Ατομική Διπλωματική Εργασία

ΕΞΟΜΟΙΩΤΗΣ ΠΕΔΙΟΥ ΒΟΛΗΣ ΣΕ Unity3D

Παναγιώτης Κυριάκου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εξομοιωτής Πεδίου Βολής σε Unity3D

Παναγιώτης Κυριάκου

Επιβλέπων Καθηγητής
Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2014

Ευχαριστίες

Ξεκινώντας, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου, Δρ. Γιώργο Χρυσάνθου ο οποίος μου έδωσε την ευκαιρία να εργαστώ με τη παρούσα εργασία καθώς και τα άτομα του εργαστηρίου Γραφικών και Πολυμέσων για την καθοδήγηση και τις συμβουλές τις οποίες μου έδωσαν κατά τη διάρκεια της μελέτης μου.

Περίληψη

Η διπλωματική μου εργασία ασχολείται με τη ανάπτυξη ενός προσομοιωτή πεδίου βολής χρησιμοποιώντας τη μηχανή παιχνιδιών (game engine) Unity 3D και αποτελεί ένα από τα βασικά κομμάτια ενός μεγαλύτερου συνόλου το οποίο θα απαρτίζει ένα ολοκληρωμένο προσομοιωτή πεδίου βολής ο οποίος θα μπορεί να χρησιμοποιηθεί για εκπαίδευση στρατιωτών της Εθνικής Φρουράς.

Κατά τη διάρκεια ανάπτυξης της παρούσας διπλωματικής εργασίας, επενδύθηκε αρκετός χρόνος στην κατανόηση των κανόνων της βαλλιστικής και των νόμων της φυσικής που καθορίζουν την πορεία μιας βολίδας και ως επέκταση, το αποτέλεσμα μιας βολής.

Επίσης, δόθηκε μεγάλη σημασία στη δημιουργία ενός ρεαλιστικού περιβάλλοντος πεδίου βολής, τόσο σε παρουσίαση αλλά και σε λειτουργία. Παρέχονται διάφορα ρεαλιστικά σενάρια καθώς και διευκολύνσεις για τους τελικούς χρήστες (π.χ. αποθήκευση και ανάληψη αποτελεσμάτων βολών από βάση δεδομένων).

Τέλος, πάρθηκαν διάφορες μετρήσεις των τιμών που υπολογίζονται και χρησιμοποιούνται για τη διεκπεραίωση βολής από την υλοποίηση μας και έγιναν συγκρίσεις με προϊόντα τρίτων.

Να σημειωθεί πως έχει υλοποιηθεί και διεπαφή δικτύου η οποία συνδέει τον εξομοιωτή σε σύστημα laser motion detection (το οποίο αποτελούσε δεύτερη ΑΔΕ) δίνοντας τη δυνατότητα σε μετέπειτα στάδια να προστεθεί τρόπος σκόπευσης με πραγματικά όπλα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Ζούμε σε ένα κόσμο όπου τα πάντα σιγά σιγά αυτοματοποιούνται, εκμοντερνίζονται και η τεχνολογία συνεχώς βελτιώνει τη ποιότητα της ζωής μας. Το κίνητρο το οποίο μας ώθησε στην ανάπτυξη του συγκεκριμένου συστήματος προήλθε από το γεγονός ότι η εκτέλεση βολών στο νησί μας συνεχίζει να γίνεται με τον ίδιο, παραδοσιακό, χρονοβόρο, κουραστικό, επικίνδυνο και αρκετά σπάταλο τρόπο που γινόταν πάντοτε: με την εκτέλεση βολών με πραγματικά πυρά σε πεδία βολής.

Η ιδέα ήταν πως χρησιμοποιώντας τεχνολογίες τρισδιάστατων γραφικών όπως αυτές που χρησιμοποιούνται σε ηλεκτρονικά παιχνίδια καθώς και άλλες τεχνολογίες της πληροφορικής (π.χ. computer vision), θα μπορούσαμε να επιλύσουμε αυτό το πρόβλημα, βελτιώνοντας την ποιότητα της ζωής μας και δίνοντας την ευκαιρία να ανοιχθεί μια νέα πόρτα ανάπτυξης.

1.2 Στόχος

Στόχος της διπλωματικής εργασίας μου είναι η μελέτη των κανόνων της βαλλιστικής και των νόμων της φυσικής που καθορίζουν τη πορεία μιας βολίδας ενώσω αυτή βρίσκεται εν πτήση και η μετέπειτα χρήση τους για τη δημιουργία ενός συστήματος όπου να μπορούν οι βολές να εκτελούνται εικονικά, με ρεαλιστική ακρίβεια.

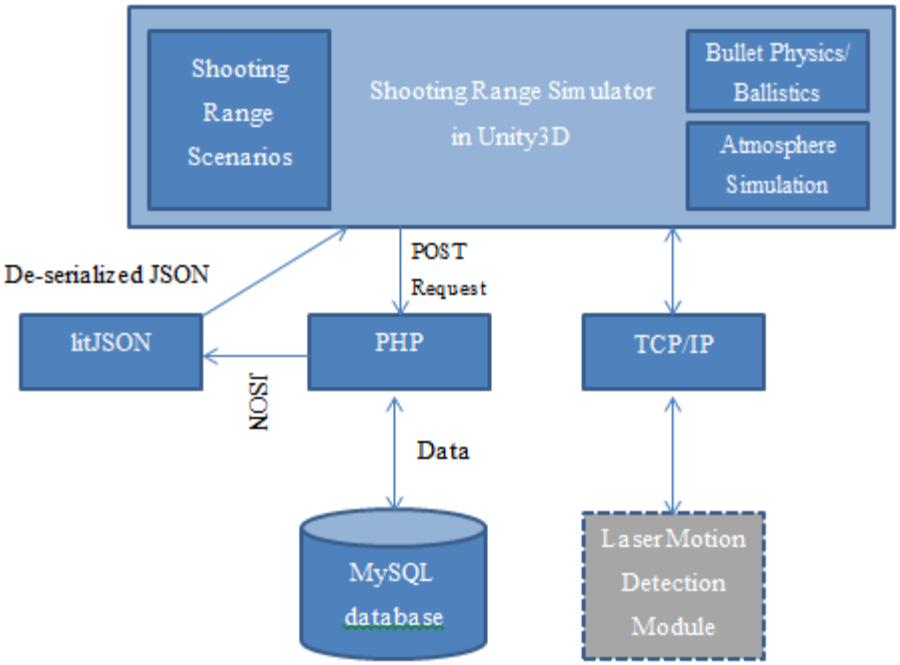
Επίσης, το σύστημα έχει ως στόχο του να δημιουργήσει μια γερή βάση και ένα καλό εναρκτήριο σημείο για τη μετέπειτα δουλειά που πιθανόν να ακολουθήσει. Το σύστημα προσπαθεί να παρέχει λειτουργικότητα η οποία να είναι φιλική προς το χρήστη και να μπορεί να βοηθήσει άτομα έτσι ώστε να μάθουν να χρησιμοποιούν σωστά το όπλο τους.

1.3 Τεχνολογίες που χρησιμοποιήθηκαν

1.3.1 Unity3D

Για τη δημιουργία του εξομοιωτή χρησιμοποιήθηκε η δωρεάν έκδοση της μηχανής κατασκευής παιχνιδιών Unity3D. Η συγκεκριμένη μηχανή είναι σχετικά νεαρή τεχνολογία η οποία έχει ανθήσει ιδιαίτερα τα τελευταία χρόνια, προσφέροντας μεγάλη ευκολία χρήσης αλλά και τη δυνατότητα εξαγωγής παιχνιδιών σε διάφορες πλατφόρμες.
Το Unity3D είναι πολύ εύκολο στη χρήση, με μια ήδη αρκετά μεγάλη κοινότητα έτοιμη να σε βοηθήσει, δίνοντας τη δυνατότητα σε όποιο θελήσει, να σχηματίσει με σχετική ευκολία ένα δικό του παιχνίδι και να το μοιραστεί με τον υπόλοιπο κόσμο.
Τελειώνοντας, το Unity3D δεν έχει τίποτα να ζηλέψει από άλλες μηχανές παιχνιδιών, καθώς προσφέρει όλα τα απαραίτητα χαρακτηριστικά όπως physics engine βασισμένο στο Nvidia PhysX, animation editor (mecanim), soft shadows, particle effects, και πολλά άλλα μικρά και μεγάλα εργαλεία.

Η γλώσσα προγραμματισμού που χρησιμοποίησα με το Unity3D ήταν η C# η οποία παρέχεται στο Unity3D μέσω του Open-Source .NET Framework-compatible project το οποίο ονομάζεται Mono. Εκτός από τη C#, το Unity παρέχει τη δυνατότητα προγραμματισμού σε μια ειδική μορφή JavaScript (UnityScript) καθώς και μια γλώσσα βασισμένη στην Python, η οποία ονομάζεται Boo.



Εικόνα 1.1 Το scope της ΑΔΕ

Κεφάλαιο 2

Βαλλιστική

2.1 Γενικά

Βαλλιστική είναι η επιστήμη η οποία μελετά τα χαρακτηριστικά που διέπουν την κίνηση μιας βολίδας που έχει ριχθεί. Χωρίζεται σε δύο είδη:

Εσωτερική Βαλλιστική: Τι συμβαίνει στην βολίδα από τη στιγμή που το όπλο εκπυρσοκροτεί μέχρι να εξέλθει από την κάννη.

Εξωτερική Βαλλιστική: Τι συμβαίνει στην βολίδα εφόσον έχει εξέλθει από την κάννη του όπλου.

Λόγω της φύσης του προβλήματος που καλούμαστε να μελετήσουμε, το οποίο ασχολείται με την πορεία μιας βολίδας ενόσω αυτή βρίσκεται εν πτήση, θα δώσουμε μεγαλύτερο βάρος στην εξωτερική βαλλιστική. Θεωρούμε πως η βολή γίνεται βάση των χαρακτηριστικών που παρέχονται για το όπλο το οποίο εκτελεί τη βολή.

Για μια ρεαλιστική προσέγγιση, χρησιμοποιούμε τα θεωρητικά χαρακτηριστικά που παρέχονται για το τυφέκιο G3 το οποίο χρησιμοποιείται εκτενώς από την Εθνική Φρουρά:

Εικόνα 2.1 Το στρατιωτικό τυφέκιο G3A3

Χαρακτηριστικό	**Τιμή**
Διαμέτρημα	7.62mm
Θεωρητική Ταχυβολία	500-600 βολίδες το λεπτό
Αρχική Ταχύτητα Βολίδας	780-800 m/s
Μήκος Κάννης	45cm
Αριθμός Αυλακώσεων Κάννης	4
Περιστροφή Κάννης	1 περιστροφή ανα 305mm
Μέγιστο Βεληνεκές	3700m
Δραστικό Βεληνεκές	400m
Απόσταση Ασφαλείας	4000m

Πίνακας 2.1 Χαρακτηριστικά του τυφεκίου G3A3

2.2 Δυνάμεις

Κατά τη πτήση μιας βολίδας, η βολίδα αλληλεπιδρά με τα σωματίδια που βρίσκονται στον αέρα, τα οποία ασκούν με τη σειρά τους διάφορες δυνάμεις σ'αυτή. Λόγω της ρευστής μορφής του αέρα, η κίνηση διέπεται από τους νόμους της ρευστοδυναμικής.

Πιο κάτω θα ακολουθήσει μια παράθεση των δυνάμεων [1] που ασκούνται σε μια βολίδα κατά την κίνηση της:

2.2.1 Αντίσταση του Αέρα:

Η δύναμη της αντίστασης του αέρα ασκείται στην αντίθετη κατεύθυνση από την κατεύθυνση της ταχύτητας της βολίδας και προκαλεί την σταδιακή μείωση της. Αποτελεί (μετά από την βαρύτητα) την πιο σημαντική δύναμη από το σύνολο των δυνάμεων που θα μελετήσουμε.

Η δύναμη αντίστασης του αέρα καθορίζεται από την πιο κάτω εξίσωση:

$$\vec{F}_{drag} = -\frac{1}{2}\rho S C_{drag} \vec{V} V$$

Εξίσωση 2.1 Δύναμη αντίστασης του αέρα [1]

Σύμβολο	Νόημα
ρ	Πυκνότητα του αέρα
S	Επιφάνεια αναφοράς
C_{drag}	Συντελεστής αντίστασης αέρα της βολίδας
$\vec{V}$	Διάνυσμα ταχύτητας της βολίδας.
V	Μέγεθος της ταχύτητας της βολίδας.

Πίνακας 2.2 Εξήγηση συμβόλων εξίσωσης αντίστασης του αέρα

Η επιφάνεια αναφοράς μπορεί να χαρακτηριστεί ως το εμβαδό της διατομής της βολίδας στο σημείο όπου τελειώνει η καμπύλη της μύτης της. Λόγω του κυλινδρικού σχήματος των βολίδων, το σχήμα το οποίο προκύπτει είναι κύκλος με διάμετρο ίση με το διαμέτρημα της βολίδας. Ξέρουμε πως το εμβαδό ενός κύκλου υπολογίζεται από την εξίσωση:

$$A_{circle} = \pi r^2$$

Εξίσωση 2.2 Εξίσωση υπολογισμού εμβαδού κύκλου.

Εφόσον ξέρουμε πως: $r = \frac{d}{2}$, (d διαμέτρημα της σφαίρας) τότε μπορούμε να υπολογίσουμε πως η επιφάνεια αναφοράς έχει εμβαδό ίσο με:

$$S = \frac{\pi d^2}{4}$$

Εξίσωση 2.3 Εξίσωση υπολογισμού επιφάνειας αναφοράς για κυλινδρικές βολίδες.

Ο συντελεστής αντίστασης αέρα της βολίδας αποτελείται από δυο συστατικά κομμάτια:

$$C_D = C_{D_0} + C_{D_{\delta^2}} \delta^2$$

Εξίσωση 2.4 Εξίσωση υπολογισμού συντελεστή αντίστασης του αέρα.

Όπου C_{D_0}ο συντελεστής αντίστασης του αέρα για μηδενική γωνιά και $C_{D_{\delta^2}}$ ο συντελεστής του αέρα για γωνιά. Επίσης:

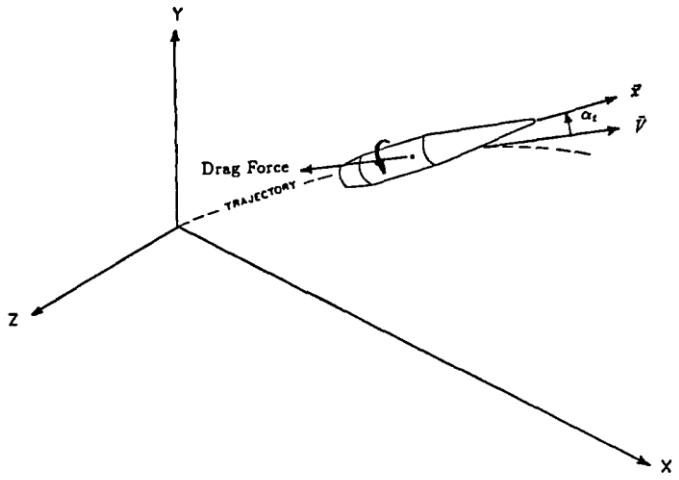
$$\delta = \sin(a)$$

Εξίσωση 2.5 Εξίσωση υπολογισμού δ.

όπου a η γωνιά που σχηματίζεται μεταξύ του διανύσματος της ταχύτητας και του άξονα συμμετρίας της βολίδας.

Να σημειωθεί πως οι δυο συντελεστές αλλάζουν ανάλογα με την ταχύτητα με την οποία κινείται η βολίδα. Συγκεκριμένα, αλλάζουν ανάλογα με το λόγο της ταχύτητας της βολίδας σε σχέση με τη ταχύτητα του αέρα.

Η τιμή αυτή ονομάζεται Mach number ($M = \frac{V}{Speed\ of\ sound}$).



Εικόνα 2.2 Η δύναμη αντίστασης του αέρα [1]

2.2.2 Δύναμη Ανύψωσης

Η δύναμη ανύψωσης [1] ασκείται όταν η μύτη της βολίδας σχηματίζει γωνιά σε σχέση με το διάνυσμα της ταχύτητας και προκαλεί την ανύψωση της βολίδας στην κατεύθυνση που δείχνει η μύτη της. Η δύναμη αυτή ασκείται κάθετα στο διάνυσμα της ταχύτητας.

Σε περίπτωση που η μύτη βλέπει προς τα κάτω σε σχέση με το διάνυσμα της ταχύτητας, τότε η δύναμη ανύψωσης θα την ωθεί προς τα κάτω, ενώ θα συμβεί το αντίθετο σε περίπτωση που η μύτη βλέπει προς τα πάνω. Η γωνιά μεταξύ του άξονα συμμετρίας της βολίδας και του διανύσματος της ταχύτητας είναι πολύ σημαντική.

Η εξίσωση που χαρακτηρίζει την δύναμη ανύψωσης είναι η ακόλουθη:

$$\vec{F}_{lift} = \frac{1}{2}\rho S C_{L_a}[\vec{V} \times (\vec{x} \times \vec{V})]$$

Εξίσωση 2.6 Εξίσωση υπολογισμού δύναμης ανύψωσης. [1]

Σύμβολο	**Νόημα**
ρ	Πυκνότητα του αέρα.
S	Επιφάνεια αναφοράς.
C_{L_a}	Συντελεστής ανύψωσης της βολίδας.
$\vec{V}$	Διάνυσμα ταχύτητας της βολίδας.
$\vec{x}$	Διάνυσμα παράλληλο με τον άξονα συμμετρίας της βολίδας.

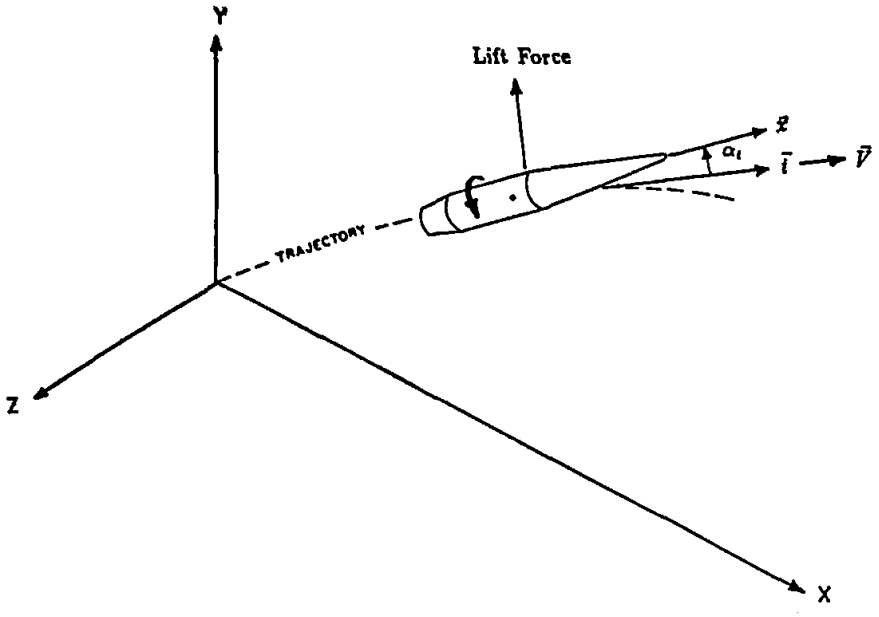
Πίνακας 2.3 Εξήγηση παραμέτρων εξίσωσης δύναμης ανύψωσης.

Ο συντελεστής ανύψωσης εξαρτάται από δύο παράγοντες:

$$C_{L_a} = C_{L_{\alpha_0}} + C_{L_{\alpha_2}}\delta^2$$

Εξίσωση 2.7 Εξίσωση υπολογισμού συντελεστή ανύψωσης. [1]

Όπου το $C_{L_{\alpha_0}}$ είναι ανάλογο της τιμής δ (βλ. εξίσωση 2.5) και το $C_{L_{\alpha_2}}$ είναι ανάλογο του δ^3.



Εικόνα 2.3Δύναμη Ανύψωσης [1]

2.2.3 Δύναμη Magnus

Η δύναμη Magnus παρουσιάζεται όταν μια περιστρεφόμενη βολίδα, λόγω της περιστροφικής της κίνησης, προκαλεί άνισες πιέσεις στις δυο πλευρές του περιστρεφόμενου άξονα. Η δύναμη αυτή είναι κάθετη στο επίπεδο (cross product) του διανύσματος της ταχύτητας και του διανύσματος του άξονα συμμετρίας του σχήματος.

Χαρακτηρίζεται από την ακόλουθη εξίσωση:

$$\vec{F}_{magnus} = \frac{1}{2}\rho V^2 S\left(\frac{pd}{V}\right) C_{N_{Pa}}(\vec{\imath} \times \vec{x})$$

Εξίσωση 2.8 Εξίσωση υπολογισμού δύναμης Magnus. [1]

Σύμβολο	Νόημα
$\boldsymbol{\rho}$	Πυκνότητα του αέρα.
$\boldsymbol{V}$	Μέγεθος της ταχύτητας της βολίδας.
$\boldsymbol{S}$	Επιφάνεια αναφοράς.
$\boldsymbol{p}$	Γωνιακή ταχύτητα περιστροφής της βολίδας. (radians)
$\boldsymbol{d}$	Διαμέτρημα της βολίδας
$\boldsymbol{C_{N_{Pa}}}$	Συντελεστής δύναμης Magnus
$\vec{\boldsymbol{x}}$	Διάνυσμα παράλληλο με τον άξονα συμμετρίας της βολίδας.
$\vec{\boldsymbol{\imath}}$	Κανονικοποιημένο διάνυσμα της ταχύτητας της βολίδας.

Πίνακας 2.4 Εξήγηση παραμέτρων δύναμης Magnus

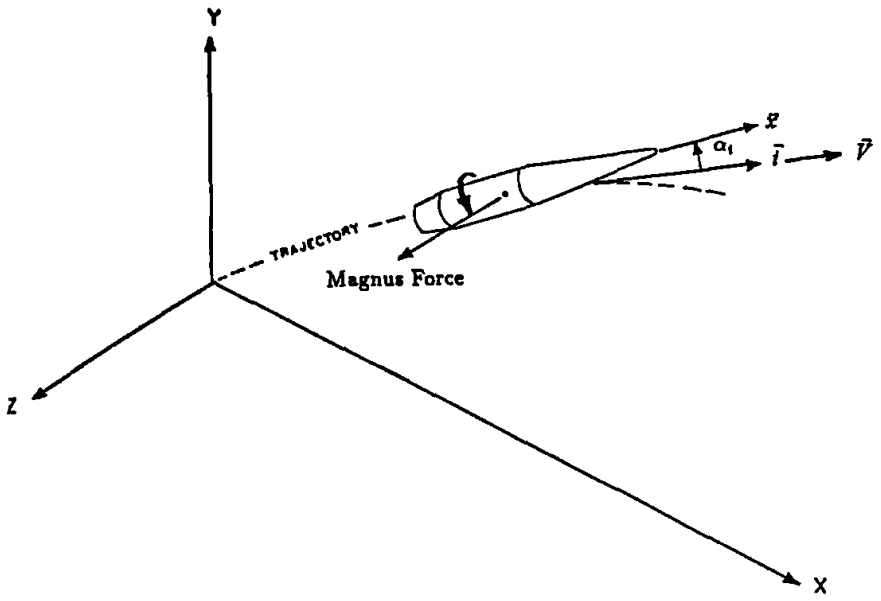
Ο συντελεστής Magnus εξαρτάται από δύο παράγοντες:

$$\boldsymbol{C_{N_{Pa}} = C_{N_{P\alpha_0}} + C_{N_{P\alpha_2}}\delta^2}$$

Εξίσωση 2.9 Εξίσωση υπολογισμού συντελεστή Magnus. [1]

Όπου το $\boldsymbol{C_{N_{P\alpha_0}}}$ είναι ανάλογο της τιμής δ (βλ. εξίσωση 2.5) και το $\boldsymbol{C_{N_{P\alpha_2}}}$είναι ανάλογο του δ^3.

Λόγω του μικρού μεγέθους της συγκεκριμένης δύναμης, θα μπορούσε να αγνοηθεί.



Εικόνα 2.4 Δύναμη Magnus

2.2.4 Δύναμη Απόσβεσης Κλίσης

Η συγκεκριμένη δύναμη είναι αρκετά μικρή και θα μπορούσε να αγνοηθεί.

$$\vec{F}_{p.d.} = \frac{1}{2}\rho S d C_{N_q} V \left(\frac{d\vec{x}}{dt}\right) + \frac{1}{2}\rho S d C_{N_a} V \left[\left(\frac{d\vec{x}}{dt}\right) - \left(\frac{d\vec{i}}{dt}\right)\right]$$

Εξίσωση 2.9 Εξίσωση υπολογισμού δύναμης απόσβεσης κλίσης. [1]

Σύμβολο	Νόημα
$\boldsymbol{\rho}$	Πυκνότητα του αέρα.
$\boldsymbol{S}$	Επιφάνεια αναφοράς.
$\boldsymbol{d}$	Διαμέτρημα της βολίδας
$\boldsymbol{C_{M_q}}$	Συντελεστής δύναμης απόσβεσης κλίσης για τη γωνιακή ταχύτητα κλίσης.
$\boldsymbol{C_{M_a}}$	Συντελεστής δύναμης απόσβεσης κλίσης για τη γωνιακή ταχύτητα εκτροπής (yaw).
$\boldsymbol{V}$	Μέγεθος της ταχύτητας της βολίδας.

Πίνακας 2.5 Εξήγηση συμβόλων της δύναμης απόσβεσης της κλίσης

2.2.5 Δύναμη της Βαρύτητας

Η βαρύτητα είναι μια από τις πιο σημαντικές δυνάμεις που ασκούνται σε μια βολίδα. Υπολογίζεται πολύ απλά ως:

$$\vec{\boldsymbol{F}}_{\boldsymbol{gravity}} = \boldsymbol{m}\vec{\boldsymbol{g}}$$

Εξίσωση 2.10 Εξίσωση υπολογισμού δύναμης βαρύτητας.

2.3 Ροπές

2.3.1 Ροπή απόσβεσης περιστροφής

Η συγκεκριμένη ροπή έχει την τάση να μειώνει την γωνιακή ταχύτητα της βολίδας στον άξονα περιστροφής της. Ο συντελεστής της είναι πάντοτε αρνητικός.

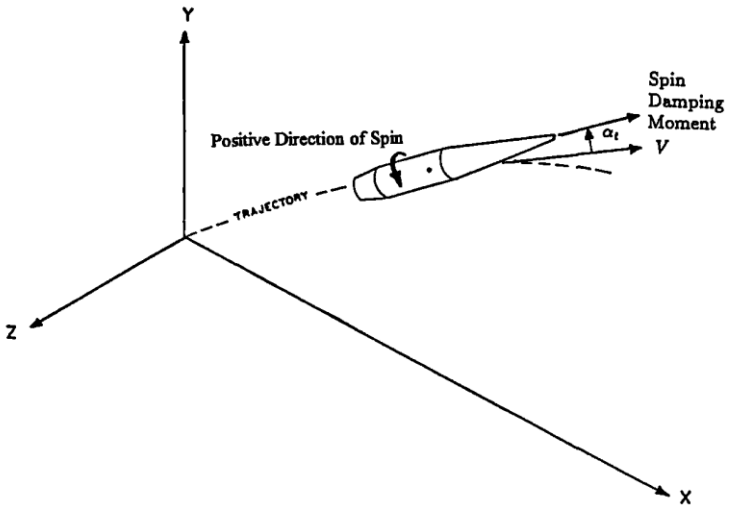
Η εξίσωση της είναι η ακόλουθη:

$$\boldsymbol{T_{S.D} = \frac{1}{2}\rho V^2 S d \left(\frac{pd}{V}\right) C_{l_p \vec{x}}}$$

Εξίσωση 2.11 Εξίσωση υπολογισμού ροπής απόσβεσης περιστροφής. [1]

Σύμβολο	**Νόημα**
$\boldsymbol{\rho}$	Πυκνότητα του αέρα.
$\boldsymbol{S}$	Επιφάνεια αναφοράς.
$\boldsymbol{d}$	Διαμέτρημα της βολίδας
$\boldsymbol{p}$	Γωνιακή ταχύτητα περιστροφής της βολίδας. (radians)
$\boldsymbol{C_{l_p}}$	Συντελεστής ροπής απόσβεσης περιστροφής. (πάντα αρνητικό)
$\boldsymbol{\vec{x}}$	Διάνυσμα παράλληλο με τον άξονα συμμετρίας της βολίδας.
$\boldsymbol{V}$	Μέγεθος της ταχύτητας της βολίδας.

Πίνακας 2.6 Εξήγηση μεταβλητών εξίσωσης ροπής απόσβεσης περιστροφής.



Εικόνα 2.5 Ροπή απόσβεσης περιστροφής. Σε αυτό το παράδειγμα, η γωνιακή ταχύτητα περιστροφής αυξάνεται.

2.3.2 Ροπή Ανατροπής

Η συγκεκριμένη ροπή τείνει να μεταβάλλει την κλίση της βολίδας ανάλογα με τη γωνιά που σχηματίζει ο κεντρικός άξονας συμμετρίας με το διάνυσμα της ταχύτητας. Αυτό σημαίνει πως αν η μύτη της βολίδας έχει κάποια γωνιά σε σχέση με την τροχιά της, τότε η ροπή ανατροπής θα αυξήσει τη γωνιά αυτή. Η συγκεκριμένη ροπή μπορεί να συσχετιστεί με τη δύναμη άνωσης.

Η εξίσωση της έχει ως ακολούθως: [1]

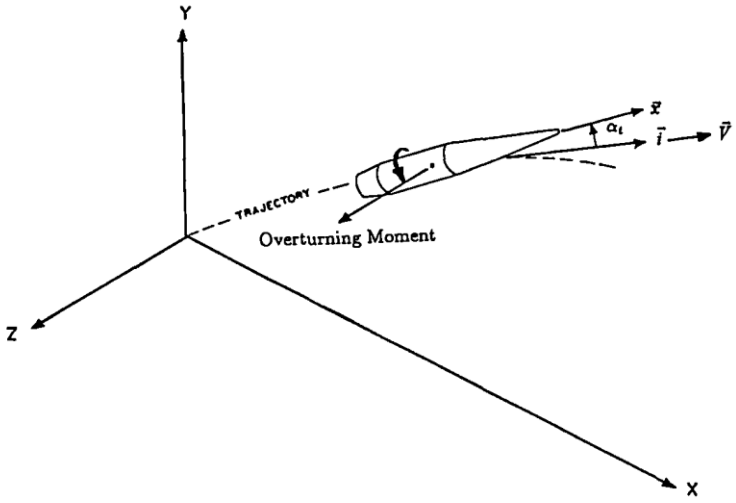
$$\boldsymbol{T_{OT} = \frac{1}{2}\rho S d C_{M_\alpha} V^2 (\vec{\imath} \times \vec{x})}$$

Εξίσωση 2.12 Εξίσωση υπολογισμού ροπής ανατροπής. [1]

$$\boldsymbol{C_{M_\alpha} = C_{M_{\alpha_0}} + C_{M_{\alpha_2}} \delta^2}$$

Εξίσωση 2.13 Εξίσωση υπολογισμού συντελεστή ροπής ανατροπής. [1]

Όπου το $\boldsymbol{C_{M_{\alpha_0}}}$ είναι ανάλογο της τιμής δ (βλ. εξίσωση 2.5) και το $\boldsymbol{C_{M_{\alpha_2}}}$ είναι ανάλογο του δ^3.



Εικόνα 2.6 Η ροπή ανατροπής. [1]

2.3.3 Ροπή Magnus

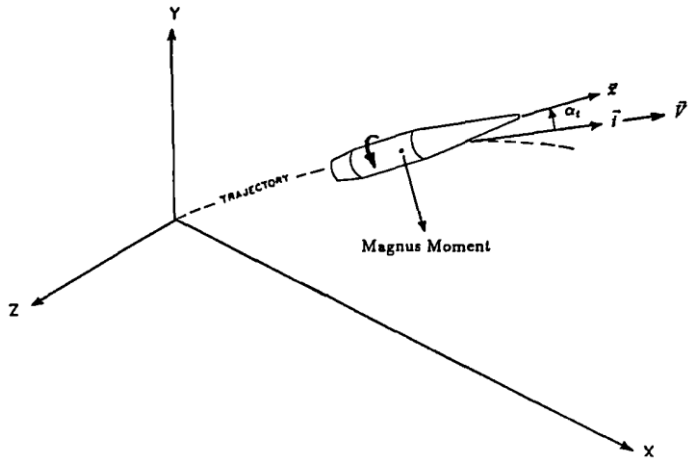
$$\boldsymbol{T_M = \frac{1}{2}\rho S d V^2 S \left(\frac{pd}{V}\right) C_{M_{P\alpha}}[\vec{x} \times (\vec{\imath} \times \vec{x})]}$$

Εξίσωση 2.14 Εξίσωση υπολογισμού ροπής Magnus. [1]

$$\boldsymbol{C_{M_{P\alpha}} = C_{M_{P\alpha_0}} + C_{M_{P\alpha_2}} \delta^2}$$

Εξίσωση 2.15 Εξίσωση υπολογισμού συντελεστή ροπής Magnus. [1]

Όπου το $\boldsymbol{C_{M_{P\alpha_0}}}$ είναι ανάλογο της τιμής δ (βλ. εξίσωση 2.5) και το $\boldsymbol{C_{M_{P\alpha_2}}}$ είναι ανάλογο του δ^3.



Εικόνα 2.7 Η ροπή Magnus. [1]

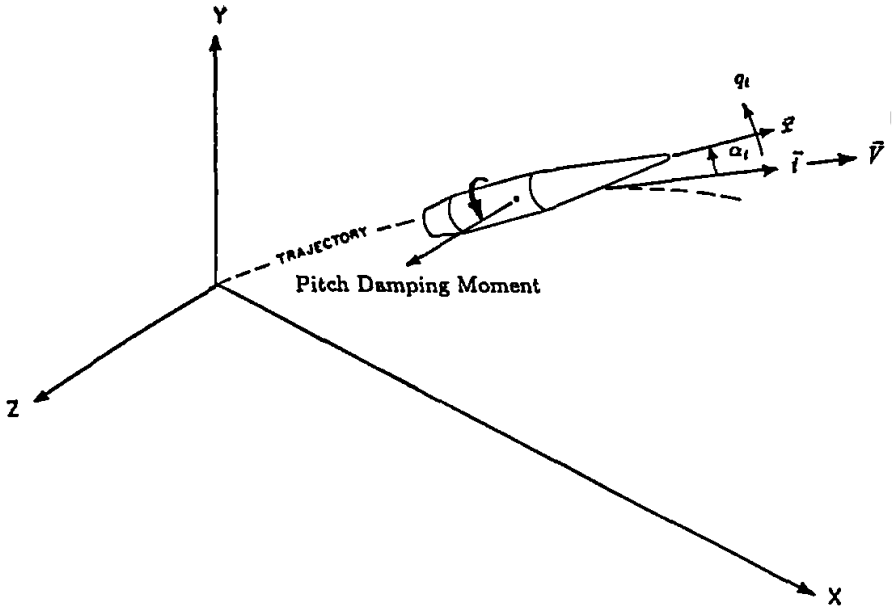
2.3.4 Ροπή απόσβεσης κλίσης

Αντιθέτως με τη δύναμη απόσβεσης κλίσης, η ροπή απόσβεσης κλίσης είναι σημαντική και πρέπει να υπολογίζεται καθώς παίζει μεγάλο αρνητικό ρόλο στη σταθερότητα, αποσταθεροποιώντας τη βολίδα.

Η εξίσωση της έχει ως ακολούθως: [1]

$$\boldsymbol{T_{p.d.} = \frac{1}{2}\rho S d^2 C_{M_q} V \left(\vec{x} \times \frac{d\vec{x}}{dt}\right) + \frac{1}{2}\rho S d^2 C_{M_a} V \left[\left(\vec{x} \times \frac{d\vec{x}}{dt}\right) - \left(\vec{x} \times \frac{d\vec{i}}{dt}\right)\right]}$$

Εξίσωση 2.15 Εξίσωση υπολογισμού ροπής απόσβεσης κλίσης. [1]



Εικόνα 2.8 Ροπή απόσβεσης κλίσης. Οι συντελεστές είναι ανάλογοι των γωνιών a και q, και το άθροισμα τους είναι αρνητικό. [1]

2.4 Πρόγραμμα MC Drag [2]

Το πρόγραμμα MC Drag φτιάχτηκε από τον Robert L McCoy το 1981 με στόχο να μπορεί να υπολογίζει το συντελεστή αντίστασης του αέρα για μηδενική γωνιά για οποιαδήποτε σφαίρα την οποία μπορούμε με σαφήνεια να χαρακτηρίσουμε.

Ο τρόπος με τον οποίο δουλεύει το συγκεκριμένο πρόγραμμα είναι πως σπάζει τη σφαίρα στα συστατικά της κομμάτια και υπολογίζει για το καθένα απ'αυτά το συντελεστή αντίστασης του. Προσθέτοντας όλους τους συντελεστές που υπολογίζονται, βρίσκουμε τον συντελεστή αντίστασης του αέρα για μηδενική γωνιά.

Οι συντελεστές που υπολογίζονται είναι οι ακόλουθοι:

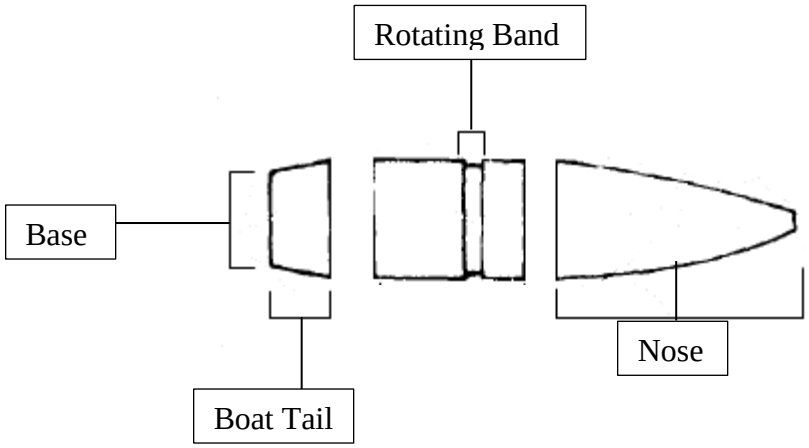
$$C_{D_0} = C_{D_H} + C_{D_{BT}} + C_{D_B} + C_{D_{RB}} + C_{D_{SF}}$$

Εξίσωση 2.16 Εξίσωση υπολογισμού συντελεστή αντίστασης του αέρα για μηδενική γωνιά. [2]

Σύμβολο	**Σημασία**
C_{D_0}	Συντελεστής αντίστασης του αέρα για μηδενική γωνιά.
C_{D_H}	Συντελεστής αντίστασης του αέρα για το κεφάλι/μύτη της βολίδας.
$C_{D_{BT}}$	Συντελεστής αντίστασης του αέρα για την ουρά της βολίδας.
C_{D_B}	Συντελεστής αντίστασης του αέρα για την βάση της βολίδας.
$C_{D_{RB}}$	Συντελεστής αντίστασης του αέρα για τον δακτύλιο που την περιβάλλει.
$C_{D_{SF}}$	Συντελεστής αντίστασης του αέρα για το wetted surface της βολίδας (εκτός από τη βάση).

Πίνακας 2.7 Επεξήγηση επιμέρους συντελεστών αντίστασης του αέρα τους οποίους υπολογίζει το πρόγραμμα MC Drag

Εικόνα 2.9 Βολίδα NATO M80 7.62mm



Εικόνα 2.10 Τα επιμέρους κομμάτια που απαρτίζουν τη βολίδα M80.

2.5 Υπολογισμός πορείας

Τα χαρακτηριστικά της πορείας μιας σφαίρας μπορούν να υπολογισθούν με 3 διαφορετικούς τρόπους, χρησιμοποιώντας τις ακόλουθες μεθόδους:

2.5.1 Flat-Fire Point-Mass Trajectory

Υποθέσεις:

1) Σταθερός συντελεστής αντίστασης του αέρα.
2) Η πορεία της βολίδας μπορεί να χαρακτηριστεί ως ευθεία καθώς η κλίση είναι αρκετά μικρή και θεωρείται αμελητέα.
3) Οι μόνες δυνάμεις που ασκούνται πάνω στη βολίδα είναι η βαρύτητα και η αντίσταση του αέρα.

Η μέθοδος αυτή παρέχει διάφορες εξισώσεις υπολογισμού παραμέτρων (π.χ. ταχύτητα, ώρα πτήσης, ύψος, γωνιά πρόσκρουσης) μέσω των οποίων μπορούμε να κάνουμε approximation τη κατάσταση της βολίδας μας βάση της απόσταση που έχει διανύσει στο πεδίο βολής.

2.5.2 Η μέθοδος Siacci

Υποθέσεις:

1) Σταθερή πυκνότητα του αέρα, λόγω χαμηλών διαφορών μεταξύ των υψών στα οποία ταξιδεύει η βολίδα.
2) Σταθερή θερμοκρασία καθ' όλη τη πορεία της βολίδας.
3) Η ταχύτητα V μπορεί να προσεγγιστεί ως $V = V_x \sec \varphi_0$, όπου φ_0 η γωνιά αναχώρησης

Η μέθοδος αυτή, αν και αρκετά παλιά, χρησιμοποιείται πολύ στον πολιτικό τομέα. Χρησιμοποιεί προϋπολογισμένους πίνακες για να υπολογίσει τα χαρακτηριστικά της πορείας της βολίδας.

2.5.3 6-DOF

Η μέθοδος 6 DOF λαμβάνει υπόψη της όλες τις δυνάμεις και ροπές που ασκούνται στη βολίδα και τις εκφράζει σε 6 άξονες, τρείς για την ταχύτητα και τρείς για υπολογισμό της γωνιακής ταχύτητας. Λόγω της φύσεως τους, οι διαφορικές εξισώσεις χρειάζονται εξειδικευμένους αλγορίθμους για να λυθούν, όπως για παράδειγμα ο αλγόριθμος Runge-Kutta ο οποίος είναι αρκετά ακριβός από θέμα επίδοσης.

Κεφάλαιο 3

Ατμόσφαιρα

3.1 International Standard Atmosphere

Μετά από δεκαετίες συλλογής μετρήσεων, η παγκόσμια κοινότητα δημιούργησε διάφορα μοντέλα που μπορούν να προσομοιώνουν τα χαρακτηριστικά της ατμόσφαιρας. Μερικά από αυτά τα μοντέλα είναι τα ακόλουθα:

A) ICAO Standard Atmosphere
B) Army Standard Metro
C) US Standard Atmosphere

Το μοντέλο που θα χρησιμοποιήσουμε για να προσομοιώσουμε την ατμόσφαιρα στον εξομοιωτή μας ονομάζεται International Standard Atmosphere[3], και έχει κυκλοφορήσει το 1975 ως ISO-2533. Όπως και το US Standard Atmosphere, το ISA θέτει τη βάση του πάνω στο ICAO Standard Atmosphere, προσθέτοντας την ελλείπων έννοια της σχετική υγρασίας στο επίπεδο της θάλασσας.

Το ISA παρέχει συγκεκριμένες σταθερές τις οποίες χρησιμοποιώ για να υλοποιήσω τη λειτουργικότητα που χρειάζεται. Αυτές τις τιμές τις παραθέτω στον πιο κάτω πίνακα:

Σύμβολο	**Τιμή**	**Εξήγηση**
g_n	9.80665 m/s^2	Η επιτάχυνση της βαρύτητας στο γεωγραφικό πλάτος $45° 32' 33''$

r	6356766 m	Ακτίνα της Γης
T_n	288.15 °K	Θερμοκρασία στο επίπεδο της θάλασσας.
T_{11}	216.65 °K	Θερμοκρασία στο επίπεδο της τροπόπαυσης (11km)
H_n	0 m	Υψόμετρο στο επίπεδο της θάλασσας.
H_{11}	11000 m	Υψόμετρο στο επίπεδο της τροπόπαυσης.
p_n	101325 Pascal	Πίεση του αέρα στο επίπεδο της θάλασσας.
p_{11}	22632 Pascal	Πίεση στο επίπεδο της τροπόπαυσης.
β	-0.0065	Ρυθμός πτώσης της θερμοκρασίας ανα μέτρο στην τροπόσφαιρα.
R	287.05287	Specific Gas Constant

3.2 Λειτουργίες που υποστηρίζονται

Λειτουργίες που έχουν υλοποιηθεί και ο τρόπος υπολογισμού τους:

3.2.1 Εύρεση Βαρύτητας:

$$g = g_n(\frac{r}{r+h})^2$$

Εξίσωση 3.1Εξίσωση υπολογισμού επιτάχυνσης της βαρύτητας σε υψόμετρο.

Η επιτάχυνση της βαρύτητας είναι ανάλογη του τετραγώνου του λόγου της ακτίνας της γης r ως προς το υψόμετρο το οποίο βρισκόμαστε (r + h) επί την καθιερωμένη standard επιτάχυνση της βαρύτητας.

Το αποτέλεσμα έχει μονάδες μέτρησης m/s^2

3.2.2 Εύρεση θερμοκρασίας:

$$T = T_b + \beta(H - H_b)$$

Εξίσωση 3.2 Εξίσωση υπολογισμού της θερμοκρασίας σε υψόμετρο.

Στο ISA, η ατμόσφαιρα χωρίζεται σε στρώματα.

Ο δείκτης β αναγνωρίζει σε πιο στρώμα είμαστε και συγκεκριμένα καθορίζει πως T_b είναι η βασική θερμοκρασία (σε Κέλβιν) στο συγκεκριμένο στρώμα, H_bείναι το υψόμετρο στο οποίο ξεκινά το συγκεκριμένο στρώμα και β είναι ο ρυθμός μεταβολής της θερμοκρασίας ανά μέτρο.

Π.χ. Σε υψόμετρο 10 μέτρα, ξέρουμε πως βρισκόμαστε στην τροπόσφαιρα η οποία ξεκινά σε υψόμετρο 0 μέτρα, έχει βασική θερμοκρασία 287.15°K και ο ρυθμός μεταβολής της θερμοκρασίας είναι -0.0065 βαθμοί ανά μέτρο. Κάνοντας ένα απλό υπολογισμό βλέπουμε πως στο υψόμετρο που είμαστε έχουμε 288.15 – 0.0065 * 10 = 288.085 °K

Το στρώμα το οποίο μας ενδιαφέρει εμάς ονομάζεται τροπόσφαιρα η οποία ξεκινά στο επίπεδο της θάλασσας ($H_b = 0\ m$) και τελειώνει στην τροπόπαυση η οποία βρίσκεται σε υψόμετρο 11km, όπου παρατηρείται πως η θερμοκρασία σταματά να διαφοροποιείται μέχρι και τα 20km.

Το αποτέλεσμα έχει μονάδα μέτρησης σε °K.

3.2.3 Εύρεση Πίεσης:

$$p = p_b \exp\left[1 + \frac{\beta}{T_b}(H - H_b)\right]$$

Εξίσωση 3.3 Εξίσωση υπολογισμού πίεσης του αέρα σε υψόμετρο.

Όπως και στη περίπτωση της θερμοκρασίας, η εύρεση της πίεσης του αέρα γίνεται βάση του υψομέτρου στο οποίο βρισκόμαστε, το οποίο καθορίζει το στρώμα της ατμόσφαιράς στο οποίο είμαστε. Βρίσκοντας το στρώμα της ατμόσφαιρας, τότε χρησιμοποιούμε τα βασικά χαρακτηριστικά (p_b, T_b, β, H_b) του συγκεκριμένου στρώματος για τον υπολογισμό της πίεσης του αέρα.

Το αποτέλεσμα έχει μονάδα μέτρησης σε Pascal.

3.2.4 Εύρεση Πυκνότητας:

$$\rho = \frac{p}{RT}$$

Εξίσωση 3.4 Εξίσωση υπολογισμού πυκνότητας του αέρα σε υψόμετρο.

Ο υπολογισμός της πυκνότητας του αέρα για 0% υγρασία καθορίζεται από το πιο πάνω τύπο ο οποίος ονομάζεται και ως «νόμος ιδανικών αερίων».

3.2.5 Humidity Correction:

$$\rho_{hc} = \rho\,(1\ -0.00378\ *relative\ humidity(\%)\ *\left(\frac{P_{wv}}{P_{std}}\right))$$

Εξίσωση 3.5 Εξίσωση υπολογισμού ποσοστού διόρθωσης πυκνότητας του αέρα για σχετική υγρασία. [1]

Ο τρόπος με τον οποίο γίνεται ο υπολογισμός της πυκνότητας του αέρα με υγρασία γίνεται με τον πιο πάνω τύπο, ο οποίος εξαρτάται άμεσα από την πίεση υδρατμού σε κορεσμό στη συγκεκριμένη θερμοκρασία. Το 0.00378 προκύπτει από την εξίσωση 1% * (1 – Μοριακή μάζα υδρατμών νερού / Μοριακή μάζα αέρα) [4].

Το ποσοστό rh καθορίζει την ποσότητα του υδρατμού που υπάρχει στην ατμόσφαιρα σε σχέση με το πόσο φορτίο υδρατμών μπορεί να αποθηκευθεί στην ατμόσφαιρα πριν αρχίσουν να δημιουργούνται σωματίδια νερού. Με λίγα λόγια, για rh = 1 ή 100%, τότε σε πραγματικές συνθήκες θα είχαμε βροχή.

3.2.6 Πίεση υδρατμού σε κορεσμό

Φανταστείτε ένα υγρό μέσα σε ένα ερμητικά κλειστό δοχείο το οποίο αφήνεται να ηρεμήσει. Το συγκεκριμένο υγρό λέμε πως βρίσκεται σε κορεσμό όταν παρουσιάζει ισορροπία στον αριθμό από σωματίδια τα οποία εξαερίζονται από την επιφάνεια του και στα σωματίδια τα οποία έρχονται πίσω από την αέρια τους μορφή. Αυτή η ισορροπία συμβαίνει λόγω του ότι ο αέρας έχει φτάσει στα όρια χωρητικότητας του σε σωματίδια του υγρού. Η μέτρηση της πίεσης αυτών των σωματιδίων, στην περίπτωση που το υγρό είναι νερό, καθορίζει τη ποσότητα των σωματιδίων νερού (υδρατμού) που μπορούν να αποθηκευτούν στην ατμόσφαιρα έτσι ώστε να έχουμε 100% humidity.

Αρχικά είχε μελετηθεί η εξίσωση (A1.1) [4] η οποία καθορίζει τη πίεση υδρατμού σε κορεσμό για τις θερμοκρασίες 15-27°C, ωστόσο λόγω του συγκεκριμένου ορίου το οποίο ήταν πολύ περιοριστικό, χρειάστηκε να βρούμε ένα άλλο τύπο με μεγαλύτερο εύρος εφαρμογής.

Εν τέλει επιλέχθηκε η φόρμουλα των W. Wagner και A. Pruss [5] για υπολογισμό της πίεσης υδρατμού σε κορεσμό. Η συγκεκριμένη φόρμουλα έχει εύρος από 0 - 360°C.

Κεφάλαιο 4

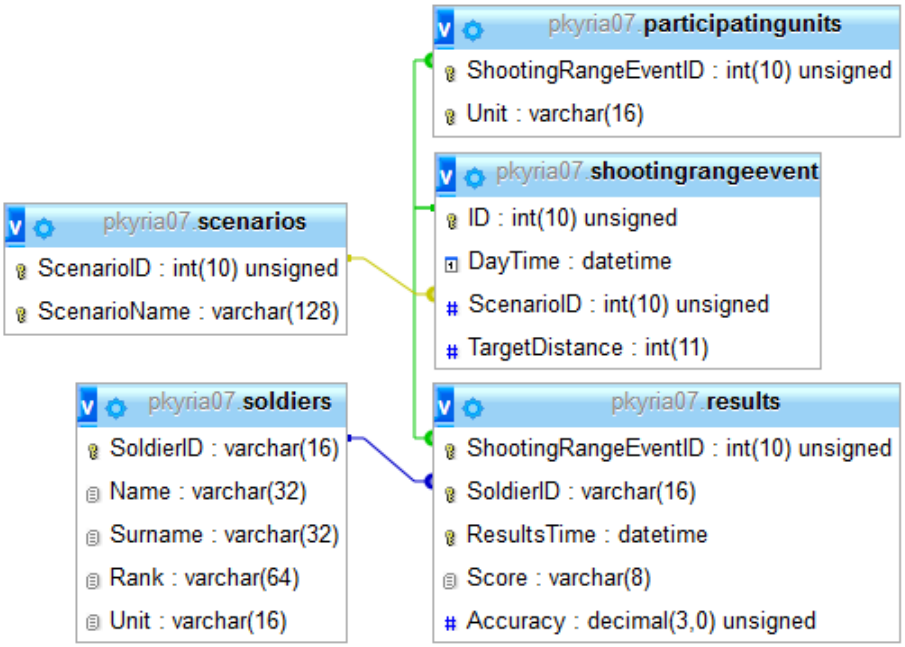
Βάση Δεδομένων MySQL και επικοινωνία μέσω PHP και JSON

4.1 Γενικά

Ο λόγος που επιλέγηκε να γίνει η δημιουργία μιας βάσης δεδομένων για το συγκεκριμένο project ήταν το γεγονός ότι ένα σύστημα τέτοιου βαθμού θα πρέπει να μπορεί χρησιμοποιείται από πολλά άτομα και μονάδες ταυτόχρονα.

Ως εκ τούτου, ο μόνος τρόπος για να δώσουμε συνοχή στα δεδομένα που θα αποθηκεύονται είναι με τη χρήση μιας κεντρικής βάσης δεδομένων η οποία να μπορεί να εξυπηρετήσει πλειάδα χρηστών παράλληλα.

4.2 Διάγραμμα Βάσης Δεδομένων



Εικόνα 4.1 Διάγραμμα πινάκων βάσης δεδομένων

Όπως μπορείτε να δείτε και στο σχήμα, η βάση δεδομένων αποτελείται από 5 πίνακες, ο καθένας από τους οποίους εκτελεί ξεχωριστή λειτουργία.

Στον πίνακα scenarios φυλάσσονται τα διάφορα σενάρια τα οποία παρέχει ο εξομοιωτής μας, τα οποία χαρακτηρίζονται από ένα μοναδικό αριθμό και περιέχουν το όνομα του σεναρίου.

Ο πίνακας ShootingRangeEvent περιέχει τα δεδομένα των βολών που έχουν διενεργηθεί μέχρι στιγμής. Κάθε βολή χαρακτηρίζεται από ένα μοναδικό αριθμό, ο οποίος αυξάνεται αυτόματα. Για κάθε βολή φυλάμε την ημέρα και ώρα που είχε εκτελεστεί καθώς και το είδος του σεναρίου και την απόσταση του στόχου από τον σκοπευτή.

Ο πίνακας ParticipatingUnits περιέχει τις στρατιωτικές μονάδες που έχουν λάβει μέρος σε βολές. Μια στρατιωτική μονάδα μπορεί να πάρει μέρος σε πολλές βολές και σε μια βολή μπορούν να λάβουν μέρος πολλές μονάδες.

Ο πίνακας Soldiers περιέχει τα δεδομένα των στρατιωτών. Ο κάθε στρατιώτης χαρακτηρίζεται από τον αριθμό ταυτότητας του ο οποίος αποτελείται από αριθμούς και γράμματα. Για κάθε στρατιώτη, φυλάμε το ονοματεπώνυμο του, το βαθμό του καθώς και την στρατιωτική μονάδα στην οποία ανήκει.

Τέλος, ο πίνακας Results περιέχει τα αποτελέσματα για τη βολή που εκτελέστηκε από κάθε στρατιώτη. Τα αποτελέσματα μιας βολής χαρακτηρίζονται από το αναγνωριστικό της βολής στην οποία είχε συμμετάσχει ο στρατιώτης, την ταυτότητα του στρατιώτη καθώς και το πότε συλλέχθηκαν τα αποτελέσματα. Τα δεδομένα μιας καταχώρησης result είναι το σκορ του στρατιώτη καθώς και το ποσοστό ακριβείας των βολών του.

Το είδος του collation που χρησιμοποιείται είναι **utf8_bin** για να μας δίνει τη δυνατότητα να φυλάμε δεδομένα σε ότι γλώσσα θέλουμε, χωρίς το φόβο πως οι συγκρίσεις των τιμών θα είναι λάθος λόγω άγνωστων χαρακτήρων.

Επίσης, οι πίνακες είναι διασυνδεδεμένοι μεταξύ τους και έτσι η οποιαδήποτε μεταβολή ή διαγραφή ενός στοιχείου θα διαδίδεται σε όλους, κρατώντας το σύνολο των δεδομένων συνεπές.

4.3 Διασύνδεση μέσω PHP

Καθώς το Unity3D δεν παρέχει κάποια native βιβλιοθήκη για την επίτευξη επικοινωνίας με βάση δεδομένων, χρειαζόμαστε κάποιο διαμεσολαβητή ο οποίος να εκτελεί αυτή την επικοινωνία μεταξύ των δυο, σε μορφή που και οι δύο καταλαβαίνουν.

Η PHP είναι μια γλώσσα προγραμματισμού ιστοσελίδων η οποία παρέχει τεράστιες δυνατότητες, όπως για παράδειγμα αντικειμενοστρεφές προγραμματισμό. Είναι αρκετά εύκολη στη χρήση, και σου δίνει τη δυνατότητα να μπορείς να δημιουργείς γρήγορα αυτό που θες να πετύχεις. Αυτό είναι σημαντικό, καθώς σημαίνει πως σε μετέπειτα στάδια θα μπορούμε με σχετική ευκολία να μεταβάλουμε και να αναπτύξουμε τον τρόπο λειτουργίας του συστήματος μας.

Για την επικοινωνία μεταξύ των αρχείων PHP και της βάσης δεδομένων χρησιμοποιείται αντικειμενοστρεφής προγραμματισμός. Συγκεκριμένα, χρησιμοποιείται ένα αντικείμενο της κλάσης mysqli το οποίο ανοίγει ένα δίαυλο επικοινωνίας μεταξύ της PHP και του MySQL εξυπηρετητή και προσφέρει διάφορους τρόπους επικοινωνίας.

Η επικοινωνία μεταξύ τους γίνεται μέσω queries και prepared statements. Τα δυο τους είναι ισοδύναμα, ωστόσο, τα prepared statements είναι πιο ασφαλή καθώς διενεργούν έλεγχο στα δεδομένα που πρόκειται να αποστείλουν έτσι ώστε να προλαμβάνονται κακόβουλες επιθέσεις ενάντια στη βάση δεδομένων. Επίσης, γίνεται χρήση και transactions τα οποία επιτρέπουν την εκτέλεση πολλαπλών εντολών σε μια, χωρίς να χρειάζεται να ανοίγουν και να κλείνουν πολλές συνδέσεις προς τη βάση δεδομένων.

Η επικοινωνία της C# με την PHP επιτυγχάνεται μέσω HTTP POST request. Η συγκεκριμένη λειτουργικότητα παρέχεται από την κλάση WWW του UnityEngine. Ένα ωφέλημα των POST request είναι πως δεν επιτρέπουν στον χρήστη να εισαγάγει δεδομένα ή να αποστείλει δεδομένα στη βάση δεδομένων χρησιμοποιώντας το URL στον internet browser του, δίνοντας μια μικρή ασφάλεια από επιτήδειους.

4.4 JSON και βιβλιοθήκη LitJSON

Το JSON είναι ένα ακρωνύμιο για το όνομα Javascript Object Notation και αποτελεί μια standardized μέθοδο αντιπροσώπευσης δομών δεδομένων η οποία είναι εύκολη στο διάβασμα και την κατανόηση. Πιο κάτω παρατίθεται μια καταχώρηση από τον πίνακα Soldiers η οποία επιστρέφεται από τη βάση δεδομένων:


```
{
        "SoldierID":"21344/09/09B",
        "Name":"Ιωάννης",
        "Surname":"Ιωάννου",
        "Rank":"Οπλίτης",
        "Unit":"213ΤΠ"
}
```

Ο τρόπος μετατροπής δεδομένων από τα δεδομένα που παίρνουμε από την βάση δεδομένων σε JSON είναι πολύ απλός. Χρησιμοποιώντας associative arrays τα οποία παρέχονται από την PHP μπορούμε να συνδέσουμε κάθε τιμή που παίρνουμε με ένα αναγνωριστικό. Ο πιο κάτω κώδικας σε PHP θα μας δημιουργούσε την πιο πάνω εικόνα:

Εικόνα 4.2 JSON output

```
$soldier = array();
$soldier[“SoldierID”] = "21344/09/09B",
$soldier["Name"] = "Ιωάννης";
$soldier["Surname"] = "Ιωάννου";
$soldier["Rank"] = "Οπλίτης",
$soldier["Unit"] = "213ΤΠ";
echo json_encode($soldier)
```

Εικόνα 4.3 PHP JSON encoding

Πρέπει να σημειωθεί πως λόγω της χρήσης utf8_bin collation από τη βάση δεδομένων, τα strings τα οποία επιστρέφονται είναι κωδικοποιημένα σε μορφή UTF8. Με λίγα λόγια, δυστυχώς δεν θα είναι εύκολο για κάποιο να διαβάσει απευθείας ελληνικούς χαρακτήρες από το JSON που θα επιστρέφεται. Π.χ. η λέξη «τεστ» θα επιστρέφεται ως «\u03c4\u03b5\u03c3\u03c4». Ωστόσο, τα strings της C# υποστηρίζουν UTF8 και θα δείξουν σωστά τους χαρακτήρες.

Η βιβλιοθήκη litJSON αποτελείται από ένα αρχείο dll το οποίο τοποθετείται στο φάκελο που περιέχει τα assets για το Unity3D project. Μετά από την εισαγωγή της βιβλιοθήκης, μπορείς να την χρησιμοποιήσεις πολύ απλά σε C#:

```
using LitJSON;
….
string json = “…soldierJSON…”
JsonData soldierData = JsonMapper.ToObject(json);
string soldierName = (string) soldierData[“Name”];
                    OR
class Soldier { ….. }
Soldier soldier = JsonMapper.ToObject<Soldier>(json);
```

4.5 Λειτουργίες που υποστηρίζονται

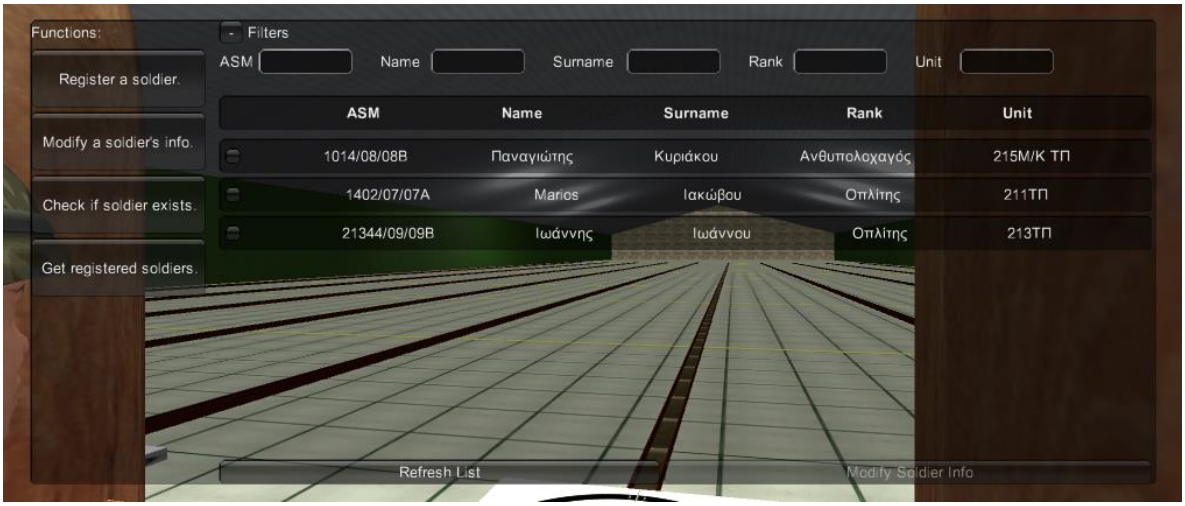
Χρησιμοποιώντας την PHP, έχουν ετοιμαστεί 12 διαφορετικά κομμάτια κώδικα τα οποία διαχειρίζονται τις λειτουργίες τις οποίες χρειάζεται να εκτελούνται για να παρέχεται η βασική λειτ Εικόνα 4.4 Μετατροπή από JSON σε δεδομένα της C#

Οι λειτουργίες που υποστηρίζονται είναι οι ακόλουθες:

- Καταχώρηση νέου στρατιώτη
- Τροποποίηση καταχώρησης στρατιώτη
- Λήψη όλων των καταχωρημένων στρατιωτών.
- Έλεγχος αν ένας στρατιώτης είναι καταχωρημένος.
- Λήψη σεναρίων
- Καταχώρηση νέας βολής
- Λήψη καταχωρημένων βολών
- Καταχώρηση αποτελεσμάτων βολής ενός στρατιώτη
- Λήψη των στρατιωτών που έλαβαν μέρος σε μια συγκεκριμένη βολή
- Λήψη καταχωρημένων αποτελεσμάτων ενός στρατιώτη για όλες τις βολές του.
- Λήψη καταχωρημένων αποτελεσμάτων ενός στρατιώτη για συγκεκριμένη βολή.

Για περαιτέρω πληροφορίες παρακαλώ αναφερθείτε στο παράρτημα Α.

4.6 Παρουσίαση δεδομένων σε Unity3D



Εικόνα 4.5 Παρουσίαση καταχωρημένων στρατιωτών στη βάση δεδομένων μας, παρέχοντας και τη δυνατότητα άμεσου φιλτραρίσματος/αναζήτησης

Όπως δείχνει η εικόνα 4.5, η επικοινωνία μεταξύ των διαφόρων συστατικών έχει επιτευχθεί και με την χρήση του GUI το οποίο παρέχεται από το Unity3D, τα δεδομένα που βρίσκονται αποθηκευμένα στη βάση δεδομένων μας μπορούν να παρουσιάζονται με φιλικό προς το χρήστη τρόπο. Επίσης, δίνεται η δυνατότητα αναζήτησης εντός του συνόλου στρατιωτών που έχουμε λάβει, ευκολύνοντας τη χρήση της εφαρμογής.

Εικόνα 4.6 Επιλέγοντας το στρατιώτη που θέλουμε και πατώντας στο κουμπί Modify Soldier Info, μας εμφανίζεται η πιο πάνω φόρμα όπου μπορούμε να αλλάξουμε όλα τα στοιχεία του στρατιώτη, εκτός από τον αριθμό στρατιωτικής του ταυτότητας.

Για τη δημιουργία του γραφικού περιβάλλοντος εντός του Unity, χρησιμοποιήθηκε η συνάρτηση OnGUI() η οποία χρησιμοποιείται αποκλειστικά από το Unity3D για παρουσίαση του γραφικού περιβάλλοντος που παρουσιάζεται στον χρήστη. Μπορούμε να ορίσουμε την συνάρτηση OnGUI σε πολλαπλά scripts ταυτόχρονα και να έχουμε πολλαπλά αντικείμενα που να διαχειρίζονται το γραφικό περιβάλλον του χρήστη.

Οι εικόνες 4.5 και 4.6 είναι πρωτότυπα για το τι μπορεί να επιτευχθεί με το GUI του Unity3D.

Κεφάλαιο 5

Διασύνδεση με σύστημα laser motion detection

5.1 Γενικά

Το σύστημα του laser motion detection έχει αναπτυχθεί παράλληλα με την παρούσα ατομική διπλωματική εργασία από τη Μαρία Παρασκευά, φοιτήτρια του Πανεπιστημίου Κύπρου. Στόχος μας ήταν να συνδέσουμε τα δυο συστήματα, τα οποία αποτελούν επιμέρους κομμάτια του εξομοιωτή πεδίου βολής.

Το σύστημα laser motion detection θα χρησιμοποιηθεί για δίνεται στους χρήστες του συστήματος μια ακόμη πιο ρεαλιστική εμπειρία. Η χρήση του motion detection θα μας επιτρέψει σε μετέπειτα στάδια να χρησιμοποιήσουμε αδρανή όπλα τα οποία θα έχουν εξοπλιστεί με συστήματα ανάκρουσης για να εκτελούμε πραγματικές βολές στον εξομοιωτή.

Με πολύ απλά λόγια, το σύστημα motion detection θα μας δίνει τη θέση στην οποία στοχεύει ο χρήστης εντός του εξομοιωτή για να μπορούμε να εκτελέσουμε τη βολή.

Προτού δούμε τον τρόπο διασύνδεσης που χρησιμοποιήθηκε, ας εξηγήσουμε τον τρόπο με τον οποίο επιτυγχάνεται η σκόπευση εντός του εξομοιωτή.

5.2 Στόχευση με τη χρήση RayCast

Για να μπορέσουμε να στοχεύσουμε στον εξομοιωτή μας, πρώτα, παίρνουμε ένα σημείο στην οθόνη μας. Στη παρούσα φάση, το σημείο αυτό δύναται να καθορίζεται από τη θέση του mouse μας, ή από το σύστημα motion detection.
Ακολούθως, χρησιμοποιούμε τη συνάρτηση Camera.ScreenPointToRay για να μετατρέψουμε το δισδιάστατο σημείο σε σημείο στον τρισδιάστατο χώρο. Η συγκεκριμένη συνάρτηση δημιουργεί μια ακτίνα η οποία ξεκινά από το backplane της κάμερας που χρησιμοποιούμε για να δούμε το τρισδιάστατο κόσμο του εξομοιωτή και περνά μέσα από το σημείο της οθόνης στο οποίο αναφερόμαστε. Στη συνέχεια, μετακινιόμαστε απόσταση 400* μέτρα προς την κατεύθυνση που βλέπει η ακτίνα, ξεκινώντας από το σημείο έναρξης της ακτίνας και καταλήγουμε στο σημείο στο κόσμο όπου έχουμε στοχεύσει.

Για να μετακινήσουμε το όπλο ώστε να βλέπει προς το συγκεκριμένο σημείο, καλούμε τη συνάρτηση transform.LookAt(Vector3 position) στο transform το όπλου μας, με δεδομένο εισόδου το σημείο το οποίο έχουμε υπολογίσει.

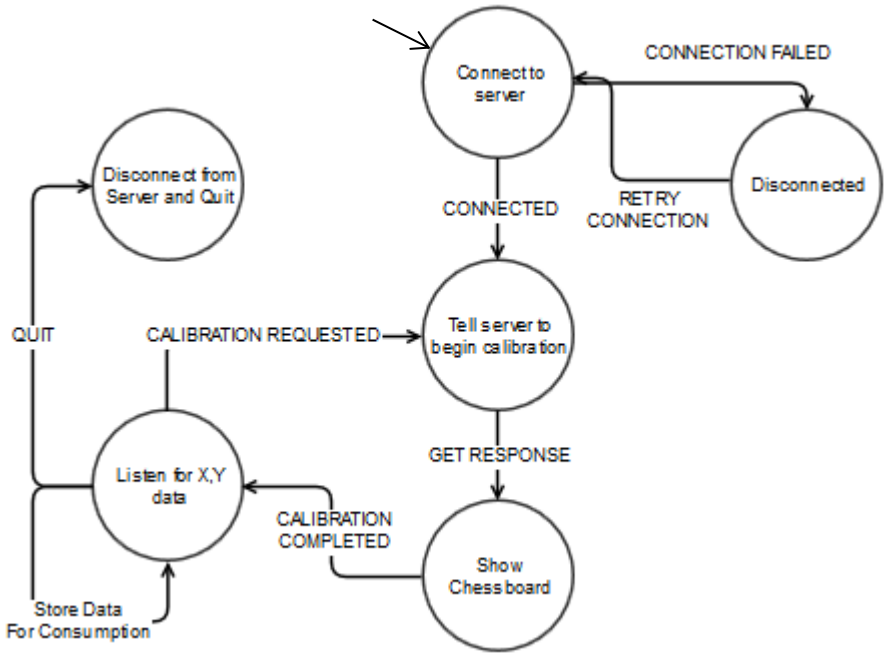
* Ο λόγος που χρησιμοποιούμε τα 400 μέτρα είναι γιατί το στρατιωτικό τυφέκιο G3 υποστηρίζει στόχευση μέσω του κλισιοσκοπίου του σε αποστάσεις των 400 μέτρων.

5.3 Διασύνδεση των δυο συστημάτων

Η διασύνδεση των δυο συστημάτων επιτεύχθηκε με τη χρήση του πρωτοκόλλου TCP/IP. Ο λόγος που επιλέξαμε το TCP/IP είναι γιατί δεν θέλαμε να έχουμε καθόλου πιθανότητα απώλειας δεδομένων.
Στην υλοποίηση μας, το σύστημα motion tracking ξεκινά ένα server και περιμένει κάποιο client να ενωθεί. Ο εξωμοιοτής λειτουργεί ως ο client και ενώνεται με τον server. Το πρωτόκολλο έχει ως ακολούθως:

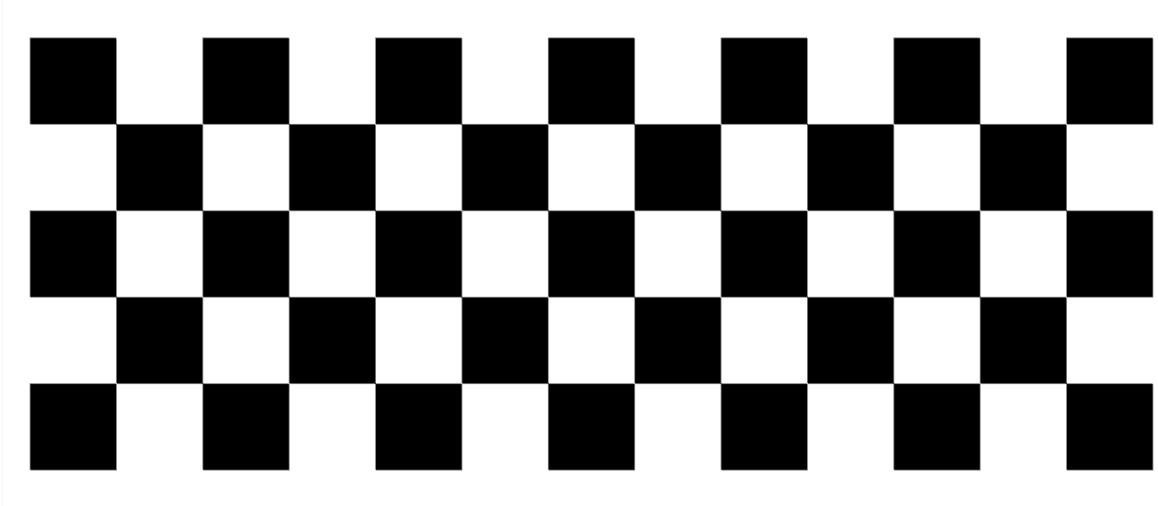
1) Ο client ενώνεται με τον server.

2) Ο client αποστέλλει μήνυμα CALIBRATE στον server και περιμένει ανταπόκριση.
3) Με την λήψη του μηνύματος απόκρισης, ο client ενεργοποιεί ένα chessboard στην οθόνη του, αποστέλλει ένα μήνυμα μορφής «X:Y», όπου X και Y οι διαστάσει της οθόνης και αναμένει απάντηση από τον server για να προχωρήσει. Με την ολοκλήρωση του calibration, ο client απενεργοποιεί το chessboard και μπαίνει σε κατάσταση listen.
4) Μόλις ο client παρατηρήσει πως υπάρχουν δεδομένα για λήψη, τότε τα λαμβάνει και τα αποκωδικοποιεί και απαντά πίσω πως τα έχει παραλάβει. Τα δεδομένα τα οποία λαμβάνει έχουν τη μορφή «x:y», όπου x,y το σημείο στην οθόνη στο οποίο έχει στοχεύσει ο χρήστης. Με τη λήψη των δεδομένων, αποθηκεύει τις τιμές σε μια μεταβλητή η οποία είναι προστατευμένη με locks, έτσι ώστε να είναι thread safe.
5) Ανα πάσα στιγμή, ο client, μέσω επιλογής του χρήστη, μπορεί να αποστείλει σήμα CALIBRATE, στην οποία περίπτωση η ροή επιστρέφει πίσω στο βήμα 3.
6) Όταν ο εξομοιωτής κλείσει, αποστέλλεται ένα μήνυμα QUIT το οποίο ενημερώνει τον server ότι μπορεί να κλείσει.
7) Σε περίπτωση που ο client στο βήμα 1 δεν καταφέρει να συνδεθεί με τον server, τότε μπαίνει σε κατάσταση DISCONNECTED και περιμένει από το χρήστη να ζητήσει επαναπροσπάθεια της σύνδεσης.



Εικόνα 5.1 FSM του client

Το chessboard αποτελείται από τετραγωνάκια με πλευρά 100px. Ότι ποσοστό της οθόνης δεν επαρκεί για τη δημιουργία του τελευταίο τετραγώνου, τότε χρησιμοποιείται ως border. Σε περίπτωση που το border είναι ίσο με 0, τότε θυσιάζεται ένα τετράγωνο έτσι ώστε να δημιουργηθεί border. Για τη δημιουργία του chessboard, δημιουργείται ένα texture μεγέθους 1x1 του οποίου μεταβάλλουμε το χρώμα του με τη συνάρτηση texture.SetPixels() και εφαρμόζοντας την αλλαγή με texture.Apply().



Εικόνα 5.2 Το chessboard

Ο κώδικας του client βασίζεται πάνω σε βασικά κομμάτια που παρέχονται από τη C#. Συγκεκριμένα, χρησιμοποιούμε ένα αντικείμενο TCPClient το οποίο εγκαθιδρύει τη σύνδεση με τον server και προσφέρει πρόσβαση στο NetworkStream της σύνδεσης, το οποίο χρησιμοποιούμε για να διαβάσουμε και να γράψουμε στο socket.

Ένα πρόβλημα το οποίο έπρεπε να επιλυθεί είναι πως οι κλήσεις των συναρτήσεων που έχουν να κάνουν με το διάβασμα από το socket είναι blocking, και ως εκ τούτου δε θα ήταν σοφό να τοποθετήσουμε αυτές τις συναρτήσεις απευθείας μέσα στο Unity. Ο τρόπος με τον οποίο μπορούμε να επιλύσουμε αυτό το πρόβλημα είναι με την χρήση thread, τα οποία είναι ανεξάρτητα από το Unity, και ως εκ τούτου δε μπορούν να προκαλέσουν hang.

Ο client, με τη λήψη νέων δεδομένων, τα αποθηκεύει σε αντικείμενο τύπου Vector3 το οποίο είναι προσβάσιμο από μέρη του Unity. Το διάβασμα και το γράψιμο στις συγκεκριμένες μεταβλητές γίνεται πάντοτε με locks για να μην υπάρξουν race conditions.

Κεφάλαιο 6

Υλοποίηση

6.1 Βαλλιστική

Όπως είδαμε και στο κεφάλαιο 2, η κίνηση μιας βολίδας εξαρτάται από ένα σύνολο δυνάμεων και ροπών που ασκούνται πάνω της κατά τη πτήση της. Έχοντας αναλύσει τις διάφορες επιλογές βαλλιστικών μοντέλων που μπορούσα να υλοποιήσω κατέληξα στο συμπέρασμα ότι η χρήση του μοντέλου 6-DOF θα μας έφερνε τα πιο αληθοφανή αποτελέσματα. Το μοντέλο 6-DOF είναι το μόνο μοντέλο που λαμβάνει υπόψη του όλες τις δυνάμεις και ροπές που ασκούνται πάνω στη βολίδα και παρέχει κίνηση σε 6 διαφορετικούς άξονες. Επίσης, είναι το μόνο που θεωρεί πως οι τιμές μεταβάλλονται σε συνάρτηση με το χρόνο παρά με την απόσταση που είχε διανυθεί.

Ξεκινώντας τη προσπάθεια μου να λύσω αυτό το πρόβλημα παρατήρησα πως το πρόβλημα θα μπορούσε να λυθεί χρησιμοποιώντας τη συνάρτηση FixedUpdate(). Η συνάρτηση FixedUpdate() είναι μια συνάρτηση η οποία εκτελείται σε συγκεκριμένα διαστήματα στο χρόνο δίνοντας μας την έννοια dT (ή, όπως την ονομάζουν στο Unity, Time.fixedDeltaTime). Το διάστημα αυτό είναι προσαρμόσιμο και ως εκ τούτου η FixedUpdate() μπορεί να τρέχει πολύ περισσότερες φορές απ'ότι η Update() η οποία

εξαρτάται από το framerate. Με λίγα λόγια, η FixedUpdate() είναι φτιαγμένη για ανάλυση εξισώσεων φυσικής.

Έχοντας καταλήξει πως η λύση βρισκόταν με τη χρήση της FixedUpdate(), άφησα τη λύση του προβλήματος 6-DOF και έστρεψα την σκέψη μου στις φυσικές εξισώσεις που παρουσίασα στο 2^{o} κεφάλαιο. Προσεγγίζοντας το πρόβλημα, εν τέλει κατέληξα στους ακόλουθους μετασχηματισμούς:

	Νόημα	**Αντιστοιχία στο Unity3D**
$\vec{V}$	Διάνυσμα της ταχύτητας	rigidbody.velocity
$\vec{\imath}$	Κανονικοποιημένο διάνυσμα της ταχύτητας	rigidbody.velocity.normalized
V	Μέγεθος Διανύσματος ταχύτητας	rigidbody.velocity.magnitude
$\vec{x}$	Διάνυσμα άξονα συμμετρίας	transform.forward
p	Ταχύτητα περιστροφής γύρω από τον άξονα συμμετρίας	rigidbody.angularVelocity.z
$\frac{d\vec{x}}{dt}$		(transform.forward – previousForward) / Time.fixedDeltaTime
$\frac{d\vec{\imath}}{dt}$		(prevVelocity.normalized – rigidbody.velocity.normalized) / Time.fixedDeltaTime

Εικόνα 6.1 Πίνακας μετατροπής συμβόλων εξισώσεων δυνάμεων (2^{o} κεφ.) σε κώδικα Unity3D

Με τη χρήση των πιο πάνω μετασχηματισμών, έγινε δυνατή η υλοποίηση των φυσικών εξισώσεων οι οποίες υπολογίζουν τις δυνάμεις και τις ροπές που ασκούνται πάνω στις βολίδες κατά την ώρα της πτήσης τους.

Η εφαρμογή των δυνάμεων και των ροπών γίνεται μέσω των συναρτήσεων

rigidbody.AddForce() ή rigidbody.velocity = calculatedVelocity

rigidbody.AddTorque()

Αρχικά είχε γίνει χρήση της εξίσωσης του δεύτερου νόμου του Νεύτωνα για τον υπολογισμό της νέας ταχύτητας βάση των εξισώσεων:

$$\vec{F} = m * \vec{a}$$

$$\vec{a} = \frac{\vec{F}}{m}$$

$$\left(\frac{d\vec{v}}{dt}\right) = \frac{\vec{F}}{m}$$

$$d\vec{v} = \overrightarrow{v_2} - \overrightarrow{v_1}$$

$$\overrightarrow{v_2} = \frac{\vec{F}}{m} * dt + \overrightarrow{v_1}$$

Εξίσωση 6.1 Υπολογισμός ταχύτητας χρησιμοποιώντας τον δεύτερο νόμο του Νεύτωνα.

Ωστόσο, μετά από σύγκριση της πιο πάνω μεθόδου με το αποτέλεσμα της συνάρτησης AddForce(), βρέθηκε πως το τελικό αποτέλεσμα είναι το ίδιο.

Παραπάνω στοιχεία για τον τρόπο λειτουργίας των βολίδων μπορείτε να δείτε στο Annex A.

6.2 Υλοποίηση collision detection με χρήση LineCast

Λόγω του μικρού μεγέθους των βολίδων που βάλλονται καθώς και το γεγονός ότι κινούνται σε αρκετά υψηλές ταχύτητες, ήταν συχνό το φαινόμενο οι σφαίρες να περνάνε μέσα από colliders.
Η λύση που υπερίσχυσε ήταν η δημιουργία μιας συνάρτησης collision detection, χρησιμοποιώντας LineCast για την εύρεση collision. Ο τρόπος λειτουργίας του συγκεκριμένου αλγορίθμου είναι απλός:

Εκτελείται σε κάθε κάλεσμα της fixedUpdate, και σε κάθε εκτέλεση φυλάμε την θέση μας. Στην αμέσως επόμενη εκτέλεση της fixedUpdate, ελέγχεται κατά πόσο υπάρχει κάποιο collider στη γραμμή που σχηματίζεται μεταξύ των δύο θέσεων και αν ναι, τότε έχουμε collision και εκτελούνται οι ανάλογες ενέργειες.

Είναι αρκετά γρήγορος και απλός αλγόριθμός με πολύ καλά αποτελέσματα. Δυστυχώς, όμως, υπάρχει και πάλι η πιθανότητα το collision detection να αποτύχει, συνήθως σε περιπτώσεις με πολύ ψηλές γωνιακές ταχύτητες.

Ένα ακόμη μικρό πρόβλημα είναι πως λόγο του μικρού timestep το οποίο έχει οριστεί για την FixedUpdate() και λόγω των βίαιων δυνάμεων που ασκούνται πάνω στη βολίδα, γίνεται collide με τον εαυτό του. Αυτό μπορεί να φτιαχτεί πολύ εύκολα, αγνοώντας συγκρούσεις τέτοιου είδους.

6.3 Υλοποίηση Όπλου

Το όπλο, ως συμπεριφορά είναι υλοποιημένο στο C# αρχείο WeaponBehaviour.cs
Για να ανταποκρίνεται στον αλγόριθμο RayCasting, πρέπει να γίνει attach και το script RayCasting.cs πάνω στο game object.

Περιέχει τα ακόλουθα χαρακτηριστικά:

Ταχυβολία: (rpm)

Αριθμός σφαιρών ανα λεπτό.

Default: 400 (βλ. πίνακα 2.1)

Ταχύτητα Σφαίρας: (m/s)

Η ταχύτητα που θα έχει η σφαίρα με την έξοδο της από την κάννη.

Default: 800 (βλ. πίνακα 2.1)

Λόγος περιστροφής: (mm)

Χρησιμοποιείται για να υπολογιστεί η γωνιακή ταχύτητα της σφαίρας. Είναι η απόσταση που πρέπει να διανύσει μια σφαίρα εντός της κάννης για να εκτελέσει μια περιστροφή.

Default: 305

Ημιαυτόματο:

Χαρακτηρίζει αν το όπλο θα ρίχνει τις σφαίρες βολή κατά βολή ή βολή κατά ρυπάς.

Η γωνιακή ταχύτητα της σφαίρας (στο μετρικό σύστημα) υπολογίζεται ως:

$$w = \left(\frac{1000}{\lambda\acute{o}\gamma o\varsigma\ \pi\varepsilon\rho\iota\sigma\tau\rho o\varphi\acute{\eta}\varsigma}\right) * \ \tau\alpha\chi\acute{\upsilon}\tau\eta\tau\alpha\ \sigma\varphi\alpha\acute{\iota}\rho\alpha\varsigma$$

Εξίσωση 6.2 Υπολογισμός γωνιακής ταχύτητας της βολίδας

Η εκπυρσοκρότηση του όπλου ξεκινά να γίνεται με τη χρήση της συνάρτησης pressTrigger(). Αν το όπλο είναι αυτόματο, τότε θα αρχίσει να βάλλει κατά ρυπάς μέχρι να

εκτελεστεί η συνάρτηση releaseTrigger(). Αν το όπλο είναι ημι-αυτόματο, τότε θα βάλει μόνο μια σφαίρα.

Η σφαίρα γίνεται instantiate από ένα συγκεκριμένο prefab πριν να ριχθεί, ωστόσο, έχει προστεθεί και η λειτουργία γεμιστήρα (+ bullet characteristics loader from XML + magazine loader GUI) σε περίπτωση που μετέπειτα θέλουμε να έχουμε συγκεκριμένο αριθμό και είδος σφαιρών.

Η βολίδα εκτοξεύεται στην κατεύθυνση forward σε σχέση με το transform του όπλου. Εφόσον είναι ενεργοποιημένο το RayCasting, τότε το όπλο θα βλέπει πάντοτε εκεί που δείχνει ο δείκτης του mouse ή το laser του motion tracking.

Σημείωση: Τόσο το RayCasting όσο και crosshair χρειάζεται να συνδεθούν με το Motion Capture GameObject και να ενεργοποιηθεί η επιλογή Use Network για να χρησιμοποιήσουν το Motion Capture αντί του mouse.

Σημείωση 2: Ο ήχος που ακούγεται όταν το όπλο εκπυρσοκροτεί προέρχεται από τον χρήστη του YouTube GoodSoundForYou

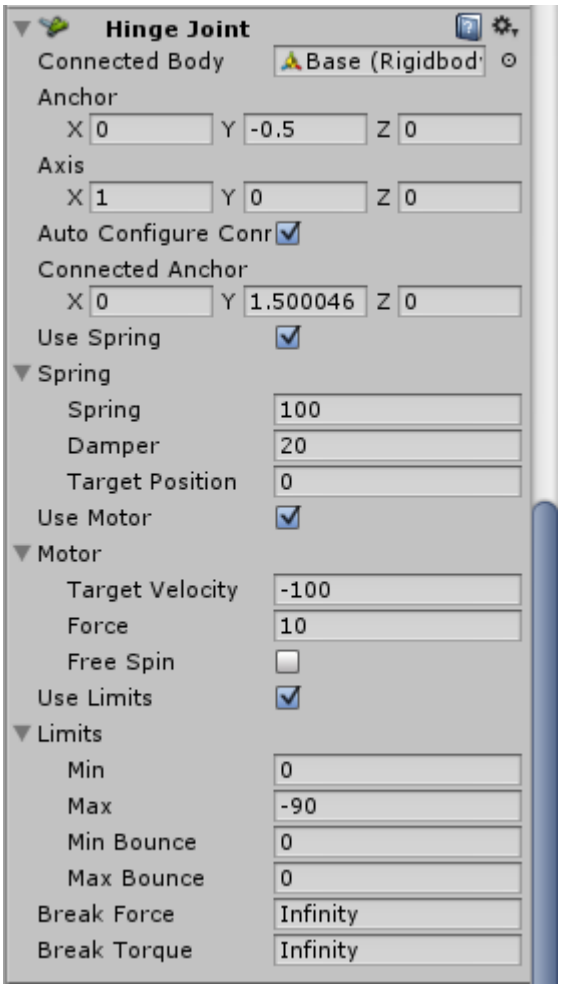
6.4 Υλοποίηση Στόχων

Έχουν υλοποιηθεί 2 διαφορετικά είδη στόχων.

A) Στατικός στόχος

B) Κινούμενος στόχος

Και τα δύο είδη στόχων αποτελούν πτυσσόμενους στόχους οι οποίοι έχουν δημιουργηθεί με hinge joint και ένα motor το οποίο αναλαμβάνει να τους ανυψώσει.



Εικόνα 6.2 Το hinge joint το οποίο χρησιμοποιείται για να αναπτύξει τους στόχους, το όριο καθορίζει τη γωνιά του στόχου

Εικόνα 6.3 Υπάρχουν δυο σχέδια στόχων, το καθένα με διαφορετική βαθμολόγηση.

Εικόνα 6.4 Στα αριστερά υπάρχει ο κινούμενος στόχος και στα δεξιά ο στατικός.

Οι κινούμενοι στόχοι μπορούν να ρυθμιστούν ανάλογα με το τι τους θέλουμε να κάνουν. Συγκεκριμένα, το μέγεθος της ράγα τους είναι προσαρμόσιμο, και μπορούν να ταλαντεύονται στα δυο άκρα της. Εκτός από ταλάντευση μπορούν να παραμείνουν στάσιμοι σε σημείο της επιλογής μας ή να μεταβούν σε ένα από τα άκρα της επιλογής μας. Επίσης, μπορούν να παρακολουθούν τον χρήστη έτσι ώστε η επιφάνεια του στόχου να είναι πάντοτε απέναντι απ' το χρήστη.

Η κίνηση των κινούμενων στόχων επιτυγχάνεται με τη βοήθεια της συνάρτησης Vector3.MoveTowards(Vector3 from, Vector3 to, float maxDistanceDelta).

Οι στάσιμοι στόχοι είναι σχετικά απλοί, χωρίς κάποια εξειδικευμένη λειτουργικότητα.

Τα δυο σχέδια στόχων έχουν διαφορετική βαθμολόγηση, η οποία καθορίζεται από την απόσταση του σημείου επαφής της βολίδας προς το κέντρο του στόχου. Ο τρόπος βαθμολόγησης θεωρεί πως ο κύκλος χωρίζεται σε ν κομμάτια. Βρίσκοντας την απόσταση του σημείο επαφής -> κέντρο στόχου, μπορούμε να υπολογίσουμε, βάση της ακτίνας του κύκλου, το ποσοστό ακρίβειας αλλά και το βαθμό που αντιστοιχεί για το συγκεκριμένο σημείο.

Η καταμέτρηση των στόχων γίνεται ως ακολούθως:

1) Η βολίδα κτυπά στην επιφάνεια του στόχου και ενημερώνει τον στόχο (script ScoreFinder.cs) για το συμβάν.
2) Ο στόχος υπολογίζει το σκορ και το αποστέλλει στον ScoreManager της σκηνής.
3) Το ScoreGUI λαμβάνει το νέο σκορ από τον ScoreManager και το δείχνει στην οθόνη.

Ο υπολογισμός της ακρίβειας του χρήστη υπολογίζεται ως ακολούθως:

$$accuracy = \frac{(targetRadius - distance)}{targetRadius}$$

Εξίσωση 6.3 Υπολογισμός ακρίβειας του χρήστη ανάλογα με την απόσταση της βολίδας από το κέντρο του κύκλου.

Ο Score Manager εν τέλει υπολογίζει την ολική ακρίβεια ως ακολούθως:

$$finalAccuracy = \frac{successes}{misses + successes} * \frac{totalAccuracy}{successes}$$

Εξίσωση 6.4 Υπολογισμός τελικής ακρίβειας της βολής για τον χρήστη

6.5 Διαχείριση του καιρού

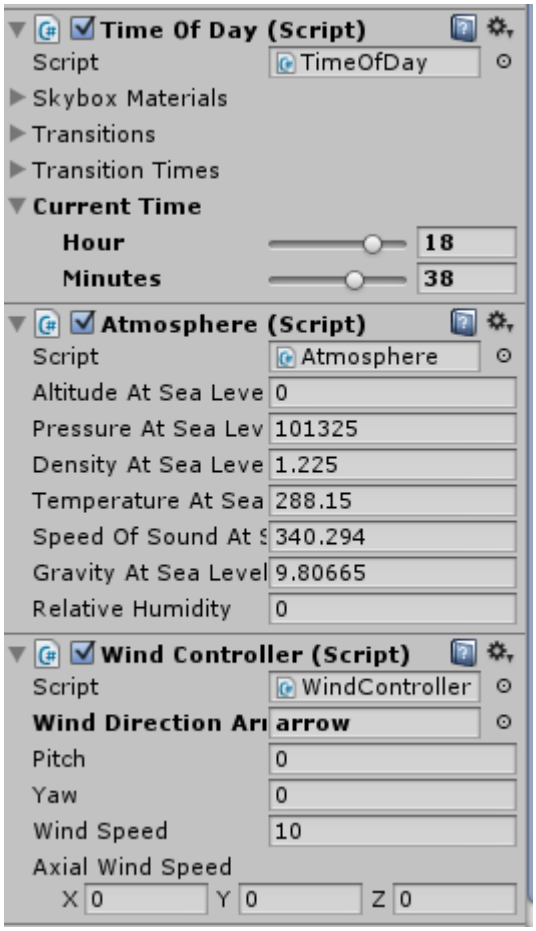
Έχει υλοποιηθεί η δυνατότητα αλλαγής του φωτισμού και του ουρανού ανάλογα με την ώρα της ημέρας. Η αλλαγή από το ένα είδος ουρανού προς το άλλο γίνεται με την βοήθεια ενός shader ο οποίος επιτρέπει το blending μεταξύ δυο skyboxes. Ακόμη, η ένταση του φωτός είναι ανάλογη της ώρας της ημέρας, η οποία έχει χωριστεί σε αριθμό από κομμάτια.

Επίσης, έχει υλοποιηθεί τόξο το οποίο επιδεικνύει την κατεύθυνση (pitch, yaw) και την ταχύτητα του αέρα. Η κατεύθυνση του τόξου εξαρτάται από το rotation του χαρακτήρα μας στο κόσμο. Επίσης, η ταχύτητα του αέρα λαμβάνεται υπόψη στους υπολογισμούς της πορείας των βολίδων.

Εικόνα 6.5 Το Χαμάμ η ώρα 23:15

Εικόνα 6.6 Το Χαμαμ η ώρα 19:15

Εικόνα 6.7 Το Χαμαμ η ώρα 18:38



Εικόνα 6.8 Οι ρυθμίσεις των script του καιρού

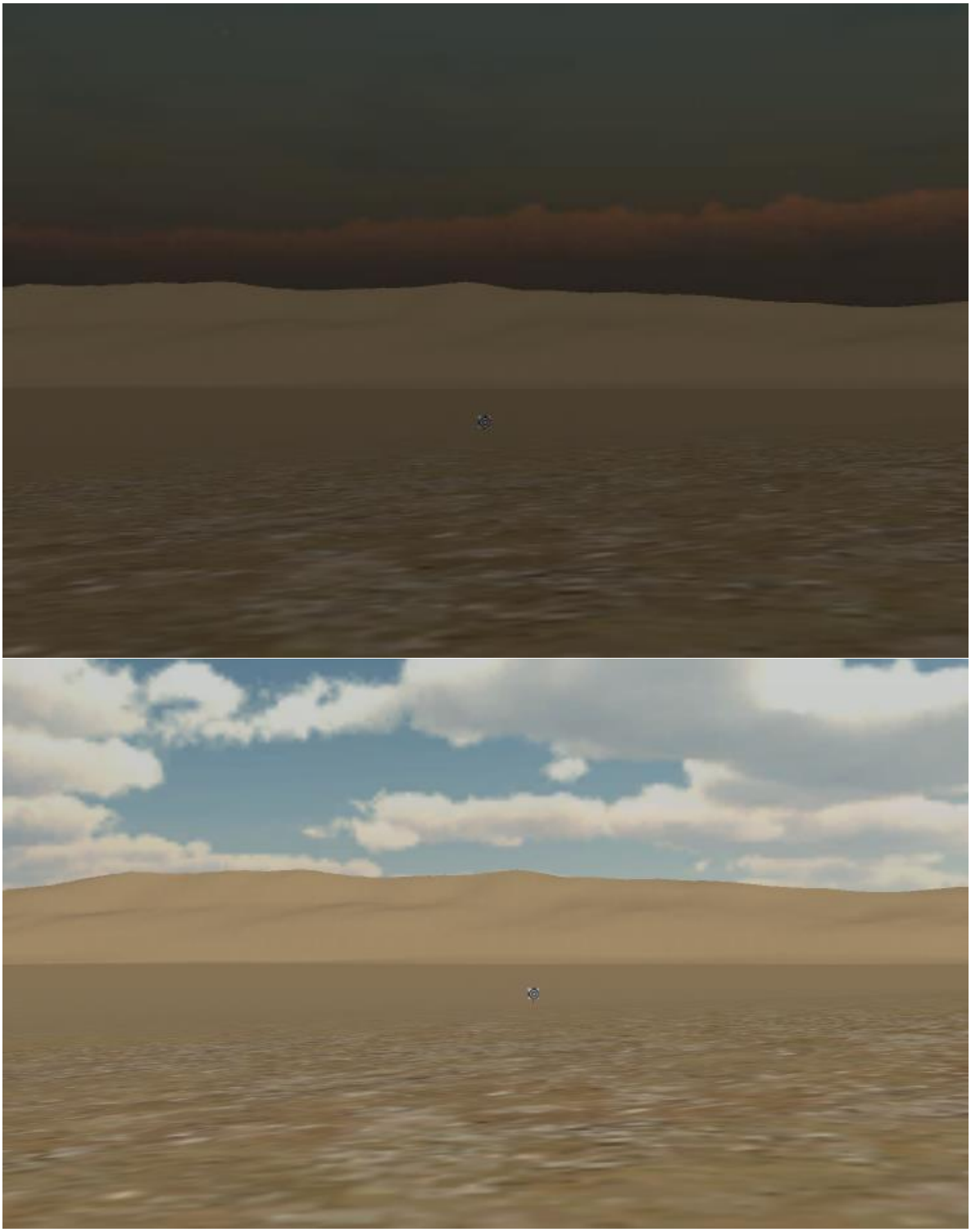
6.6 Σενάρια που έχουν υλοποιηθεί

Μέχρι στιγμής έχουν υλοποιηθεί 2 σενάρια:

Α) Πεδίο βολής για μακρινές βολές

Β) Εσωτερικό πεδίο βολής για βολές μέχρι και 100 μέτρα.

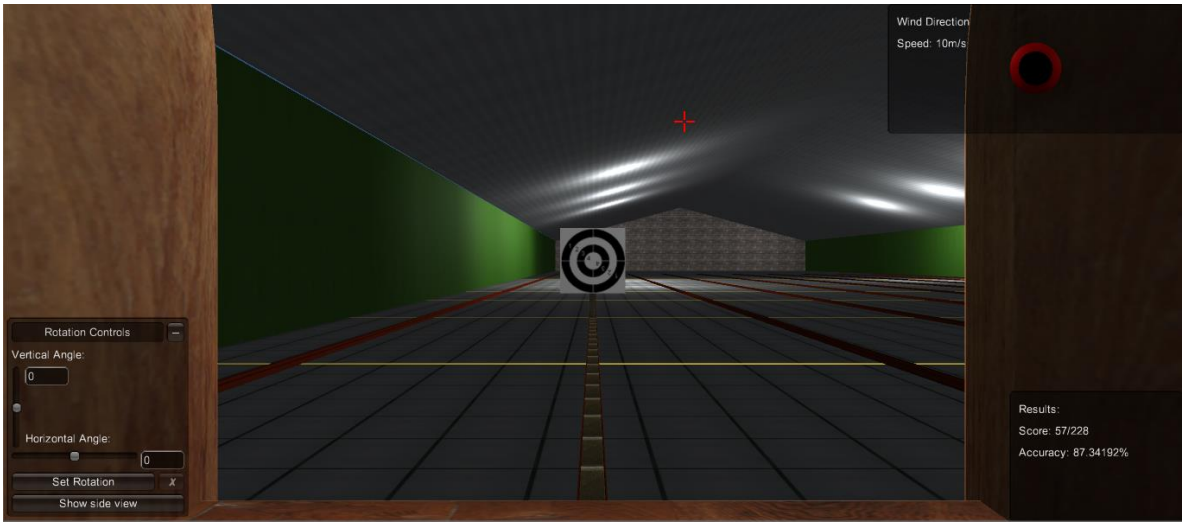
Στο πεδίο για μακρινές βολές ο χρήστης ξαπλώνει πρηνηδόν σε ένα μικρό ύψωμα και σημαδεύει στόχους που βρίσκονται κάτω στη πεδιάδα.



Εικόνα 6.9 Πεδίο μακρινών βολών, στόχος στα 200μέτρα. Η πάνω φωτογραφία πάρθηκε η ώρα 5:15 ενώ η δεύτερη η ώρα 8:15

Εικόνα 6.10 Ο στρατιώτης σε θέση πρηνηδόν

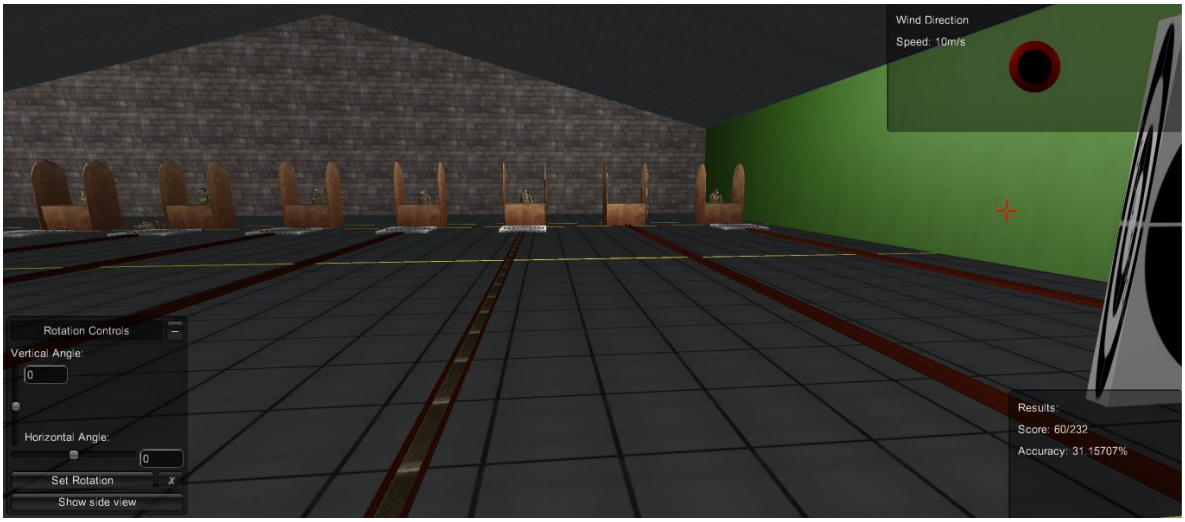
Εσωτερικό πεδίο βολής



Εικόνα 6.11 Εκτέλεση βολής στο εσωτερικό πεδίο βολής, ο στόχος βρίσκεται στα 20 μέτρα μακριά.

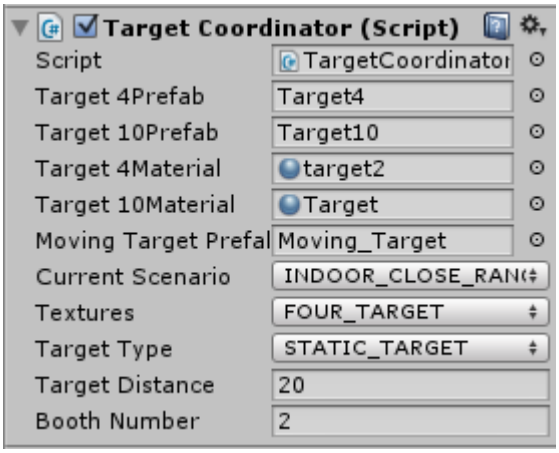
Το εσωτερικό πεδίο βολής μικρών αποστάσεων μπορεί να χωρέσει μέχρι και 10 στρατιώτες που να εκτελούν ταυτόχρονα βολή. Συγκεκριμένα, παρέχει 10 booths, το κάθε ένα από τα

οποία περιέχει το δικό του κινούμενο στόχο ο οποίος μπορεί να ρυθμιστεί ανάλογα με τις απαιτήσεις το χρήστη.



Εικόνα 6.12 Εσωτερικό πεδίο βολής μικρών αποστάσεων.

Η ρύθμιση του κάθε σεναρίου γίνεται από τον TargetCoordinator, ο οποίος αναλαμβάνει να ντύσει, να τοποθετήσει και να ρυθμίσει το στόχο που θα χρησιμοποιείς ανάλογα με τις ρυθμίσεις που του θέτεις.



Εικόνα 6.13 Ρυθμίσεις του Target Coordinator

Κεφάλαιο 7

Ανάλυση Αποτελεσμάτων

7.1 Μελέτη δυνάμεων

Μετά τη υλοποίηση των εξισώσεων των δυνάμεων που ασκούνται στις βολίδες, δοκίμασα μερικά απλά τεστ για να ελέγξω την ορθότητα τους.

Στο τέστ που ακολουθεί, είχα αρχικά ρίξει μια βολίδα στην οποία ασκείτο μόνο η δύναμη της βαρύτητας. Στη συνέχεια, δοκίμασα να ρίξω βολίδες στις οποίες θα ασκούνταν οι διάφορες δυνάμεις που είχα υλοποιήσει. Οι βολές που έγιναν είχαν μηδενική γωνιά αναχώρησης.

Τα ακόλουθα είναι τα αποτελέσματα μιας τέτοιας βολής:

Εικόνα 7.1 Αποτελέσματα βολής ελέγχου ορθότητας δυνάμεων

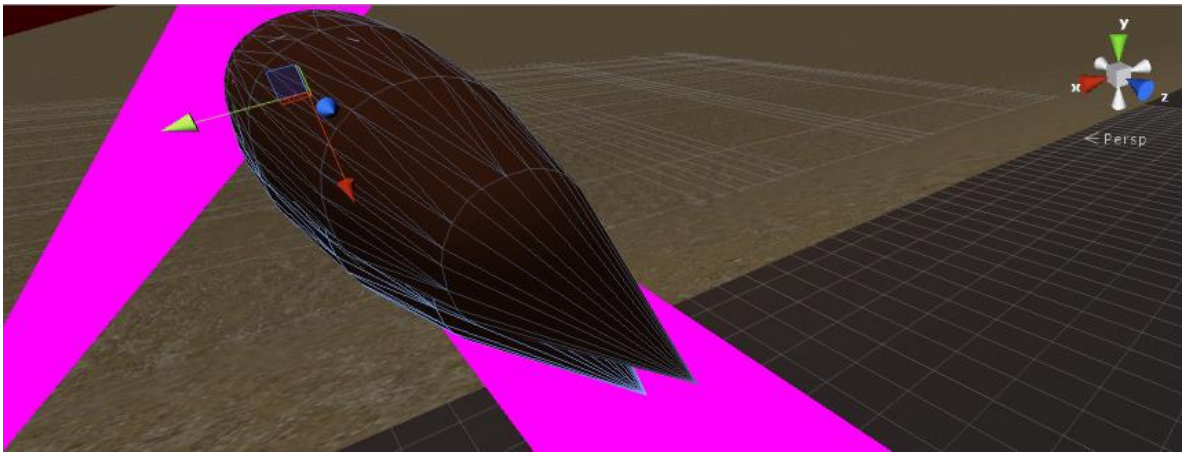
Στην βολίδα 1, ασκείτο μόνο η δύναμη της βαρύτητας και παρατηρούμε μια σταδιακή μικρή κλίση προς τα κάτω.

Στην βολίδα 2, εκτός από τη δύναμη της βαρύτητας, ασκείτο και η δύναμη της αντίστασης του αέρα. Από την εξίσωση της δύναμης αντίστασης του αέρα, ξέρουμε πως είναι μια δύναμη η οποία ασκείται αντίθετη ως προς τη κίνηση της βολίδας. Αυτό εξηγεί την απότομη κλίση προς τα κάτω και το γεγονός ότι η δεύτερη βολίδα διάνυσε πολύ μικρότερη απόσταση.

Στην βολίδα 3, όπου ασκείτο η δύναμη άνωσης, παρατηρούμε πως είχε ταξιδέψει πιο μακριά από τη βολίδα 1. Αυτό οφείλεται στο γεγονός ότι η δύναμη άνωσης ασκεί δύναμη που τείνει να ανυψώσει ή να καθηλώσει τη βολίδα ανάλογα με τη γωνιά που σχηματίζει η μύτη της.

Εικόνα 7.2 Βαρύτητα vs Magnus

Στην εικόνα 7.2, παρατηρούμε πως η βολίδα στην οποία ασκείτο η δύναμη Magnus έχει βγεί εκτός τροχιάς σε σχέση με τη βολίδα όπου ασκείτο μόνο η βαρύτητα. Αυτό είναι αναμενόμενο καθώς η δύναμη Magnus τείνει να κινήσει την βολίδα προς το πλάι, λόγω της διαφοράς πίεσης που δημιουργεί η κυκλική κίνηση της βολίδας στα σωματίδια του αέρα που την περιτριγυρίζουν. Η διαφορά μεταξύ των δύο σημείων ξεπερνά τα 30 εκατοστά και μιλάμε μόνο για μια δύναμη!

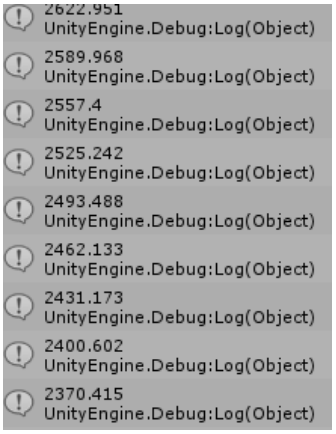


Εικόνα 7.3 Βαρύτητα vs Pitch Damping Force

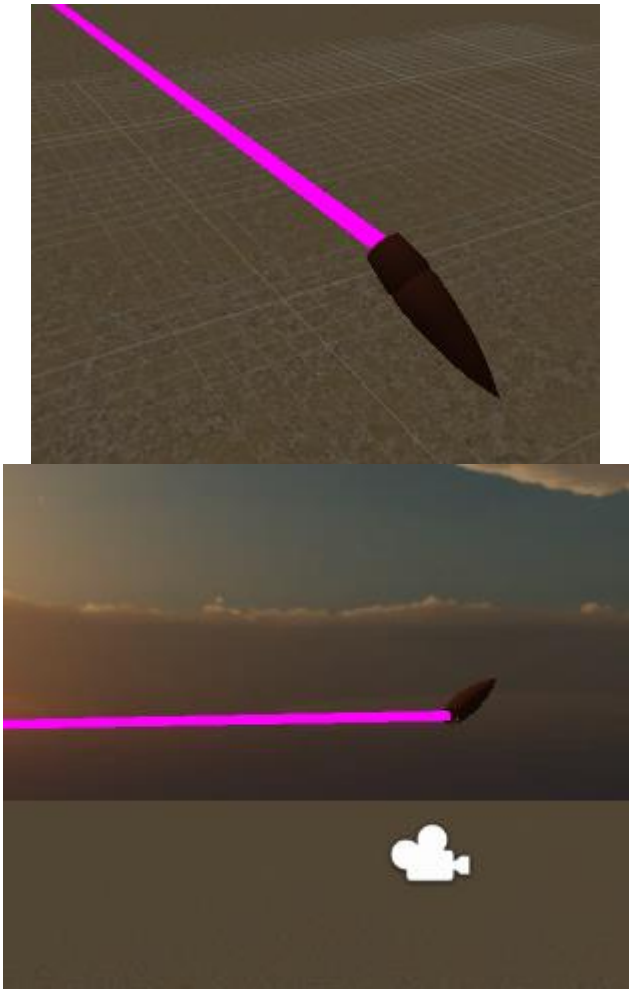
Στην εικόνα 7.3, παρατηρούμε πως οι δυο βολίδες σχεδόν εφάπτονται, αλλά όχι πλήρως, καθώς η δύναμη απόσβεσης κλίσης έχει αλλάξει την κλίση της δεύτερης βολίδας.

7.2 Μελέτη Ροπών

Για να ελέγξουμε την ροπή απόσβεσης ταχύτητας περιστροφής (spin damping moment), παρακολουθήσαμε το μέτρο της γωνιακής ταχύτητας της βολίδας μας. Τα αποτελέσματα ήταν προφανές.



Εικόνα 7.4 Μείωση γωνιακής ταχύτητας λόγω της ροπής απόσβεσης γωνιακής ταχύτητας

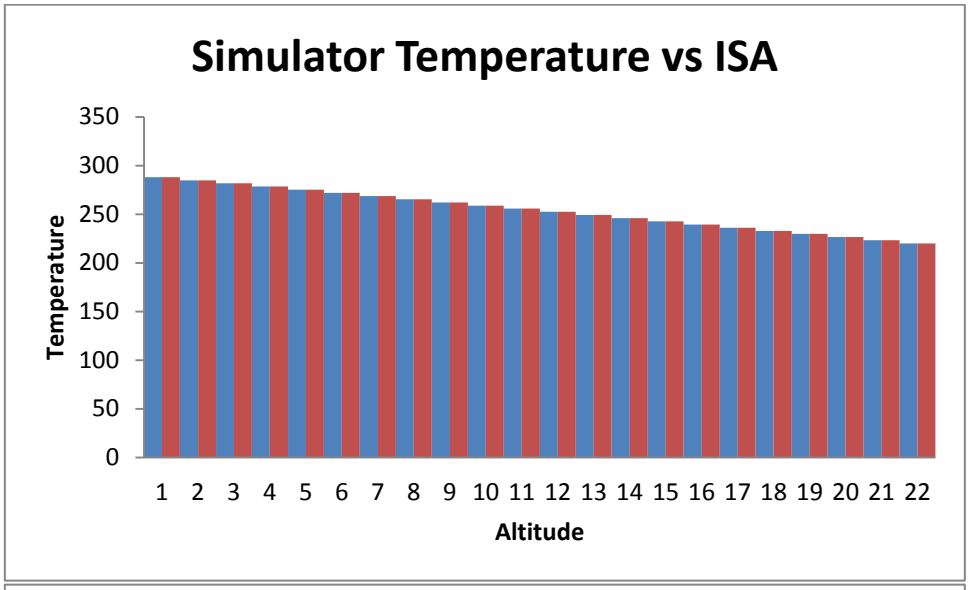


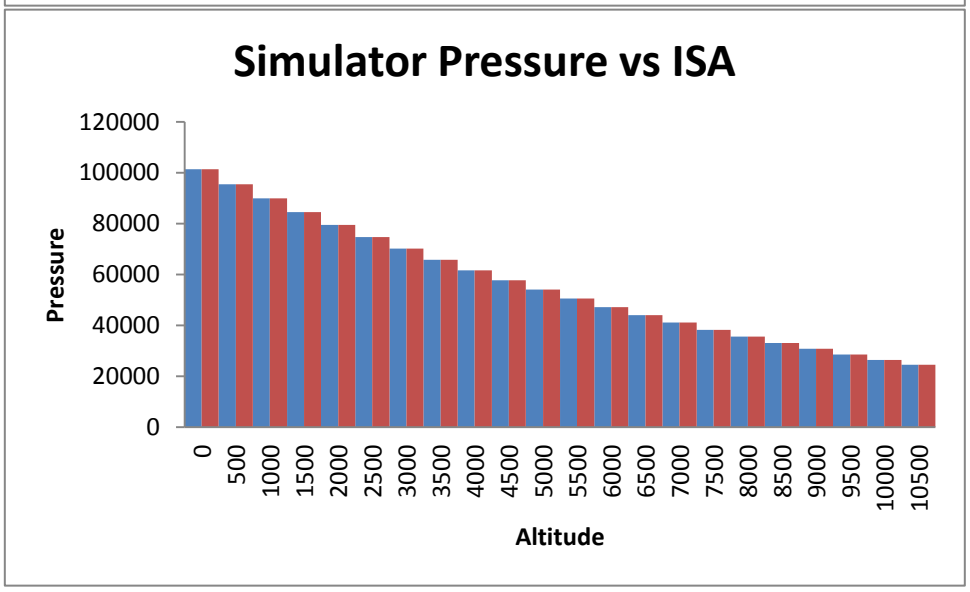
Εικόνα 7.5 Ροπή ανατροπής

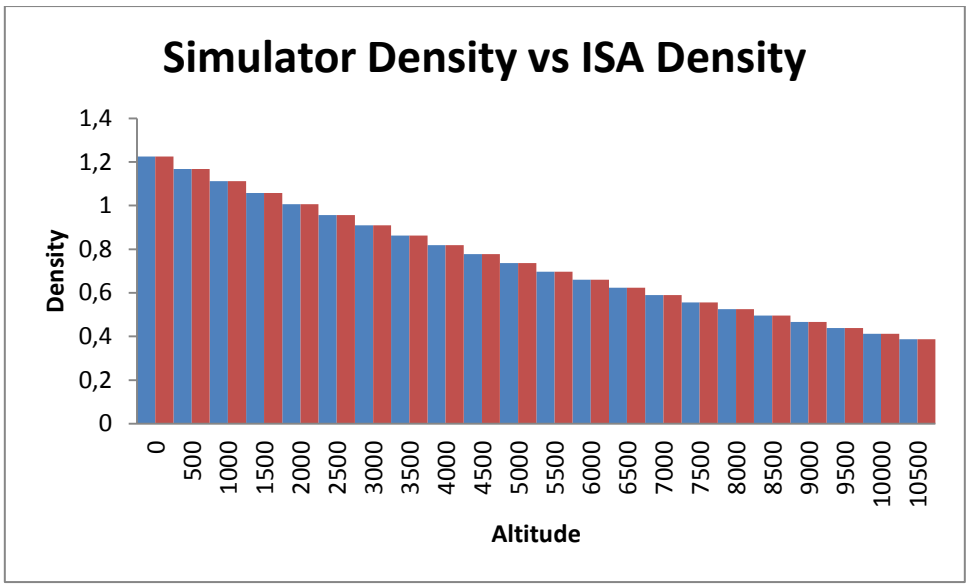
Μελετώντας την ροπή ανατροπής, βρίσκουμε τη βολίδα να ανατρέπεται.

Δυστυχώς δεν μπορούμε να πούμε το ίδιο και για τις ροπές Magnus και Pitch Damping, όπου δυστυχώς δεν μπορέσαμε να εντοπίσουμε κάποια σημαντική διαφορά ή απόκλιση στη κίνηση.

7.3 Σύγκριση Ατμόσφαιρας Εξομοιωτή με τα αναμενόμενα δεδομένα της ISA







Εικόνα 7.6 Τα αποτελέσματα της ατμόσφαιρας που έχουμε υλοποιήσει είναι σχεδόν τα ίδια με αυτά που αναμένουμε από το μοντέλο ISA.

Κεφάλαιο 8

Επίλογος

8.1 Συμπεράσματα

Μέσα από την παρούσα διπλωματική εργασία καταφέραμε να προσομοιώσουμε τόσο τις δυνάμεις που ασκούνται στις βολίδες κατά την πτήση τους, όσο και την ατμόσφαιρα η οποία τις προκαλεί. Επίσης, δημιουργήθηκαν μερικά απλά, αλλά ρεαλιστικά σενάρια τα οποία θα μπορούν να χρησιμοποιηθούν και σε μετέπειτα στάδια. Επιπρόσθετα, έχει επιτευχθεί η σύνδεση με το σύστημα motion detection το οποίο σχεδιάστηκε από την φοιτήτρια Μαρία Παρασκευά. Επίσης, έχει δημιουργηθεί ένα MySQL - PHP framework το οποίο θα μπορεί να υποστηρίξει την χρήση του συστήματος από πολλά άτομα.

8.2 Μελλοντικό έργο

Η συγκεκριμένη διπλωματική εργασία αποτελεί ισχυρή βάση για τη ρεαλιστική προσομοίωση ενός πεδίου βολής. Μέσω της περαιτέρω ανάπτυξης του συστήματος και τον εμπλουτισμό της προσομοίωσης τότε θα μπορούμε να παρέχουμε μια πιο ρεαλιστική και σωστή εμπειρία στους χρήστες. Ένα χαρακτηριστικό παράδειγμα βελτίωσης του συστήματος αυτού αφορά τον σωστό υπολογισμό των συντελεστών των διαφόρων δυνάμεων που ασκούνται κατά τη πτήση μιας βολίδας.

Βιβλιογραφία

1. McCoy, R.L., *Modern Exterior Ballistics*. 2nd Edition ed. 2012: Schiffer Publishing.
2. McCoy, R.L., *"MC DRAG" - A computer program for estimating the drag coefficients of projectiles*. 1981, US Army Ballistic Research Laboratory.
3. Standardization, I.O.f., *ISO 2533*. 1975: International Organization for Standardization.
4. Picard, A., et al., *Revised formula for the density of moist air (CIPM-2007).* Metrologia, 2008. **45**(2): p. 149.
5. Wagner, W. and A. Pruss, *International equations for the saturation properties of ordinary water substance. Revised according to the international temperature scale of 1990. Addendum to J. Phys. Chem. Ref. Data 16, 893 (1987).* Journal of Physical and Chemical Reference Data, 1993. **22**(3): p. 783-787.

Παράρτημα Α

Οδηγός χρήσης υπάρχοντος PHP framework

Καταχώρηση νέου στρατιώτη

Εκτελεί καταχώρηση νέου στρατιώτη στη βάση δεδομένων.

Όνομα αρχείου: register_soldier.php

Τρόπος Επικοινωνίας: POST Request

Input:

- SoldierID
- Name
- Surname
- Rank
- Unit

Output:

- "success":
    - "0" αν η καταχώρηση απέτυχε
    - "1" αν η καταχώρηση ήταν επιτυχής
- "message":
    - Μήνυμα σφάλματος ή επιτυχίας

Τροποποίηση καταχώρησης στρατιώτη

Εκτελεί τροποποίηση της καταχώρησης ενός στρατιώτη στη βάση δεδομένων.

Όνομα αρχείου: modify_soldier.php

Τρόπος Επικοινωνίας: POST Request

Input:

- SoldierID
- Name
- Surname
- Rank
- Unit

Output:

- "success":
    - "0" αν η τροποποίηση απέτυχε
    - "1" αν η τροποποίηση ήταν επιτυχής
- "message":
    - Μήνυμα σφάλματος ή επιτυχίας

Λήψη όλων των καταχωρημένων στρατιωτών.

Εκτελεί λήψη όλων των καταχωρημένων στρατιωτών.

Όνομα αρχείου: get_soldiers.php

Τρόπος Επικοινωνίας: POST Request

Input:

No inputs

Output:

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχής

"soldiers"[]:

- Αντικείμενα τύπου soldier
  - "SoldierID"
  - "Name"
  - "Surname"
  - "Rank"
  - "Unit"

Έλεγχος αν ένας στρατιώτης είναι καταχωρημένος.

Ελέγχει κατά πόσο ένας στρατιώτης είναι καταχωρημένος

Όνομα αρχείου: soldier_exists.php

Τρόπος Επικοινωνίας: POST Request

Input:

SoldierID

Output:

"success":

- "0" αν το ερώτημα απέτυχε ή ο στρατιώτης δεν υπάρχει

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχής και ο στρατιώτης υπάρχει.

"message":

- Μήνυμα επιτυχίας

Λήψη σεναρίων

Εκτελεί λήψη όλων των καταχωρημένων σεναρίων.

Input:

No inputs

Output:

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχής

"scenarios"[]:

- Αντικείμενα τύπου scenario
  - "ScenarioID"
  - "ScenarioName"

Καταχώρηση νέας βολής

Εκτελεί καταχώρηση μιας νέας βολής.

Όνομα αρχείου: create_shooting_range_event.php

Τρόπος Επικοινωνίας: POST Request

Input:

ScenarioID

TargetDistance

Units[] τα ονόματα των μονάδων πρέπει να είναι χωρισμένα με κόμμα

Output:

"success":

- "0" αν η καταχώρηση απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν η καταχώρηση ήταν επιτυχής

" shootingrangeevent":

- Αντικείμενο τύπου ShootingRangeEvent
  - "ID"
  - "DayTime"
  - "ScenarioID"
  - "TargetDistance"
  - "Units"[]

Λήψη καταχωρημένων βολών

Λήψη όλων των βολών που είναι καταχωρημένες στη βάση δεδομένων

Όνομα αρχείου: get_shooting_range_events.php

Input:

No input

Output:

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα εκτελέστηκε επιτυχώς

" shootingrangeevents"[]:

- Αντικείμενα τύπου ShootingRangeEvent
  - "ID"
  - "DayTime"
  - "ScenarioID"
  - "TargetDistance"
  - "Units"[]

Καταχώρηση αποτελεσμάτων βολής ενός στρατιώτη

Εκτελεί καταχώρηση των αποτελεσμάτων ενός στρατιώτη

Όνομα αρχείου: create_soldier_result.php

Τρόπος Επικοινωνίας: POST Request

Input

EventID

SoldierID

Score

Accuracy

Output

"success":

- "0" αν η καταχώρηση απέτυχε ή "1" αν πέτυχε

"message":

- Μήνυμα σφάλματος ή επιτυχίας

Λήψη των στρατιωτών που έλαβαν μέρος σε μια συγκεκριμένη βολή

Λήψη στρατιωτών που έλαβαν μέρος σε μια συγκεκριμένη βολή

Όνομα αρχείου: get_shooting_range_event_participants.php

Τρόπος Επικοινωνίας: POST Request

Input

EventID

Output

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχές

"soldiers"[]:

- Αντικείμενα τύπου Soldier
  - "SoldierID"
  - "Name"
  - "Surname"
  - "Rank"
  - "Unit"

Λήψη καταχωρημένων αποτελεσμάτων ενός στρατιώτη για όλες τις βολές του

Λήψη όλων των αποτελεσμάτων βολών που αφορούν τον συγκεκριμένο στρατιώτη

Όνομα αρχείου: get_soldier_results.php

Τρόπος Επικοινωνίας: POST Request

Input

SoldierID

Output

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχές

"results"[]:

- Αντικείμενα τύπου Result
  - "EventID"
  - "ScenarioName"
  - "TargetDistance"
  - "Time"
  - "Score"
  - "Accuracy"

Λήψη καταχωρημένων αποτελεσμάτων ενός στρατιώτη για συγκεκριμένη βολή

Λήψη αποτελεσμάτων ενός στρατιώτη για μια συγκεκριμένη βολή

Όνομα αρχείου: get_soldier_results_for_event.php

Τρόπος Επικοινωνίας: POST Request

Input

SoldierID

EventID

Output

"success":

- "0" αν το ερώτημα απέτυχε

"message":

- Μήνυμα σφάλματος

Ή

"success":

- "1" αν το ερώτημα ήταν επιτυχές

"results"[]:

- Αντικείμενα τύπου Result
  - "EventID"
  - "ScenarioName"
  - "TargetDistance"
  - "Time"
  - "Score"
  - "Accuracy"

Annex

Δημιουργία ενός σεναρίου

Για τη δημιουργία ενός σεναρίου, στη σκηνή θα πρέπει να υπάρχουν τα πιο κάτω αντικείμενα (prefabs ή GameObjects με επισυνημμένα scripts):

1. **Ο χαρακτήρας τον οποίο θα χειριζόμαστε**: Αυτό μπορεί να είναι μοντέλο χαρακτήρα ή και απλά μια κάμερα που να αιωρείται στον αέρα. Ο χαρακτήρας μας θα πρέπει να έχει επιλεγμένο το tag "Player" έτσι ώστε να μπορούν να τον ανακαλύψουν τα διάφορα scripts που θα έχουμε στη σκηνή μας.

2. **Το όπλο**: Αυτό πρέπει να είναι επισυνημμένο πάνω στη κάμερα του χαρακτήρα μας και να φέρει τα παρακάτω scripts / components / tags:
    - **a.** Weapon Behaviour
        i. Στην επιλογή chambered bullet, τοποθετήστε το bullet prefab.
    - **b.** Ray Casting
        i. Στη μεταβλητή Network θα πρέπει να μπει σύνδεσμος προς το script Network Manager, το οποίο συνδέεται με το motion capture system.
        ii. Επιλέξτε Use network για να μπορείτε να χειριστείτε το όπλο με σύστημα motion capture.
    - **c.** Audio Source
    - **d.** Tag "Weapon"

3. **Την ατμόσφαιρα και τον έλεγχο του καιρού:** Δημιουργήστε ένα κενό GameObject και μετονομάστε το σε Weather Control. Πάνω σ'αυτό το GameObject, τοποθετήστε τα scripts / components / tag:
    a. Time of Day (προαιρετικό)
        i. Σημείωση: Στη σκηνή σας θα πρέπει να υπάρχει ένα directional light με το όνομα και tag "Sun".
        ii. Αν δεν θέλετε να το χρησιμοποιήσετε, δεν είναι απαραίτητο, απλά προσθέστε ένα δικό σας skybox από τα Render Settings και θα είναι εντάξει.
        iii. Σε περίπτωση που θέλετε να το χρησιμοποιήσετε, βάλτε τα materials με τα ομώνυμα ονόματα στα slots των skybox και τα ανάλογα blended skybox materials στη θέση των transitions. Αν δυσκολεύεστε να βρείτε τα αρχεία, τα ονόματα των μεταβλητών είναι 1-1 τα ονόματα των αρχείων.
        iv. Στα transition times μπορείτε να καθορίσετε πότε ξεκινά και πότε τελειώνει το κάθε κομμάτι της ημέρας. Αν παρατηρήσετε πως δε δουλεύει σωστά τότε το πιο πιθανό είναι πως είναι πρόβλημα του script.
    b. Atmosphere (drag and drop)
    c. Wind Controller
        i. Αυτό το script χειρίζεται το πάνω δεξιά τόξο που καθορίζει την κατεύθυνση του αέρα. Στο slot wind direction arrow, τοποθετήστε το τόξο το οποίο θα τοποθετήσουμε στο βήμα **4.**
    d. "WeatherControl" tag!

4. **Κάμερα παρουσίασης κατεύθυνσης αέρα:** Δημιουργήστε μια νέα camera (αφαιρέστε τα audio listener, κλπ) και ρυθμίστε την ως ακολούθως:
    a. Clear flags: Depth only
    b. Culling Mask: WindDirectionArrow
    c. Projection: Orthographic

d. Size: 0.03

e. Clipping Planes: 0.0 και 0.03

f. Depth: 1

g. Viewport Rect: X = 0.75, Y = 0.75, W = 1, H = 1

h. Κάντε drag and drop πάνω της το τόξο, έτσι ώστε η κάμερα να γίνει γονιός του τόξου. Στο τόξο θα πρέπει να τοποθετήσετε το layer (πάνω δεξιά) WindDirectionArrow.

i. Μετακινήστε το τόξο μέχρις ότου βρίσκεται στο κέντρο της κάμερας μας.

j. Μη ξεχάσετε να τοποθετήσετε το τόξο στο slot του βήματος *3.c.i*

5. **Score Manager:** Δημιουργήστε ένα κενό GameObject και ονομάστε το ScoreManager. Βάλτε του το tag ScoreManager και τοποθετήστε πάνω του το script … ScoreManager. Αυτό το αντικείμενο θα είναι υπεύθυνο για τον υπολογισμό του τελικού score και accuracy.

6. **Motion Tracking:** Δημιουργήστε ένα κενό GameObject και ονομάστε το MotionTracking. Πάνω σ'αυτό το GameObject, κάντε attach το script NetworkManager και Chessboard. Θα πρέπει να κάνετε drag and drop το ίδιο το αντικείμενο στο slot Chessboard του Network Manager.

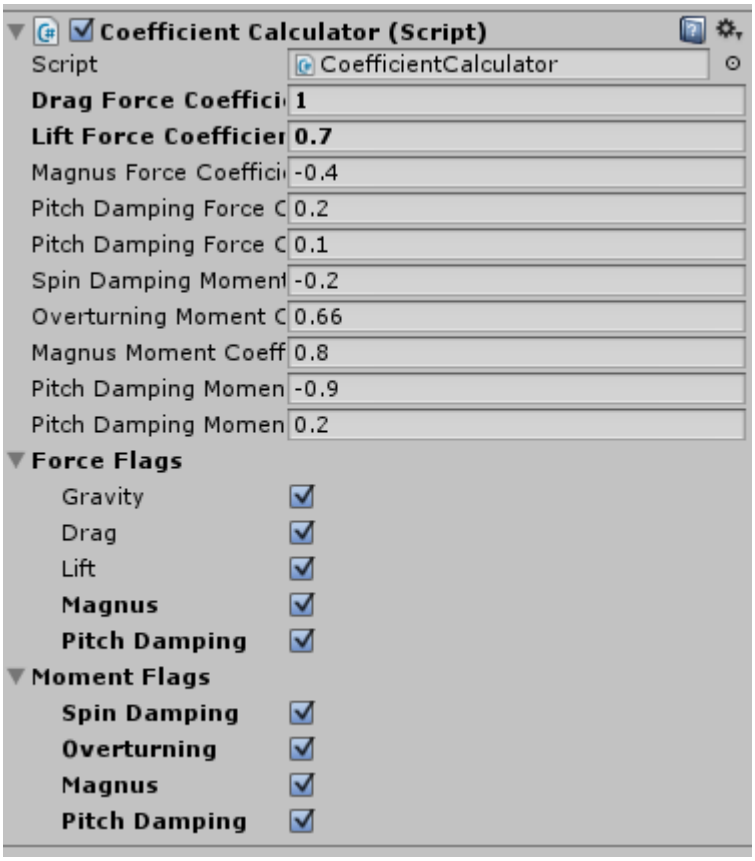
7. **Keyboard Controls:** Δημιουργήστε ένα κενό GameObject και ονομάστε το KeyboardControls. Κάντε attach πάνω του το script Controls.cs.

    a. Μπορείτε να πιέζετε το κουμπί R για να διαγράφετε όλες τις βολίδες από τη σκηνή

8. **GUI Controls:** Δημιουργήστε ένα κενό GameObject και ονομάστε το GUIControls. Πάνω του, τοποθετήστε τα ακόλουθα:
    a. **Σημείωση:** Θα χρειαστεί να παίξετε λίγο με τα Rects των GUI για να τοποθετηθούν σωστά.
    b. RotationGUI για να μπορείτε να θέτετε συγκεκριμένη γωνιά.
        i. Εδώ θα πρέπει να δημιουργήσετε μια κάμερα, να της βάλετε το tag SideViewCamera, να της κάνετε attach το script SideCameraBehaviour και να την τοποθετήσετε στο slot του RotationGUI. Επίσης, θα πρέπει να συνδέσετε την main camera σας με το Rotation GUI.
    c. Crosshair για να μπορείτε να δείτε που στοχεύετε
        i. Μην ξεχάσετε να κάνετε attach το crosshair texture.
    d. Wind Camera GUI
        i. Εδώ συνδέεται η κάμερα που δείχνει το τόξο του αέρα καθώς και το WeatherControl.
    e. ScoreGUI
        i. Εδώ συνδέεται ο ScoreManager για να σας παρουσιάζει στο σκορ στην οθόνη
9. **Coefficients:** Το τελευταίο και πιο σημαντικό GameObject στο παιχνίδι, καθώς χωρίς αυτό δε θα μπορούσαμε να κάνουμε βολές. Σου παρέχει τη δυνατότητα να αλλάξεις τα coefficients που χρησιμοποιούνται για τις βολές.

Συναρτήσεις των δυνάμεων που ασκούνται στις βολίδες

Οι συναρτήσεις που χρησιμοποιούνται για τον υπολογισμό των δυνάμεων που ασκούνται στις βολίδες βρίσκονται στο αρχείο BulletBehaviour.cs. Για τον έλεγχο των συντελεστών που χρησιμοποιούνται καθώς και για τη δυνατότητα να ενεργοποιούμε/απενεργοποιούμε τις δυνάμεις που θέλουμε, μπορούμε να χρησιμοποιήσουμε το αντικείμενο Coefficients ή Coefficient Calculator. Πιο κάτω παρουσιάζεται η διεπαφή που μας παρέχει το script αυτό:



Διεπαφή του Coefficient Calculator, αν και το όνομα του είναι παραπλανητικό, θα μπορούσε σε μετέπειτα στάδια να υπολογίζει όντως τους διάφορους συντελεστές.

Κώδικας Συναρτήσεων των Δυνάμεων

```
/*************************************************************************/
// FORCES
/*************************************************************************/

   Vector3 dragForce(float airDensity, Vector3 velocity, float dragCoefficient)
   {
       return -0.5f * airDensity * characteristics.dragReferenceArea *
                                                dragCoefficient * velocity * velocity.magnitude;
   }

   Vector3 liftForce(float airDensity, Vector3 velocity, float liftCoefficient)
   {
      return 0.5f * airDensity * characteristics.dragReferenceArea * liftCoefficient * Vector3.Cross(velocity,
Vector3.Cross(transform.forward, velocity));
   }

   Vector3 magnusForce(float airDensity, Vector3 velocity, float axialSPin, float magnusCoefficient)
   {
       return 0.5f * airDensity * velocity.magnitude * characteristics.dragReferenceArea * (axialSPin) *
characteristics.getCalibreInMeters() * magnusCoefficient * Vector3.Cross(velocity.normalized,
transform.forward);
   }

   Vector3 pitchDampingForce(float airDensity, Vector3 velocity, float yawDampingCoefficient, float
pitchDampingCoefficient, Vector3 deltaX, Vector3 deltaI)
   {
      return 0.5f * airDensity * characteristics.dragReferenceArea * characteristics.getCalibreInMeters() *
velocity.magnitude * (pitchDampingCoefficient * (deltaX / Time.fixedDeltaTime) + yawDampingCoefficient
* ((deltaX / Time.fixedDeltaTime) - (deltaI / Time.fixedDeltaTime)));
   }
```

```
/*************************************************************************************/
// MOMENTS
/*************************************************************************************/

Vector3 spinDampingMoment(float airDensity, Vector3 velocity, float axialSpin, float spinDampingCoefficient)
{
    return 0.5f * airDensity * velocity.magnitude * characteristics.dragReferenceArea * characteristics.getCalibreInMeters() * axialSpin * characteristics.getCalibreInMeters() * spinDampingCoefficient * transform.forward;
}

Vector3 overturningMoment(float airDensity, Vector3 velocity, float overturningMomentCoefficient)
{
    return 0.5f * airDensity * characteristics.dragReferenceArea * characteristics.getCalibreInMeters() * overturningMomentCoefficient * velocity.sqrMagnitude * Vector3.Cross(velocity.normalized, transform.forward);
}

Vector3 magnusMoment(float airDensity, Vector3 velocity, float axialSpin, float magnusMomentCoefficiet)
{
    return 0.5f * airDensity * characteristics.dragReferenceArea * characteristics.getCalibreInMeters() * velocity.magnitude * characteristics.dragReferenceArea * axialSpin * characteristics.getCalibreInMeters() * magnusMomentCoefficiet * Vector3.Cross(transform.forward, Vector3.Cross(velocity.normalized, transform.forward));
}

Vector3 pitchDampingMoment(float airDensity, Vector3 velocity, float yawDampingCoefficient, float pitchDampingCoefficient, Vector3 deltaX, Vector3 deltaI)
{
    return 0.5f * airDensity * characteristics.dragReferenceArea * characteristics.getCalibreInMeters() * characteristics.getCalibreInMeters() * velocity.magnitude * (pitchDampingCoefficient * Vector3.Cross(transform.forward, deltaX / Time.fixedDeltaTime) + yawDampingCoefficient * (Vector3.Cross(transform.forward, deltaX / Time.fixedDeltaTime) - Vector3.Cross(transform.forward, deltaI / Time.fixedDeltaTime)));
}
```