ΑΠΟΜΑΚΡΙΣΜΕΝΟΣ ΕΛΕΓΧΟΣ ΑΕΡΟΣΚΑΦΩΝ, ΣΕ ΠΕΡΙΒΑΛΛΟΝ ΠΡΟΣΟΜΟΙΩΣΗΣ

ΛΕΑΝΔΡΟΣ ΜΠΟΥΖΙΩΤΗΣ
Τμ. ΗΛΕΚΤΡΟΝΙΚΗΣ,
ΤΕΙ ΚΡΗΤΗΣ
Εισηγητής Δρ. Δ. Α. ΠΛΙΑΚΗΣ

Περιεχόμενα

ΣΥΝΟΨΗ

Στην παρούσα πτυχιακή εργασία θα προσπαθήσουμε να εμβαθύνουμε στον τομέα της σχεδίασης των μη επανδρωμένων συστημάτων που είναι υπεύθυνος για τον σχεδιασμό πτήσης. Για να υλοποιήσουμε τους αλγορίθμους και να δοκιμάσουμε την αποτελεσματικότητα τους δημιουργίσαμε ένα πλαίσιο αποτελούμενο από υπάρχοντα συστήματα σε συνδιασμό με κώδικα που δημιουργίσαμε συγκεκριμένα για την διεξαγωγή των δοκιμών. Η όλη σχεδίαση έχει σκοπό την ρεαλιστικότερη δοκιμή του αλγορίθμου του Dubins, και την ταυτόχρονη διαχείριση πλήθους αεροσκαφών από ένα χρήστη.

ABSTRACT

In this diploma thesis we review the design of autonomous aircraft systems with emphasis in path planning. This framework aims first to provide a realistic testbed for the Dubins Path algorithm and second to provide us the ability of simultaneous control of multiple aircrafts by a single user. In order to test the efficiency and robustness of the algorithms, we have developed a framework consisting of already available software and custom written code designed for this specific task.

Κεφάλαιο 1

Εισαγωγή

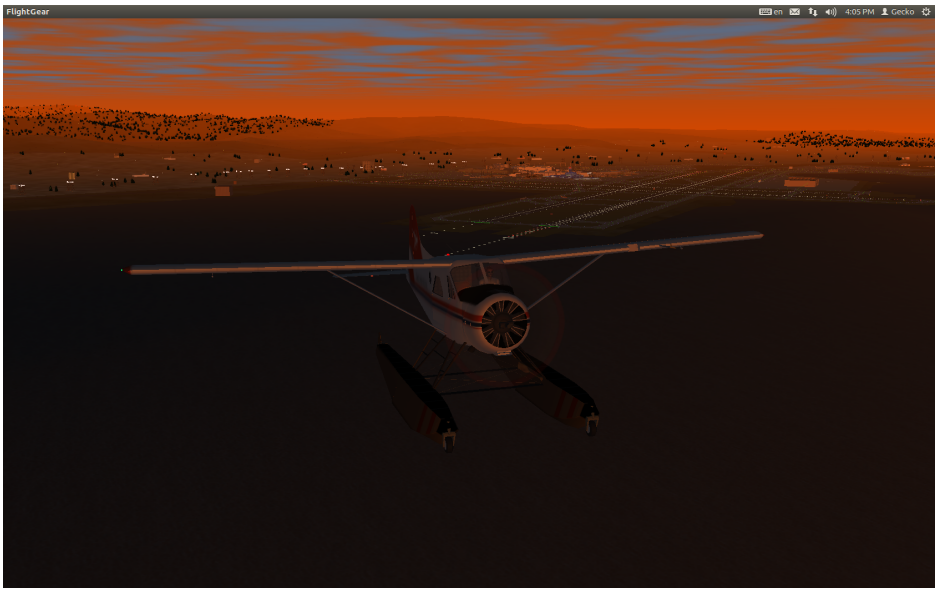
Η παρούσα πτυχιακή εργασία χωρίζεται σε δύο μέρη. Στο πρώτο μέρος αναλύονται οι βασική άξονες στους οποίους βασίστηκε η υλοποίηση της εφαρμογής. Στο πρώτο κεφάλαιο παρουσιάζουμε τα *μη-επανδρωμένα συστήματα* (UAS) και εξετάζουμε τις δυνατότητες τους σήμερα. Στο δεύτερο κεφάλαιο μελετάμε ένα συγκεκριμένο πρόβλημα στον τομέα των UAS, *την εύρεση της συντομότερης διαδρομής μεταξύ δύο θέσεων*, και παρουσιάζουμε τη δημοφιλή μέθοδο επίλυσης του Dubins Path.

Στα επόμενα δύο κεφάλαια κάνουμε μια εισαγωγή στα εργαλεία που χρησιμοποιήσαμε για να δημιουργήσουμε ένα περιβάλλον στο οποίο μπορούμε να υλοποιήσουμε την προσέγγιση που παρουσιάσαμε πριν. Για να το επιτύχουμε αυτό κάναμε χρήση του εξομοιωτή πτήσεων Flight Gear, που παρουσιάζεται στο τρίτο κεφάλαιο, και της γλώσσας προγραμματισμού Python, που παρουσιάζεται στο τέταρτο και τελευταίο κεφάλαιο του πρώτου μέρους.

Στο δεύτερο μέρος γίνεται παρουσίαση της εφαρμογής που υλοποιήθηκε για την δοκιμή του Dubins Path, και αναλύονται ορισμένα μέρη του κώδικα. Στο τέλος παρουσιάζονται εικόνες από τις δοκιμές της υλοποίησης.

Το αεροσκάφος που επιλέχτηκε ήταν το Dhc2. Οι λόγοι που μας οδήγησαν στην επιλογή του Dhc2 ήταν τρείς. Η αντοχή του σε απότομους χειρισμούς, κάτι που βοήθησε κατά την διάρκεια των δοκιμών. Η ύπαρξη υποσυστημάτων για το συγκεκριμένο αεροσκάφος στον εξομοιωτή πτήσεων Flight Gear. Τέλος η δυνατότητα προσθαλάσσωσης του Dhc2 μας επιτρέπει να προσομοιώσουμε μεγαλύτερο εύρος αποστολών.

Όλο το περιεχόμενο της παρούσας πτυχιακής εργασίας στηρίχθηκε αποκλειστικά σε προγράμματα και πόρους που είναι ελεύθερα από περιοριστικούς όρους χρήσης. Επιτρέπεται η οποιαδήποτε χρήση των πληροφοριών που υπάρχουν στην παρούσα πτυχιακή εργασία.

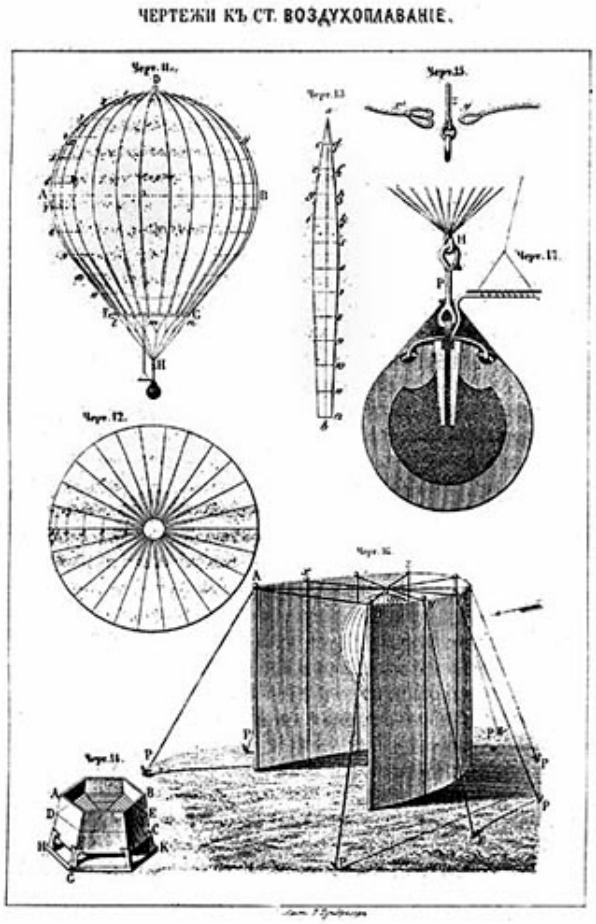


Σχήμα 1.1: Το Dhc2 πάνω από το KSFO.

Η όλη εφαρμογή δημιουργήθηκε σε περιβάλλων Linux και σαν προτεραιότητα έχει να είναι χρηστικό σε λειτουργικά συστήματα Unix. Το Flight Gear και η Python είναι cross-platform, έτσι είναι δυνατή και η χρήση τους σε περιβάλλον Windows.

Κεφάλαιο 2

Μη επανδρωμένα εναέρια συστήματα.



Σχήμα 2.1: Μια από τις πρώτες απόπειρες κατασκευής μη επανδρωμένου εναέριου συστήματος περίπου το 1850, αποτυπωμένο από Ρώσο καλλιτέχνη.

2.1 Ιστορία

Με τον όρο μη επανδρωμένο αεροσκάφος (UAV- Unmanned Air Vehicle) περιγράφουμε ένα αεροσκάφος το οποίο δεν έχει φυσική παρουσία πιλότου επί του αεροσκάφους. Η πτήση του αεροσκάφους ελέγχεται είτε από ένα αυτοματοποιημένο σύστημα ή μέσω τηλεχειρισμού.

Οι περισσότεροι άνθρωποι ταυτίζουν τα μη επανδρωμένα αεροσκάφη με τα πρόσφατα επιτεύγματα αυτής της τεχνολογίας όπως το Predator και το Global Hawk. Η αλήθεια όμως είναι ότι οι προσπάθειες για την κατασκευή μη επανδρωμένων αεροσκαφών ξεκινάει πολύ πιο πριν.

Στις 22 Αυγούστου του 1849, οι Αυστριακοί οι οποίοι ήλεγχαν ένα μεγάλο μέρος τις Ιταλίας, αποφασίσανε να στείλουν περί τα 200 μη επανδρωμένα αερόστατα στην πόλη της Βενετίας. Αυτά τα αερόστατα ήταν εξοπλισμένα με εκρηκτικά τα οποία πυροδοτούνταν με χρονοδιακόπτες.

Το όλο σχέδιο ήταν να ελευθερώσουν αυτά τα αερόστατα τα οποία με την σειρά τους, παρασυρόμενα από τα ρεύματα αέρα θα κατευθύνονταν προς την πόλη όπου και θα εκρήγνυντο. Για κακή τύχη όμως των Αυστριακών κατά την διάρκεια αυτής της εναέριας επιδρομής τα ρεύματα αέρα άλλαξαν με αποτέλεσμα αρκετά αερόστατα να ρίξουν τις βόμβες τους πίσω από τις γραμμές του Αυστριακού στρατού.

Σχήμα 2.2: Αναπαράσταση της επίθεσης στην Βενετία.

Ένας ακόμα επιστήμονας ο οποίος ήταν μπροστά από την εποχή του ήταν ο Νίκολα Τέσλα. Ο οποίος όχι απλά προέβλεψε την επανάσταση των μη επανδρωμένων αεροσκαφών, την οποία θεωρούσε και φυσική εξέλιξη, αλλά έκανε και επίδειξη ενός ασύρματα τηλεχειριζόμενου σκάφους το 1893. Ο ίδιος ο Τέσλα το 1921 στο *"Οι εφευρέσεις μου"* αναφέρει,

" Τα τηλεαυτόματα θα κατασκευαστούν, ικανά να δρουν ως κατέχοντα την δική τους νοημοσύνη, και η έλευση τους θα είναι ικανή να ξεκινήσει μια επανάσταση".

Στην εποχή που διανύουμε τα μη επανδρωμένα αεροσκάφη χρησιμοποιούνται σε ένα μεγάλο εύρος αποστολών, και ο αριθμός τους τείνει αυξανόμενος. Μπορούμε να τα συναντήσουμε σε αποστολές αναγνώρισης, επιτήρησης, ακόμα και σε αποστολές προσβολής στόχων.

2.2 Κατάταξη

Τα μη επανδρωμένα αεροσκάφη μπορούν να κατηγοριοποιηθούν χοντρικά στις παρακάτω κατηγορίες:

- *Εναέριοι στόχοι* έχουν ως σκοπό την εξομοίωση στόχων για την εκπαίδευση των αντιαεροπορικών συστημάτων, επίγειων ή εναέριων.
- *Αναγνώρισης*, σκοπός τους είναι η συλλογή πληροφοριών στο πεδίο της μάχης.
- *Μάχης*, παρέχουν την δυνατότητα εμπλοκής στόχων υψηλού ρίσκου.
- *Μεταφορά και ανεφοδιασμός* χρησιμοποιούνται σε αποστολές μεταφοράς και ανεφοδιασμού.
- *Έρευνας* αυτά τα συστήματα χρησιμοποιούνται σαν πλατφόρμες για την περαιτέρω εξέλιξη τεχνολογιών που χρησιμοποιούνται στα μη επανδρωμένα αεροσκάφη.
- *Πολιτικά και εμπορικά* χρησιμοποιούνται σε πολιτικού τύπου εφαρμογές όπως η πυρανίχνευση ή ακόμα και στην εξεύρεση κοπαδιών ψαριών βοηθώντας σημαντικά τους αλιείς (Fulmar).

Καθώς πλέον κατασκευάζονται πλατφόρμες οι οποίες συμπεριλαμβάνουν περισσότερες από μια των παραπάνω κατηγοριών, υπάρχει ένας ακόμα τρόπος κατηγοριοποίησης. Αυτός είναι η κατηγοριοποίηση σύμφωνα με την οροφή πτήσης / απόστασης που ένα μη επανδρωμένο όχημα μπορεί να διανύσει. Έχουμε λοιπόν:

1. *Χειρός* οροφή 2000 πόδια (600 μέτρα) / περίπου 2 km εμβέλεια.
2. *Κοντινής απόστασης*, οροφή 5000 πόδια (1500 μέτρα) / μέχρι 10 km εμβέλεια.
3. Τύπου *NATO*, οροφή 10.000 πόδια (3000 μέτρα) / μέχρι 50 km εμβέλεια.
4. *Τακτικά*, οροφή 18000 πόδια (5500 μέτρα) / μέχρι 160 km εμβέλεια.
5. *MALE, (medium altitude / long endurance)* μεσαίου υψομέτρου / μεγάλης ακτίνας δράσης, μέχρι 30000 πόδια (9000 μέτρα) / πάνω από 200 km εμβέλεια.
6. *HALE (high altitude / long endurance)* μεγάλου υψομέτρου / μεγάλης ακτίνας δράσης, πάνω από 30000 πόδια (9100 μέτρα) / απροσδιόριστη εμβέλεια.
7. *HYPERSONIC* (υπερηχητικά) υψηλής ταχύτητας, υπερηχητικά (1-5 Mach), υπερυπερηχητικά (5+ Mach) / υψόμετρο 50000 ποδια (15200 μέτρα) ή υποτροχιακό υψόμετρο, εμβέλεια 200 km.
8. *ORBITAL* (τροχιακά) χαμηλή περί τη γη τροχιά (LEO, Low Earth Orbit) ταχύτητα 25+ Mach.

Σχήμα 2.3: Το βλήμα Tactic Rainbow

2.3 Επίπεδα αυτονομίας

Ως επίπεδο αυτονομίας ενός μη επανδρωμένου αεροσκάφους ορίζεται η δυνατότητα που έχει το αεροσκάφος να πραγματοποιεί χωρίς ανθρώπινη παρέμβαση ή έλεγχο μία συγκεκριμένη αποστολή, ή συγκεκριμένα σκέλη αυτής.

Τα πρώτα μη επανδρωμένα αεροσκάφη που χρησιμοποιήθηκαν με ορισμένη αυτονομία χρησιμοποιήθηκαν στον πόλεμο του Βιετνάμ από τις Η.Π.Α. Αυτά τα αεροσκάφη ήταν εξοπλισμένα με συσκευές καταγραφής εικόνας (φωτογραφικές μηχανές, μηχανές κινηματογράφησης), και χρησιμοποιήθηκαν σε αποστολές αναγνώρισης.

Η λειτουργία τους ήταν ανοιχτού βρόχου (Open loop), αυτό περιόριζε τις δυνατότητες τους καθώς δεν μπορούσαν να έχουν δυναμική συμπεριφορά. Αυτό σημαίνει ότι οποιαδήποτε αλλαγή στις περιβαλλοντικές παραμέτρους, για παράδειγμα η κατεύθυνση και η ένταση των ανέμων, μπορούσαν να οδηγήσουν την αποστολή σε αποτυχία.

Για παράδειγμα η αποστολές αναγνώρισης γινόταν βασισμένες σε προκαθορισμένες συμπεριφορές πτήσης. Τα αεροσκάφη πετούσαν είτε σε ευθεία γραμμή ή σε κυκλικές τροχιές, καταγράφοντας το υλικό σε αποθηκευτικά μέσα που βρίσκονταν επί του αεροσκάφους. Όταν τα καύσιμα τελείωναν το αεροσκάφος "προσγειωνόταν", και το υλικό ανακτάτο από το προσωπικό εδάφους. Όπως είναι προφανές αυτός ο τρόπος λειτουργίας δεν ήταν και πολύ αποτελεσματικός.

Με την εξέλιξη της τεχνολογίας συστήματα τηλεκατεύθυνσις ενσωματώθηκαν στα αεροσκάφη δίνοντας έτσι την δυνατότητα μεγαλύτερου ελέγχου της πορείας που τα αεροσκάφη ακολουθούσαν. Ένα ακόμα βήμα ήταν η απευθείας μετάδοση των εικόνων που κατέγραφαν τα καταγραφικά του αεροσκάφους σε επίγειους σταθμούς ελέγχου. Αν και ο τηλεχειρισμός ήταν ένα βήμα προς την αυτονομία, σε καμία περίπτωση δεν μπορούμε να μιλήσουμε για πραγματικά αυτόνομα αεροσκάφη.

Ένα δημοφιλές αεροσκάφος που διαμορφώθηκε για αποστολές εξομοίωσης στόχου ήταν το F-4 Phantom το οποίο και μετονομάστηκε σε QF-4E. Το πρόθεμα Q πριν την κωδική ονομασία του αεροσκάφους είναι πάγια τακτική της αμερικανικής αεροπορίας να δηλώνει την μετατροπή ενός αεροσκάφους σε τηλεχειριζόμενο.

Σχήμα 2.4: Το QF-4E εν πτήση, παρατηρείστε την απουσία πιλότου.

Τα σημερινά μη επανδρωμένα αεροσκάφη έχουν την δυνατότητα να εκτελούν αυτοματοποιημένα πολλές από τις λειτουργίες που χρειάζονται, από συστήματα που είναι εγκατεστημένα επί του αεροσκάφους (σταθεροποίηση, έλεγχος ορθής λειτουργίας υποσυστημάτων κ.α), ή λαμβάνοντας οδηγίες από τον σταθμό ελέγχου (τροχιά που ακολουθείται, αλλαγή τρόπου λειτουργίας κ.α.). Αυτές οι λειτουργίες χρησιμοποιούν βρόχους ανάδρασης καθιστώντας το σύστημα ελέγχου στην κατηγορία κλειστού βρόχου (Closed loop).

Πέραν όμως αυτών των βασικών λειτουργιών όπως είναι η καθοδήγηση του αεροσκάφους, υπάρχουν πολλές ακόμα λειτουργίες που συντρέχουν κατά την διάρκεια μιας αποστολής, όπως η επιλογή στόχων ή η επιλογή του οπλικού συστήματος που θα χρησιμοποιηθεί για την προσβολή του στόχου. Αυτές οι λειτουργίες απαιτούν αυτονομία στην λήψη αποφάσεων, σε πραγματικό χρόνο από τα συστήματα του αεροσκάφους. Για να καταστεί αυτό εφικτό δεν ήταν αρκετή η υπάρχουσα τεχνολογία.

Η εξέλιξη συστημάτων που είναι ικανά να φέρουν εις πέρας τις παραπάνω λειτουργίες γέννησε την κατηγορία των μη επανδρωμένων εναέριων συστημάτων.

2.4 Μη επανδρωμένο εναέριο σύστημα

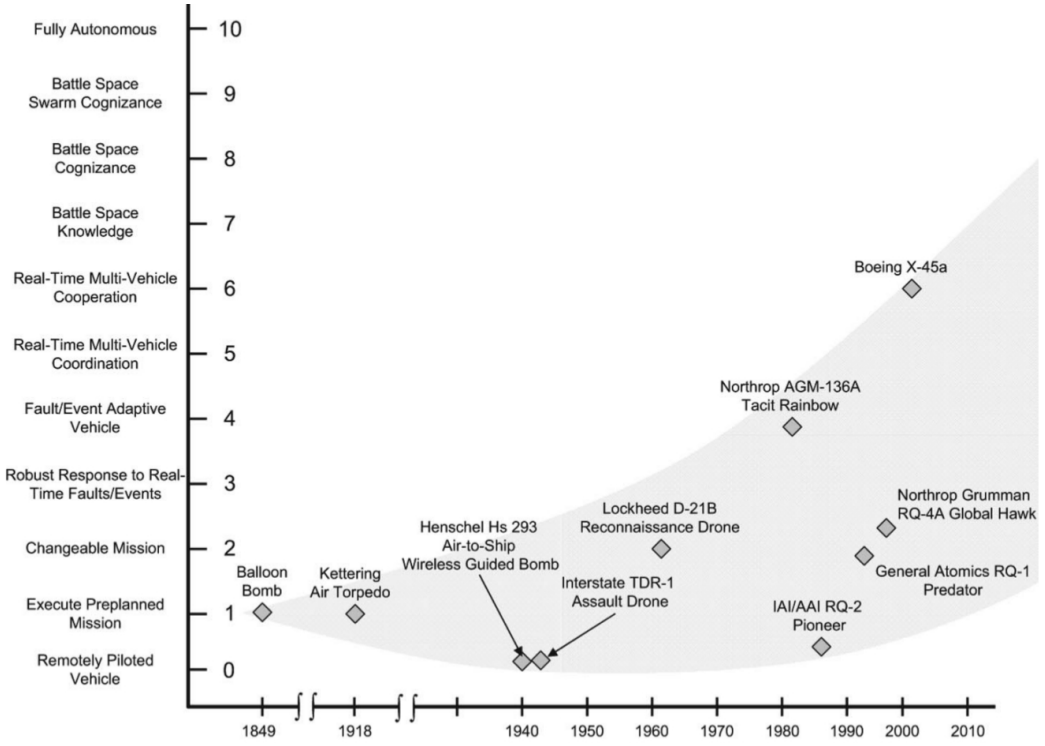
Το 2004 η Ομοσπονδιακή διοίκηση αεροπλοΐας των Η.Π.Α εισήγαγε τον όρο UAS (Unmaned Aircraft System), μη επανδρωμένο εναέριο σύστημα, για να περιγράψει την αύξουσα πολυπλοκότητα που υπάρχει στα σύγχρονα συστήματα, καθώς αυτά δεν αποτελούνται πλέον μόνο από ένα μη επανδρωμένο αεροσκάφος αλλά από αρκετά επιμέρους στοιχεία. Αυτά μπορεί να είναι σταθμοί εδάφους, πλήθος εναέριων μέσων που το κάθε ένα έχει διακριτό ρόλο στην αποστολή και διάφορα άλλα υποσυστήματα. Πρόσφατα αυτός ο όρος υιοθετήθηκε και από τον ICAO (International Civil Aviation Organization), Διεθνής οργανισμός πολιτικής αεροπορίας, για την περιγραφή αυτών των συστημάτων.

Η κύρια ώθηση για την ανάπτυξη μη επανδρωμένων εναέριων συστημάτων είναι η στρατιωτικές εφαρμογές. Ένα ενδιαφέρον στοιχείο είναι οι ώρες πτήσεις που έχουν καταγράψει τα σύγχρονα συστήματα όπως το Predator σε ρόλους υποστήριξης στις επιχειρήσεις Enduring Freedom, Iraqi Freedom. Μέχρι το 2005 ανέρχονται σε 100000. Όπως καταλαβαίνουμε η εμπλοκή τέτοιων συστημάτων στα πεδία των μαχών είναι ήδη μεγάλη και προβλέπεται ότι ο αριθμός και η κλίμακα που θα

χρησιμοποιούνται συνεχώς θα αυξάνεται.

Η συνεχιζόμενη πρόοδος της τεχνολογίας σε καίριους τομείς, έχουν καταστήσει δυνατή την δημιουργία οπλικών συστημάτων ικανών να φέρουν μεγάλη ποικιλία αισθητήρων και οπλικών συστημάτων σε όλο και μικρότερου μεγέθους οχήματα. Επίσης παρέχεται η δυνατότητα σχεδίασης συστημάτων που μπορούν, με την χρήση μεθόδων που ανήκουν στον κλάδο της τεχνητής νοημοσύνης, να λαμβάνουν αποφάσεις για την σειρά που θα εμπλέξουν τους στόχους σύμφωνα με την κρισιμότητα τους, ακόμα και την διαδρομή που θα ακολουθήσουν μέχρι τον στόχο βασιζόμενοι σε διάφορες παραμέτρους.

Η παράλληλα αυξανόμενη πολυπλοκότητα των UAS, σε διάφορα πεδία όπως, αντίληψη-έλεγχος κατάστασης, ανάλυση-συγχρονισμός, λήψη αποφάσεων και ικανότητες συστήματος, έκαναν την αξιολόγηση του επιπέδου αυτονομίας ενός συστήματος δύσκολη. Υπάρχουν αρκετές κλίμακες για την αξιολόγηση του επιπέδου αυτονομίας ενός συστήματος. Μία προσέγγιση είναι η αξιολόγηση της αυτονομίας μέσα σε ένα δοσμένο πλαίσιο που ορίζεται από την εκάστοτε αποστολή. Σε αυτήν την κατηγορία ανήκουν οι δείκτες, ACL (Autonomous Control Level) και ALFUS (Autonomy Levels For Autonomous Systems), και άλλες προσεγγίσεις. Στο παρόν κεφάλαιο θα δούμε σαν παράδειγμα τον δείκτη ACL 2.5.



Σχήμα 2.5: Ο δείκτηςACL.

Στον άξονα των y, ο ACL έχει 11 διαβαθμίσεις, ξεκινώντας από το τηλεκατευθυνόμενο όχημα, μέχρι το πλήρως αυτόνομο σύστημα. Για να καταλάβουμε περίπου την κλίμακα,στον άξονα των x έχουμε την χρονική τοποθέτηση του εκάστοτε συστήματος. Ξεκινώντας από το 1849 και τα αερόστατα των Αυστριακών, μέχρι το 2010 που συναντάμε το Boeing X-45A.

Σχήμα 2.6: Το X-45A της Boeing.

Το Boeing X-45 είναι ένα από τα πλέον σύγχρονα συστήματα. Η πρώτη του πτήση έγινε το 2002 και ο σκοπός της δημιουργίας του ήταν να εξακριβωθεί αν η ιδέα ενός αυτόνομου αεροσκάφους μάχης (UCAV), είναι εφικτή. Η τεχνογνωσία που αποκτήθηκε στο X-45 θα χρησιμοποιηθεί από την Boeing για την ανάπτυξη του ακόμα πιο εξελιγμένου Phantom Ray.

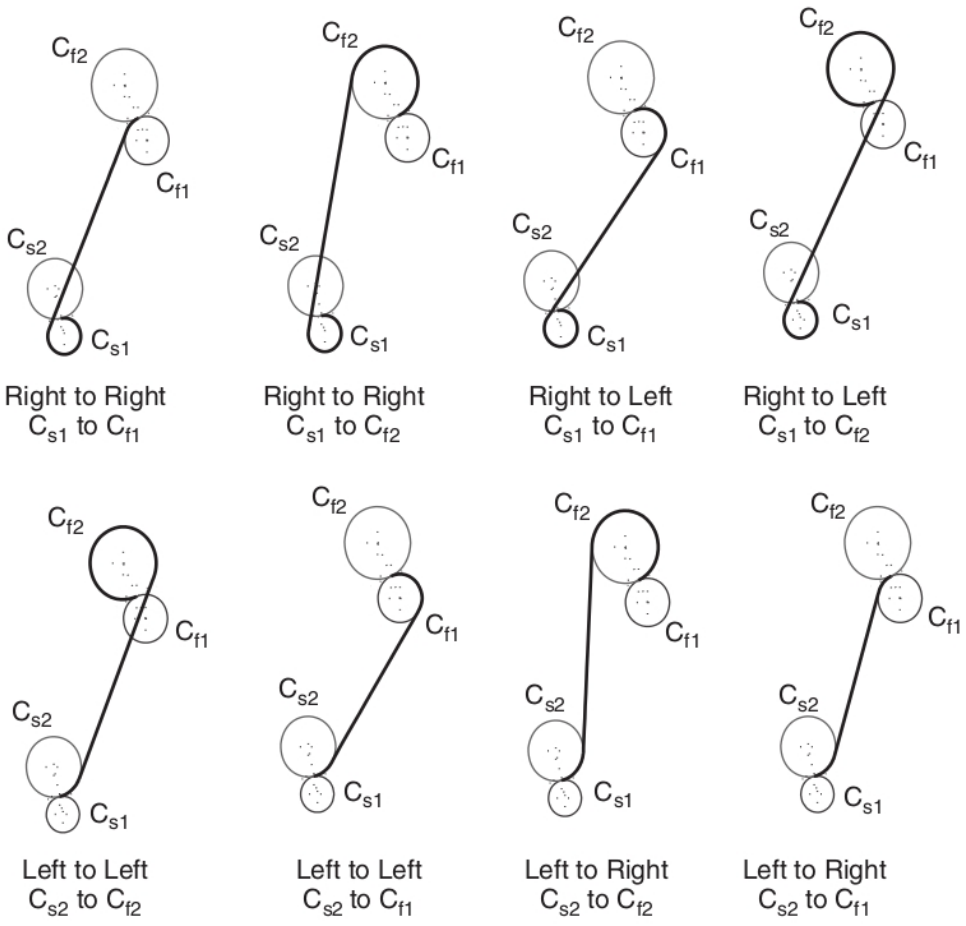
2.5 Περίληψη

Σε αυτό το κεφάλαιο περιγράψαμε την έννοια του μη επανδρωμένου αεροσκάφους UAV. Έγινε μια αναδρομή στην ιστορία των μη επανδρομένων αεροσκαφών και παρουσιάστικαν οι πιθανές εφαρμογές αυτών τον συστημάτων. Παρουσιάστικε η έννοια της αυτονομίας και πως η αύξηση της πολυπλοκώτητας μας οδήγησε στα μη επανδρωμένα συστήματα UAS. Στην συνέχεια είδαμε τις κατήγοριες των μη επανδρωμένων αεροσκαφών και τον τρόπο με τον οποίο μπορούμε να κατατάξουμε ένα μη επανδρομένο αεροσκάφος σε αυτές (κλίμακα ACL).

Στο επόμενο κεφάλαιο θα δούμε πως επιλύεται το πρόβλημα εύρεσης της συντομότερης διαδρομής μεταξύ δυο θέσεων. Ο σχεδειασμός πτήσης είναι ένα βασικό τμήμα της αυτονομίας του μη επανδρωμένου εναέριου συστήματος.

Κεφάλαιο 3

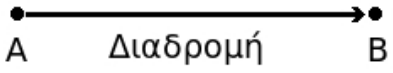
Εύρεση συντομότερης διαδρομής.



Σχήμα 3.1: Πιθανές διαδρομές μεταξύ δύο σημείων χωρίς περιορισμό κατεύθυνσης κατά την προσέγγιση του τελικού σημείο.

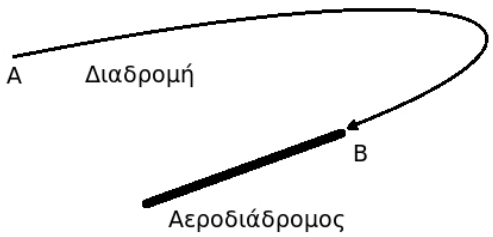
3.1 Εισαγωγή

Ένα δομικό στοιχείο για τα μη επανδρωμένα αεροσκάφη (UAV) στον δρόμο προς την αυτονομία είναι ο σχεδιασμός πτήσης. Με τον όρο σχεδιασμός πτήσης (path planning) εννοούμε την διαδικασία μέσω της οποίας χαράσσεται η διαδρομή που πρέπει να ακολουθήσει το UAV ώστε να πάει από το σημείο Α στο σημείο Β. Στην πιο απλή εκδοχή αυτή η διαδρομή θα μπορούσε να είναι μια ευθεία γραμμή που ενώνει αυτά τα δύο σημεία.



Σχήμα 3.2: Η πιο απλή διαδρομή, μια ευθεία γραμμή.

Αυτό όμως δεν είναι αρκετό. Όπως είναι προφανές η διαδρομές που πρέπει να ακολουθήσει ένα UAV είναι πολύ πιο σύνθετες από ευθείες γραμμές. Το περιβάλλον στο οποίο κινείται ένα αεροσκάφος μπορεί να περιέχει εμπόδια, ή να χρειάζεται το αεροσκάφος να προσεγγίσει αυτά τα σημεία από συγκεκριμένες κατευθύνσεις ώστε να εκτελέσει συγκεκριμένες λειτουργίες. Για παράδειγμα όταν το αεροσκάφος πρέπει να προσγειωθεί, η προσέγγιση του αεροδιαδρόμου πρέπει να είναι από μια συγκεκριμένη κατεύθυνση.



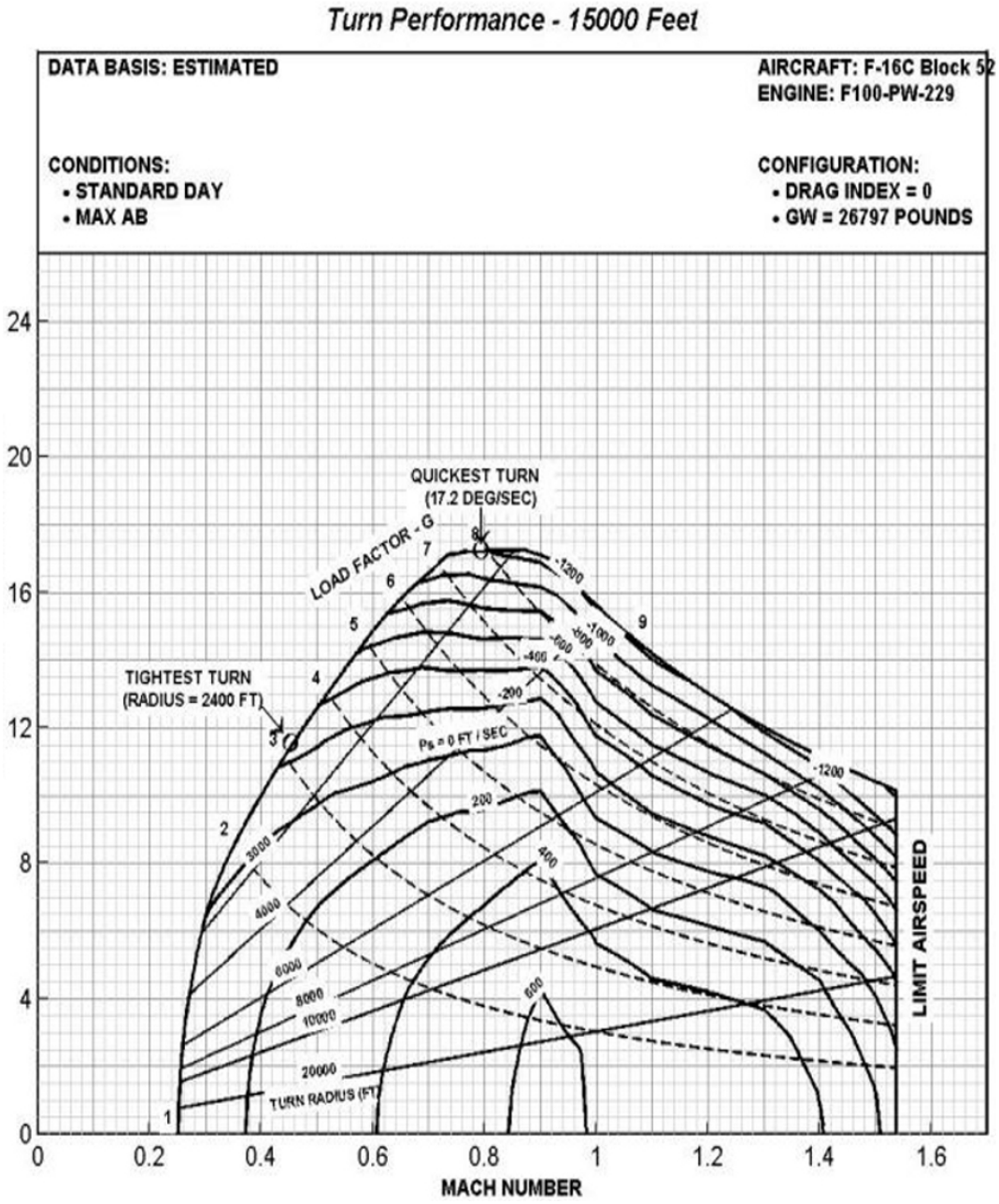
Σχήμα 3.3: Διαδρομή για συγκεκριμένη κατεύθυνση προσέγγισης.

Ο πλέον σημαντικός περιορισμός όμως για την δημιουργία των διαδρομών είναι ο περιορισμός στην ελάχιστη ακτίνα στροφής που έχει το κάθε αεροσκάφος, η οποία είναι συνάρτηση του ρυθμού στροφής που μπορεί να επιτευχθεί από το εκάστοτε αεροσκάφος.

Ο ρυθμός στροφής ενός αεροσκάφους εξαρτάται από πολλούς παράγοντες όπως, η πυκνότητα του αέρα (υψόμετρο, ατμοσφαιρικές συνθήκες), η ταχύτητα του αεροσκάφους, και το εμβαδό των επιφανειών ελέγχου του αεροσκάφους. Στην πραγματικότητα κάθε αεροσκάφος έχει τα δικά του χαρακτηριστικά, αλλά η μοντελοποίηση μιας τόσο σύνθετης κατάστασης είναι περίπλοκη. Για αυτόν τον λόγο χρησιμοποιείτε ένα ασφαλές όριο στον ρυθμό στροφής για την παραγωγή αυτών των διαδρομών.

Επίσης είναι απαραίτητο η διαδρομή που θα προκύψει να είναι και η συντομότερη δυνατή. Αυτό γιατί με το να ακολουθήσουμε την συντομότερη διαδρομή έχουμε εξοικονόμηση ενέργειας για την εκπλήρωση των στόχων της αποστολής, άρα οι απαιτήσεις μας σε καύσιμα και ο χρόνος πτήσης ελαχιστοποιούνται.

Όταν έχουμε εξασφαλίσει ότι μπορούμε να βρούμε αυτές τις διαδρομές μπορούμε να εισάγουμε και άλλες απαιτήσεις όπως είναι η αποφυγή εμποδίων που προαναφέρθηκε ή ο συγχρονισμός των UAV ώστε να φτάσουν στον στόχο τους σε καθορισμένο χρόνο.



Σχήμα 3.4: Διάγραμμα επίδοσης στροφής του F-16.

Το πρόβλημα εύρεσης της συντομότερης διαδρομής μελετάται σε διάφορα επιστημονικά πεδία όπως η εφαρμοσμένη έρευνα, η υλικοτεχνική μέριμνα (Logistics) και η υπολογιστική γεωμετρία. Ένα από τα πλέον μελετημένα προβλήματα της υπολογιστικής γεωμετρίας είναι η ανεύρεση της συντομότερης διαδρομής σε ένα δοσμένο διάγραμμα (Freedman-Tarjan 1987). Το πρόβλημα του πλανόδιου πωλητή (Travelling salesman problem, TSP) και το πρόβλημα του κινέζου ταχυδρόμου (Chinese postman

problem, CPP), είναι προβλήματα που μελετώνται σε βάθος στην επιχειρησιακή έρευνα.

Όλα τα παραπάνω προβλήματα ψάχνουν μεν την συντομότερη διαδρομή, αλλά δεν λαμβάνουν υπόψη κανέναν περιορισμό του οχήματος. Επίσης η αναζήτηση περιορίζεται σε ένα *γράφημα*.

Για την χάραξη διαδρομών που θα συμπεριλαμβάνουν τους περιορισμούς του οχήματος έχουν δημιουργηθεί αρκετές προσεγγίσεις. Για παράδειγμα ο L. E. Dubins [;] το 1957 παρουσίασε μία προσέγγιση του προβλήματος. Ο Dubins προτείνει την δημιουργία διαδρομών που αποτελούνται από ευθείες και κυκλικά τόξα συνεχούς καμπυλότητας, αυτή η προσέγγιση ονομάζεται διαδρομές Dubins.

Μία άλλη προσέγγιση είναι οι διαδρομές Clothoid που είναι στην ουσία διαδρομές Dubins με την διαφορά ότι τα κυκλικά τόξα έχουν μεταβλητή καμπυλότητα. Αυτό δίνει την δυνατότητα ομαλότερης μετάβασης του αεροσκάφους από το ευθύγραμμο τμήμα στο καμπύλο τμήμα και το αντίθετο.

Επίσης υπάρχει η λύση των οδογραφικών πυθαγόρειων διαδρομών Pythagorian Hodograph Paths, τα οποία είναι διαδρομές που σε όλο το μήκος τους, έχουν μεταβλητή καμπυλότητα η οποία παραμετροποιείται από πολυώνυμα εν συναρτήσει του μήκους της απόστασης των δύο σημείων.

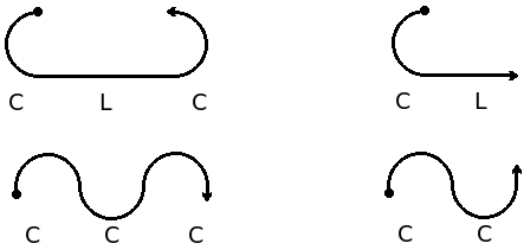
Στην παρούσα πτυχιακή εργασία υλοποιήθηκε η προσέγγιση του Dubins καθώς σε αυτή βασίζονται και οι υπόλοιπες προσεγγίσεις. Η μικρές απαιτήσεις σε επεξεργαστική ισχύ για τους υπολογισμούς, σε σύγκριση με τις άλλες προσεγγίσεις, μειώνει τις απαιτήσεις σε εξοπλισμό που θα φέρει το αεροσκάφος, κάτι που ελαχιστοποιεί τους χρόνους απόκρισης του αεροσκάφους.

3.2 Διαδρομές

Ο Dubins παρουσίασε μία μέθοδο για την παραγωγή διαδρομών που συμπεριλαμβάνει τους περιορισμούς της μέγιστης καμπυλότητας και του ελάχιστου μήκους. Στην εργασία του λέει ότι, κάθε διαδρομή μπορεί να διαχωριστεί σε ευθύγραμμα και καμπύλα τμήματα. Αυτά τα τμήματα συνδιαζόμενα μεταξύ τους μπορούν να παράξουν οποιαδήποτε διαδρομή.

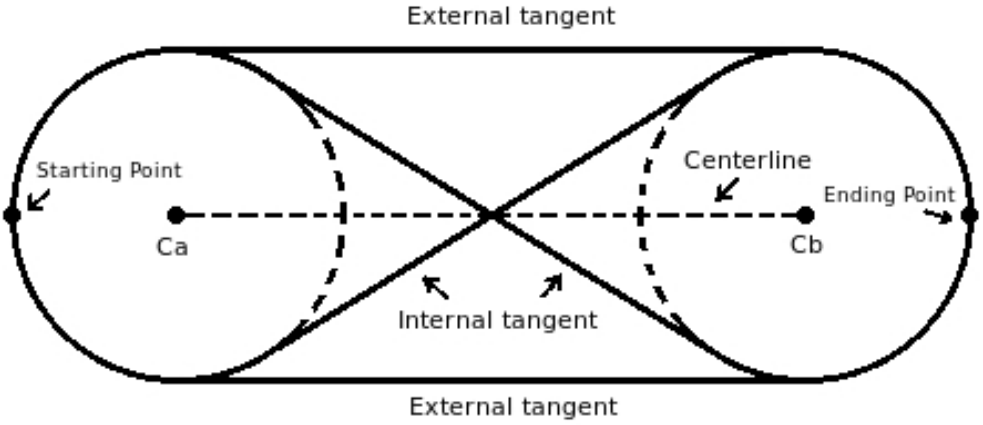
Όπως αναφέρθηκε και στην εισαγωγή μια σημαντική παράμετρος είναι η κατεύθυνση από την οποία πρέπει να προσεγγίσουμε το σημείο. Το σημείο συμπεριλαμβανομένης της κατεύθυνσης ονομάζετε στάση (pose). Έχουμε λοιπόν την απαίτηση να βρούμε την συντομότερη διαδρομή που ενώνει δύο στάσεις, την αρχική στάση (starting pose) και την τελική στάση (final pose). Ένας σύντομος ορισμός της διαδρομής Dubin είναι ο εξής:

Η συντομότερη διαδρομή η οποία ικανοποιεί τον περιορισμό της μέγιστης καμπυλότητας που μπορούμε να έχουμε μεταξύ δύο σημείων με συγκεκριμένες κατευθύνσεις σε ένα επίπεδο είναι, ή CLC ή CCC διαδρομή, όπου το C είναι ένα κυκλικό τόξο και το L είναι μία ευθεία εφαπτόμενη του C. Η CLC διαδρομή δημιουργείται ενώνοντας δύο κυκλικά τόξα με μια ευθεία η οποία είναι εφαπτόμενη τους, και η CCC διαδρομή δημιουργείται από τρία διαδοχικά καμπύλα τόξα τα οποία εφάπτονται μεταξύ τους. Τα υποσύνολα αυτών των διαδρομών είναι τα CL, LC, CC.



Σχήμα 3.5: Διαδρομές CCC, CLC, CC, CL

Για την σχεδίαση των διαδρομών που προτείνει ο Dubins μπορούν να χρησιμοποιηθούν διαφορετικά μαθηματικά εργαλεία. Εμείς θα χρησιμοποιήσουμε τα εργαλεία της αναλυτικής γεωμετρίας. Όπως είδαμε προηγουμένως η δημιουργία της διαδρομής γίνεται με την σύνδεση δύο τόξων με μια ευθεία εφαπτομένη στα δύο τόξα. Αυτή η εφαπτομένη μπορεί να είναι είτε εσωτερική (internal tangent) είτε εξωτερική (external tangent), ανάλογα με την στάση που θέλουμε. Αυτό απλοποιεί το πρόβλημα στην εύρεση των κοινών εφαπτόμενων που έχουν δύο κυκλικά τόξα. Η εσωτερική εφαπτομένη τέμνει την διάκεντρο (Centerline), ενώ η εξωτερική δεν τέμνει την διάκεντρο. Αν το αρχικό και το τελικό τόξο τέμνονται, τότε μόνο η λύση της εξωτερικής εφαπτομένης υπάρχει.



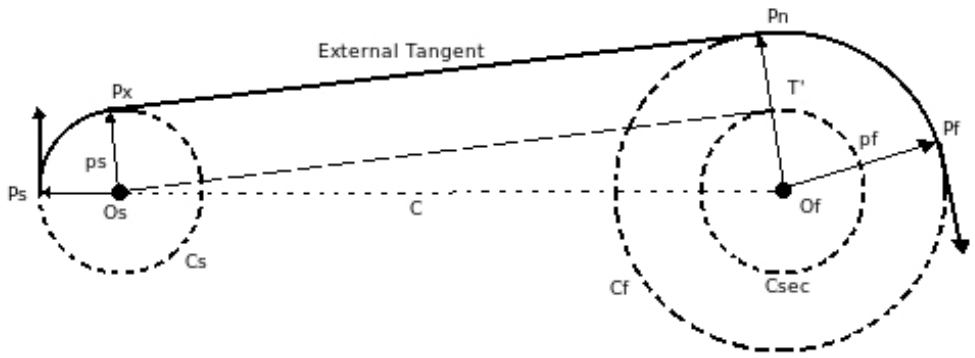
Σχήμα 3.6: Σχέδιο όλων των πιθανών διαδρομών

Όπως βλέπουμε στην 3.6 η διαδρομές που χρησιμοποιούν την λύση της εξωτερικής εφαπτομένης, ακολουθούν την διαδοχή δεξιάς - δεξιάς στροφής, ή αριστερής – αριστερής στροφής. Αντίθετα οι διαδρομές που χρησιμοποιούν την λύση της εσωτερικής εφαπτομένης ακολουθούν την διαδοχή δεξιάς – αριστερής στροφής, ή αριστερής- δεξιάς στροφής. Παρατηρούμε λοιπόν πως ένα ζευγάρι σημείων δημιουργεί τέσσερις πιθανές διαδρομές. Η επιλογή γίνεται με βάση την κατεύθυνση που έχουμε στο αρχικό σημείο (Starting Point) και εκείνη που θέλουμε να καταλήξουμε στο τελικό σημείο (Ending Point).

3.3 Λύση εξωτερικής Εφαπτομένης.

Στις παραγράφους που ακολουθούν χρησιμοποιούμε τοπικές καρτεσιανές συντεταγμένες. Η χρήση σφαιρικών γεωδαισιακών περιγράφεται στο κεφάλαιο του που αναπτύσσεται ο κώδικας.

Παρακάτω απεικονίζεται το σχεδιάγραμμα της λύσης εξωτερικής εφαπτομένης (external tangent solution). Με P_s συμβολίζεται η αρχική στάση του αεροσκάφους $P_s(X_s, Y_s, \theta_s)$. Όπου τα X_s, Y_s είναι οι συντεταγμένες του σημείου στο επίπεδο και το θ_s είναι η κατεύθυνση του αεροσκάφους στο συγκεκριμένο σημείο. Αντίστοιχα το $P_f(X_f, Y_f, \theta_f)$ είναι η τελική στάση του αεροσκάφους. Το σημείο P_x είναι το σημείο εξόδου της εφαπτομένης και το P_n είναι το σημείο εισόδου της εφαπτομένης. Με ρ_s, ρ_f συμβολίζονται οι ακτίνες του αρχικού τόξου και του τελικού τόξου αντίστοιχα.



Σχήμα 3.7: Λύση εξωτερικής εφαπτομένης

Για την σχεδίαση της διαδρομής Dubins είναι αναγκαίο να γνωρίζουμε τα εξής:

- Την αρχική στάση $P_s(X_s, Y_s, \theta_s)$
- Την τελική στάση $P_f(X_f, Y_f f, \theta_f)$
- Την αρχική καμπυλότητα $K_s \leq K_{max}$, όπου $K_s = \frac{1}{\rho_s}$ όπου ρ_s είναι η ακτίνα του αρχικού τόξου.
- Την τελική καμπυλότητα $K_f \leq K_{max}$, όπου $K_f = \frac{1}{\rho_f}$ όπου το ρ_f είναι η ακτίνα του τελικού τόξου.

Η λύση της εξωτερικής εφαπτομένης εξάγεται ακολουθώντας τα παρακάτω βήματα.

1. Βρίσκουμε το κέντρο του αρχικού κύκλου $C_s, O_s(X_{cs}, Y_{cs})$ και το κέντρο του τελικού κύκλου $C_f, O_f(X_{cf}, Y_{cf})$ χρησιμοποιώντας τις παρακάτω σχέσεις:

$$\begin{aligned} X_{cs} &= X_s - \rho_s \cos(\phi_s \pm \tfrac{\pi}{2}), \\ Y_{cs} &= Y_s - \rho_s \sin(\phi_s \pm \tfrac{\pi}{2}), \\ X_{cf} &= X_f - \rho_f \cos(\phi_f \pm \tfrac{\pi}{2}), \\ Y_{cf} &= Y_f - \rho_f \sin(\phi_f \pm \tfrac{\pi}{2}), \end{aligned}$$

όπου οι C_s, C_f ονομάζονται *πρωτεύοντες κύκλοι.*

2. Σχεδιάζουμε έναν *δευτερεύοντα κύκλο* C_{sec} ακτίνας $|\rho_f - \rho_s|$ στο σημείο O_f για $\rho_f \leq \rho_s$.

3. Ενώνουμε τα κέντρα C_s, C_f με μία γραμμή C, την οποία ονομάζουμε *διάκεντρο*. Το μήκος της διακέντρου είναι $|c| = \sqrt{(X_{cs} - X_{cf})^2 + (Y_{cs} - Y_{cf})^2}$.

4. Σχεδιάζουμε στο σημείο O_f μια κάθετη στην C. Η κάθετη τέμνει τον κύκλο C_{sec} στο σημείο T', και τον κύκλο C_f στο σημείο P_n. Το σημείο P_n ονομάζεται *σημείο εισόδου της εφαπτομένης*.

5. Ενώνουμε τα σημεία O_s, T'.

6. Σχεδιάζουμε μια ευθεία από το σημείο O_s, παράλληλη της O_f, P_n η οποία τέμνει τον C_s στο σημείο P_x. Το σημείο P_x ονομάζετε σημείο εξόδου της εφαπτομένης.

7. Ενώνουμε τα σημεία P_x, P_n με μια ευθεία που είναι παράλληλη με την γραμμή O_s, T'. Αυτή η ευθεία P_x, P_n είναι η εξωτερική εφαπτομένη.

8. Σχεδιάζουμε ένα τόξο ακτίνας πς από το σημείο P_s μέχρι το σημείο P_x, μια γραμμή από το σημείο P_x μέχρι το σημείο P_n και ένα τόξο ακτίνας ρ_f από το σημείο P_n μέχρι το σημείο P_f. Αυτά τα τρία τμήματα όταν τα ενώσουμε σχηματίζουν διαδρομές *Δυβινς* εξωτερικής εφαπτομένης.

Από το σχήμα προκύπτει η λύση της διαδρομής *Δυβινς*. Το ορθογώνιο τρίγωνο O_s, O_f, T' έχει κάθετες πλευρές τις O_f, T' και O_s, T', και υποτείνουσα την O_s, O_f, όπου $||O_s T'|| = |\rho_f - \rho_s|$. Τώρα μπορούμε να υπολογίσουμε την γωνία που σχηματίζεται από τις πλευρές O_s, O_f και OsT' με τον τύπο:

$$\alpha = \arcsin\left(\frac{\rho_f - \rho_s}{|c|}\right)$$

Η κλίση της διακέντρου είναι:

$$\beta = \arctan\left(\frac{Y_{cf} - Y_{cs}}{X_{cf} - X_{cs}}\right)$$

Τώρα μπορούμε να υπολογίσουμε τα σημεία εισόδου $P_x(X_{px}, Y_{px})$ του κύκλου C_s και εξόδου $P_n(X_{pn}, Y_{pn})$ της εφαπτομένης στον κύκλο C_f:

$$\begin{aligned} X_{px} &= X_{cs} + \rho_s \cos(\phi) \\ Y_{px} &= Y_{cs} + \rho_s \sin(\phi) \\ X_{pn} &= X_{cf} + \rho_f \cos(\phi) \\ Y_{pn} &= X_{cf} + \rho_f \sin(\phi) \end{aligned}$$

Το πρόσημο της στροφής ϕ είναι ανάλογα με την στροφή που κάνουμε, δεξιά ή αριστερά. Στον παρακάτω πίνακα βλέπουμε πως υπολογίζεται.

ϕ	Αρχική	Τελική
ϕ_+	$\alpha + \beta + \frac{\pi}{2}$	$\alpha + \beta + \frac{\pi}{2}$
ϕ_-	$\beta - \alpha + \frac{3\pi}{2}$	$\beta - \alpha + \frac{3\pi}{2}$
Εάν $k_s = k_f$ τότε $\alpha = 0$ και $O_s O_f // P_x P_N$ ηειγητ		

Οι γωνίες των τόξων από την αρχική στάση μέχρι το σημείο εξόδου της εφαπτομένης, και από το σημείο εισόδου της εφαπτομένης μέχρι την τελική στάση, υπολογίζονται με τους παρακάτω τύπους:

1. Για την γωνία από την αρχική στάση P_s μέχρι το σημείο εξόδου της εφαπτομένης P_x είναι: $\Psi_s = \phi_s \pm \frac{\pi}{2} + \phi$

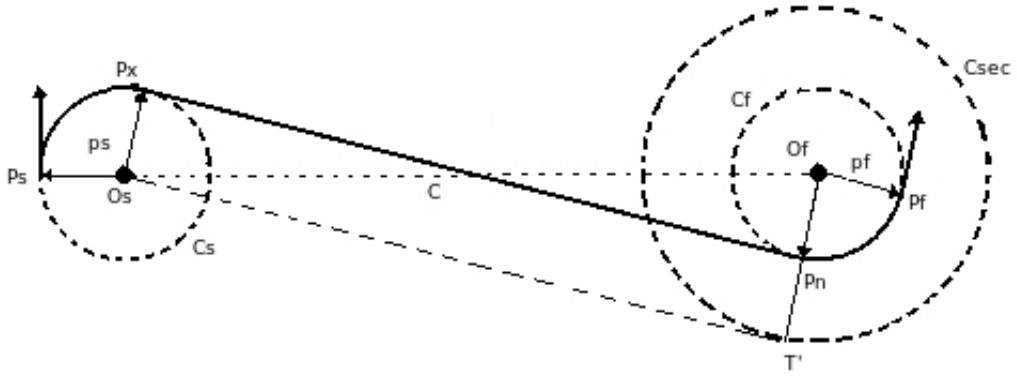
2. Από το σημείο P_x μέχρι το σημείο P_n δεν έχουμε στροφή.

3. Η γωνία από το σημείο εισόδου της εφαπτομένης P_n μέχρι την τελική στάση P_f είναι: $\Psi_f = \phi_f \pm \frac{\pi}{2} + \phi$ Το μήκος του τμήματος P_x, P_n βρίσκεται με το πυθαγόρειο θεώρημα στα σημεία εισόδου και εξόδου της εφαπτομένης.

$$L = \sqrt{(X_{P_x} - X_{P_n})^2 + (Y_{P_x} - Y_{P_n})^2}$$

3.4 Λύση εσωτερικής Εφαπτομένης

Η λύση της εσωτερικής εφαπτομένης (internal tangent solution)δεν διαφέρει από την λύση εξωτερικής εφαπτομένης, παρά μόνο στην γωνία της περιστροφής. Παρακάτω δίνεται το σχήμα της διαδρομής.



Σχήμα 3.8: Λύση εσωτερικής εφαπτομένης

Τα βήματα που ακολουθούμε για να βρούμε την διαδρομή είναι τα εξής:

1. Βρίσκουμε το κέντρο του αρχικού κύκλου $C_s, O_s(X_{cs}, Y_{cs})$ και το κέντρο του τελικού κύκλου $C_f, O_f(X_{cf}, Y_{cf})$ χρησιμοποιώντας τις παρακάτω σχέσεις:

$$\begin{aligned} X_{cs} &= X_s - \rho_s \cos(\Phi_s \pm \tfrac{\pi}{2}) \\ Y_{cs} &= Y_s - \rho_s \sin(\Phi_s \pm \tfrac{\pi}{2}), \\ X_{cf} &= X_f - \rho_f \cos(\Phi_f \pm \tfrac{\pi}{2}), \\ Y_{cf} &= Y_f - \rho_f \sin(\Phi_f \pm \tfrac{\pi}{2}), \end{aligned}$$

όπου οι C_s και C_f ονομάζονται *πρωτεύοντες κύκλοι.*

2. Σχεδιάζουμε έναν δευτερεύοντα κύκλο C_{sec} ακτίνας $|\rho_f - \rho_s|$ στο σημείο O_f για $\rho_f \leq \rho_s$

3. Ενώνουμε τα κέντρα C_s, C_f με μία γραμμή C, την οποία ονομάζουμε *διάκεντρο*. Το μήκος της διακέντρου είναι
$$|c| = \sqrt{(X_{cs} – X_{cf})^2 + (Y_{cs} - Y_{cf})^2}$$

4. Σχεδιάζουμε στο σημείο O_f μια κάθετη στην C. Η κάθετη τέμνει τον κύκλο C_{sec} στο σημείο T', και τον κύκλο C_f στο σημείο P_n. Το σημείο P_n ονομάζεται σημείο εισόδου της εφαπτομένης.

5. Ενώνουμε τα σημεία O_s και T'.

6. Σχεδιάζουμε μια ευθεία από το σημείο O_s, παράλληλη της O_f, P_n η οποία τέμνει τον C_s στο σημείο P_x. Το σημείο P_x ονομάζεται σημείο εξόδου της εφαπτομένης.

7. Ενώνουμε τα σημεία P_x και P_n με μια ευθεία που είναι παράλληλη με την γραμμή O_sT'. Αυτή η ευθεία P_xP_n είναι η εξωτερική εφαπτομένη.

8. Σχεδιάζουμε ένα τόξο ακτίνας ρ_s από το σημείο P_s μέχρι το σημείο P_x, μια γραμμή από το σημείο P_x μέχρι το σημείο P_n και ένα τόξο ακτίνας ρ_f από το σημείο P_n μέχρι το σημείο P_f. Αυτά τα τρία τμήματα όταν τα ενώσουμε σχηματίζουν τη διαδρομή Dubins εσωτερικής εφαπτομένης.

Από την 3.8 προκύπτει η λύση της διαδρομής Dubins. Το ορθογώνιο τρίγωνο $\Delta(O_sO_fT')$ έχει κάθετες πλευρές τις O_fT' και O_sT', και υποτείνουσα την O_sO_f , όπου $||O_sT'|| = |\rho_f + \rho_s|$. Τώρα μπορούμε να υπολογίσουμε την γωνία που σχηματίζεται από τις πλευρές O_sO_f και O_sT' με τον τύπο:

$$\alpha = \arcsin\left(\frac{\rho_f + \rho_s}{|c|}\right)$$

Η κλίση της διακέντρου είναι:

$$\beta = \arctan\left(\frac{Y_{cf} - Y_{cs}}{X_{cf} - X_{cs}}\right)$$

Τώρα μπορούμε να υπολογίσουμε τα σημεία εισόδου $P_x(X_{px}, Y_{px})$ του κύκλου C_s και εξόδου $P_n(X_{pn}, Y_{pn})$ της εφαπτομένης στον κύκλο C_f:

$$\begin{aligned} X_{px} &= X_{cs} + \rho_s \cos(\phi) \\ Y_{px} &= Y_{cs} + \rho_s \sin(\phi) \\ X_{pn} &= X_{cf} + \rho_f \cos(\phi) \\ U_{pn} &= X_{cf} + \rho_f \sin(\phi) \end{aligned}$$

Το πρόσημο της ϕ είναι ανάλογα με την στροφή που κάνουμε, δεξιά ή αριστερά. Μπορούμε να τα υπολογίζουμε με τον παρακάτω πίνακα.

ϕ	Αρχική	Τελική
ϕ_+	$\alpha + \beta$	$\beta - \alpha + \pi$
ϕ_-	$\beta - \alpha + 2\pi$	$\beta + \alpha + \pi$
Εάν $k_s = k_f$ τότε $\alpha = 0$ και $O_sO_f//P_xP_N$ ηειγητ		

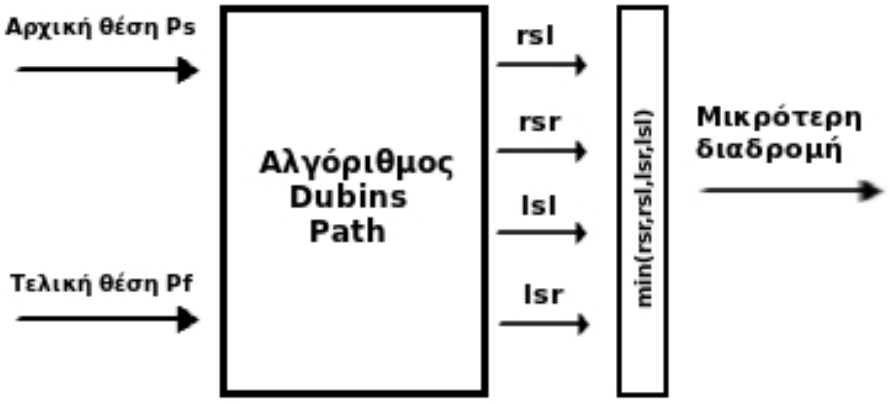
Οι γωνίες των τόξων από την αρχική στάση μέχρι το σημείο εξόδου της εφαπτομένης, και από το σημείο εισόδου της εφαπτομένης μέχρι την τελική στάση, υπολογίζονται με τους παρακάτω τύπους:

1. Για την γωνία από την αρχική στάση P_s μέχρι το σημείο εξόδου της εφαπτομένης P_x είναι: $\Psi_s = \Phi_s \pm \frac{\pi}{2} + \Phi$

2. Από το σημείο P_x μέχρι το σημείο P_n δεν έχουμε στροφή.

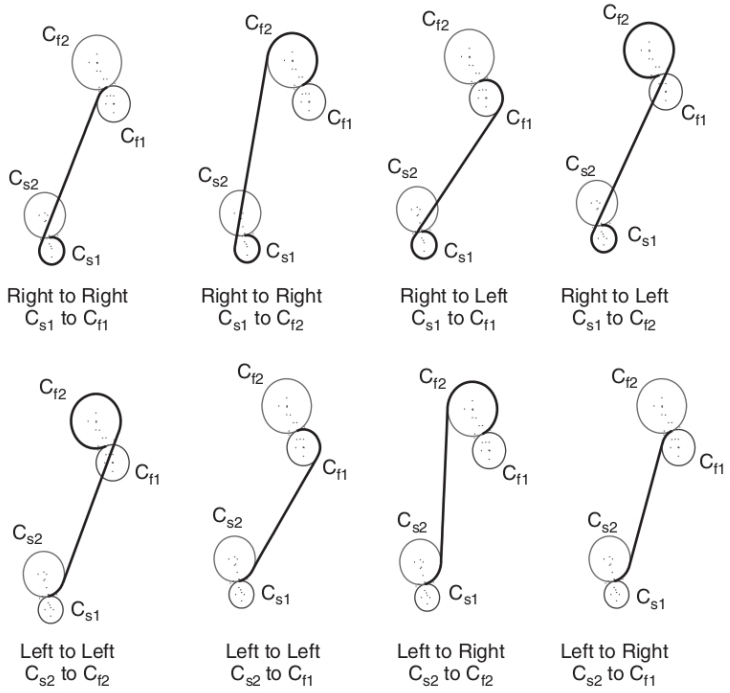
3. Η γωνία από το σημείο εισόδου της εφαπτομένης P_n μέχρι την τελική στάση P_f είναι: $\Psi_f = \Phi_f \pm \frac{\pi}{2} + \Phi$

Στο παρακάτω διάγραμμα 3.9 βλέπουμε τον τρόπο με τον οποίο επιλέγεται η διαδρομή Dubins. Δίνουμε ως ορίσματα την αρχική και τελική στάση (P_s, P_f). Ορίζουμε τα όρια της καμπυλότητας K_s και K_f. Ο αλγόριθμος δημιουργεί τις τέσσερις διαδρομές. Στο επόμενο βήμα επιλέγουμε την συντομότερη διαδρομή από τις τέσσερις.



Σχήμα 3.9: Αλγόριθμος επιλογής συντομότερης διαδρομής

Αν δεν έχουμε απαίτηση το αεροσκάφος να έχει συγκεκριμένη διεύθυνση στο τελικό σημείο, το τελικό τόξο μπορεί να απαλειφθεί και να τερματιστεί η διαδρομή στο άκρο του ευθύγραμμου τμήματος όπου και θα είναι το τελικό σημείο P_f. Αν στο τελικό σημείο έχουμε μόνο την διεύθυνση ως απαίτηση και όχι την κατεύθυνση, τότε έχουμε οκτώ πιθανές διαδρομές. Αν αυτό ισχύει και στην αρχική θέση, τότε οι πιθανές διαδρομές φτάνουν τις δεκαέξι.



Σχήμα 3.10: Πιθανές διαδρομές μεταξύ δύο σημείων, χωρίς περιορισμό κατεύθυνσης στο τελικό σημείο.

3.5 Ύπαρξη διαδρομών

Η εύρεση των διαδρομών Dubins εμπεριέχει την εύρεση των σημείων P_x, P_n μεταξύ των δύο κυκλικών τόξων. Αν δεν είναι δυνατό να βρούμε αυτά τα δύο σημεία τότε η διαδρομή Dubins δεν υπάρχει. Ωστόσο αν τα δύο αυτά σημεία συμπίπτουν, το ευθύγραμμο τμήμα της διαδρομής έχει μηδενικό μήκος. Αυτό έχει ως αποτέλεσμα να έχουμε μια διαδρομή τύπου CC. Αυτό σημαίνει ότι τα δύο τόξα εφάπτονται. Η λύση της εξωτερικής εφαπτομένης δεν υπάρχει όταν ο ένας κύκλος περιστροφής εμπεριέχεται στον άλλο, ενώ η λύση της εσωτερικής εφαπτομένης δεν υπάρχει όταν οι δύο κύκλοι τέμνονται. Για τον έλεγχο ύπαρξης των λύσεων δύνονται οι παρακάτω εξισώσεις: Εξωτερική εφαπτομένη: $|\rho_f - \rho_s| < |c|$ Εσωτερική εφαπτομένη: $|\rho_f + \rho_s| < |c|$

Μήκος διαδρομής Dubins Μια διαδρομή Dubins όπως είδαμε παραπάνω αποτελείται από τρία ξεχωριστά τμήματα. Δύο τόξα, το αρχικό και το τελικό, και μία ευθεία την εφαπτομένη. Το συνολικό της μήκος λοιπόν είναι το άθροισμα των μηκών αυτών των επιμέρους τμημάτων.

$$S_{Dubins} = S_{arc,start} + S_{tangent} + S_{arc,finish}$$

Χρησιμοποιώντας την αναλυτική γεωμετρία έχουμε,

$$S_{Dubins} = \rho_s \Phi_s + S_t + \rho_f \Phi_f = f(K_s, K_f)$$

όπου το S_{Dubins} είναι το συνολικό μήκος της διαδρομής,τα Φ_s, Φ_f είναι οι γωνίες που τις βρίσκουμε από τους αντίστοιχους πίνακες και το $S_t = ||P_x P_n||$.

Από την παραπάνω εξίσωση βλέπουμε ότι το μήκος είναι συνάρτηση της αρχικής και τελικής ακτίνας. Δεν πρέπει όμως να ξεχάσουμε πως και το μήκος της εφαπτομένης εξαρτάται από την αλλαγή των ακτίνων, άρα θα πρέπει να υπολογίσουμε και αυτήν την αλλαγή στους υπολογισμούς μας.

3.6 Περίληψη

Σε αυτό το κεφάλαιο παρουσιάστικε η ανάγκη που υπάρχει, στα μη επανδρωμένα αεροσκάφοι, για την εξεύρεση των συντομότερων διαδρομών μεταξύ δύο σημείων που θα μπορεί ένα αεροσκάφος να ακολουθήσει.

Ο Dubins πρότεινε μια λύση η οποία λαμβάνει υπόψη τον περιορισμό στην κατεύθυνση που έχουμε στην αρχική και τελική θέση, και την μέγιστη καμπυλώτητα που μπορούν να έχουν τα κυλικά τόξα μίας διαδρομής.

Παρουσιάστικε η λύση της εξωτερικής και της εσωτερικής εφαπτομένης και πώς γίνεται η επιλογή της λύσης που θα χρησιμοποιηθεί. Τέλος παρουσιάζεται η διαδικασία που μας επιτρέπει να δούμε αν είναι εφικτή αυτή η λύση ανάλογα με την θέση μας στον χώρο.

Στο επόμενο κεφάλαιο θα δούμε τι είναι ο εξομοιωτής πτήσεως και από τι αποτελείτε. Στην συνέχεια θα παρουσιάσουμε τον εξομοιωτή πτήσεως Flight Gear, στον οποίο θα υλοποιήσουμε την λύση που προτείνει ο Dubins.

Κεφάλαιο 4

Εξομοιωτής Πτήσεως

Σχήμα 4.1: Χορεύοντας με τα σύννεφα...

4.1 Εξομοιωτής Πτήσεως

Εξομοιωτής πτήσεως ονομάζεται ένας μηχανισμός που έχει ως σκοπό την προσομοίωση της συμπεριφοράς ενός αντικειμένου όταν βρίσκεται εν πτήση, καθώς και των υπολογισμό διαφόρων συνθηκών που διαμορφώνονται στο περιβάλλον της προσομοίωσης.

Αναλυτικότερα ένας εξομοιωτής πτήσεων είναι υπεύθυνος μεταξύ άλλων, για τον υπολογισμό των εξισώσεων κίνησης που διέπουν την συμπεριφορά του σκάφους εν πτήση, πώς αντιδρά το σκάφος σε μεταβολές των επιφανειών ελέγχου, πως αντιδρά το σκάφος σε εξωτερικούς παράγοντες όπως οι άνεμοι, η πυκνότητα του αέρα, η βροχή και άλλα.

Ένας εξομοιωτής πτήσεων μπορεί να χρησιμοποιηθεί σε ένα μεγάλο φάσμα εφαρμογών. Μπορούμε να βρούμε εξομοιωτές σχεδιασμένους απλά για ψυχαγωγία, εξομοιωτές που χρησιμοποιούνται για

έρευνα, μέχρι στρατιωτικούς εξομοιωτές που χρησιμοποιούνται στην βασική εκπαίδευση των πιλότων και στην εξοικείωση με νέους τύπους αεροσκαφών.

Η πολυπλοκότητα και ο ρεαλισμός του εκάστοτε εξομοιωτή είναι ανάλογα με την χρήση για την οποία προορίζεται. Θα μπορούσαμε να συναντήσουμε εξομοιωτές βασισμένους μόνο σε έναν προσωπικό υπολογιστή, μέχρι εξομοιωτές που κάνουν χρήση συστημάτων αναπαράστασης εξωτερικού χώρου ευρείας γωνίας (wide-field outside-world visual systems), σε συνδυασμό με κινούμενες πλατφόρμες έξι βαθμών ελευθερίας 6 (DoF).

Σχήμα 4.2: Ένας από τους πρώτους εξομοιωτές πτήσεως

Σχήμα 4.3: Ο εξομοιωτής πτήσεως του TU Delft

4.2 Ο εξομοιωτής πτήσεων Flight Gear

Για να υλοποιηθεί η παρούσα πτυχιακή εργασία έπρεπε να αναζητηθεί ένας εξομοιωτής γύρο από τον οποίο θα χτιζόταν η εφαρμογή. Οι αρχικές απαιτήσεις ήταν, να είναι έξι βαθμών ελευθερίας με δυνατότητα μοντελοποίησης ρευμάτων αέρα.

Έπειτα από έρευνα και αξιολόγηση των διαθέσιμων λογισμικών επιλέχτηκε το Flight Gear.Αυτό γιατί ήταν ο μοναδικός εξομοιωτής που ενσωμάτωνε κάποια πολύ χρήσιμα χαρακτηριστικά. Μπορεί να

τρέξει σχεδόν σε όλα τα λειτουργικά συστήματα (cross-platform) όπως Linux, MacOS, BSD, Windows και Solaris. Έχει μικρές απαιτήσεις σε στοιχεία μηχανών (hardware). Είναι εύκολα επεκτάσιμος και παραμετροποιήσιμος.

Το κυριότερο χαρακτηριστικό όμως ήταν ότι το Flight Gear έχει δημιουργηθεί από μια ομάδα ανθρώπων που μοιράζονται το πάθος για την πτήση και την πίστη στην κοινότητα του ελεύθερου κώδικα (Free source). Αυτό πρακτικά σημαίνει ότι υπάρχει, ένας πλήρως λειτουργικός εξομοιωτής πτήσεων και όλα τα υποσυστήματα αυτού διαθέσιμα σε μορφή πηγαίου κώδικα (source code), ελεύθερος για χρήση από όποιον τα χρειάζεται.

Το τελευταίο χαρακτηριστικό μας έδωσε την δυνατότητα στην ουσία να ελέγξουμε της εσωτερικές μεταβλητές της προσομοίωσης και έτσι να αξιοποιήσουμε χιλιάδες γραμμές κώδικα που έχουν γραφτεί από πολλά άλλα άτομα.

4.3 Δένδρο καταγραφής μεταβλητών.

Το Flight Gear έχει ως κύριο άξονα σχεδίασης την εύκολη πρόσβαση και διαχείριση των επιμέρους χαρακτηριστικών της εξομοίωσης μέσω ενός δένδρου καταγραφής των μεταβλητών (Property Tree). Αυτό το δένδρο μεταβλητών θεωρείτε το "κεντρικό νευρικό σύστημα" του εξομοιωτή. Αυτό γιατί δίνει την δυνατότητα πρόσβασης σε χαμηλού επιπέδου (low level) μεταβλητών κατάστασης (state variables) σε πραγματικό χρόνο (real time). Το Flight Gear Property Tree είναι ο κοινός παρανομαστής για την λειτουργία της εξομοίωσης και για την πρόσβαση στις εσωτερικές μεταβλητές της εξομοίωσης.

Η δομή του είναι παρόμοια με το σύστημα αρχείων (File system) του Unix. Η περιήγηση στις διάφορες μεταβλητές του συστήματος γίνεται με τις εντολές cd, ls, set, get κτλ. Η πρόσβαση στο Property Tree μπορεί να γίνει μέσο Telnet (το οποίο όμως προτείνεται μόνο για "περιήγηση" στην δομή του δένδρου), ή με την σύνδεση μέσω socket σε μια πόρτα του συστήματος (localhost), και την χρήση πρωτοκόλλων μεταφοράς (TCP , UDP κτλ.).

Το Telnet είναι ένα πρωτόκολλο στο επίπεδο των εφαρμογών (OSI application layer), που επιτρέπει τον απομακρυσμένο έλεγχο ενός συστήματος. Το Telnet κατασκευάστηκε το 1969, για χρήση στα τοπικά δίκτυα πανεπιστημίων ή μεγάλων ερευνητικών κέντρων. Πλέον η χρήση του στο διαδίκτυο είναι περιορισμένη λόγο της ανύπαρκτης ασφάλειας που παρέχει, και έχει αντικατασταθεί από το SSH. Για να ξεκινήσουμε το Flight Gear με την δυνατότητα πρόσβασης στις μεταβλητές γράφουμε στο Terminal του συστήματος την παρακάτω εντολή.

```
gecko@ubuntu:  fgfs -- telnet=5400
```

Με αυτήν την σύνταξη λέμε στο σύστημα να ξεκινήσει το Flight Gear με ανοιχτή πόρτα για επικοινωνία μέσω Telnet στην πόρτα 5400. Μόλις η σύνδεση είναι επιτυχής θα δούμε στο Terminal κατά την διάρκεια τις εκκίνησης του Flight Gear το παρακάτω μήνυμα.

Property server started on port 5400

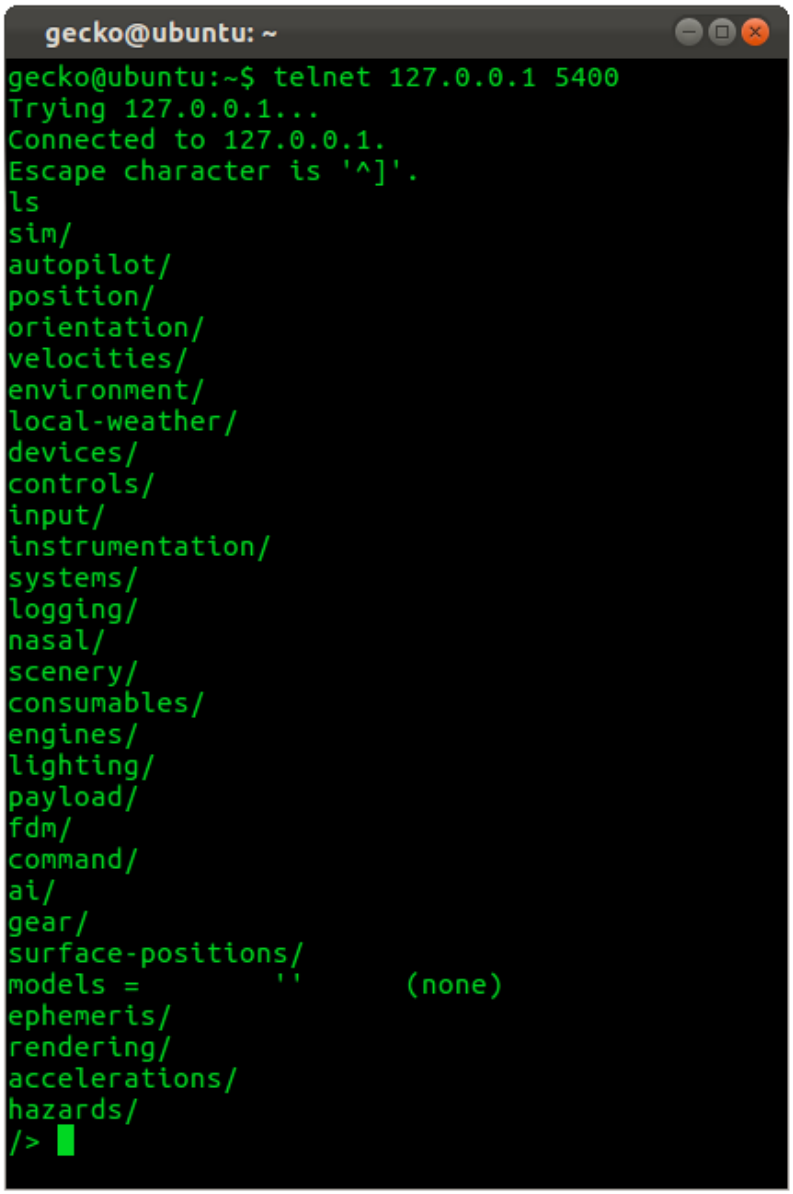
Για να συνδεθούμε τώρα στο Flight Gear εκκινούμε από ένα άλλο Terminal το πρωτόκολλο Telnet με την παρακάτω εντολή.

```
gecko@ubuntu:  telnet 127.0.0.1 5400
```

Οι παράμετροι 127.0.0.1 και 5400 είναι η διεύθυνση του μηχανήματος (στην παρούσα του λέμε να συνδεθεί στο παρών μηχάνημα (localhost) στην πόρτα 5400. Μόλις η σύνδεση είναι επιτυχής θα δούμε στο Terminal που έχουμε ανοίξει το telnet το παρακάτω μήνυμα.

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'
```

Αυτό σημαίνει ότι είμαστε συνδεδεμένοι με το Flight Gear. Κάτω από αυτό το μήνυμα θα είναι ο δρομέας που περιμένει την εντολή που θέλουμε να εκτελέσουμε. Αν γράψουμε την εντολή ls θα παρουσιαστεί το Property Tree του Flight Gear 4.4.



Σχήμα 4.4: Το Property Tree του FG

Ας υποθέσουμε ότι ενδιαφερόμαστε για την παρούσα θέση του αεροσκάφους στον χώρο. Για να βρούμε τις συγκεκριμένες μεταβλητές θα κάναμε το εξής.

```
/>cd position
```

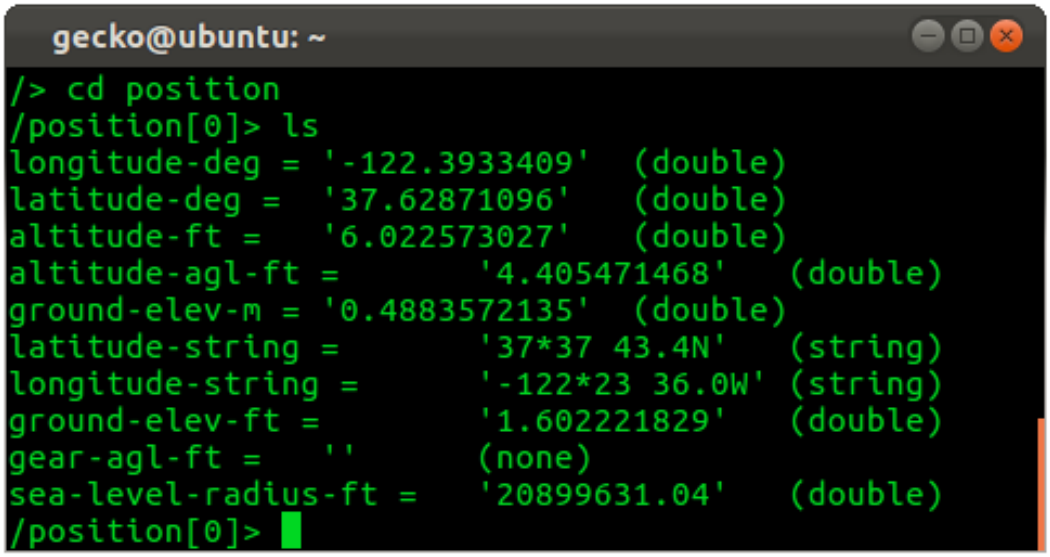
Αυτό μας πάει στον φάκελο position

```
/position[0]>
```

Είμαστε στον φάκελο position

```
/position[0]>ls
```

Αυτό θα μας επιστρέψει τις μεταβλητές που αφορούν την θέση του αεροσκάφους 4.5 (στις παρενθέσεις βλέπουμε τον τύπο της κάθε μεταβλητής).



Σχήμα 4.5: Κόμβος position

Με παρόμοιο τρόπο μπορούμε να έχουμε πρόσβαση σε οποιαδήποτε μεταβλητή επιθυμούμε. Βλέπουμε λοιπόν πόσο εύκολο είναι να έχουμε πρόσβαση σε πραγματικό χρόνο στις μεταβλητές της εξομοίωσης. Παρά την ευκολία στον συγκεκριμένο τρόπο διασύνδεσης για την πρόσβαση και ανάγνωση των μεταβλητών, αυτός ο τρόπος δεν είναι και ο καταλληλότερος όταν θέλουμε να έχουμε πρόσβαση για ανάγνωση ή θέση πολλών μεταβλητών ταυτόχρονα.

Για να είναι δυνατή αυτή η λειτουργία πρέπει να αναζητήσουμε έναν διαφορετικό τρόπο διασύνδεσης με το Flight Gear.

4.4 Διασύνδεση μέσω Socket και του πρωτοκόλλου TCP

Το Flight Gear μας δίνει την δυνατότητα να δημιουργήσουμε διαύλους επικοινωνίας μέσω Socket και των γνωστών πρωτοκόλλων TCP, UDP. Στην παρούσα πτυχιακή χρησιμοποιήθηκε το πρωτόκολλο TCP καθώς πρώτος στόχος είναι να έχουμε αξιόπιστη μεταφορά των δεδομένων από και προς τον εξομοιωτή. Πέραν του καθορισμού του πρωτοκόλλου που θα χρησιμοποιήσουμε για το επίπεδο μεταφοράς (OSI transfer layer), θα πρέπει να καθορίσουμε και το πρωτόκολλο στο επίπεδο εφαρμογών (OSI application layer).

Το Flight Gear μας δίνει την δυνατότητα να δημιουργήσουμε δικά μας πρωτόκολλα για αυτό το επίπεδο με την γενική ονομασία generic. Σε ένα generic πρωτόκολλο εμείς ορίζουμε ποιες μεταβλητές θα αποστέλλονται και ποιες μεταβλητές θα λαμβάνονται από το Flight Gear για επεξεργασία.

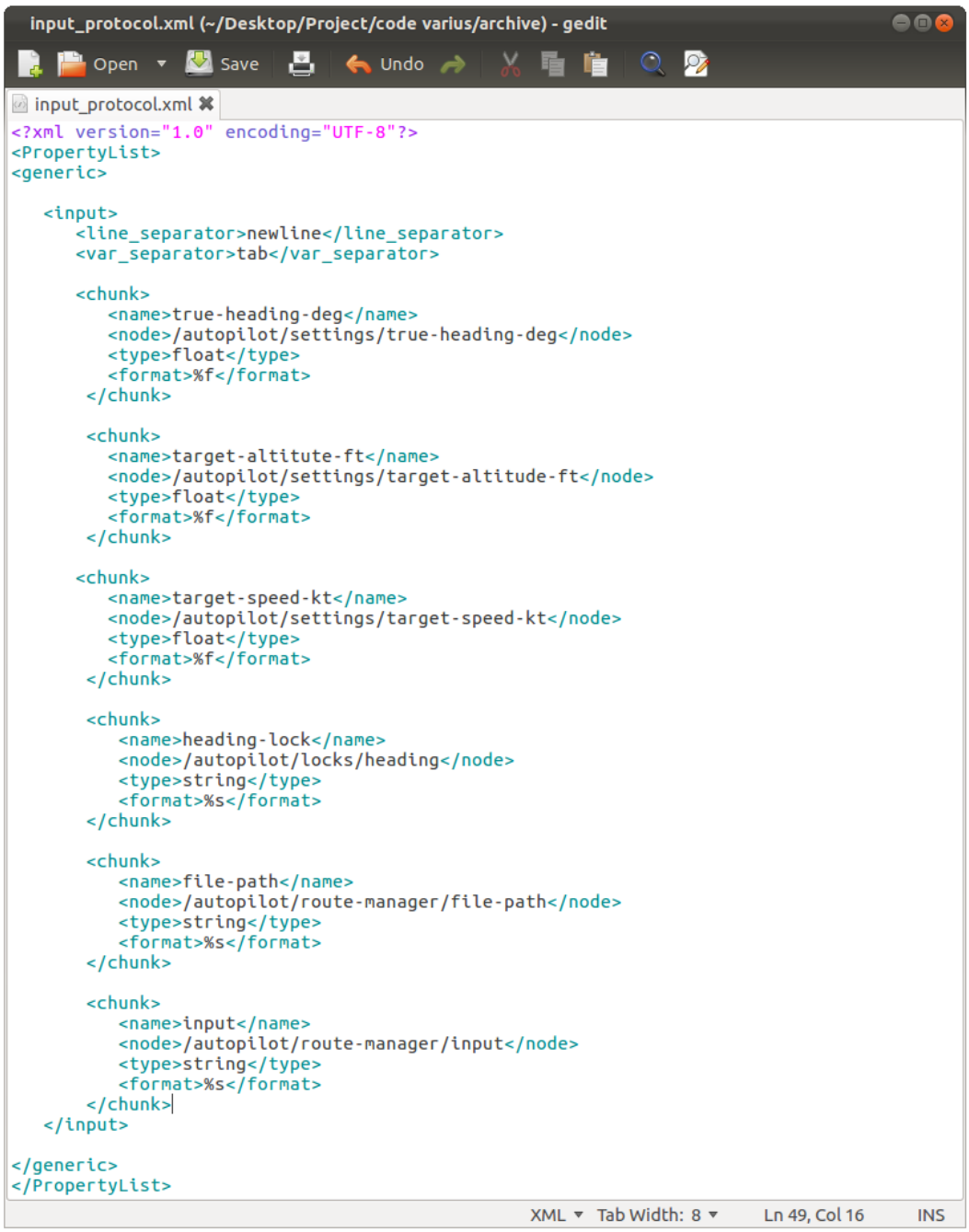
Τα πρωτόκολλα γράφονται σε μορφή .xml (Extensible Markup Language).Η XML είναι μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Μπορούμε να δημιουργήσουμε ξεχωριστά αρχεία, ένα για το πρωτόκολλο εισόδου και ένα για το πρωτόκολλο εξόδου, ή ένα κοινό αρχείο που θα καθορίζονται και τα δύο πρωτόκολλα. Το Flight Gear θα χρησιμοποιήσει μόνο τα δεδομένα που περικλείονται για το εκάστοτε κανάλι ανάλογα με το πως έχει κληθεί το πρωτόκολλο.

Εμείς δημιουργήσαμε δύο πρωτόκολλα, ένα για τις μεταβλητές που θέλουμε να διαβάζουμε και ένα για τις μεταβλητές που θα στέλνουμε. Αυτό έγινε για να είναι ποιο εύκολη η αποσφαλμάτωση κατά την διάρκεια των δοκιμών. Παρατίθεται το πρωτόκολλο με κατεύθυνση από την εφαρμογή μας στον εξομοιωτή 4.6, με την ονομασία input_protocol.xml.

Στην πρώτη γραμμή ενημερωνόμαστε για την έκδοση της xml (η 1.0 είναι η πέμπτη έκδοση) και την κωδικοποίηση που χρησιμοποιούνται. Η ετικέτα input ενημερώνει το Flight Gear ότι είναι πρωτόκολλο εισόδου. Κάθε μεταβλητή που μας ενδιαφέρει είναι και ένα "κομμάτι" (chunk) μέσα στο αρχείο. Κάθε κομμάτι έχει ένα όνομα (name), για δική μας διευκόλυνση. Το σημείο στο Property Tree που παρέχεται αυτή η πληροφορία (node). Τον τύπο της μεταβλητής (type) και την διαμόρφωση που θα έχει (format). Με τον ίδιο τρόπο αλλά με ετικέτα output δημιουργούμε και το πρωτόκολλο εξόδου (με ανάλογη ονομασία).

Βλέπουμε λοιπόν ότι η δημιουργία ενός πρωτοκόλλου επικοινωνίας δεν είναι δύσκολη, αρκεί να γνωρίζουμε ποιες είναι οι μεταβλητές που μας ενδιαφέρουν.

Ένας τρόπος να βρούμε την θέση των μεταβλητών που μας απασχολούν είναι να συνδεθούμε μέσω Τελνετ όπως περιγράφηκε στην προηγούμενη ενότητα. Όταν τις εντοπίσουμε μπορούμε να φτιάξουμε τα πρωτόκολλα που μας ενδιαφέρουν. Βλέπουμε ότι αυτός ο τρόπος σύνταξης και δημιουργίας του πρωτοκόλλου μας επιτρέπει να διαφοροποιούμε γρήγορα και εύκολα το πρωτόκολλο ανάλογα με τις ανάγκες μας.



Σχήμα 4.6: Το πρωτόκολλο εισόδου `input_protocol`

Αποθηκεύοντας το αρχείο στον φάκελο `/FG_ROOT/Protocol` το κάνουμε διαθέσιμο στο Flight Gear για χρήση. Για να καλέσουμε το Flight Gear με το πρωτόκολλο που έχουμε φτιάξει γράφουμε την εντολή.

```
gecko@ubuntu: $ fgfs -- generic=socket,in,10,localhost,5401,tcp,input_protocol
```

Αυτή η εντολή καλεί το Flight Gear να ξεκινήσει δημιουργώντας ένα κανάλι επικοινωνίας (socket),

με κατεύθυνση εισόδου (in), συχνότητας 10 hz (10), στο παρών μηχάνημα (localhost), στην πόρτα (5401), με πρωτόκολλο μεταφοράς (tcp) και το πρωτόκολλο εφαρμογής (`input_protocol`).

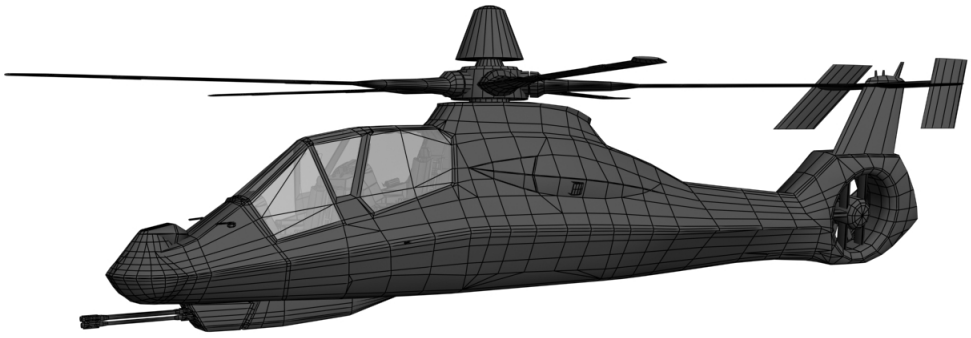
4.5 Δυναμικό μοντέλο πτήσης, FDM.

Κάθε εξομοιωτής πτήσης έχει στην "καρδιά" του ένα μοντέλο το οποίο είναι υπεύθυνο για την προσομοίωση των χαρακτηριστικών πτήσης ενός δοσμένου αντικειμένου σύμφωνα με τους νόμους της φυσικής. Αυτά τα μοντέλα ονομάζονται FDM (flight dynamics model).

Τα FDM χωρίζονται σε δύο βασικές κατηγορίες. Η πρώτη κατηγορία αναπαριστά το αντικείμενο με ένα πίνακα τιμών. Κατά την διάρκεια της εξομοίωσης το μοντέλο ανασύρει, ανάλογα με τις συνθήκες της προσομοίωσης, το ανάλογο σετ τιμών από τον πίνακα και το επιστρέφει στον εξομοιωτή. Αυτοί οι πίνακες έχουν δημιουργηθεί εξ αρχής από μετρήσεις που έχουν γίνει κατά την διάρκεια πραγματικών πτήσεων του αντικειμένου. Αυτά τα μοντέλα ονομάζονται *look-up table models*.

Η δεύτερη κατηγορία είναι αυτή που το μοντέλο για τον υπολογισμό των τιμών χρησιμοποιεί μία γεωμετρική αναπαράσταση του αντικειμένου. Αυτή η γεωμετρική αναπαράσταση δημιουργείται αναλύοντας το όλο αντικείμενο σε πολύ μικρά πολύγωνα. Στην συνέχεια υπολογίζονται οι δυνάμεις που ασκούνται σε κάθε πολύγωνο ξεχωριστά. Η ολική συμπεριφορά του αντικειμένου είναι το αποτέλεσμα της άθροισης του συνόλου των επιμέρους πολυγώνων. Αυτά τα μοντέλα ονομάζονται *geometric models*.

Το πόσο ρεαλιστικό είναι ένα μοντέλο εξαρτάται, στο *look-up table model* από την ποιότητα του πίνακα τιμών. Με αυτό εννοούμε το πλήθος τον διακριτών καταστάσεων και την ποσότητα των παραμέτρων που είναι διαθέσιμες. Στο *geometric model* ο ρεαλισμός εξαρτάται από το πόσο ακριβής είναι η αναπαράσταση του αντικειμένου, καθώς και η ανάλυση των πολυγώνων που έχουμε.



Σχήμα 4.7: Γεωμετρική αναπαράσταση του RAH-66 Comanche

Το Flight Gear έχει την δυνατότητα να χρησιμοποιεί τρία βασικά μοντέλα προσομοίωσης. Αυτά είναι το JSBSim, το YASim και το UIUC.Τα μοντέλα YASim και UIUC έχουν δημιουργηθεί εξολοκλήρου για το Flight Gear. Στις πρώτες του εκδόσεις, το Flight Gear χρησιμοποιούσε ένα FDM

βασισμένο στο LaRCsim που είχε αναπτύξει η NASA, το οποίο όμως αντικαταστάθηκε από πιο ευέλικτα FDM. Για πιο εξειδικευμένα μοντέλα όπως η εξομοίωση υπέρελαφρών αεροσκαφών μπορούν είτε να δημιουργηθούν νέα FDM ή να χρησιμοποιηθούν τα δεδομένα που παράγονται εξωτερικά από έτοιμα εξειδικευμένα FDMs.

– *JSBSim*: Είναι το προεπιλεγμένο FDM που χρησιμοποιεί το Flight Gear από το 2000 και μετά. Δημιουργήθηκε από τον *Jon S. Berndt* το 1996 και η βασικές αρχές της σχεδίασης του ήταν να μην έχει μεγάλες απαιτήσεις σε πόρους του συστήματος (lightweight), να είναι μη γραμμικός (non-linear), και να έχει έξι βαθμούς ελευθερίας (6 DoF). Είναι γραμμένο σε C++ και για την παραμετροποίηση του χρησιμοποιεί XML αρχεία. Αυτό το μοντέλο είναι τύπου *look-up table model*.

–Το YASim: Είναι ένα FDM που έχει στόχο την ευκολία στην χρήση. Η πρώτη του κυκλοφορία ήταν το Σεπτέμβριο του 2002, από τον Andy Ross, στην συνέχεια προστέθηκε στην ομάδα και ο Maik Justus. Μέχρι στιγμής είναι το μοναδικό FDM το οποίο μπορεί να παρέχει δεδομένα προσομοίωσης για αεροσκάφη που για την παραγωγή άνωσης χρησιμοποιούν ρότορα. Ο διαχωρισμός αυτών των αεροσκαφών γίνεται από τον ICAO με την φράση, " πτήση υποστηριζόμενη από την αντίσταση του αέρα σε έναν ή περισσότερους ρότορες" Η κύρια διαμόρφωση σε αυτήν την κατηγορία είναι τα ελικόπτερα. Σε αυτή την κατηγορία περιλαμβάνονται όμως και αρκετές, λιγότερο γνωστές, διαμορφώσεις όπως, το αυτόγυρο (autogyro) και το πιο φουτουριστικό ελικοπλάνο (heliplane). Αυτό το μοντέλο είναι τύπου *geometrical model*.

Σχήμα 4.8: Ένα ελικοπλάνο!

–Το UIUC: Είναι ένα FDM που αναπτύχθηκε από το Εργαστήριο Εφαρμοσμένης Αεροδυναμικής UIUC στο University of Illinois at Urbana-Champaign ο οποίος βασίστικε στον LaRCsim. Δεν χρησιμοποιείται πολύ πλέον.

4.6 Περίληψη

Σε αυτό το κεφάλαιο είδαμε τι είναι ο εξομοιωτής πτήσεως και από τι αποτελείτε. Παρουσιάσαμε τον εξομοιωτή πτήσεως Flight Gear. Τον τρόπο που είναι δομημένος και τους τρόπους που μπορούμε να εισάγουμε και να εξάγουμε δεδομένα από και πρός το Flight Gear.

Τέλος είδαμε πως γίνεται η μοντελοποίηση ενός αεροσκάφους στο εσωτερικό ενός εξομοιωτή, με τα μοντέλα *look up table model* και *geometric model*, και είδαμε μερικά FDM.

Στο επόμενο κεφάλαιο θα δούμε την γλώσσα προγραμματισμού Python και τις βιβλιοθήκες που θα χρησιμοποιήσουμε.

Κεφάλαιο 5

Η γλώσσα προγραμματισμού Python

Σχήμα 5.1: Ο πύθωνας έτοιμος για δράση.

5.1 Η γλώσσα προγραμματισμού Python

Η Python σχεδιάστηκε από τον *Guido van Rossum* και εμφανίστηκε το 1991. Είναι εύκολη στην σύνταξή της και ευανάγνωστη. Είναι δυνατή γλώσσα καθώς υποστηρίζει πολλά είδη προγραμματισμού, όπως αντικειμενοστραφής, διαδικαστικός, συναρτησιακός και διαθέτει μια ποικιλία βιβλιοθηκών για διάφορες χρήσεις. Μπορεί επίσης να εκτελεστεί σε διάφορα λειτουργικά συστήματα χωρίς περιορισμούς (cross-platform) και έχει μέχρι την έκδοση v3.0 συμβατότητα προς τα πίσω. Πρέπει να αναφερθεί ότι η Python αναπτύσσεται και διαχειρίζεται από το μη κερδοσκοπικό ίδρυμα *Python Software Foundation*. Ο κώδικας διανέμεται ελεύθερα με την άδεια *Python Software Foundation License* που είναι συμβατή με την GPL.

Σχήμα 5.2: Το logo της Python

5.2 Εγκατάσταση και χρήση της Python

Η εγκατάσταση της Python είναι πολύ εύκολη στο λειτουργικό GNU/Linux. Μπορεί να γίνει με δύο τρόπους. Ο πρώτος είναι μέσω του Package Manager να επιλέξουμε την εφαρμογή και να την εγκαταστήσουμε και ο δεύτερος είναι μέσω του Terminal γράφοντας:

```
gecko@ubuntu:   sudo apt-get install python
```

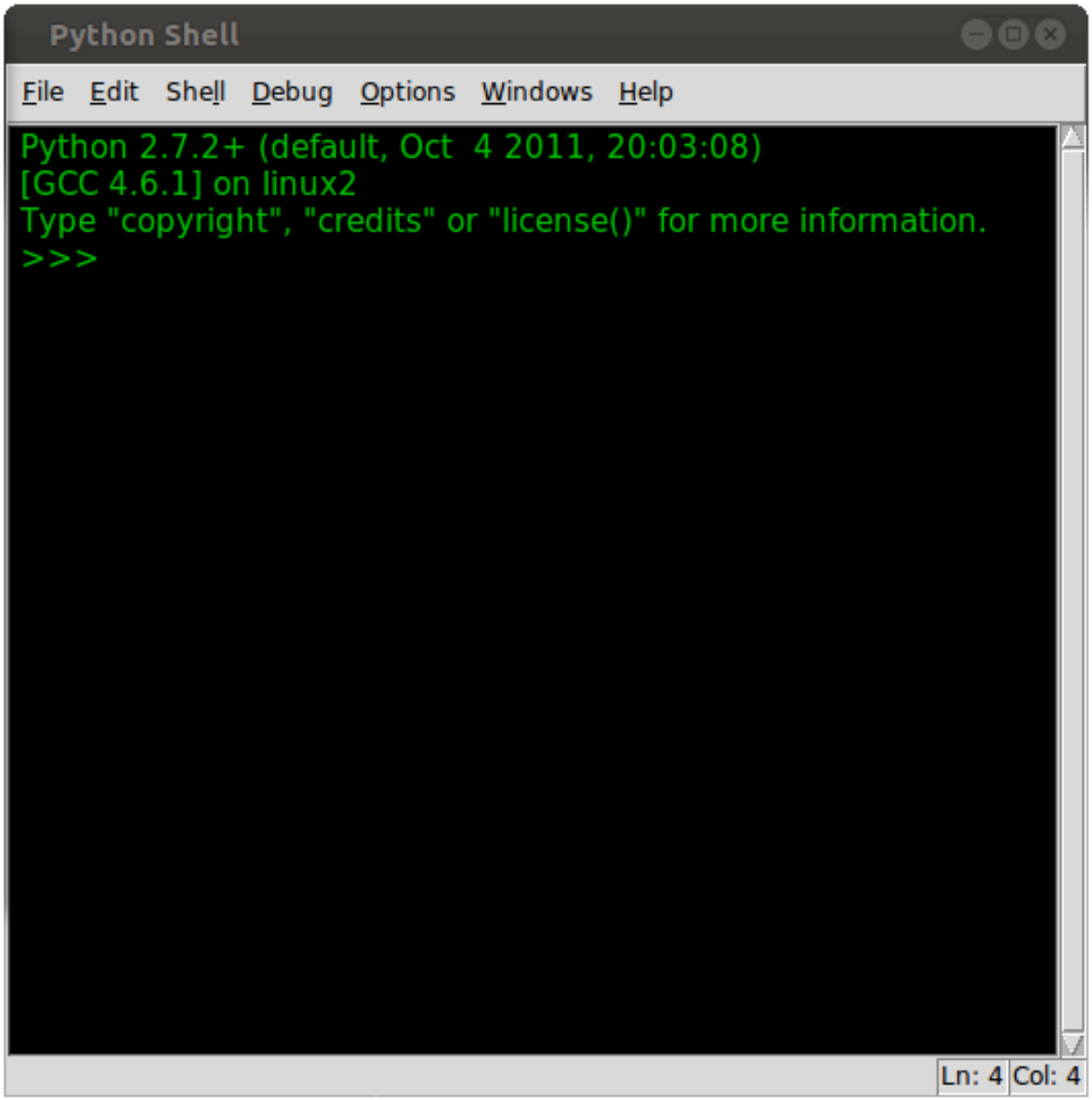
Πληροφορίες για την εγκατάστση σε άλλα λειτουργικά συστήματα μπορούν να βρεθούν στην διεύθηνση `http://www.python.org/`. Για να γράψουμε το πρώτο μας πρόγραμμα πρέπει να εγκαταστήσουμε και έναν IDE (Integrated Development Enviroment) ένα πρόγραμμα δηλαδή που μας παρέχει όλα τα απαραίτητα εργαλεία για να γράψουμε μια εφαρμογή. Το βασικό IDE της Python είναι ο IDLE. Για να εγκαταστήσουμε το IDLE ανοίγουμε ένα Terminal και εισάγουμε την εντολή:

```
gecko@ubuntu:   sudo apt-get install idle
```

Για να ξεκινήσουμε το IDLE ανοίγουμε ένα Terminal και εισάγουμε την εντολή:

```
gecko@ubuntu:   idle
```

Στην 5.3 βλέπουμε ένα shell της Python που έχουμε ανοίξει με την βοήθεια του IDLE

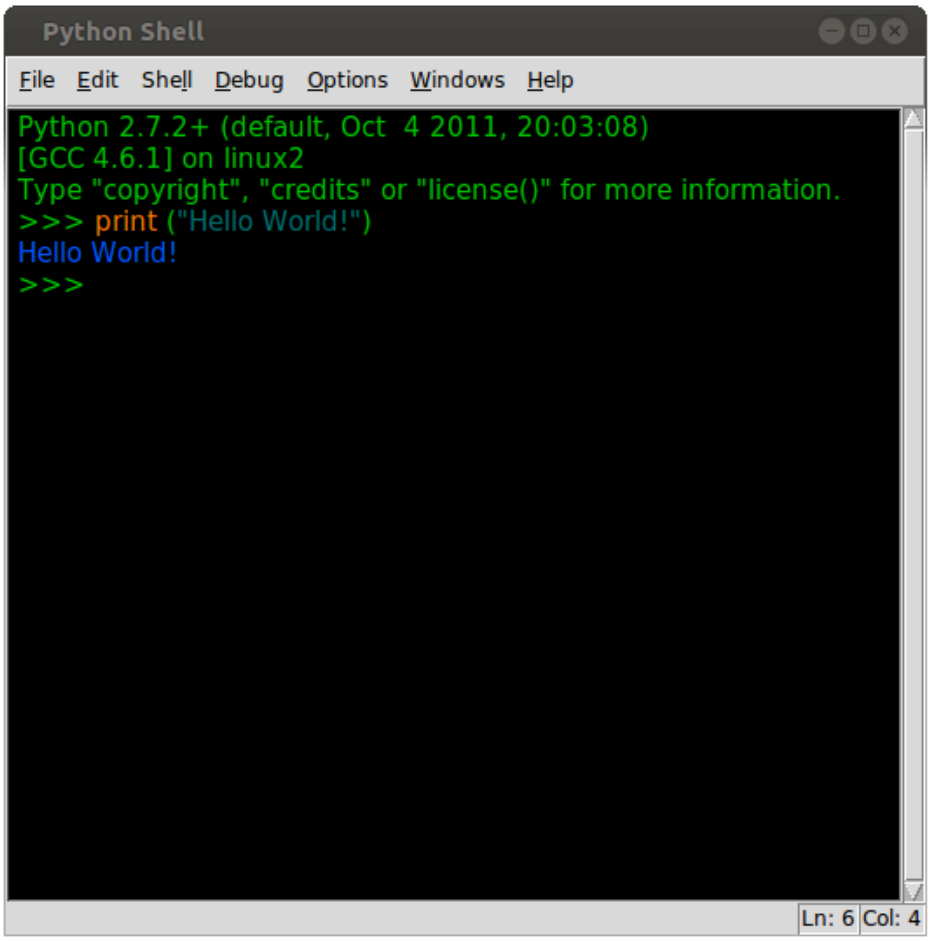


Σχήμα 5.3: Ο IDLE

5.3 Ενα απλό πρόγραμμα στην Python

Το κυρίως παράθυρο του IDLE είναι ένας Interpreter της Python. Αυτό σημαίνει ότι η εντολές εκτελούνται με το που πατήσουμε το πλήκτρο enter. Το κλασσικό παράδειγμα που χρησιμοποιείται για την επίδειξη διαφόρων γλωσσών προγραμματισμού είναι αυτό του προγράμματος *Hello World*, αλλά το να απεικονιστεί ένα μήνυμα στην οθόνη για την Pythonείναι πολύ σύντομο...

Ας δούμε λοιπόν κάτι πιο "σύνθετο".



Σχήμα 5.4: Το πρόγραμμα *Hello World!* στην Python

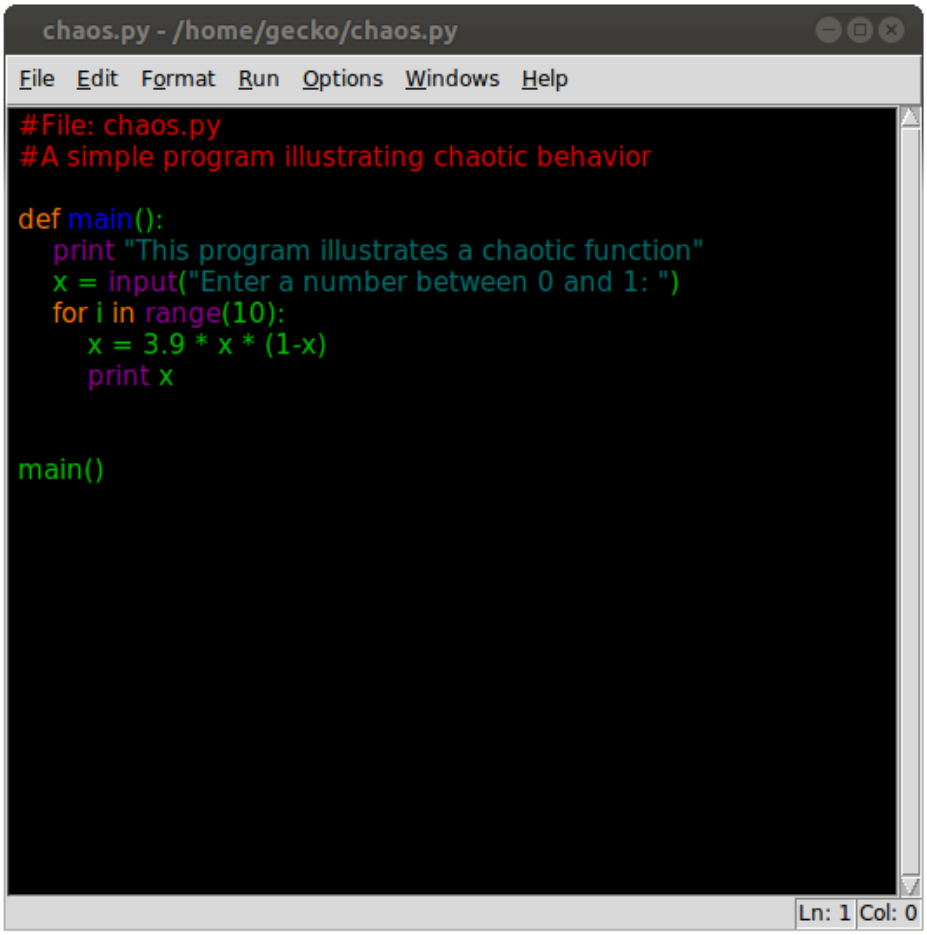
Για να εκτελέσουμε μια σειρά από εντολές μπορούμε να ορίσουμε μια συνάρτηση. Όσο έχουμε ανοιχτό το παράθυρο θα μπορούμε να την καλούμε, αλλά όταν κλείσουμε τον interpreter θα χάσουμε και την συνάρτηση που είχαμε δημιουργήσει.

Για να μπορούμε να κρατήσουμε τις συναρτήσεις και τα προγράμματα που γράφουμε, θα πρέπει να ανοίξουμε ένα νέο αρχείο τύπου .py. Πάμε στην καρτέλα File και επιλέγουμε New Window. Στο καινούριο παράθυρο που άνοιξε γράφουμε τον κώδικα που θέλουμε και στην συνέχεια τον αποθηκεύουμε με ένα όνομα. Για να αποθηκεύσουμε το αρχείο πάμε στην καρτέλα File και επιλέγουμε Save As... . Για να εκτελέσουμε το αρχείο γράφουμε στο Terminal.

```
gecko@ubuntu: $ python <nameoffile>.py
```

Μπορούμε να εντάξουμε ένα αρχείο σε ένα άλλο. Αυτό γίνεται με την εντολή ιμπορτ ακολουθούμενη, από το όνομα του αρχείου προς ένταξη, στο κυρίως αρχείο.

Στην 5.5 βλέπουμε τον κώδικα ενός προγράμματος ο οποίος όταν κληθεί ζητάει να εισάγουμε έναν αριθμό ανάμεσα στο 0 και το 1. Στην συνέχεια επιστρέφει 10 αριθμούς που αποτυπώνουν την συμπεριφορά μιας χαοτικής συνάρτησης.

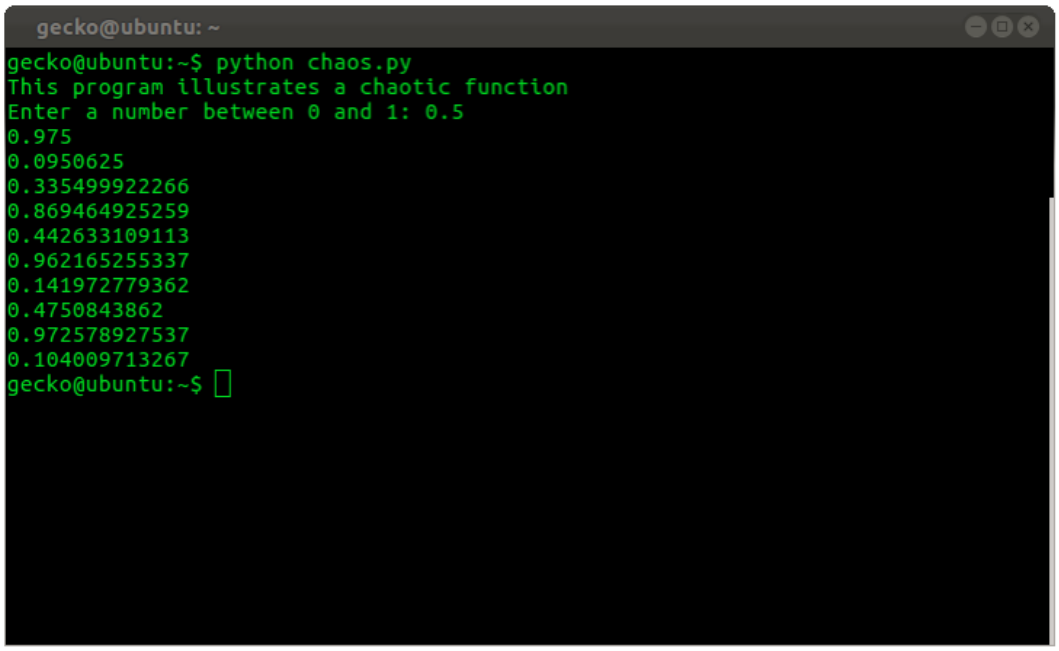


Σχήμα 5.5: Ο κώδικας του chaos.py

Για να καλέσουμε το πρόγραμμα chaos.py ανοίγουμε ένα Terminal και γράφουμε:

```
gecko@ubuntu: $ python chaos.py
```

Στην 5.6 βλέπουμε τα αποτελέσματα του προγράμματος chaos.py



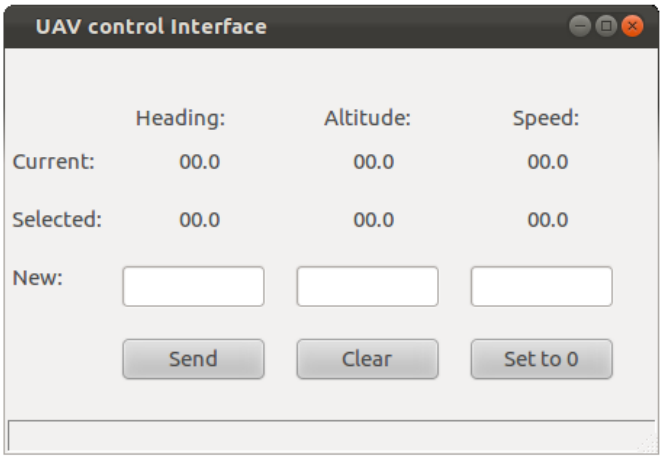
Σχήμα 5.6: Τα αποτελέσματα του chaos.py

5.4 Βιβλιοθήκες που χρησιμοποιήθηκαν

Οι βιβλιοθήκες που έχει η κάθε γλώσσα είναι συλλογές εντολών και στοιχείων που μας επιτρέπουν να χειριστούμε συγκεκριμένες καταστάσεις. Το πλέον κατανοητό παράδειγμα είναι η κατασκευή ενός GUI (Graphical User Interface). Το GUI για να σχεδιαστεί χρειάζεται επιγραμματικά ένα πλαίσιο το οποίο θα περιλαμβάνει κουμπιά και άλλα χειριστήρια καθώς ίσως και κάποιους ενδείκτες για την απεικόνιση των αποτελεσμάτων. Όλα τα απαιτούμενα γραφικά και οι συναρτήσεις ελέγχου περικλείονται σε μια βιβλιοθήκη. Για να μπορέσουμε να κάνουμε χρήση μιας βιβλιοθήκης αρκεί να την συμπεριλάβουμε στον κώδικα το όνομα της μετά από την εντολή import.

5.5 Η βιβλιοθήκη `wxpython`

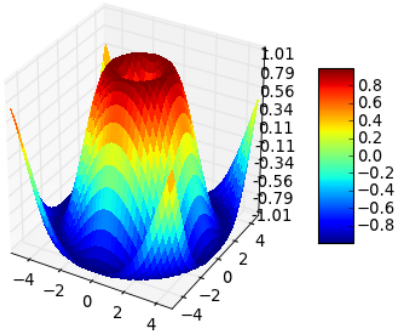
Η βιβλιοθήκη `wxpython` είναι η πλέον διαδεδομένη βιβλιοθήκη γραφικών της Python. Μας επιτρέπει να δημιουργήσουμε γραφικά για σχεδόν οποιαδήποτε εφαρμογή, καθώς επιτρέπει την επανασχεδίαση του παραθύρου σε πολύ μικρούς χρόνους. Αυτό μας δίνει την δυνατότητα σχεδίασης κινούμενων γραφικών. Επίσης έχει πολλές δυνατότητες για αυτόματη στοίχιση των γραφικών που σχεδιάζουμε σε ένα παράθυρο. Ακόμα ενσωματώνει την αυτόματη αλλαγή διαστάσεων σε περίπτωση αλλαγής των διαστάσεων του κεντρικού παραθύρου από τον χρήστη.



Σχήμα 5.7: Μια πρώτη προσέγγιση για το GUI επικοινωνίας με το FG

5.6 Η βιβλιοθήκη `matplotlib`

Η βιβλιοθήκη `matplotlib` είναι μια βιβλιοθήκη σχεδίασης κυρίως σε δύο διαστάσεις (μετά την ενσωμάτωση κώδικα του *John Porter* έχει την δυνατότητα τρισδιάστατης σχεδίασης. Μας επιτρέπει να σχεδιάσουμε ποικιλία γραφικών παραστάσεων όπως ιστογράμματα, διαγράμματα με μπάρες κ.α σε πολύ καλή ποιότητα. Επίσης έχουμε την δυνατότητα σχεδίασης διαδραστικών γραφικών για τον καλύτερο χειρισμό τους από τον χρήστη.



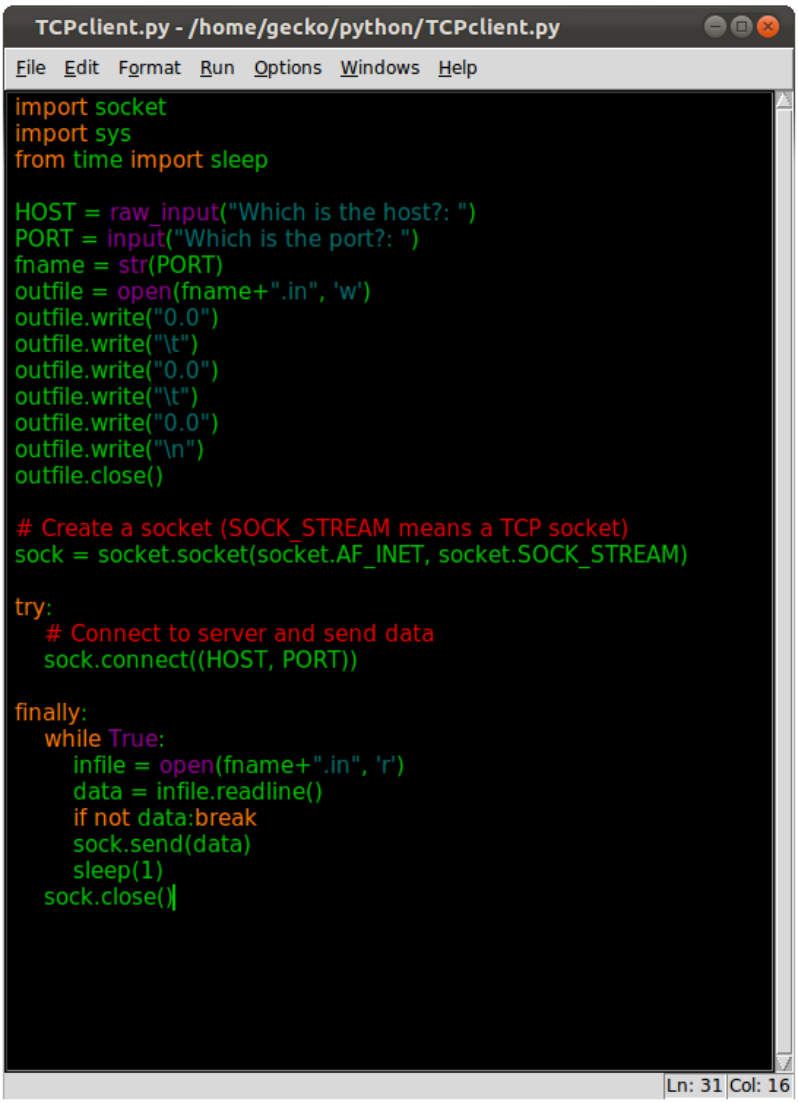
Σχήμα 5.8: 3D Διάγραμμα

5.7 Η βιβλιοθήκη διασύνδεσης socket

Η βιβλιοθήκη `socket` της Python είναι μια μεταφορά της εντολής κλήσης UNIX και της βιβλιοθήκης διασύνδεσης, στην αντικειμενοστραφή προσέγγιση της `Python`. Η συνάρτηση `socket()` επιστρέφει ένα αντικείμενο τύπου socket. Οι μέθοδοι που έχει το αντικείμενο socket χειρίζονται τις λειτουργίες που μπορούμε να εκτελέσουμε. Οι τύποι των παραμέτρων είναι υψηλότερου επιπέδου από ότι στην

διεπαφή που έχει η C. Για παράδειγμα με τις συναρτήσεις `read()` και `write()` το buffer allocation γίνεται αυτόματα.

Οι διευθύνσεις τύπου IF_INET ορίζονται με μία σύνταξη τύπου (Host, Port) όπου το string Host είναι η διεύθυνση στο διαδίκτυο και το string Port είναι μια ακέραια τιμή πόρτας στο σύστημα. Στην 5.9 βλέπουμε το πρόγραμμα TCPclient.py.



Σχήμα 5.9: Το TCPclient.py

Αυτό το πρόγραμμα στην αρχή διαβάζει από τον χρήστη τα ορίσματα Host και Port. Στην συνέχεια δημιουργεί ένα αρχείο με το όνομα <port>.out και γράφει ένα μήνυμα σαν προεπιλογή. Μετά δημιουργεί μια διεπαφή (socket) και προσπαθεί να συνδεθεί στον server. Όταν συνδεθεί διαβάζει και και στέλνει κάθε δευτερόλεπτο το μήνυμα που βρίσκεται στο αρχείο μέσο της σύνδεσης.

5.8 Η βιβλιοθήκη SocketServer

Στην 5.10 έχουμε το πρόγραμμα TCPserver.py το οποίο βρίσκεται στο άλλο άκρο της γραμμής.

Σχήμα 5.10: Το TCPserver.py

Για την υλοποίηση του server είναι απαραίτητη η χρήση της βιβλιοθίκης SocketServer. Αυτή η βιβλιοθήκη μας δίνει τα βασικά εργαλεία για να φτιάξουμε τον server. Η σχεδίαση ενός server είναι κάπως περίπλοκη. Ο σκοπός όμως της πτυχιακής δεν είναι η σχεδίαση server και για αυτό δεν θα προχωρήσουμε σε λεπτομέρειες .Αυτά που πρέπει να γνωρίζουμε είναι ότι υπάρχουν τέσσερα είδη σερερ που μπορούμε να υλοποιήσουμε, εκ των οποίων τα χρησιμότερα είναι ο TCP server και ο UDP server. Το ποιόν θα επιλέξουμε εξαρτάται από την εφαρμογή που υλοποιούμε. Ένα ακόμα σημαντικό σημείο είναι ο handler που θα χρησιμοποιήσουμε με τον πλέον απλό να είναι ο BaseRequestHandler.

Αυτό το πρόγραμμα αρχικά παίρνει από τον χρήστη τα ορίσματα Host και Port. Στην συνέχεια απευθυνόμαστε στον handler λέγοντάς του να διαχειριστεί όποιο αίτημα παρουσιαστεί. Ο handler ακούει την πόρτα που του έχει ανατεθεί. Όταν έρθει κάποιο αίτημα ακολουθεί τα βήματα που έχουμε

ορίσει στην συνάρτηση `handle()`. Τα βήματα είναι να αποθηκεύσει την πληροφορία που έρχεται σε μία μεταβλητή (self.data), και στην συνέχεια να την γράψει σε ένα αρχείο που ονομάζεται <str(port)>.out

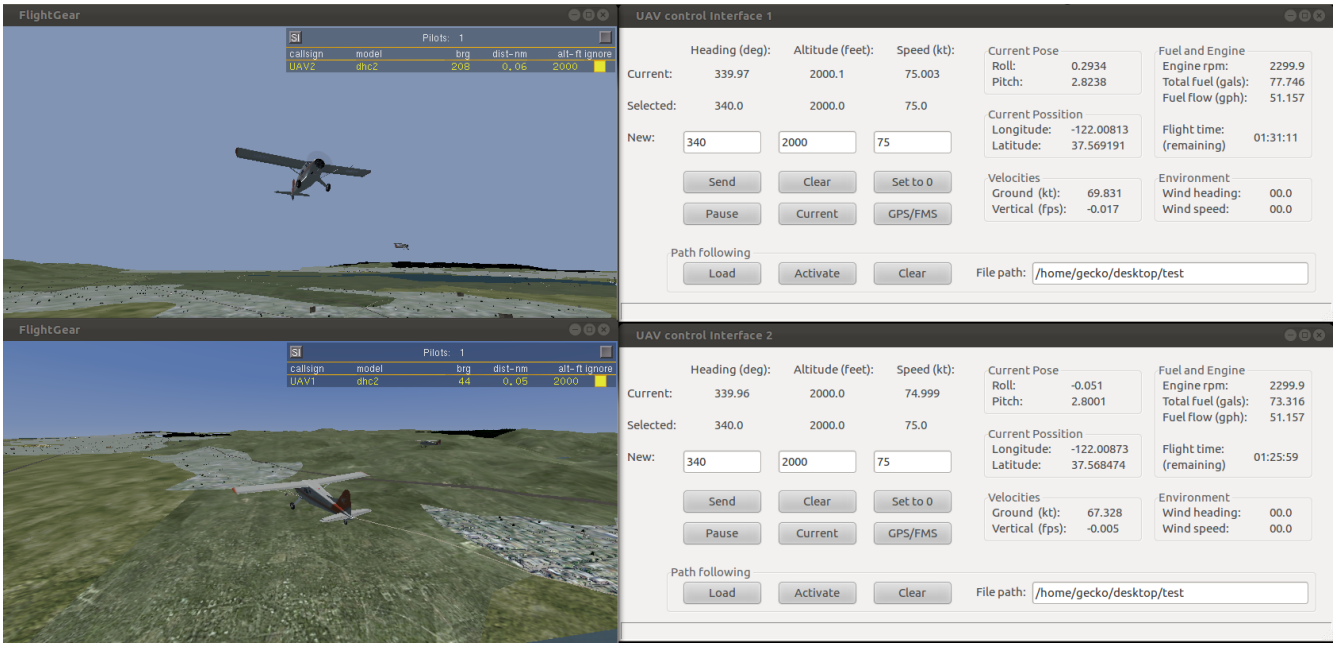
5.9 Περίληψη

Στο κεφάλαιο αυτό είδαμε την γλώσσα προγραμματισμού Python, και τις βιβλιοθήκες που χρησιμοποιήσαμε. Γνωρίσαμε το IDE της Python τον IDLE, είδαμε πώς μπορούμε να τον εγκαταστήσουμε και να τον χειριστούμε. Γράψαμε ένα στοιχειώδες πρόγραμμα και είδαμε πως το εκτελούμε. Αυτό είναι το τελευταίο κεφάλαιο του πρώτου μέρους.

Στο επόμενο κεφάλαιο θα δούμε πως με την βοήθεια της Python και του Flight Gear. Δημιουργίσαμε το περιβάλλον στο οποίο υλοποιήθικε η προσέγγιση του Dubins για τον σχεδιασμό πτήσης.

Κεφάλαιο 6

Η Εφαρμογή



Σχήμα 6.1: Πετόντας σε σχηματισμό με την βοήθεια του UAV control Interface

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δούμε πως συνδυάσαμε αυτά που περιγράψαμε στο πρώτο μέρος για να δημιουργήσουμε μια εφαρμογή που μας δίνει την δυνατότητα να ελέγξουμε πολλαπλά αεροσκάφη από απόσταση.

Η μέθοδος που ακολουθήθηκε μας επιτρέπει την μετατροπή του εκάστοτε αεροσκάφους του εξομοιωτή σε UAV. Δεν είναι αναγκαίο λοιπόν κάθε αεροσκάφος να το χειρίζεται και ένας χειριστής. Με αυτόν τον τρόπο μας δίνεται η δυνατότητα ελέγχου πολλών αεροσκαφών ταυτόχρονα από έναν χειριστή. Αυτή η λειτουργία είναι ένα από τα ζητούμενα στο σύγχρονο τρόπο διαχείρισης UAV και μας επιτρέπει να εξερευνήσουμε τις δυνατότητες ενός τέτοιου συστήματος.

Η χρήση ενός εξομοιωτή που έχει την δυνατότητα να αναπαράγει διάφορες συνθήκες πτήσης όπως μέρα / νύχτα, μετεωρολογικά φαινόμενα και πραγματικά (μη ιδανικά) αεροσκάφη βάζει σε σκληρή δοκιμασία την υλοποίηση μιας διαδρομής υπολογισμένη από τον αλγόριθμο του Dubin. Η μέχρι τώρα προσεγγίσεις για την υλοποίηση δοκιμών στην υλοποίηση διαδρομών βασίζονταν σε ιδανικές συνθήκες (συμπεριφορά αεροσκάφους, καιρός κτλ) κάτι το οποίο στερούταν σε ρεαλισμό.

Με αυτήν την εφαρμογή προσπαθούμε να δώσουμε όλες τις παραμέτρους, για την σχεδίαση συστημάτων που επιτρέπουν την εκτέλεση μιας διαδρομής, όσο ποιο ρεαλιστικά γίνεται.

6.2 Σχεδιασμός της εφαρμογής

Η εφαρμογή γράφτηκε στην γλώσσα προγραμματισμού Python. Η επιλογή της συγκεκριμένης γλώσσας έγινε γιατί η Python είναι μια πολύ ευέλικτη γλώσσα, χωρίς περίπλοκους τρόπους σύνταξης και διαθέτει μια μεγάλη και ενεργή κοινότητα που την υποστηρίζει. Επίσης η Python χρησιμοποιείται εκτενώς στην βιομηχανία στον τομέα της ανάπτυξης νέων εφαρμογών (Prototyping).

Ο βασικός στόχος ήταν η εφαρμογή να μπορεί να διαχειρίζεται ταυτόχρονα πολλούς *κλώνους* (instancies) του Fligth Gear. Αυτό μας ώθησε στην δημιουργία ξεχωριστών κομματιών τα οποία στην συνέχεια θα "κουμπώνανε" με το κάθε στιγμιότυπο ξεχωριστά. Αυτό μας επιτρέπει να έχουμε ξεχωριστά νήματα threads για την διαχείριση του κάθε αεροσκάφους, χρησιμοποιώντας έτσι μόνο τους πόρους που είναι αναγκαίοι σε κάθε περίσταση. Σε αντίθετη περίπτωση θα έπρεπε να σπαταλούμε πόρους που στην ουσία θα ήταν ανενεργοί και έτσι θα είχαμε προβλήματα στους υπολογισμούς πραγματικού χρόνου που απαιτεί η εφαρμογή.

Στην 6.2 βλέπουμε μια σχηματική αναπαράσταση της δομής της υλοποίησης στο σύνολό της. Το σύνολο χωρίζεται σε δύο υποσύνολα. Από την μία πλευρά βρίσκονται τα τμήματα του FG(πορτοκαλί πλαίσιο), και από την άλλη τα τμήματα της εφαρμογής που δημιουργήθηκε. Με τα βελάκια απεικονίζονται οι ροές πληροφοριών από και προς όλα τα επιμέρους τμήματα.

Αναλυτικότερα οι ροές από το FGMS προς τους κλώνους FG αφορούν το κομμάτι που μας επιτρέπει να δούμε όλα τα αεροσκάφη σε έναν ενιαίο χώρο πτήσης. Ο κάθε κλώνος του Flight Gear στέλνει στον FGMS πληροφορίες για την θέση του στον χώρο και κάποιες άλλες βασικές πληροφορίες όπως το όνομα του αεροσκάφους. Το FGMS από την πλευρά του στέλνει στον κάθε κλώνο, πληροφορίες για τα άλλα αεροσκάφη που βρίσκονται στην περιοχή, τα στίγματα τους στον χώρο και το όνομα τους.

Σχήμα 6.2: Διάγραμμα βαθμίδων της υλοποίησης

Στην άλλη πλευρά (κάτω) βλέπουμε τα τμήματα της εφαρμογής μας. Επιγραμματικά, με το κίτρινο πλαίσιο απεικονίζεται το παράθυρο κεντρικού ελέγχου της εφαρμογής. Εκεί δημιουργούμε τις διαδρομές που στην συνέχεια ακολουθούν τα αεροσκάφη και από εκεί ξεκινάμε ένα νέο UAV Control Interface. Ακόμα έχουμε και την δυνατότητα να παρακολουθούμε τα αεροσκάφη που ελέγχουμε μέσω ενός χάρτη της περιοχής.

Με τα πράσινα πλαίσια διακρίνονται τα ξεχωριστά πάνελ ελέγχου του κάθε αεροσκάφους. Κάθε πάνελ λαμβάνει από το Master Control την διαδρομή που πρέπει να ακολουθήσει το UAV και επιστρέφει τα στίγματα του αεροσκάφους για αναπαράσταση στον χάρτη.

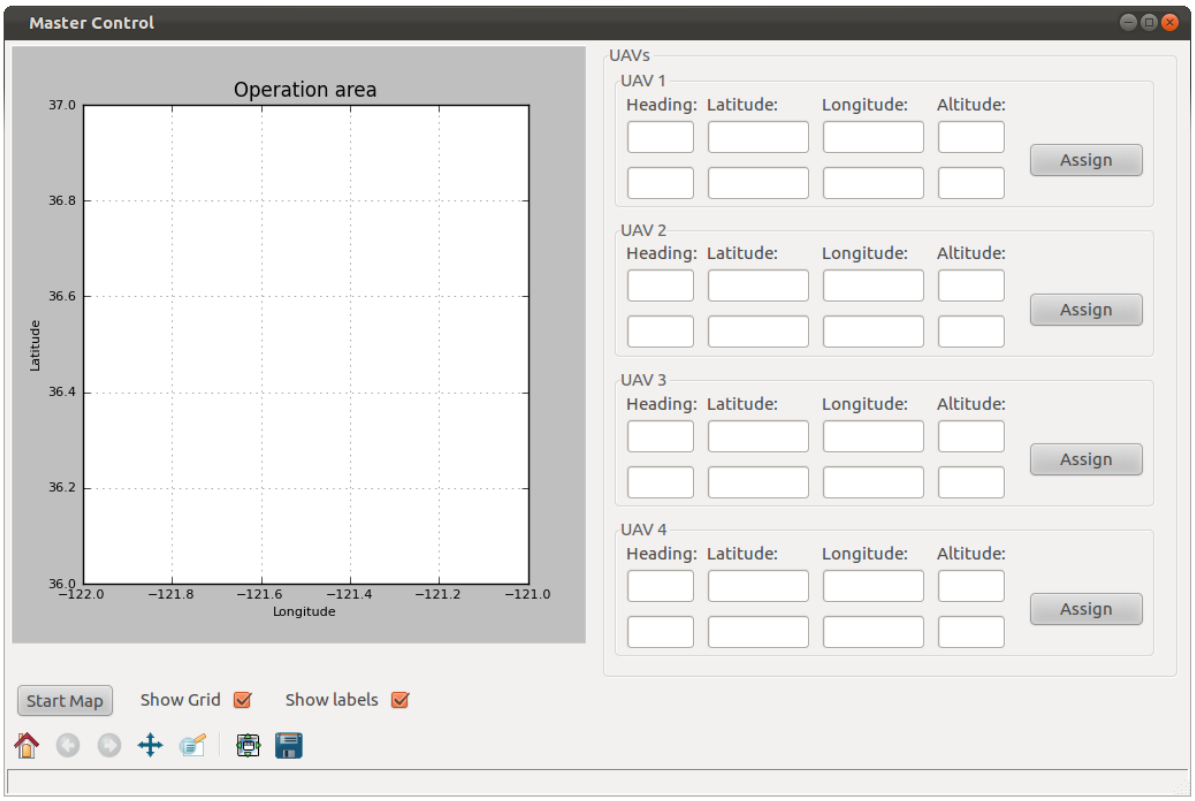
Οι κλώνοι του Flight Gear λαμβάνουν τις εντολές του αυτόματου πιλότου και τον route manager του αεροσκάφους.Από τους κλώνους προς το πάνελ ελέγχου μεταφέρονται οι τιμές των οργάνων του αεροσκάφους. Αυτή η ανταλλαγή πληροφοριών μεταξύ Control Interface και FG Instance γίνονται μέσω ξεχωριστών πελατών client και εξυπηρετητών server για το κάθε ζεύγος.

6.3 Το παράθυρο κεντρικού ελέγχου

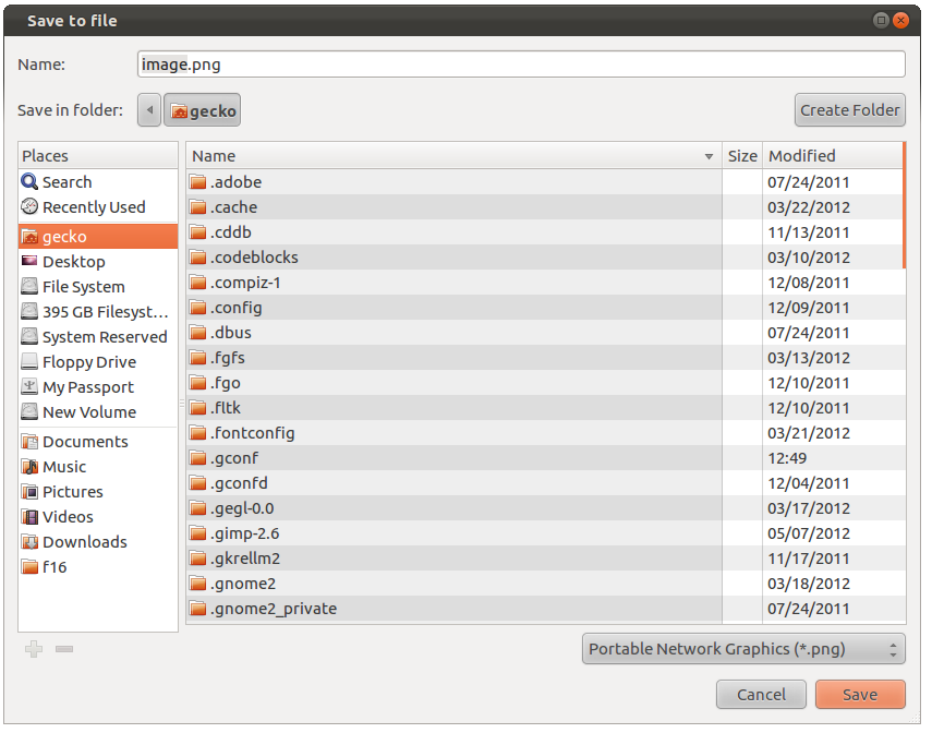
Σε αυτό το παράθυρο 6.3 βλέπουμε στην αριστερή πλευρά τον χάρτη της περιοχής που δραστηριοποιούμαστε με το όνομα Operation area. Στον άξονα των y έχουμε το γεωγραφικό πλάτος (Latitude) και στον άξονα των x έχουμε το γεωγραφικό μήκος (Longitude).

Στην δεξιά πλευρά έχουμε την κονσόλα που δημιουργεί τις διαδρομές Dubins ανάλογα με τις δύο θέσης που θα έχουμε εισάγει. Όταν βάλουμε τις θέσεις κάνουμε κλικ στο κουμπί Assign και αυτό δημιουργεί ένα αρχείο με το όνομα UAV1.pth, UAV2.pth κ.τ.λ. το οποίο περιέχει τα σημεία διαδρομής waypoints που θα φορτωθούν στον route manager του αεροσκάφους (από την καρτέλα UAV Control Interface).

Στην κάτω αριστερά γωνία του παραθύρου έχουμε την γραμμή εργαλείων που μας επιτρέπει να χειριστούμε τον χάρτη και να τον αποθηκεύσουμε 6.4. Επίσης έχουμε την επιλογή που μας επιτρέπει την αλλαγή στα όρια του χάρτη 6.5.

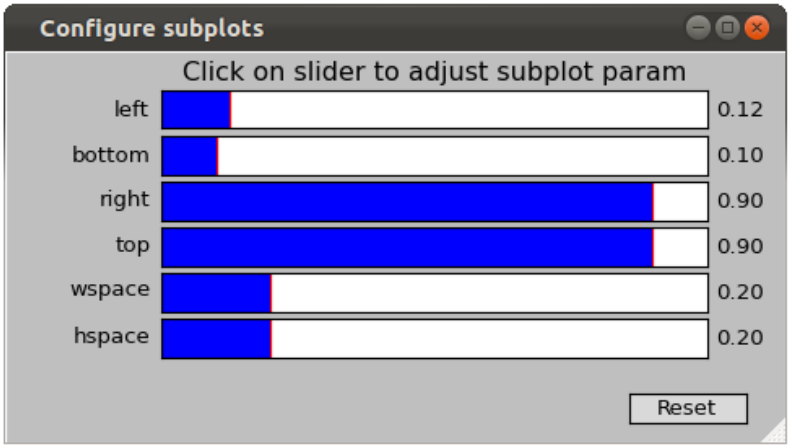


Σχήμα 6.3: Το παράθυρο κεντρικού ελέγχου Master Control



Σχήμα 6.4: Καρτέλα για την αποθήκευση του χάρτη

Για την αποθήκευση επιλέγουμε τον φάκελο που θέλουμε να αποθηκεύσουμε την εικόνα, δίνουμε ένα όνομα και επιλέγουμε `Save`, 6.4.



Σχήμα 6.5: Καρτέλα αλλαγής διαστάσεων γραφήματος

Στην καρτέλα της 6.5 σέρνουμε τις μπάρες εκεί που θέλουμε και παρατηρούμε τις αλλαγές που γίνονται στο γράφημα. Για να διατηρηθούν οι επιλογές μας απλά κλείνουμε το παράθυρο. Με το κουμπί `reset` επανέρχονται οι προεπιλεγμένες ρυθμίσεις.

Για να ανοίξουμε μία καρτέλα UAV control Interface επιλέγουμε

`File >> New UAV`.

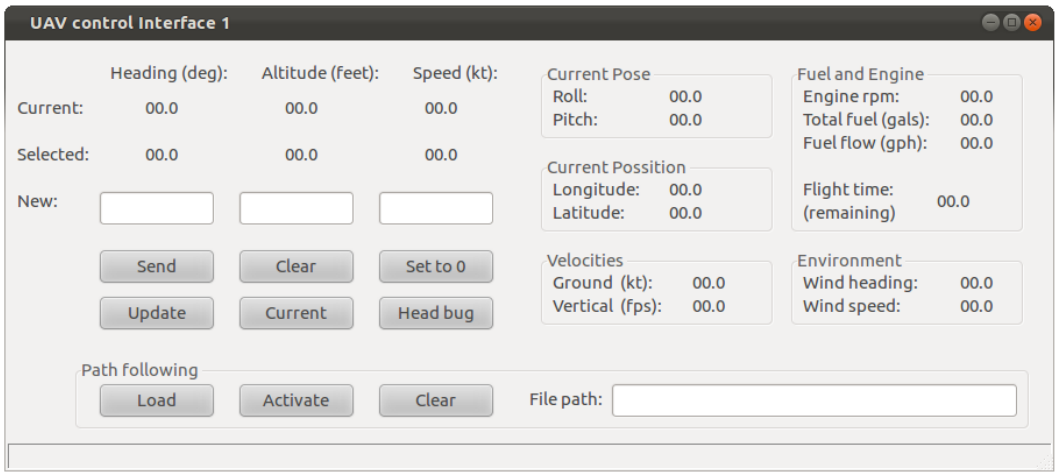
6.4 Το παράθυρο UAV control Interface

Αυτό το παράθυρο είναι η βασική κονσόλα ελέγχου για το κάθε αεροσκάφος. Διαμέσου αυτής της κονσόλας μπορούμε να δώσουμε την κατεύθυνση, το υψόμετρο και την ταχύτητα που θέλουμε να έχει το αεροσκάφος. Παράλληλα έχουμε την δυνατότητα να βλέπουμε διάφορα χαρακτηριστικά του αεροσκάφους σε πραγματικό χρόνο.

Στην πάνω αριστερή γωνία του παραθύρου στην γραμμή (Current) έχουμε τα δεδομένα που μας δείχνουν την κατεύθυνση (Heading (deg)) σε μοίρες που έχει το αεροσκάφος, το υψόμετρο (Altitude (feet» σε πόδια και την ταχύτητα (Speed (kts)) σε κόμβους την συγκεκριμένη στιγμή.Στην γραμμή Selected βλέπουμε ποιες είναι οι τιμές που έχουμε επιλέξει για να ακολουθήσει το αεροσκάφος.

Στην γραμμή New έχουμε τρία Control Boxes στα οποία εισάγουμε τις νέες τιμές που πρέπει να ακολουθήσει το αεροσκάφος. Για να στείλουμε αυτές τις τιμές επιλέγουμε `Send`. Με το κουμπί Clear καθαρίζουμε τις τιμές που έχουμε βάλει στα Control Boxes για να εισάγουμε τις νέες τιμές με ταχύτητα. Το κουμπί Set to 0 επιλέγει μηδενικές τιμές σε όλες τις παραμέτρους στην γραμμή Selected. Με το κουμπί Update ξεκινάει η διαδικασία ανανέωσης των τιμών του αεροσκάφους σε πραγματικό χρόνο. Με το κουμπί Current επιλέγονται οι τρέχοντες τιμές και προωθούνται στην γραμμή Selected.

Στην δεξιά πλευρά του παραθύρου έχουμε διάφορες χρήσιμες πληροφορίες σχετικά με την κατάσταση του αεροσκάφους.



Σχήμα 6.6: Το παράθυρο UAV control Interface

Στο πλαίσιο Current Pose βλέπουμε τις κλήσεις που έχει το αεροσκάφος. Το Roll είναι η περιστροφή στον διαμήκη άξονα x και το Pitch η περιστροφή στον πλευρικό άξονα y, ;;

Το πλαίσιο Current Possition περιέχει τις συντεταγμένες του αεροσκάφους στον χώρο. Το Longitude είναι το γεωγραφικό μήκος και το Latitude είναι το γεωγραφικό πλάτος.

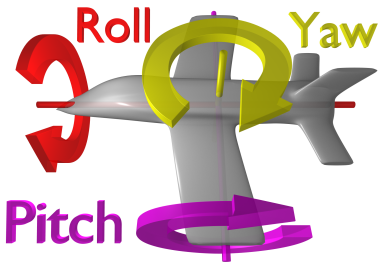
Το πλαίσιο Velocities περιέχει την ταχύτητα εδάφους σε κόμβους (Ground(kt)) και την ταχύτητα που έχουμε στον κάθετο άξονα (Vertical (fps)) σε πόδια το δευτερόλεπτο.

Το πλαίσιο Fuel and Engine περιέχει τις στροφές του κινητήρα (Engine rpm) σε στροφές το λεπτό, το συνολικό φορτίο καυσίμου που έχει το αεροσκάφος σε γαλόνια (Total fuel (gals)), την ροή καυσίμου προς τον κινητήρα σε γαλόνια την ώρα (Fuel flow (gph)) και τέλος τον εναπομείναν χρόνο πτήσης.

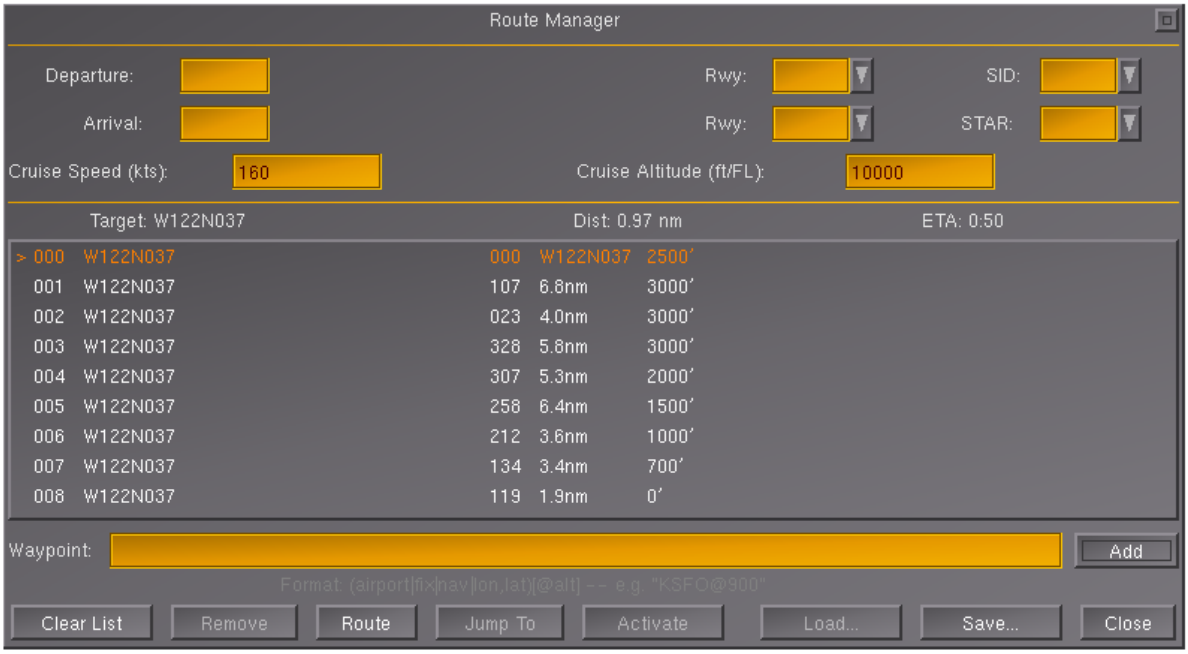
Το πλαίσιο Environment μας δίνει πληροφορίες σχετικά με την κατεύθυνση και την ένταση του αέρα.

Το τελευταίο πλαίσιο είναι αυτό που είναι υπεύθυνο για την μεταβίβαση του σχεδίου πτήσης στον route manager του αεροσκάφους. Στο Control Box με το όνομα File path βάζουμε την διαδρομή που βρίσκεται το αρχείο που που περιέχει τα waypoints έτσι όπως αυτά δημιουργήθηκαν στο παράθυρο Master Control.

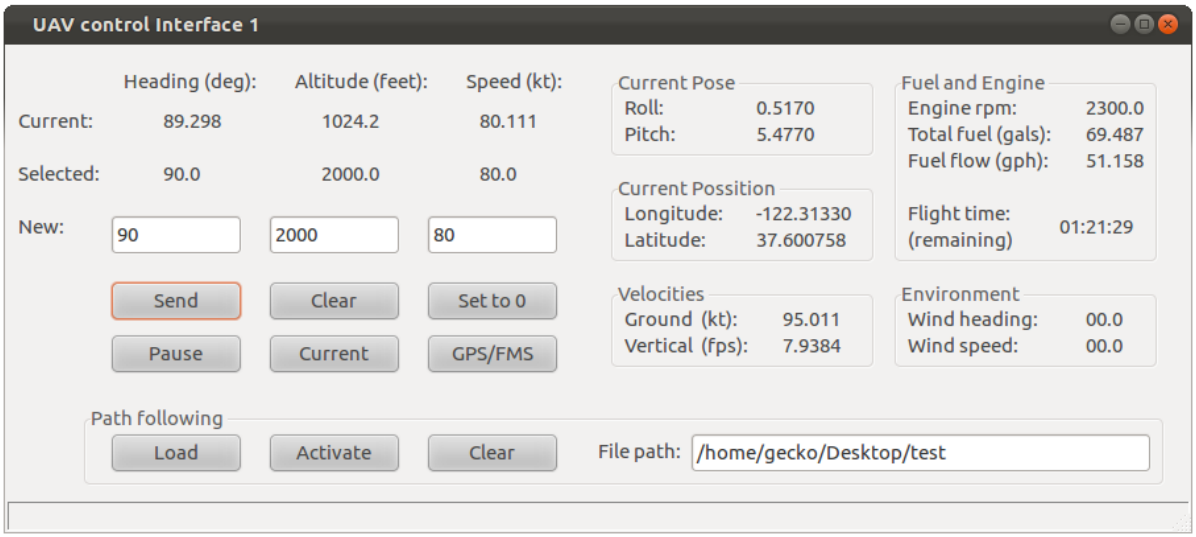
Για να φορτώσουμε το σχέδιο πτήσης στον route manager επιλέγουμε το Load και για να ενεργοποιήσουμε την λειτουργία ακολουθίας του σχεδίου επιλέγουμε το Activate. Σε περίπτωση που θέλουμε να σταματήσουμε να ακολουθούμε την διαδρομή επιλέγουμε το Clear και το αεροσκάφος αυτόματα θα ακολουθήσει τις τιμές που έχουμε στο πεδίο Current. Αν η φόρτωση είναι επιτυχής θα δούμε στον route manager τα σημεία 6.8.



Σχήμα 6.7: Άξονες περιστροφής του αεροσκάφους



Σχήμα 6.8: Ο Route Manager του Flight Gear



Σχήμα 6.9: Το UAV control Interface κατά την διάρκεια πτήσης

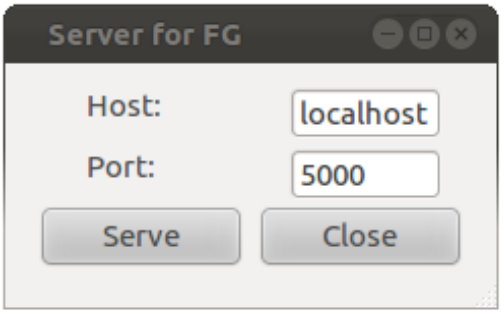
6.5 Τα παράθυρα Server και Client

Για την διασύνδεσή μας με το Flight Gear χρησιμοποιούμε δύο ξεχωριστά παράθυρα το Server for FG φ45 και το Connect to FG φ46. Για να έχουμε πρόσβαση στα παράθυρα επικοινωνίας επιλέγουμε

Interface >> Server for FG, και

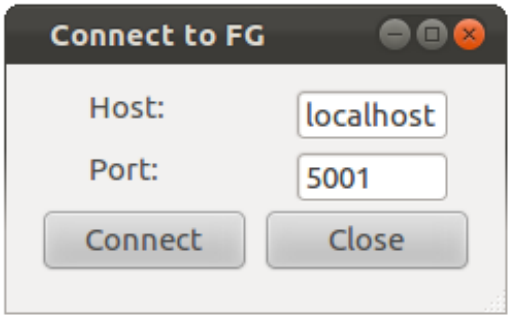
Interface >> Connect to FG.

Και τα δύο έχουν ίδιο χειρισμό. Στο Control Box του Host βάζουμε την διεύθυνση που θέλουμε να συνδεθούμε και στο Control Box του Port βάζουμε την πόρτα που ακούμε ή στέλνουμε τα δεδομένα. Μία λεπτομέρεια είναι ότι ο Server πρέπει να είναι ανοιχτός πριν εκκινήσουμε το Flight Gear ενώ το Connect πρέπει να γίνει αφότου το Flight Gear έχει ξεκινήσει.



Σχήμα 6.10: Ο Server για το Fligth Gear

Πατώντας τα πλήκτρα Serve ή Connect ενεργοποιούμε και την αντίστοιχη λειτουργία.



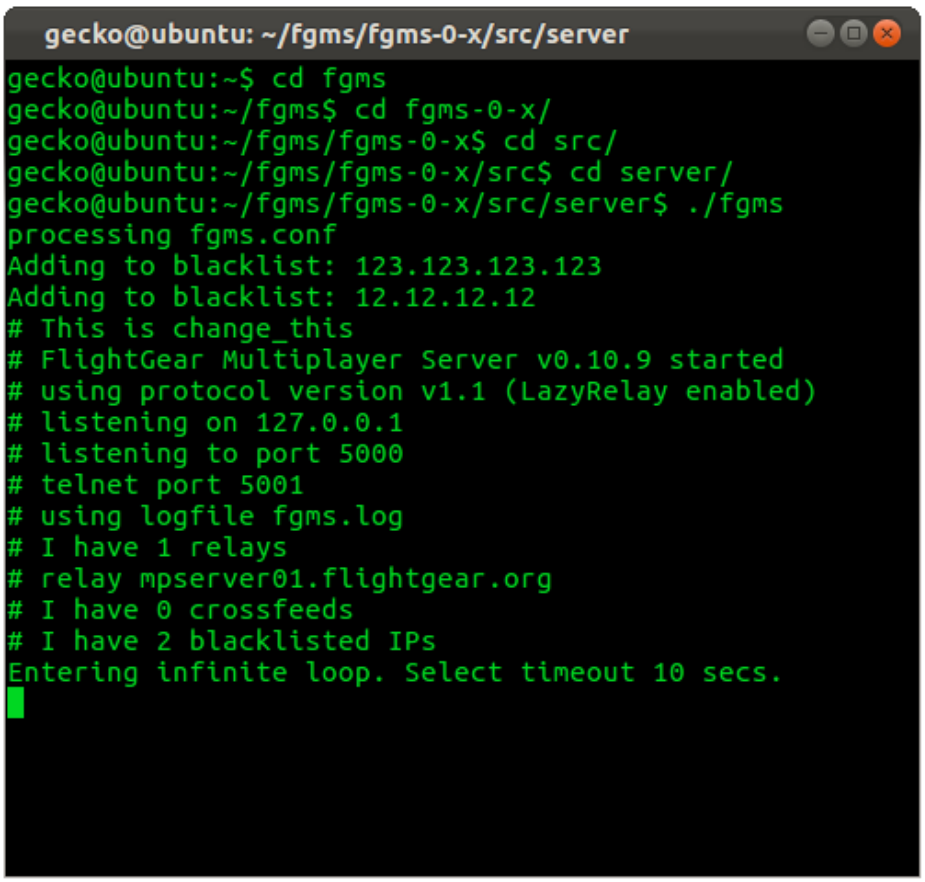
Σχήμα 6.11: Ο Client για το Flight

Μια ακόμα παρατήρηση είναι ότι για να καταχωρηθούν οι τιμές που εισάγαμε στα Control Boxes πρέπει να ακολουθούνται από το πλήκτρο Enter. Με τον συγκεκριμένο τρόπο διασύνδεσης μπορούμε να συνδεθούμε και σε ένα Instance του Flight Gear που τρέχει σε κάποιον υπολογιστή στο δίκτυο (απαιτείται γρήγορη σύνδεση) δίνοντας την IP του και ρυθμίζοντας στο firewall την πρόσβαση στην πόρτα που θέλουμε.

6.6 Τρόπος λειτουργίας και χρήσης των συστημάτων

Όπως είδαμε στα προηγούμενα κεφάλαια για να μπορέσουμε να τρέξουμε την εφαρμογή, πρέπει να συνδυάσουμε την λειτουργία πολλών διαφορετικών προγραμμάτων. Τα βήματα για την λειτουργία της συλλογής είναι τα εξής:

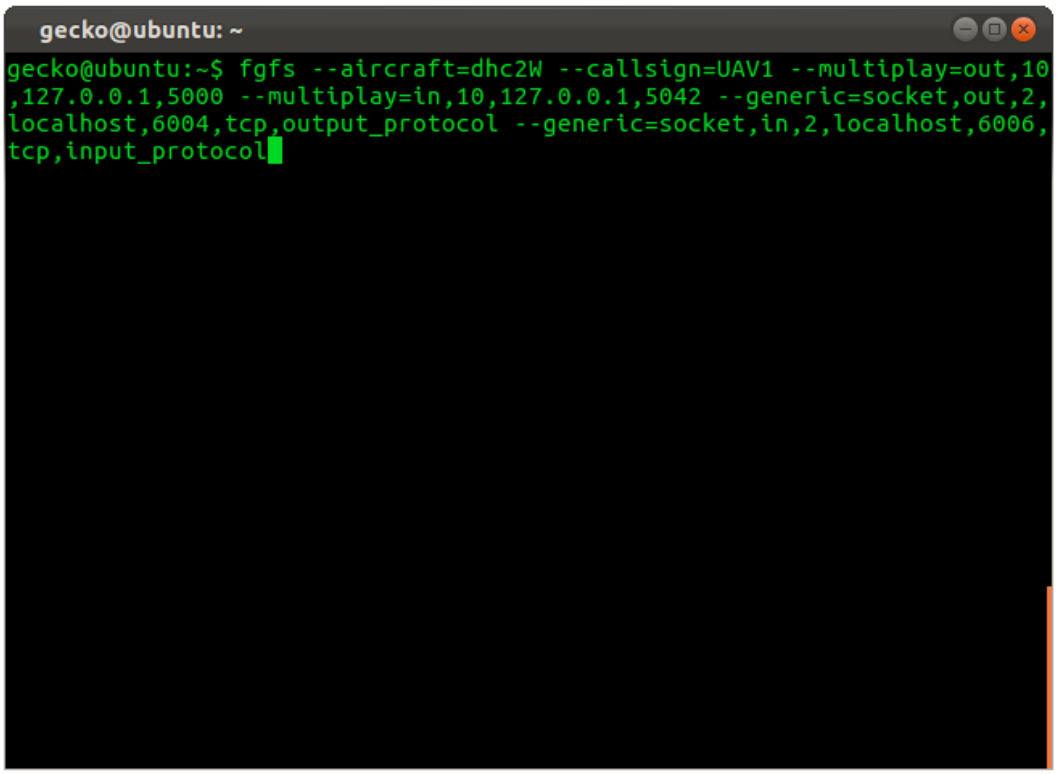
1) Ανοίγουμε ένα Terminal και εκκινούμε το FGMS. Η διαδικασία φαίνεται στην φ47.



Σχήμα 6.12: Διαδικασία εκκίνησης του FGMS

2) Ξεκινάμε την εφαρμογή μας (UAVcommander.py) και κάνουμε τις ρυθμίσεις στην καρτέλα Server for FG. Πρέπει να προσέξουμε ποιες πόρτες θα χρησιμοποιήσουμε για το κάθε Instance του Flight Gear καθώς οι πόρτες 5000 και 5001 χρησιμοποιούνται από τον Server FGMS. Προτείνεται να υπάρχει μια λίστα με γραμμένες τις εκάστοτε πόρτες που θα χρησιμοποιηθούν γιατί ο αριθμός τους είναι μεγάλος. Για να ξεκινήσουμε γράφουμε σε ένα Terminal ¿¿ python uavcommander.py

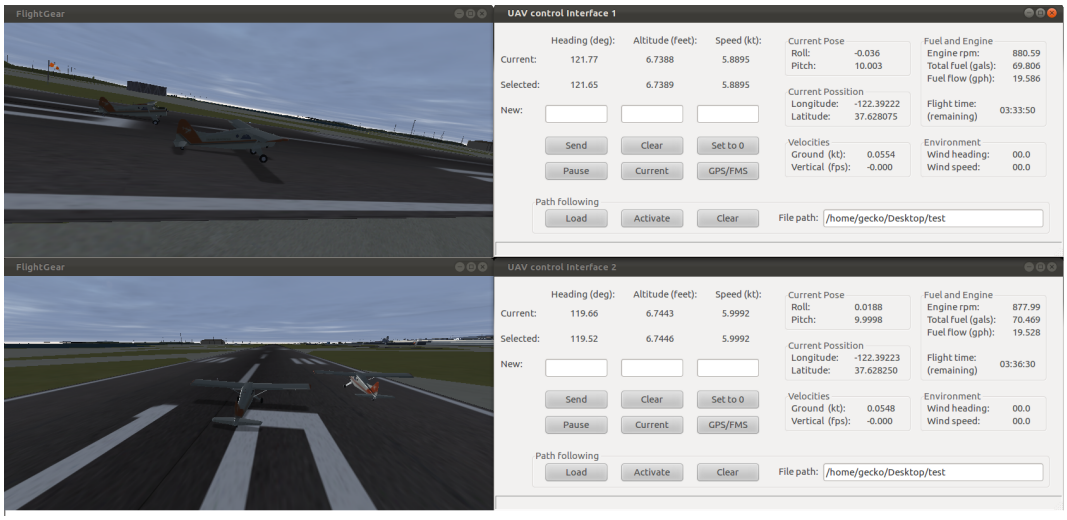
3) Η εκκίνηση των Instances του Flight Gear πρέπει να είναι σειριακή, με αυτό εννοούμε ότι πρώτα πρέπει να έχουμε φτιάξει την διασύνδεση ενός Instance και μετά να προχωρήσουμε στην εκκίνηση του επόμενου. Η εντολή που θα χρησιμοποιήσουμε για να εκκινήσουμε το Instance είναι μεγάλη για αυτό προτείνεται ο έλεγχος πριν δώσουμε το Enter για αποφυγή λαθών. Η εντολή φαίνεται στην φ48



Σχήμα 6.13: Εντολή εκκίνησης του Flight Gear

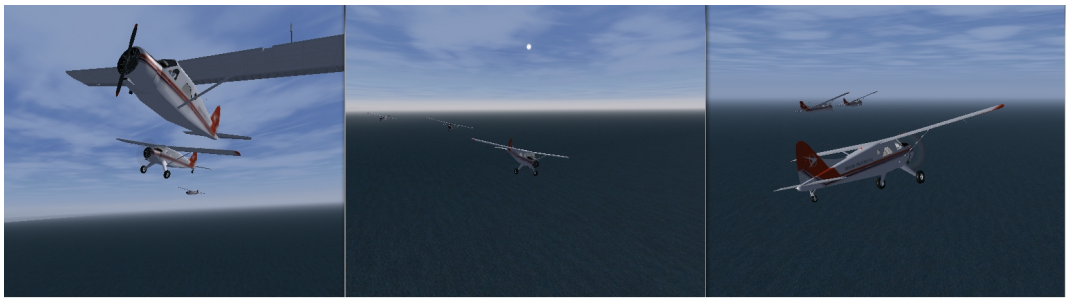
Αυτό που πρέπει να προσέξουμε εκτός από τις πόρτες είναι και όνομα (callsign) που θα έχει το κάθε Instance να είναι διαφορετικό, γιατί αλλιώς ο FGMS δεν θα μπορεί να ξεχωρίσει τα δύο αεροσκάφη.

4) Πάμε στον UAVcommander και ρυθμίζουμε την καρτέλα του Connect to FG, με τα στοιχεία που θέλουμε. Στην προκειμένη περίπτωση ο Host είναι localhost Port είναι 6006. Το αποτέλεσμα με δύο αεροσκάφη θα είναι κάπως έτσι...



Σχήμα 6.14: Ευθυγραμμισμένοι για απογείωση!

Πτήση σε σχηματισμό με τρία αεροσκάφη...



Σχήμα 6.15: Σχηματισμοί

Σχήμα 6.16: Σχηματισμοί

6.7 Περίληψη

Στο παρόν κεφάλαιο παρουσιάστικε η εφαρμογή που αναπτύξαμε, πως είναι δομημένη και ποιος ο τρόπος λειτουργίας της. Στην συνέχεια δείξαμε τα υποσυστήματα που απαιτούνται για να έχουμε πολλαπλά αεροσκάφη στον ίδιο χώρο και πως να ενεργοποιούμε. Τέλος παρατήθονται κάποιες εικόνες από τις δοκιμές.

Στο επόμενο κεφάλαιο θα αναλύσουμε τμήματα του κώδικα και θα εξηγήσουμε τον τρόπο λειτουργίας τους.

Κεφάλαιο 7

Ο ΚΩΔΙΚΑΣ

7.1 Εισαγωγή

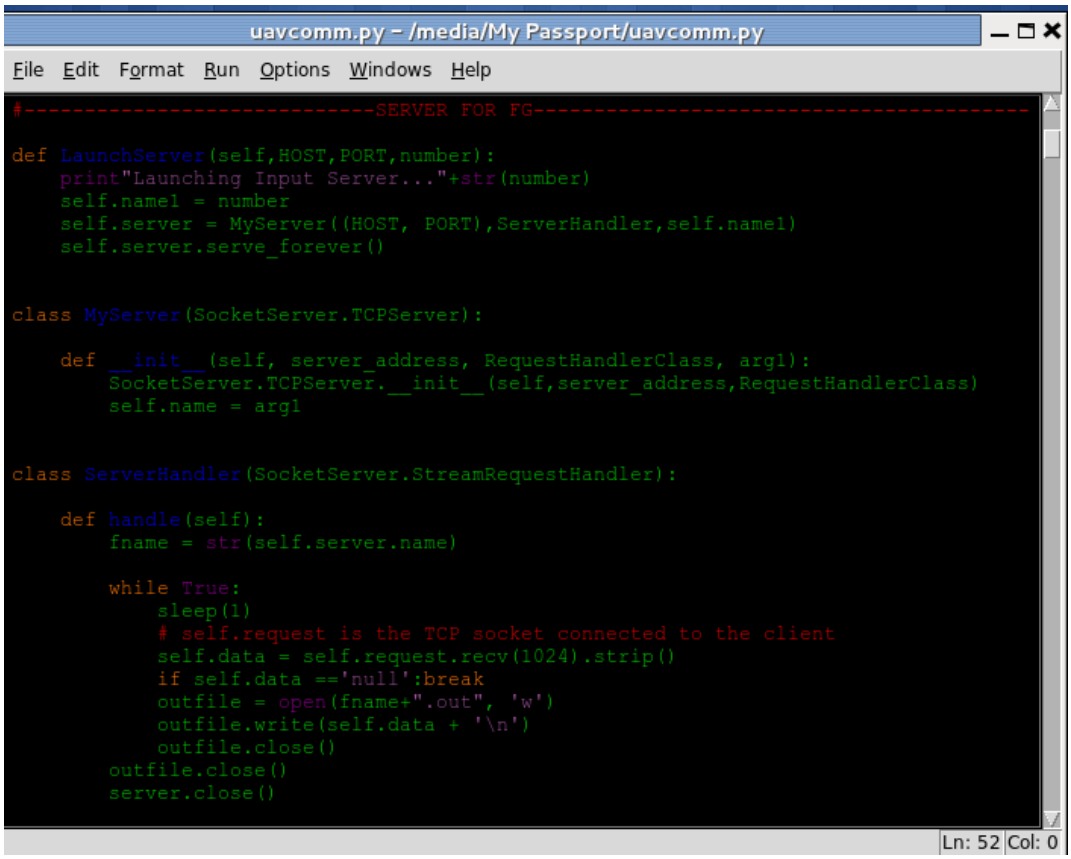
Ο κώδικας μας αποτελείται από δύο τμήματα, το πρώτο περιλαμβάνει την κατασκευή του γραφικού περιβάλλοντος (GUI, graphical user interface) για τον έλεγχο της διαδικασίας παραγωγής των διαδρομών και της επικοινωνίας με τον Flight Gear. Στο δεύτερο τμήμα δήνουμε την υλοποίηση του αλγόριθμου υπολογισμού διαδρομών τουDubins και σχολιάζουμε την λειτουργία του. Επίσης παρουσιάζουμε τις εξισώσεις που χρησιμοποιήσαμε για την μετατροπή των συντεταγμένων.

7.2 Γραφικό περιβάλλον διασύνδεσης χρήστη

Στο τρίτο κεφάλαιο είδαμε πως υλοποιείται ένας Server. Παρακάτω στο 7.1 βλέπουμε τον κώδικα που υλοποιεί τον Server του uavcomm.py. Από την κλάση uavwin καλείται η συνάρτηση LanchServer με τα ορίσματα HOST, PORT και number.

Τα HOST και PORT είναι τα απαραίτητα ορίσματα για την εκκίνηση του TCP.Server. Το όρισμα number είναι ο αριθμός αναφοράς του Server στο uavwin και προωθείται στον handler για την ονομασία του αρχείου λήψης δεδομένων.

Ο handler λαμβάνει τις πληροφορίες από το κανάλι μετάδοσης και τις αποθηκεύει σε ένα αρχείο, από το οποίο στην συνέχεια το uavwin ανασύρει τις τιμές που χρειάζεται.



Σχήμα 7.1: Ο κώδικας του Server

Στο σχήμα 7.2 βλέπουμε ένα τμήμα του κώδικα του Main control. Η συνάρτηση def __init__ είναι η συνάρτηση κατασκευής (constructor) του MainWindow. Βλέπουμε ότι για την κατασκευή του παραθύρου χρησιμοποιήσαμε μια διακριτή διαδικασία στην οποία διαχωρίστικαν τα βασικά τμήματα του γραφικού περιβάλλοντος και στην συνέχεια δημιουργήθικαν ξεχωριστές συναρτήσεις για το κάθε τμήμα. Βλέπουμε λοιπόν ότι για την ολοκλήρωση του σχεδιασμού τρέχουμε τρείς συναρτήσεις:

```
self.create.menu()
```

```
self.create_status_bar()
```

```
self.create_main_panel().
```

Κάτω από τις συναρτήσεις κατασκευής του γραφικού περιβάλλοντος δημιουργήσαμε ένα ρολόι.

```
self.redraw_timer = wx.Timer(self)
```

με το οποίο καλούμε την συνάρτηση επανασχεδίασης του παραθύρου.

```
self.Bind(wx.EVT_TIMER, self.on_redraw_timer, self.redraw_timer)
```

ώστε να γίνεται ανανέωση των στοιχείων που βλέπει ο χρήστης. Ο χρόνος του ρολογιού ορίζεται με την συνάρτηση εκκίνησης στα 100 msec

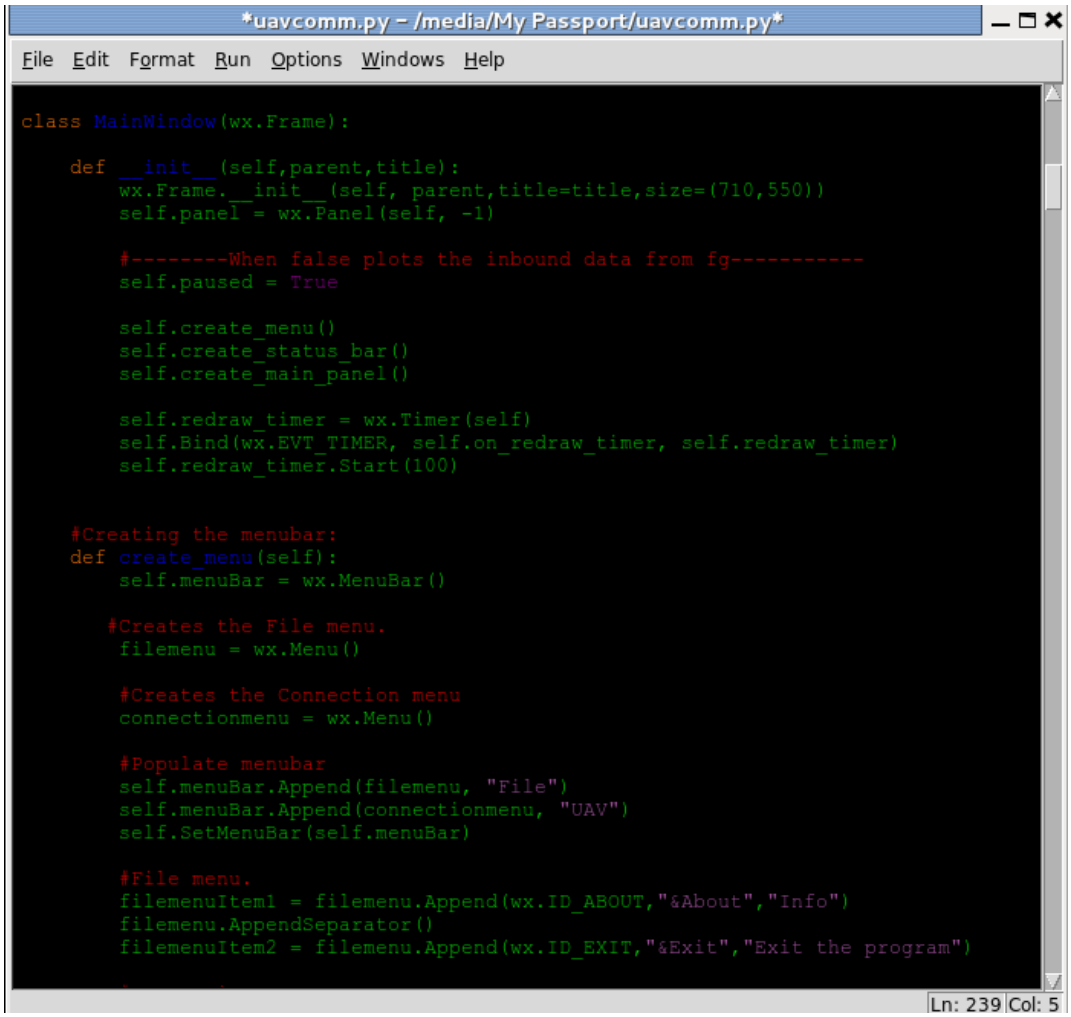
```
self.redraw_timer.Start(100)
```

Κάτω από τις διαδικασίες του ρολογιού ορίζουμε τις συναρτήσεις μας και τις δημιουργουμε. Ξεκινάμε με την συνάρτηση που δημιουργεί την πινακίδα menu:

```
def create_menu()
```

Στην αρχή δημιουργούμε την γραμμή του menu του παραθύρου. Βλέπουμε ότι χρησιμοποιούμε έτοιμα σχέδια από την βιβλιοθήκη `wxpython` βάζοντας το πρόθεμα wx. και την συνάρτηση `Menu()` της βιβλιοθήκης.

Στην συνέχεια φτιάχνουμε τις επιλογές που θα έχει, File και Connection. Στην συνέχεια δημιουργούμε τις επιλογές που θα υπάρχουν σε κάθε υπό-επιλογή. Με την ίδια λογική συνεχίζουμε σε όλες τις επιλογές.



Σχήμα 7.2: Αρχή της κλάσης MainWindow

Στο σχήμα 7.1 βλέπουμε τα τμήματα της στοίχισης και του Event Handling. Για να διευκολυνθούμε στην στοίχιση ενός παραθύρου η βιβλιοθήκη wx μας δίνει τα εργαλεία των Sizers. Οι Sizers αποτελούν πλαίσια στα οποία συνδέουμε τα διάφορα τμήματα των γραφικών. Αυτό είναι εξερετικά

χρήσιμο όταν μεταβάλουμε τις διαστάσεις του παραθύρου, γιατί τα γραφικά ακολουθούν τις αλλαγές που πραγματοποιούμε στο πλαίσιο. Ο χρήστης μπορεί έτσι να προσαρμόζει τη γεωμετρία του παραθύρου στις ανάγκες του. Οι Sizers μπορεί να είναι κάθετοι και οριζόντιοι. Όπως στα μενού και εδώ, πρώτα δημιουργούμε το πλαίσιο που περιέχει τα υπόλοιπα και στην συνέχεια προχωράμε στην κατασκευή των υπολοίπων. Στην αρχή δημιουργήσαμε ένα κάθετο πλαίσιο με την εντολή:

```
self.vbox = wx.BoxSizer(wx.VERTICAL)
```

Στην συνέχεια δημιουργήσαμε ένα οριζόντιο πλαίσιο με την εντολή:

```
self.hbox1 = wx.BoxSizer(wx.HORIZONTAL)
```

και με την εντολή

```
self.hbox1.Add...
```

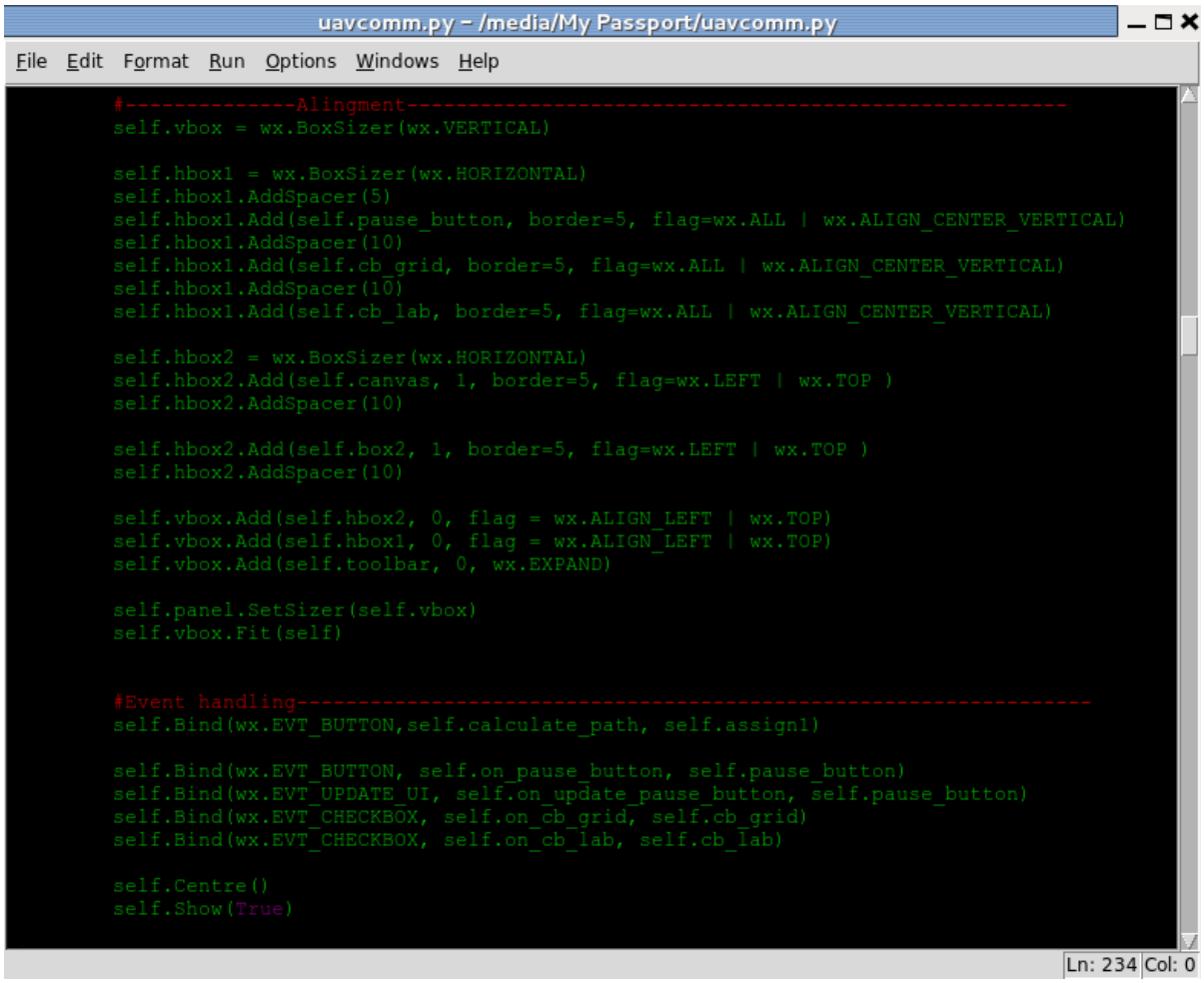
προσθέσαμε στην σειρά τα περιεχόμενα του. Μετά προχωρήσαμε στην κατασκευή του δεύτερου οριζόντιου πλαισίου και των περιεχομένων του. Στο τέλος πρωσθέτουμε τα δύο οριζόντια πλαίσια στο κάθετο.

Στην επόμενη ενότητα είναι η διαχείριση των γεγονότων Event handling. Ο σκοπός μας είναι να ενώσουμε τα σήματα που έρχονται από το πάνελ με τις συναρτήσεις που περιέχει.

Για να γίνει αυτό χρησιμοποιούμε την εντολή Bind. θα δούμε ένα παράδειγμα για να δούμε πως λειτουργεί.

```
self.Bind(wx.EVT_BUTTON, self.on_pause_button, self.pause_button)
```

Μέσα στην παρένθεση έχουμε από τα αριστερά. Το `wx.EVT_BUTTON` με το οποίο δηλώνουμε ότι το γεγονός προέρχεται από κουμπί. Με το `self.on_pause_button` επλιέγουμε τη συνάρτηση που θα κληθεί αν συμβεί το γεγονός. Τέλος δηλώνουμε το στοιχείο από το οποίο θα προκύψει το γεγονός `self.pause_button`.



Σχήμα 7.3: Στοίχιση και Event Handling

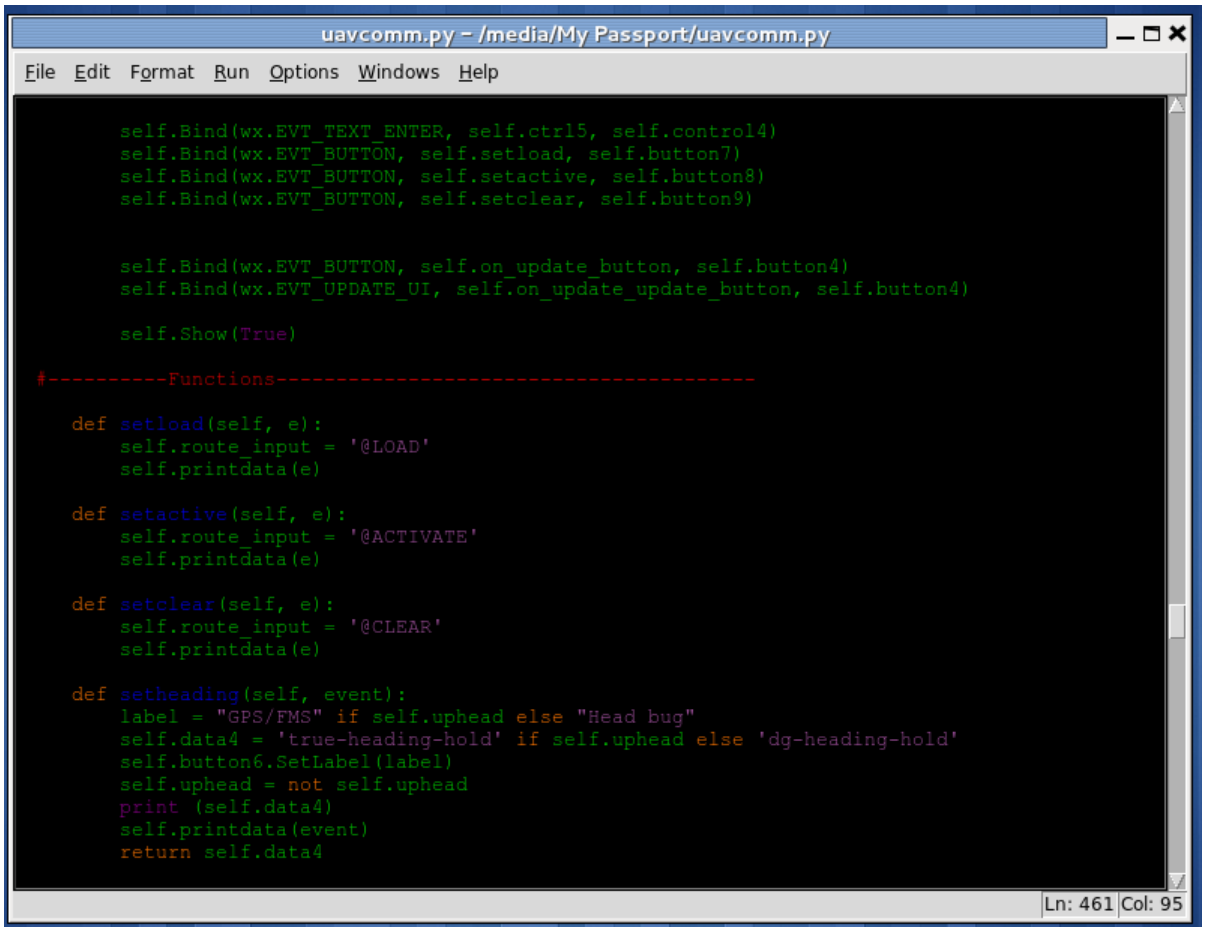
Στην 7.4 έχουμε ένα τμήμα του κώδικα στο πάνω τμήμα του οποίου έχουμε τις εντολές του Event Handling και από κάτω τις συναρτήσεις. Οι συναρτήσεις

`setload(self,e)`,

`setactive(self, e)`,

`setclear(self, e)`,

είναι υπεύθυνες για την δημιουργία των εντολών του Route Manager του Flight Gear. Με την εναλλαγή τους στη μεταβλητή self.route_input μπορούμε να φορτώσουμε, να ενεργοποιήσουμε ή να σταματήσουμε ένα σχέδιο πτήσης. Στην συνάρτηση `setheading(self, e)` βλέπουμε την υλοποίηση ενός push button με αλλαγή της ετικέτας του σε κάθε αλλαγή.



Σχήμα 7.4: Event Handling και συναρτήσεις

7.3 Διαδρομές Dubins

Ο κώδικας περιλαμβάνει την κατασκευή των απαραίτητων συναρτήσεων

- Τη μετατροπή σε γεωδαισιακές σφαιρικές συντεταγμένες, γεωγραφικό μήκος, πλάτος και ύψος (τ, λ, α) σε τοπικές καρτεσιανές συντεταγμένες (x, y, z). Οι σχέσεις αυτές είναι καθιερωμένες και τις δίνουμε εδώ

$$x = [(R+\alpha)\cos(\lambda)\cos(\tau) - R\cos(\lambda)\cos(\tau)]\sin(\tau) - [(R+\alpha)\sin(\tau) - R\sin(\tau)]\cos(\tau)$$

$$y = (R+\alpha)\sin(\lambda)\cos(\tau) - R\sin(\tau)\cos(\lambda)$$

$$z = [(R+\alpha)\cos(\lambda)\cos(\tau) - R\cos(\lambda)\cos(\tau)]\cos(\tau) - [(R+\alpha)\sin(\tau) - R\sin(\tau)]\sin(\tau)$$

και ο αντίστροφος

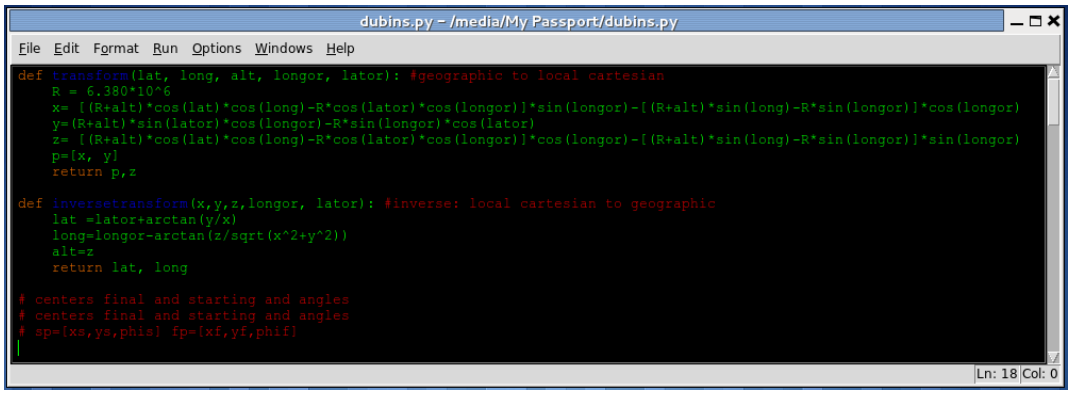
$$\lambda = \tau_0 + \arctan(\frac{y}{x})$$

$$\tau = \tau_0 - \arctan\left(\frac{z}{\sqrt{x^2+y^2}}\right)$$

$$\alpha = z$$

Εδώ λ_0, τ_0 είναι οι συντεταγμένες του σημείου έναρξης.

Οι συναρτήσεις λέγονται `transform(), inversetransform()`.



Σχήμα 7.5: Οι συναρτήσεις `transform` και `inverse transform`

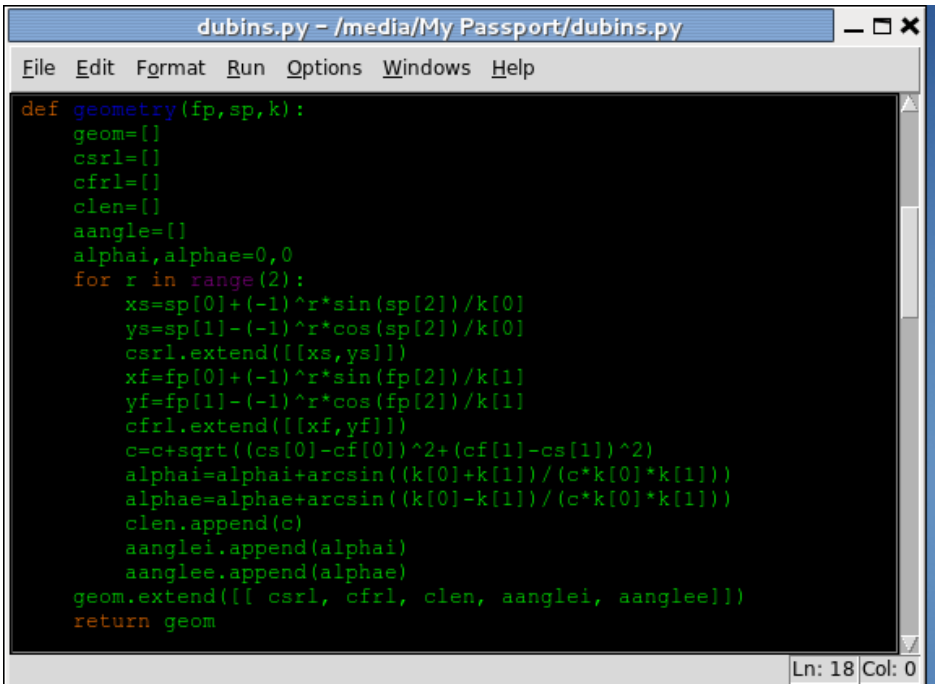
- Στη συνέχεια με βάση τα χαρακτηριστικά των τροχιών που παρέχονται στο [T] κατασκευάζουμε τις ακόλουθες συναρτήσεις που βασικά περιέχουν λίστες πινάκων στην έισοδο και στην έξοδο.

  Η συναρτήση `geometry`, σχήμα `;;`, που υπολογίζει από τα σημεία αφετηρίας, τερματισμού και μέγιστης καμπυλότητας στροφής

  `fp,sp,k`

  τη λίστα με τα στοιχεία των κύκλων και διακέντρου

  `[csrl, cfrl, clen, aanglei, aanglee]`.

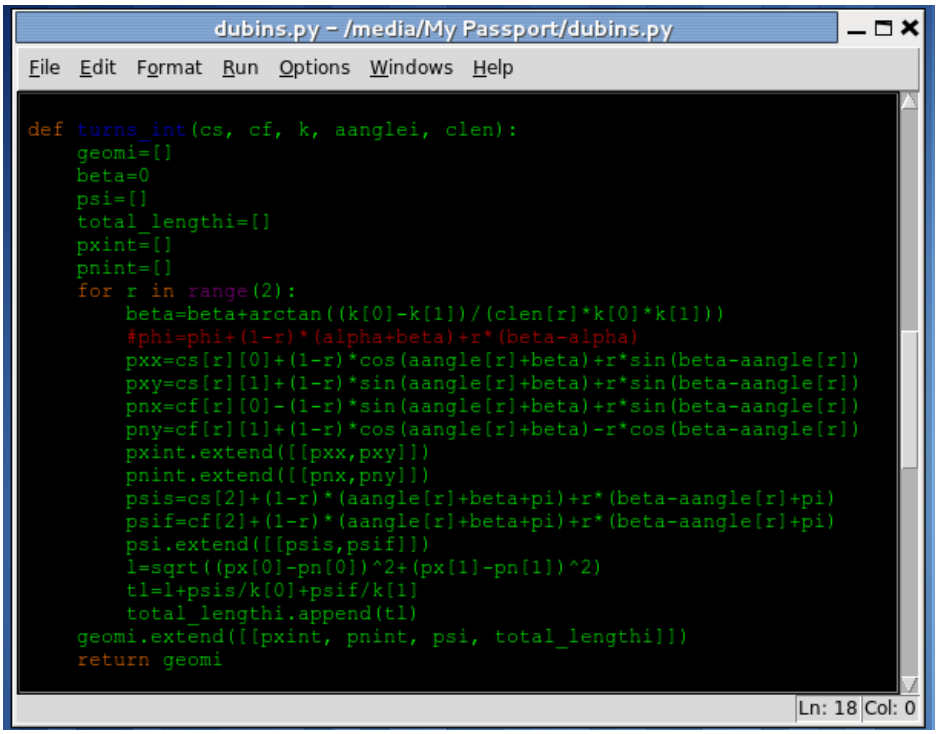


Σχήμα 7.6: Η συνάρτήση `geometry()`

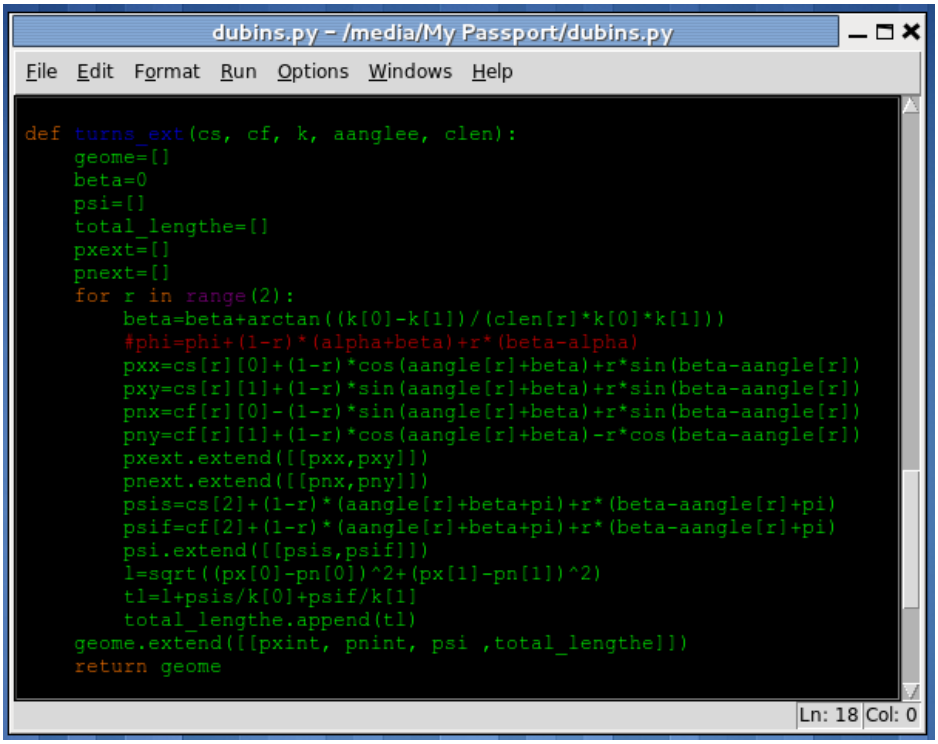
Οι συναρτήσεις στροφής δέχονται τη λίστα που περιλαμβάνει τις μεταβλήτες εξόδου απο την προηγούμενη συνάρτηση και επιστρέφει τη λίστα

`[pxint, pnint, psi, total_lengthi]`

Επεξεργάζεται τισ σχέσεις που δόθηκαν στο κεφάλαιο 2.

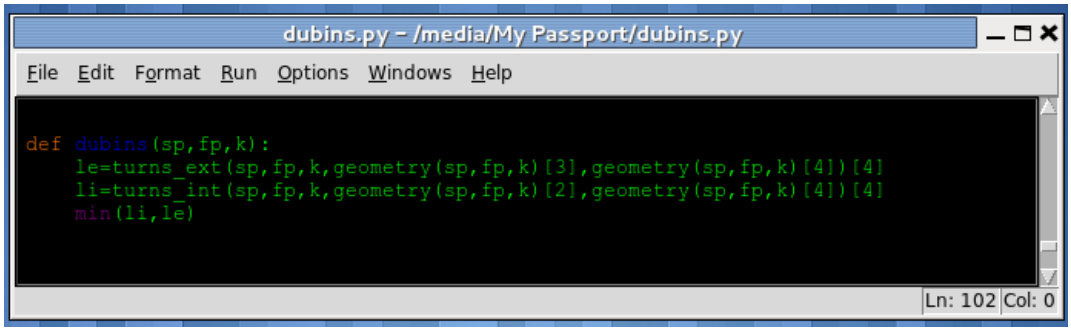


Σχήμα 7.7: Η συνάρτηση `turns_int()`



Σχήμα 7.8: Η συνάρτηση `turns_ext()`

Η συνάρτηση που επιλέγει με βάση τις τέσσερις που έχουμε υπολογίσει τη βέλτιστη διαδρομή.



Σχήμα 7.9: Η συνάρτηση `dubins()`

Μεταβλητή καμπυλότητα Η μεταβλητή καμπυλότητα μπορεί να υπολογισθεί με την επανάληψη (iteration) των συναρτήσεων που κατασκευάσαμε και φυσικά με μία επαναληπτική διαδικασία βελτιστοποίησης.

7.4 Περίληψη

Σε αυτό το τελευταίο κεφάλαιο είδαμε ορισμένα τμήματα του κώδικα που υλοποιούν βασικές λειτουργίες, σχολιάστικαν κάποιες μέθοδοι και παρουσιάστικε ο τρόπος λειτουργίας τους .

Σκοπός της πτυχιακής ήταν η εμβάθυνση σε ορισμένους τομείς της σχεδίασης των μη επανδρωμένων εναέριων συστημάτων. Ο βασικός στόχος της πτυχικής εργασίας για την υλοποίηση και δοκιμή της προσέγγισης του Dubins σε πραγματικές συνθίκες επετεύχθει. Τα αποτελέσματα των δοκιμών ήταν πολύ ικανοποιητικά καθώς έγιναν πολλές πτήσεις και το αεροσκάφος ακολούθησε την πορεία που υπολογίσαμε χωρίς πρόβλημα.

Η προσέγγιση του GUI και η επιλογή των κλάσεων ίσως δεν είναι και η καλύτερη δυνατή. Εντοπίστικαν αδυναμίες στον τρόπο που προσεγγίστικε η διαχείριση των αεροσκαφών, καθώς ο φόρτος εργασίας ήταν μεγάλος για σενάρια που ενέπλεκαν πάνω από τρία αεροσκάφη και η διαχείριση τους ήταν δύσκολη. Επίσης παρατηρήθικαν αυξημένες απαιτήσεις σε πόρους συστήματος, κάτι το οποίο θα ήταν καλό να βελτιωθεί σε κάποια επόμενη έκδοση του λογισμικού.

Μέσα από αυτήν τη διαδικασία προέκυψαν νέες απορείες και προκλήσεις που ελπίζω να διερευνηθούν στο μέλλον. Τέλος θα ήθελα να ευχαριστίσω τον εισηγητή μου, για την πολύτιμη βοήθεια του και την αμέριστη υποστίρηξή του.

Βιβλιογραφία

[B] Berndt Jon S. and the JSBSim development team *JSBSim an Open Source, platform independent, flight dynamics model in C++*, (2011)

[C] Clough B. T., *Metrics, schmetrics! How the heck do you determine a UAV's aytonomy anyway.*, (2002), Proceedings of the Performance Metrics for Intelligent Systems Workshop, Gaithershburg

[D] Dubins L.E., *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents.,* Am. J. math, **158**, (1957), 15-21

[F] Flight Gear development team, *The flight Gear manual*, v.1.9.0, (2009)

[H] Huang Hui-Min/NIST, *Autonomy Levels For Unmanned Systems (ALFUS)*, (2008), ALFUS working group, SAE AS4D Committee.

[M] Mercier Stephane, Tessier Catherine, *Some basic concepts for shared autonomy: a first report*, (2008), Proceedings of the 2008 conference on Collaborative Decision Making: Perspectives and Challenges

[S] Shima Tal, Steven J. Rasmussen, *UAV Cooperative Decision and Control*, (2009), `SIAM`

[T] Tsourdos A., White B., Shanmugavel M., *Cooperative Path Planning of Unmanned Aerial Vehicles* (2010)

[Z] Zelle John M., *Python Programming: An introduction to Computer Science*, v1.0rc2 , (2002)

[1] `http://www.ctie.monash.edu.au/hargrave/rpav_home.htmlBeginnings`

[2] `http://www.wikipedia.org`

[3] `http://eli.thegreenplace.net/2009/05/15/a-year-with-python/`

[4] `http://www.python.org/`