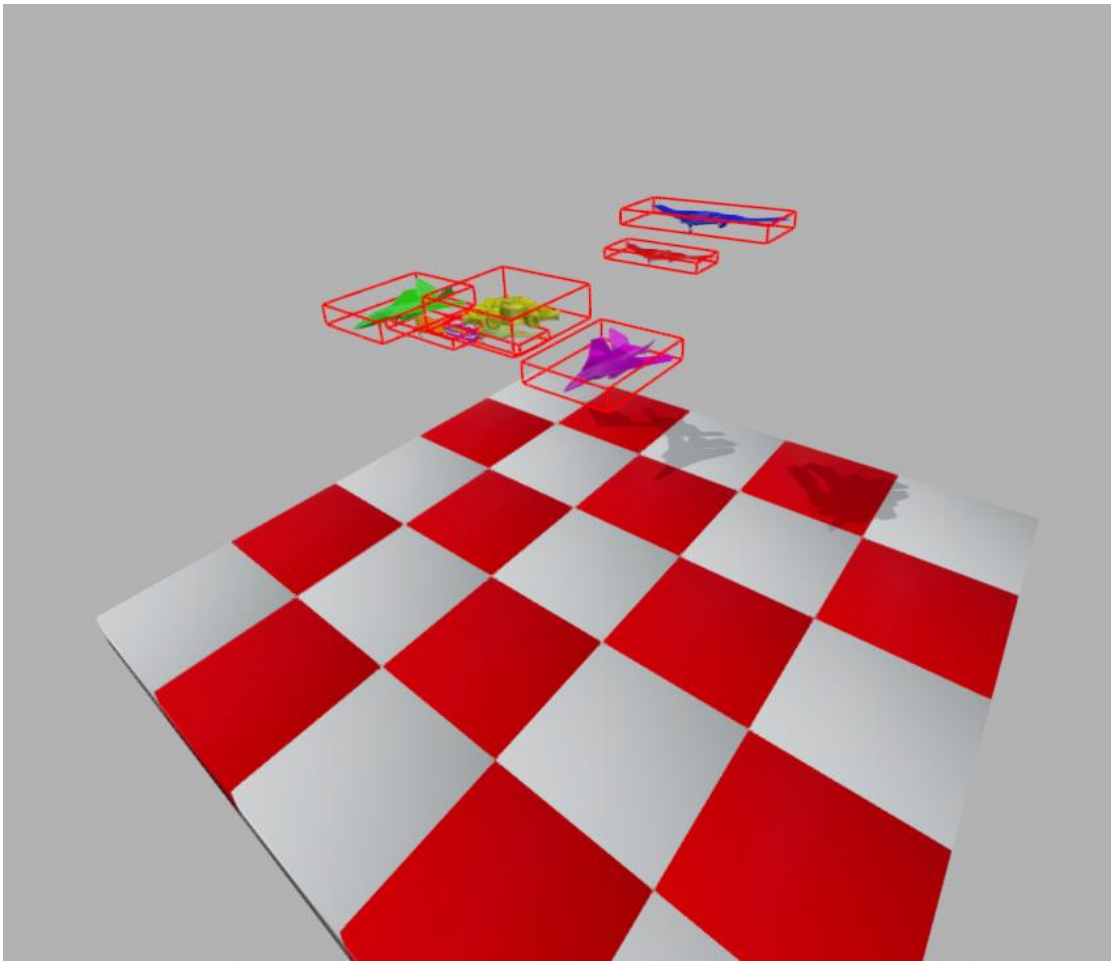
1. Ο χρήστης, αφού έχει εμφανίσει τα uavs στον χώρο, πληκτρολογώντας “X” μπορεί να εμφανίσει τα convex hulls όλων των uavs.
2. Τότε, για κάθε uav,μετατρέπεται το mesh σε o3d triangle mesh
3. Καλείται στη συνέχεια η compute_convex_hull για το triangle mesh (built in συνάρτηση).
4. Προστίθενται τα convex hulls στο scene, καθώς και στο λεξικό chullFromMesh



- AABB

1. Για την εύρεση και απεικόνιση των aabbs των uavs, πρέπει πρώτα αν υπολογίσουμε το aabb ενός mesh .
2. Μετατροπή του mesh σε open3d triangle mesh
3. Κάλεσμα της συνάρτησης get_axis_aligned_bounding_box και αποθήκευση στην μεταβλητή aabb
4. Μετατροπη του aabb σε cuboid μέσω της συνάρτησης aabb_to_cuboid. Η συνάρτηση αυτή βρίσκει την μικρότερη τιμή και την μεγαλύτερη (πιο απομακρυσμένη) συντεταγμένη του bounding_box και δημιουργεί ένα κυβοειδές με τα 2 σημεία αυτά.

5. Αποθήκευση στο λεξικό self.aabb και απεικόνιση στο scene



- KDOP

1. Το συγκεκριμένο ερώτημα δεν υλοποιήθηκε παρά τη παρατεταμένη έρευνα μου, λόγω περιορισμένης βιβλιογραφίας για την υλοποίηση στον τρισδιάστατο χώρο.

TASK-3

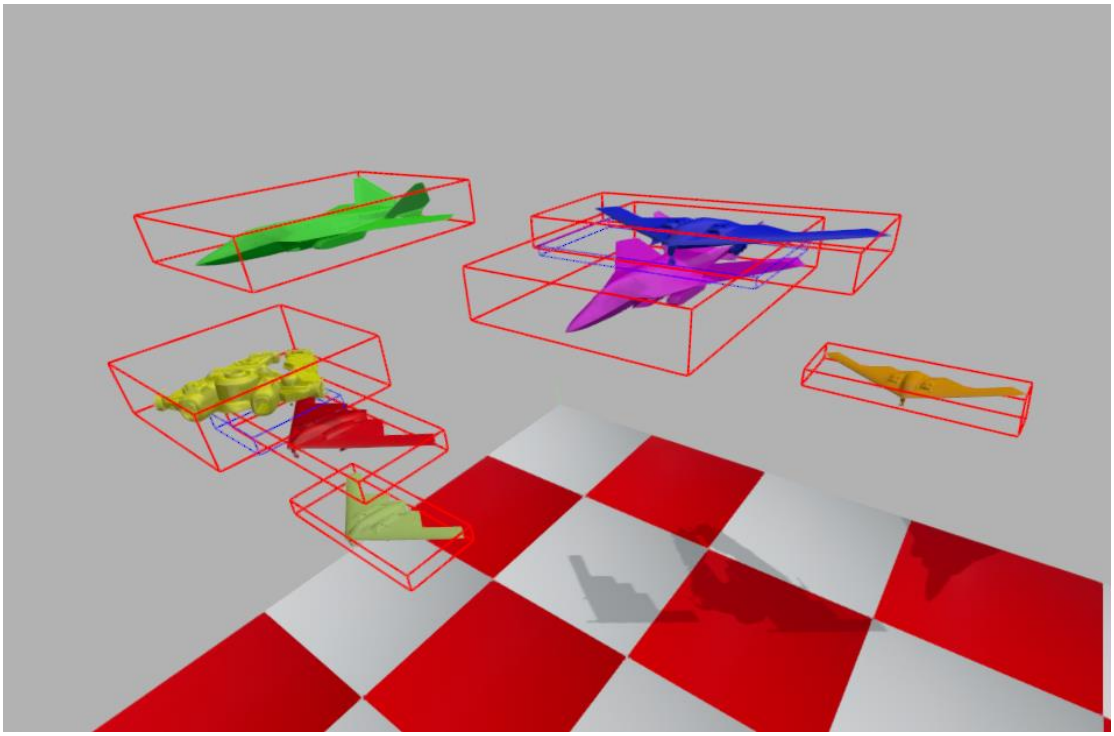
Υλοποιήστε και εκτελέστε αλγορίθμους ανίχνευσης σύγκρουσης Α βάσει μόνο των περιβαλλόντων όγκων.

Για την μέθοδο επιλογής μας, καθορίζουμε στην basiccollisionDetection (**key =”C”)** την αντίστοιχη παράμετρο π.χ. method = “aabb”

- AABB COLISSION DETECTION
    1. Για κάθε ζευγάρι aabb ((1,2),(1,3), (1, 4)…(2, 3), 2,4)…(3, 4)…), γίνεται έλεγχος σύγκρουσης.
    2. Ο έλεγχος σύγκρουσης στην ουσία ελέγχει αν κάποιο σημείο του $1^{ου}$ aabb είναι στον χώρο του $2^{ου}$. Δηλαδή αν για κάθε άξονα x, y, z γίνεται overlapping ταυτόχρονα. Για παράδειγμα στον άξονα x, πρέπει το xmin

του 1ου aabb να είναι μικρότερο ή ίσο από το xmax του 2ου aabb και το ανάποδο (xmax1 μεγαλύτερο ή ίσο από το xmin2) Αυτή η λογική ισχύει για κάθε άξονα, με τις αντίστοιχες συντεταγμένες.

3. Στην περίπτωση τομής δύο aabbs τότε εμφανίζεται το intersectingCuboid, το κυβοειδές της τομής τους. Το κυβοειδές αυτό έχει σημεία τις μικρότερες και τις μεγαλύτερες συντεταγμένες και των δύο κυβοειδών που τέμνονται, δηλαδη [xmin, ymin, zmin], [xmax, ymax, zmax], όπου xmin = min(aabb1.x_min, aabb2.x_min) κ.ο.κ.

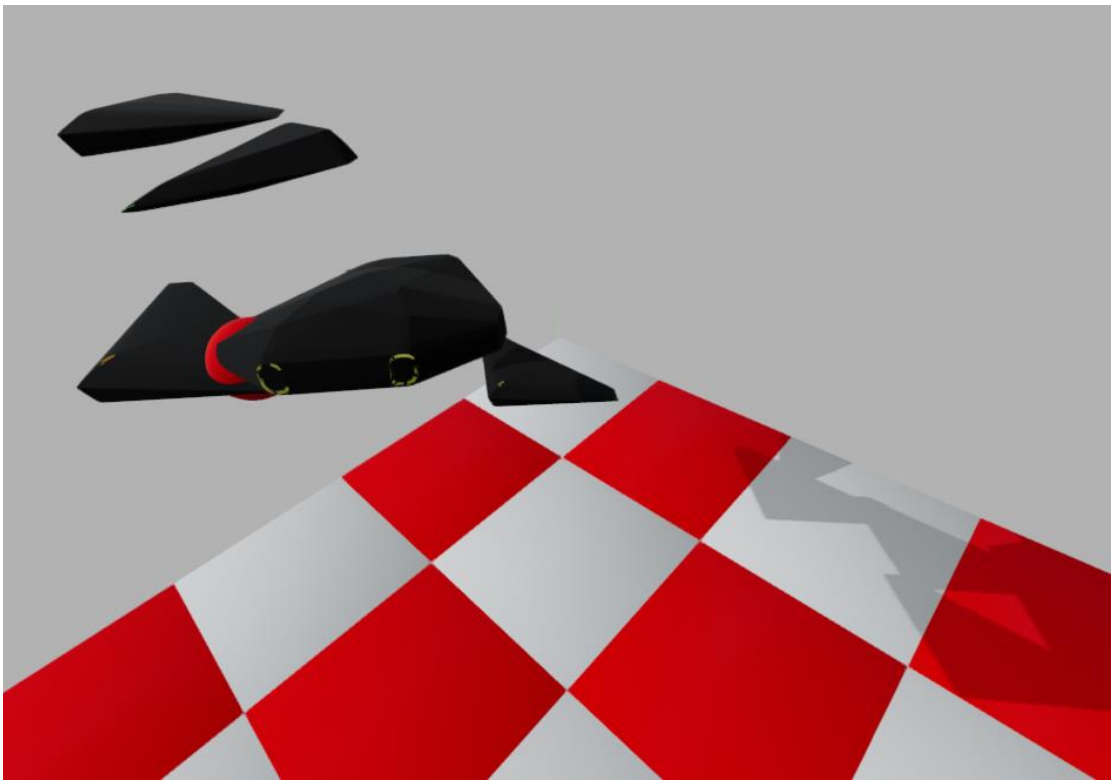


- CHULL COLLISION DETECTION
  1. Για την μέθοδο αυτή, φροντίζουμε να υπάρχουν τα convex hull όλων των uavs.
  2. Για κάθε μοναδικό και μη επαναλαμβανόμενο ζευγάρι convex hull, διατρέχεται η συνάρτηση chull_collision, με παράμετρο τα δύο αυτά convex hulls ενδιαφέροντος.
  3. Η συνάρτηση chull_collision, καλεί την mesh_intersection,η οποία
     a. Μετατρέπει τα meshes που δέχεται ως input, σε αντικείμενα trimesh.Trimesh.
     b. Καλεί την CollisionManager
     c. Καλεί την in_collision_internal η οποία επιστρέφει True, Collision_points (όταν υπάρχει collision) ή διαφορετικά False, None.

     Επομένως, η chull_collision επιστρέφει true ή false και τα σημεία επαφής (αν αυτά υπάρχουν).

  4. Στην συνέχεια,για κάθε σημείο σύγκρουσης, αυτό χρωματίζεται και αποθηκεύεται στο λεξικό intersectingPoints.

5. Γίνεται επανάληψη μέχρι να δοκιμαστούν όλα τα ζευγάρια.



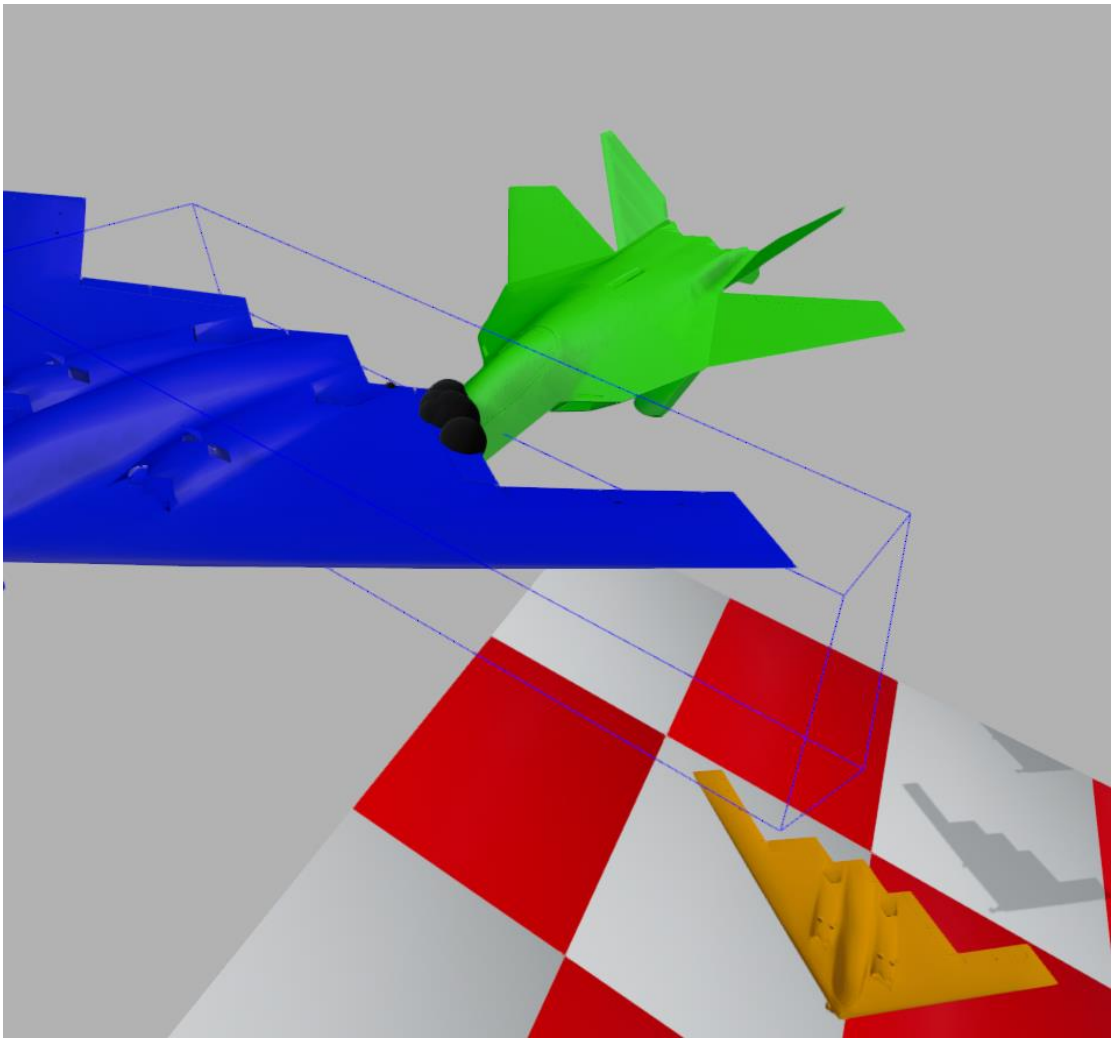
Με κόκκινο εμφανίζεται το σημείο επαφής

TASK-4

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

- ACCURATE COLLISION
  Η μέθοδος αυτή είναι η πιο ακριβής, αλλά και η πιο χρονοβόρα. Για τον λόγο αυτό πρώτα ελέγχεται για κάθε ζευγάρι αν υπάρχει aabbcollision και μετά chull collision και τέλος accurate collision.

  1. Για κάθε ζευγάρι λοιπόν, ελέγχεται το aabb collision, αν υπάρχει collision τότε ελέγχεται και το chull collision, και τότε και μόνο τότε ελέγχεται το accurate collision. Αυτό βελτιώνει σημαντικά την επίδοση του κώδικα.
  2. Η accurate collision δουλεύει ακριβώς όπως η mesh_intersection που αναλύθηκε στο task-4 chull collision detection (point 3.)

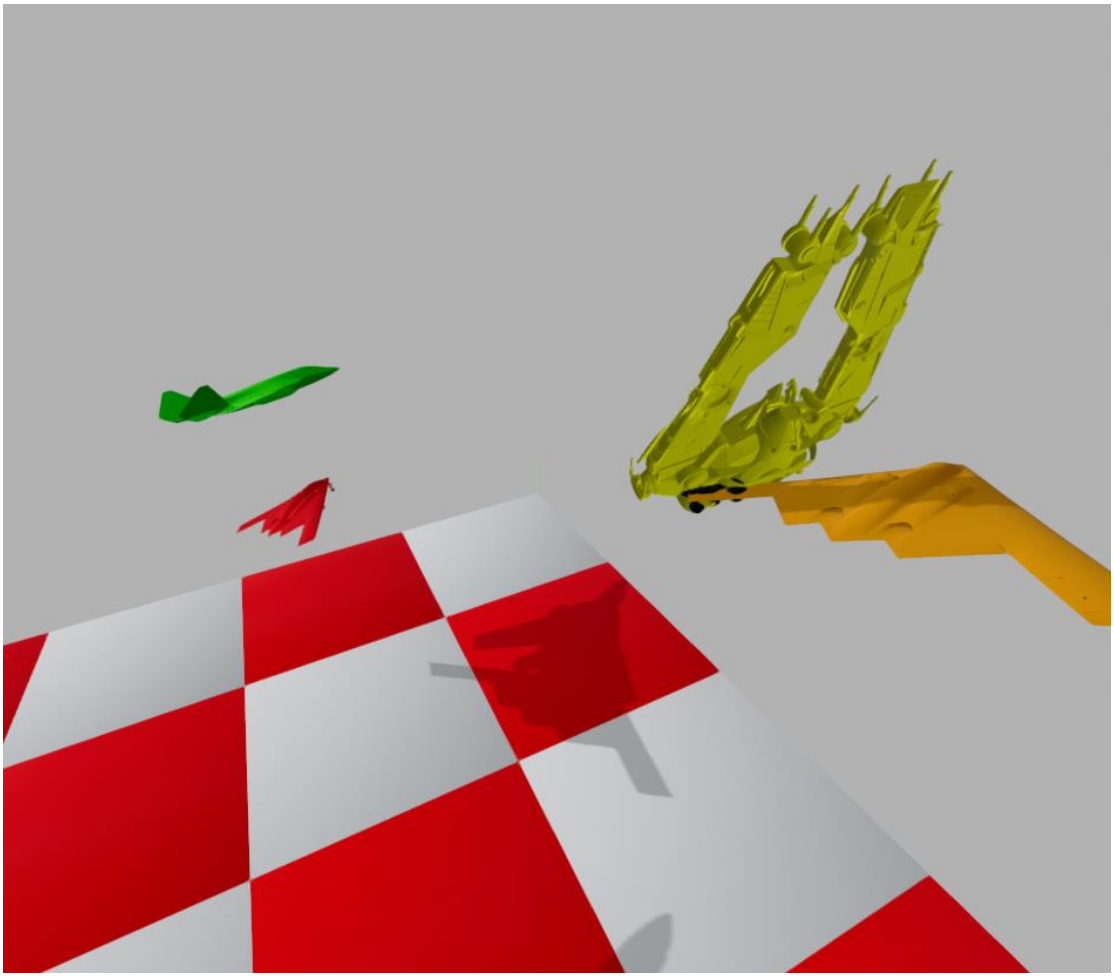


Με μαύρο εμφανίζονται τα σημεία τομής των δύο uav

TASK-5

Αρχικοποιήστε τυχαία τις θέσεις και τον προσανατολισμό των drones και απεικονίστε τις συγκρούσεις

- Τυχαία θέση και προσανατολισμός. **Πλήκτρο "R"**
  1. Στην δημιουργία των drones γίνεται η ίδια ακριβώς διεργασία με την ανωτέρα, αλλά καλείται και η συνάρτηση randomize_mesh_rotation.
  2. Δημιουργείται, δηλαδή, ένα τυχαίο rotation_matrix όπου $0<\theta<360$ μοίρες και το διάνυσμα r λαμβάνει τιμές 0 έως 1 για κάθε x, y,z
  3. Γίνεται, στη συνέχεια, εσωτερικό γινόμενο των σημείων του uav με την περιστροφή αυτή.
- Απεικόνιση συγκρούσεων.
  1. Για τα rotated uavs τώρα μπορεί να καλεστεί ακριβώς όπως και πριν η collision detection με την επιθυμητή μέθοδο.



Τα uav έχουν στραφεί, και με μαύρο απεικονίζονται τα σημεία τομής τους, με την ακριβή μέθοδο. Όλες οι μέθοδοι (aabb, convex hull και accurate collision) μπορούν να εφαρμοστούν.

TASK-6

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

Για την συνεχή ανανέωση του περιβλήματος, ο χρήστης πληκτρολογώντας το "**space**" καλείται η συνάρτηση check_moving_collision().

Σημειώνεται ότι οι **ταχύτητες** του κάθε drone είναι εντελώς **τυχαίες**. Το πρόβλημα που αντιμετωπίζουμε εδώ είναι ότι εφόσον η κίνηση των drones δεν είναι συνεχής, αλλά χωρίζεται σε frames, υπάρχει περίπτωση (σπάνια ωστόσο) να γίνει κάποιο **collision** στην θεωρητική θέση του uav **στην χρονική στιγμη μεταξυ (t, t+dt).** Για να λυθεί το πρόβλημα αυτό, ελέγχουμε τον χώρο που σαρώνει το aabb του uav για τον χρόνο αυτό. Ο χώρος αυτός θα έιναι το **convex hull του aabb** την στιγμη t και του aabb την στιγμη t+dt.

Οπότε, πρέπει να ελέγχουμε για την τομή των convex hulls. Αν αυτή υπάρχει, τότε τα colliding uavs θα μείνουν ακίνητα και τα υπόλοιπα θα πάρουν την επόμενη θέση (αυτή που ελέγξαμε)

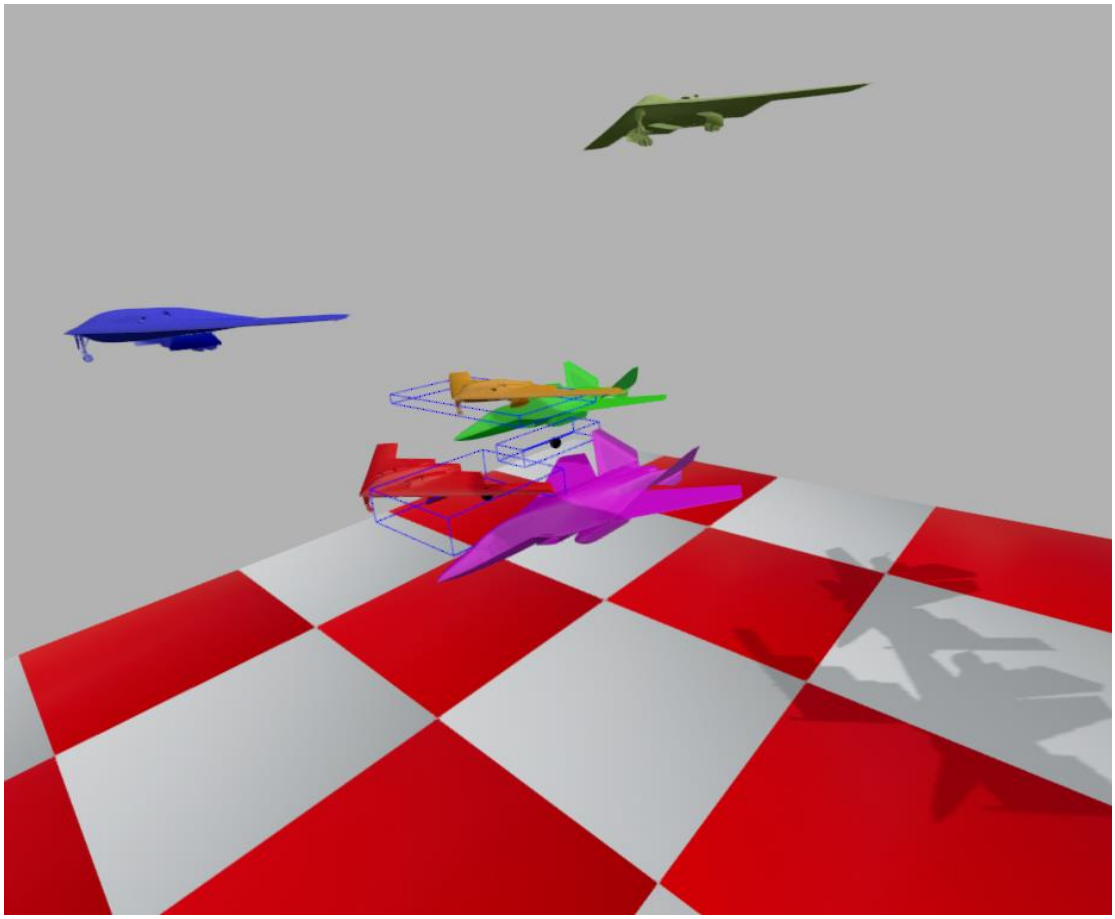
Οι βοηθητικές συναρτήσεις και λεξικό είναι οι εξής,

1. Get_velocities(uavs) επιστρέφει ένα λεξικό με random ταχύτητες για τα uavs.
2. Το fake_aabbs, στο λεξικό αυτό, βρίσκουμε το aabb στο επόμενο frame για κάθε uav.
3. Η aabbs_to_chull, η συνάρτηση αυτή λαμβάνει το τωρινό aabb και το πιθανό aabb (το aabb του επόμενου frame), και επιστρέφει το convex hull αυτών των δύο.

Τα βήματα που ακολουθούν είναι τα εξής.

1. Ορισμός μελλοντικών ταχυτήτων/θέσεων
2. Δημιουργία των μελλοντικών aabbs (t+dt) για κάθε uav
3. Δημιουργία των convex hull για τα aabb(t), aabb(t+dt)
4. Έλεγχος συγκρούσεις των convex hull.
5. Εμφάνιση του σημείου τομής των convex hull και του intersecting cuboid (αν αυτό υπάρχει. Αυτό εμφανίζεται μόνο όταν τα uavs ήδη συγκρούονταν από την αρχή. Διαφορετικά, εμφανίζεται μία μαύρη κουκκίδα στο σημείο που θα χτυπούσαν τα aabbs).
6. Ακινητοποίηση των uavs που θα χτυπήσουν
7. Μετακίνηση των uavs που δεν ανήκουν σε κάποια σύγκρουση.

Όσο το πρόγραμμα δεν είναι paused, αυτή η συνάρτηση θα τρέχει για πάντα ή μέχρι να συγκρουστούν και να ακινητοποιηθούν όλα τα uavs.



Με μαύρο εμφανίζεται το σημείο τομής των δύο convex hull (t, t+dt). Όσα uav τέμνονται, σταματούν να κινούνται. Με μπλε εμφανίζονται οι τομές μεταξύ των aabb για το επόμενο frame.

TASK-7

Ανανεώστε την ταχύτητα κάποιων οχημάτων ώστε να μην υπάρχει σύγκρουση.

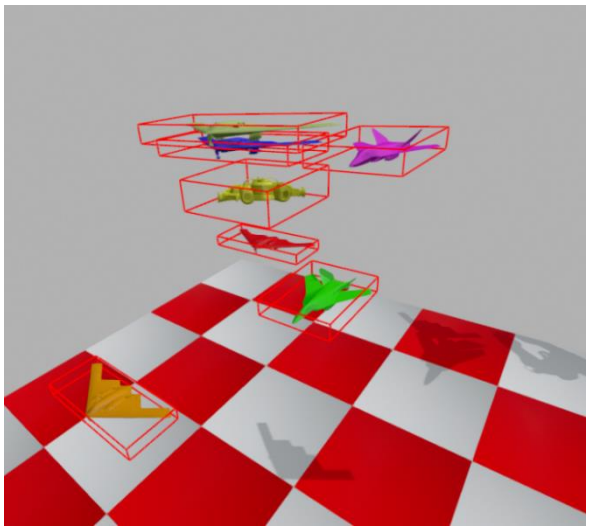
Η λογική της λειτουργίας αυτής μοιάζει αρκετά με αυτή του προηγούμενου ερωτήματος. Ωστόσο, όταν προβλέπει μελλοντική σύγκρουση, αντί να τα ακινητοποιεί, θα βρίσκει μια deflected velocity [$u_{DEFLECTED} = u_{COLLISION} - 2*(U_{COLLISION}*n)n$, όπου n νόρμα της επιφάνειας του σημείου σύγκρουσης. Ο τύπος αυτός συναντάται συχνά σε physics simulations – classical mechanics όταν κάποιο αντικείμενο συγκρούεται με κάποια επιφάνεια ή άλλο αντικείμενο.

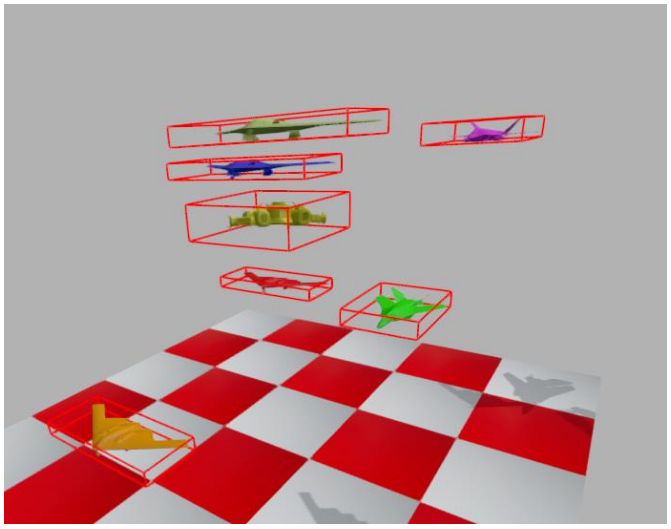
Τα βήματα που ακολουθούν είναι τα εξής.

1. Ορισμός μελλοντικών ταχυτήτων/θέσεων
2. Δημιουργία των μελλοντικών aabbs (t+dt) για κάθε uav
3. Δημιουργία των convex hull για τα aabb(t), aabb(t+dt)
4. Έλεγχος σύγκρουσης των convex hull.

5. Στην περίπτωση σύγκρουσης, για τα αντίστοιχα uavs, ανανέωσε τις μελλοντικές ταχύτητες του λεξικού, με τις deflected ταχύτητες.
6. Μετακίνηση όλων των uavs με τις μελλοντικες ταχύτητες.

Προφανώς, στην περίπτωση που τα uavs είναι ήδη σε σύγκρουση, τότε θα απομακρυνθούν. Επειδή το βήμα της ταχύτητας είναι αρκετά μικρό , η avoid collision θέλει κάποιες επαναλήψεις μέχρι την απομάκρυνση/αποφυγή.





Παρατηρούμε, ότι το μπλε και το λαχανί uav έχουν απομακρυνθεί.

TASK-8

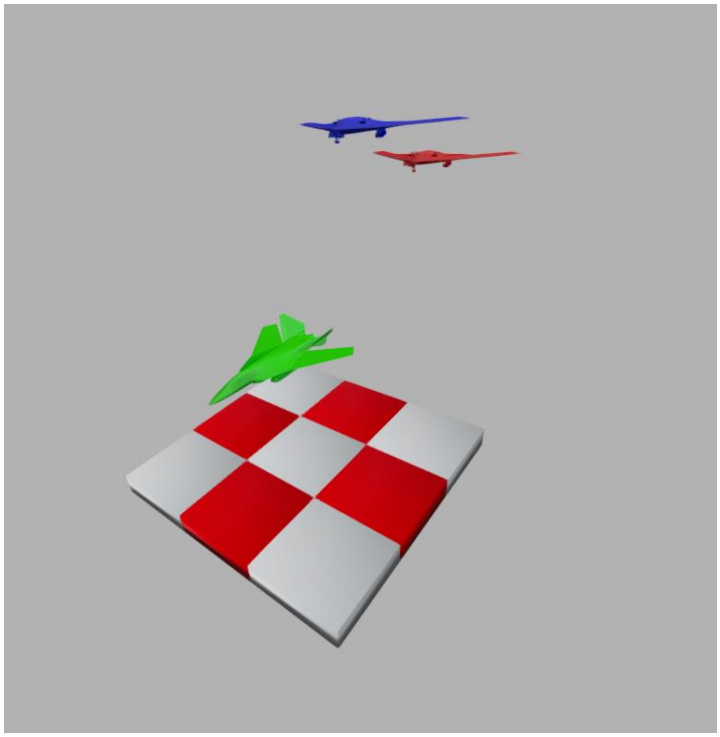
Ορίστε πρωτόκολλο προσγείωσης-απογείωσης ώστε να μην υπάρχει σύγκρουση. Το πλήθος των drones να είναι παραμετρικό όπως και η χρονική τους εμφάνιση και ταχύτητα τυχαία

Αρχικά, ορίστηκε μια συνάρτηση η οποία βρίσκει την ταχύτητα (διεύθυνση) που χρειάζεται ένα uav για να προσγειωθεί στο αντίστοιχο κυβοειδές της επιφάνειας προσγείωσης. Αυτό είναι αρκετά απλό, καθώς αρκεί να βρούμε το διάνυσμα που δείχνει από το uav (κεντρο αυτού) προς το κέντρο της επιφάνειας προσγείωσης.

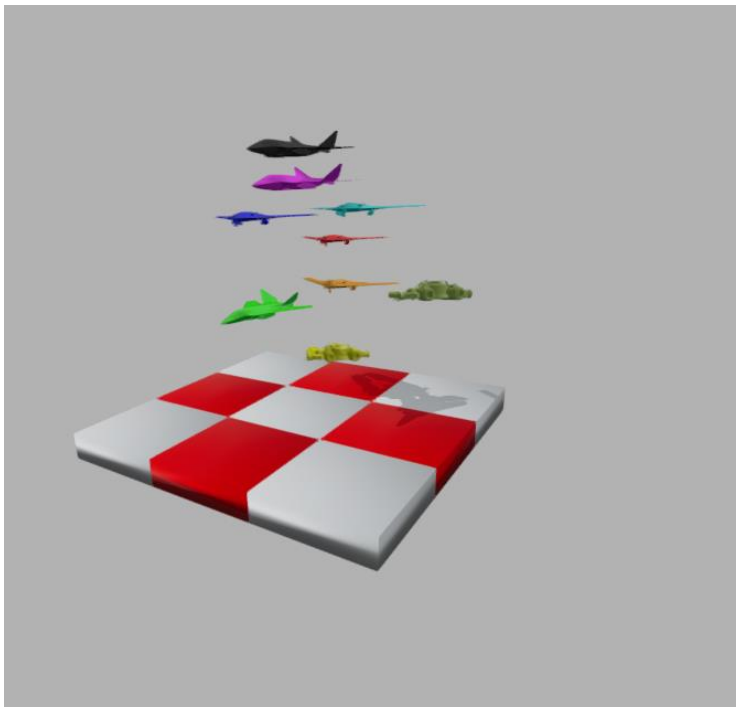
Χάρη στην ευελιξία του κώδικα αρκεί να τρέξει η avoidCollision με μοναδική αλλαγή την νέα ταχύτητα των drones. Αυτή η ταχύτητα θα είναι η κατευθυνόμενη ταχύτητα προσγείωσης.

Για την τυχαία εμφάνιση νέων drones, με την χρήση της random, όσο έχουμε λιγότερα uavs από θέσεις προσγείωσης (NxN), θα κάνουν spawn καινούργιά uavs, σε τυχαία θέση και σε τυχαίο χρόνο.

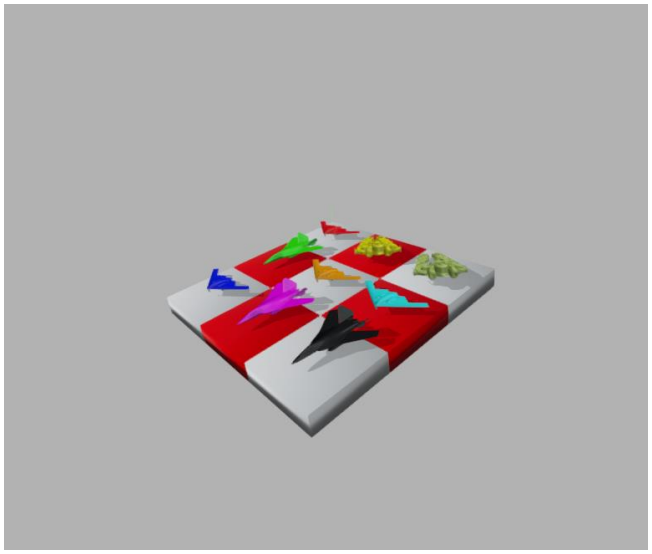
Πληκτρολογώντας το key “L” τότε όλα τα uavs απλά κινούνται κάθετα προς τα πάνω για την απογείωση τους.



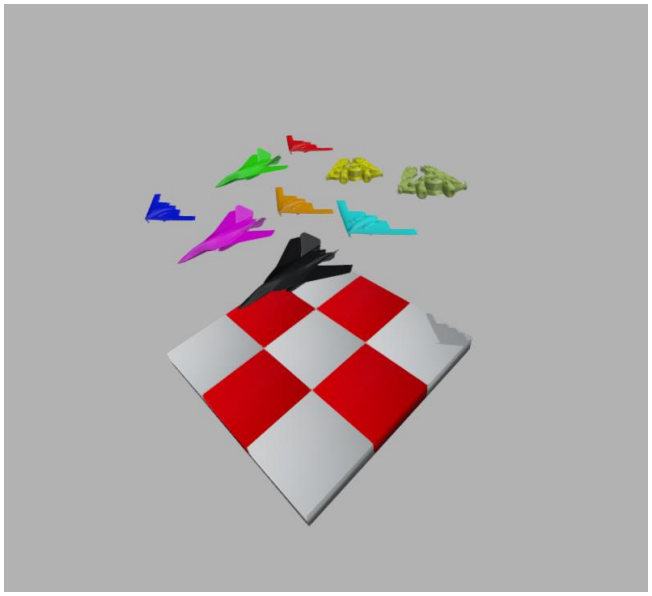
Στην αρχή εμφανίζονται 3 uav



Στην συνέχεια, εμφανίζονται και τα υπόλοιπα 3*3 – 3 = 6 uav



Ολοκληρώνεται η προσγείωση.



Πατώντας “L”, τα uavs απογειώνονται

TASK-9

Απεικονίστε την προσομοίωση και ενδιαφέροντα στατιστικά στοιχεία

Για την μέτρηση τον στατιστικών στοιχείων δημιουργήθηκε ένα λεξικό το οποίο έχει ως κλειδί τις διάφορες συναρτήσεις που καλούνται και οι τιμές είναι μία λίστα με τους χρόνους που καταγράφτηκαν. Το μήκος της λίστας αυτής αντιστοιχεί στον αριθμό των φορών που καλέστηκε η κάθε συνάρτηση. Τα στατιστικά αυτά αποθηκεύονται σε ένα json αρχείο με την μορφή που φαίνεται και παρακάτω. Για την αποθήκευση των στατιστικών που μετρήθηκαν σε ένα simulation πρέπει να πληκτρολογηθεί το πλήκτρο “Q” πριν τον τερματισμό του.

Ο κώδικας που χρειάστηκε είναι πολύ απλός. Πριν το κάλεσμα της κάθε συνάρτησεις θεωρούμε μία μεταβλητή time0 = time.time() και μετά το κάλεσμα αυτής, βρίσκουμε πάλι τον χρόνο ξανακαλώντας την time.time() και αποθηκεύοντας την διαφορά τους στο λεξικό.

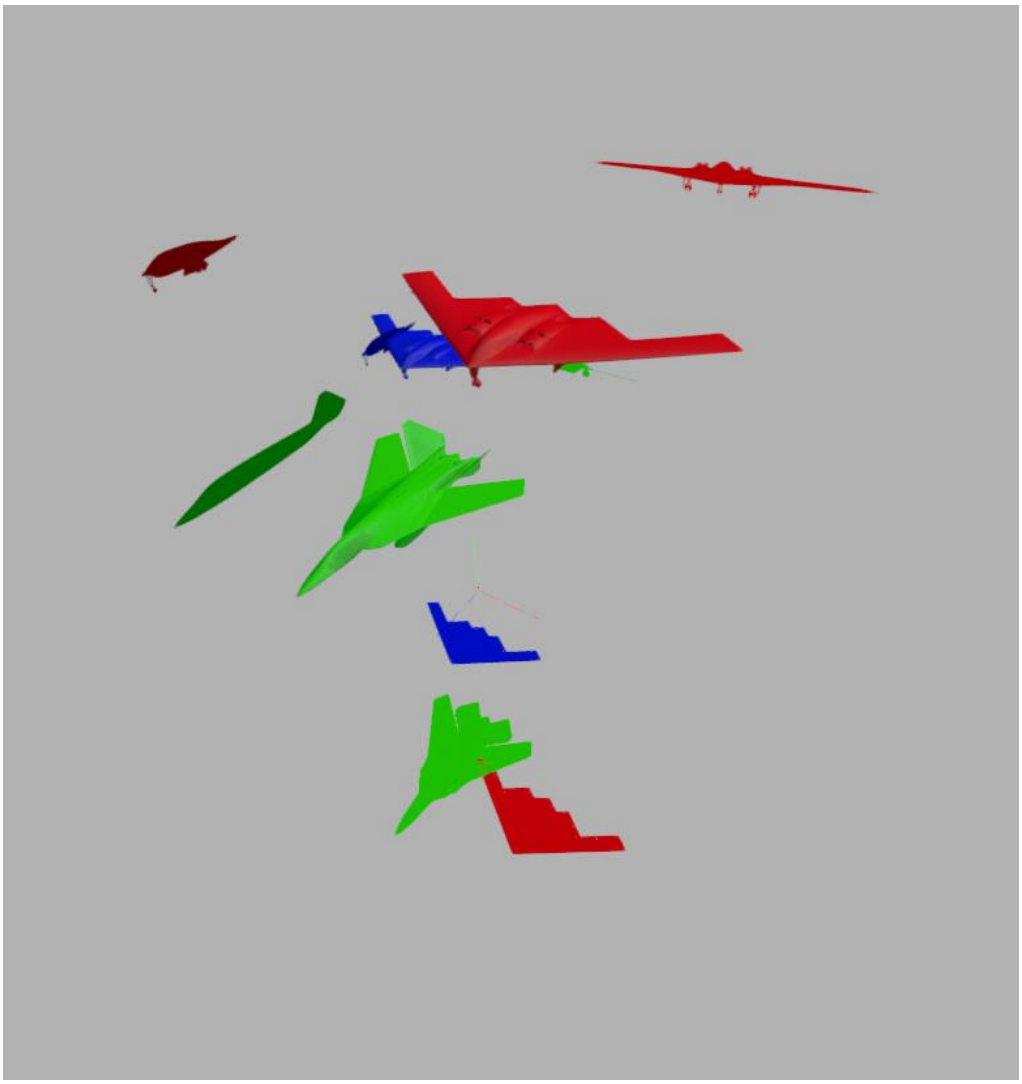
TASK-10

Προβάλλετε και απεικονίστε τα περιβλήματα των οχημάτων στα τρία επίπεδα xy, xz, yz για κάθε στιγμιότυπο. Σχολιάστε το αποτέλεσμα. Μπορεί να υπολογιστεί η τομή αποκλειστικά από την 2Δ πληροφορία στα επίπεδα;

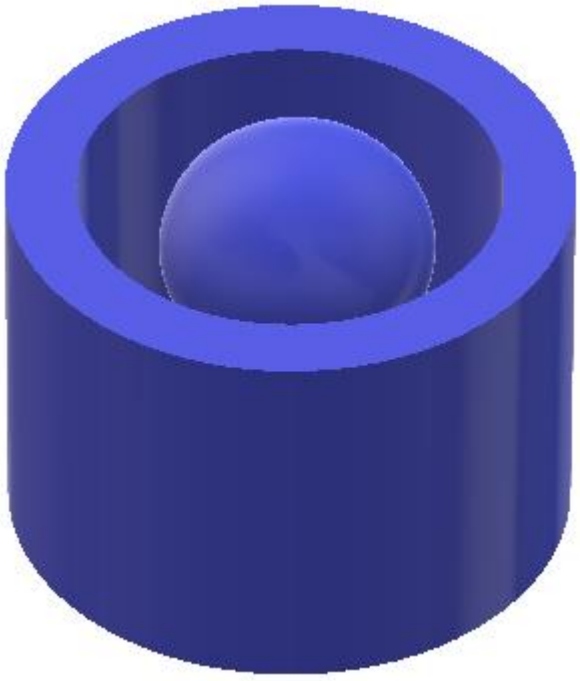
Για την προβολή των drones για τα τρία αυτά επίπεδα, έγιναν τα εξής:

1. Τα σημεία του uav χωρίζονται σε 3 άλλα σύνολα σημείων, όπου το καθένα αντίστοιχα έχει όλα τα x ή y ή z ίσα με 0.
2. Τα σύνολα αυτά μετατράπηκαν σε Mesh3D αντικείμενα.

3. Για την απεικόνιση τους, οι προβολές έχουν όλες το ίδιο χρώμα με το uav.
4. Στην περίπτωση που έχουμε τομή και στα 3 επίπεδα, για δύο uavs, τότε τυπώνεται στο terminal.

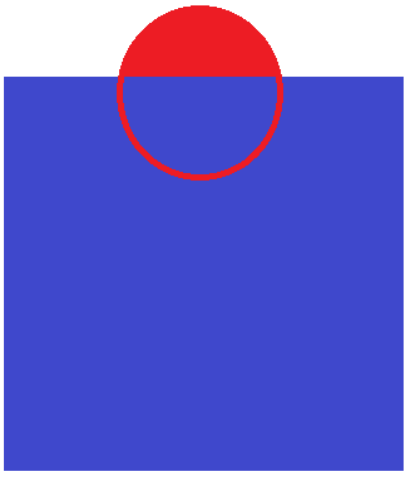


Παρατηρούμε το εξής: στην περίπτωση που τουλάχιστον για ένα από τα 3 επίπεδα δεν έχουμε τομή μεταξύ δυο προβολών, τότε είναι βέβαιο ότι δεν τέμνονται τα δύο uavs. Αυτό όμως δε σημαίνει ότι αν για όλες τις προβολές έχουμε τομή, τότε θα έχουμε και τομη των uavs. Ακολουθεί αντιπαράδειγμα που επιβεβαιώνεται η παραπάνω πρόταση.



Εικόνα 2 Δύο 3δ αντικείμενα τα οποία δεν τέμνονται.

Η όψη για κάθε επίπεδο θα είναι ίδια,



Εικόνα 3οι προβολές τέμνονται για κάθε επίπεδο xy, xz, yz

Και όπως φαίνεται, σε κάθε δισδιάστατη όψη, έχουμε τομή. Ωστόσο, δεν τέμνονται τα δύο τρισδιάστατα αντικείμενα.

Επομένως, το θεώρημα SAT (Separating axis theorem) δουλεύει μόνο ως συντηρητικός έλεγχος, καθώς ισχύει:

Αν τουλάχιστον σε ένα επίπεδο ΔΕΝ έχουμε τομή των προβολών, τότε τα τρισδιάστατα αντικείμενα ΔΕΝ τέμνονται.

Στην περίπτωση που τέμνονται οι προβολές σε όλα τα επίπεδα, τότε δεν μπορούμε να βγάλουμε συμπέρασμα, αλλά, πρέπει να γίνει περαιτέρω έλεγχος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Open3D 0.18.0 documentation

trimesh package - trimesh 4.4.9 documentation

NumPy –

3D collision detection - Game development | MDN (mozilla.org)

Object Reflecting off a Wall - Unity Engine - Unity Discussions