



ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση παιχνιδιού Εικονικής Πραγματικότητας στο
Unity.

Μελέτη της ευχρηστίας και της αλληλεπίδρασης με
τους χρήστες μέσω κίνησης, ήχου και δόνησης.

Του φοιτητή
Καλλέα Γεωργίου
Αρ. Μητρώου: 185383

Επιβλέπων
Κοκκώνης Γεώργιος
Βαθμίδα: Επίκουρος Καθηγητής

Ημερομηνία 16/05/2026

Τίτλος Π.Ε. Υλοποίηση παιχνιδιού Εικονικής Πραγματικότητας στο Unity. Μελέτη της ευχρηστίας και της αλληλεπίδρασης μεταξύ των χρηστών μέσω κίνησης, οι ήχου και δόνησης.

Κωδικός Π.Ε. 24123

Ονοματεπώνυμο φοιτητή Καλλέας Γεώργιος

Ονοματεπώνυμο εισηγητή Κοκκώνης Γεώργιος

Ημερομηνία ανάληψης Π.Ε. 14/02/2024

Ημερομηνία περάτωσης Π.Ε. 16/05/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Καλλέα Γεωργίου που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητα και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Στον καθηγητή μου κ. Κοκκώνη

Πρόλογος

Τα ηλεκτρονικά παιχνίδια αποτελούν σίγουρα έναν από τα πιο ενδιαφέροντα πεδία εφαρμογής της επιστήμης των υπολογιστών. Παρ' όλο που ο βασικός τους στόχος είναι η ψυχαγωγία, επιστημονικά, αποτελούν μία τις κύριες ατμομηχανές της εξέλιξης τόσο του λογισμικού (software) όσο και του υλικού (hardware).

Πολύ συχνά, ο τομέας των ηλεκτρονικών παιχνιδιών ενσωματώνει διάφορες τεχνολογίες που, κάποιες φορές, προκαλούν άλματα στην παραπάνω εξέλιξη. Μία από αυτές είναι και η Εικονική Πραγματικότητα (Virtual Reality). Παρέχοντας μία σειρά από διάφορες τεχνολογίες, θέτει σε νέα βάση την εμπειρία χρήστη που μπορεί να προσφέρει ένα ηλεκτρονικό παιχνίδι.

Η τρέχουσα Πτυχιακή Εργασία θέτει τις βάσεις για το ξεκίνημα ενός "ταξιδιού" στο χώρο των ηλεκτρονικών παιχνιδιών με όχημα την Εικονική Πραγματικότητα. Σκοπεύοντας στην δημιουργία ενός πραγματικού ηλεκτρονικού παιχνιδιού, με βάση την Εικονική Πραγματικότητα, μελετά την ευχρηστία και την αλληλεπίδραση με τους χρήστες μέσω κίνησης, ήχου και δόνησης.

Περίληψη

Η Εικονική Πραγματικότητα (VR), ως τεχνολογία, αντικαθιστά τον πραγματικό κόσμο με υπολογιστικά δημιουργημένα περιβάλλοντα όπου οι χρήστες βιώνουν και αλληλοεπιδρούν με ρεαλιστικό τρόπο. Τα χαρακτηριστικά της VR, όπως είναι η εμβύθιση και οι προηγμένες δυνατότητες αλληλεπίδρασης, την καθιστούν ιδανική για ηλεκτρονικά παιχνίδια, μετατρέποντας την απλή ψυχαγωγία σε καθηλωτική εμπειρία.

Η τρέχουσα Πτυχιακή Εργασία (Π.Ε.) αποτελεί μία πρακτική μελέτη για την απόδειξη της παραπάνω θέσης μέσω της ανάπτυξης ενός ηλεκτρονικού παιχνιδιού στην πλατφόρμα Unity. Το παιχνίδι αφορά την οδήγηση αγωνιστικού αυτοκινήτου σε πίστα ταχύτητας, με έμφαση στο ερώτημα κατά πόσο ο χειρισμός και η εμπειρία βελτιώνονται όταν χρησιμοποιούνται VR τεχνολογίες. Η αλληλεπίδραση γίνεται με τα γνωστά VR χειριστήρια και εμπλουτίζεται με ερεθίσματα που μόνο η VR μπορεί να προσφέρει (δηλ., κίνηση, ήχο και δονήσεις).

Οι κύριοι στόχοι της Π.Ε. είναι: i) η εμβάθυνση στις VR τεχνολογίες μέσω της κατασκευής ενός απαιτητικού παιχνιδιού, ii) η μελέτη των στοιχείων που προσδίδουν εμβύθιση, διαδραστικότητα και τηλεπαρουσία και iii) η εκτίμηση της αύξησης της ευχρηστίας και της βελτίωσης της εμπειρίας χρήστη όταν χρησιμοποιούνται VR αλληλεπιδράσεις με κίνηση, ήχο και δόνηση.

Virtual Reality game development in Unity.

Usability and user interaction study through motion, sound and vibration.

Kalleas Georgios

Abstract

Virtual Reality (VR), as a technology, replaces the real world with computer-generated environments in which users can experience and interact in realistic ways. VR features, such as immersion and advanced interaction capabilities, make it ideal for video games, transforming simple entertainment into a compelling experience.

The current thesis is a practical study aimed at demonstrating the above claim by developing a video game on the Unity platform. The game involves driving a racing car on a speed track and focuses on whether control and the overall experience improve when VR technologies are used. Interaction is performed with standard VR controllers and is enriched by stimuli that only VR can provide (i.e., motion, sound, and haptics).

The main objectives of the D.E. are: (i) to deepen knowledge of VR technologies through the creation of a demanding game, (ii) to study the elements that confer immersion, interactivity, and telepresence, and (iii) to evaluate the increase in usability and the improvement of user experience when VR interactions involving motion, sound, and haptics are employed.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Κοκκώνη για την δυνατότητα, που μου έδωσε, να εργαστώ πάνω στο θέμα της τρέχουσας Πτυχιακής Εργασίας.

Περιεχόμενα

Πρόλογος.....	v
Περίληψη	vi
Abstract	vii
Ευχαριστίες	viii
Περιεχόμενα	ix
Κατάλογος Σχημάτων	xiii
Κατάλογος Πινάκων	xiv
Συνοτομογραφίες.....	xv
Κεφάλαιο 1ο: Εισαγωγή	1
1.1 Θέμα Π.Ε.	1
1.2 Σκοπός Π.Ε.	1
1.3 Στόχοι Π.Ε.	1
Κεφάλαιο 2ο: Ανασκόπηση Βιβλιογραφίας.....	3
2.1 Εικονική Πραγματικότητα (VR)	3
2.1.1 Ιστορική Αναδρομή	3
2.1.2 Εικονικά περιβάλλοντα και εικονικοί κόσμοι	3
2.1.3 Εκτεταμένη, μικτή και επαυξημένη πραγματικότητα	4
2.1.4 Βασικά Χαρακτηριστικά Εικονικής Πραγματικότητας	4
2.1.5 Τεχνολογίες Εικονικής Πραγματικότητας.....	5
2.1.6 Εφαρμογές Εικονικής Πραγματικότητας	7
2.2 Παιχνίδια και Εικονική Πραγματικότητα	8
2.2.1 Πλεονεκτήματα Χρήσης της Εικονικής Πραγματικότητας στην Κατασκευή Παιχνιδιών.....	9
2.2.2 Τεχνικές και μέθοδοι.....	9
2.2.3 Εμπειρία Χρήστη	9
Κεφάλαιο 3ο: Ανάλυση Απαιτήσεων	11
3.1 Λειτουργικές Απαιτήσεις (Functional Requirement)	11
3.1.1 Γενικές Απαιτήσεις	11
3.1.2 Απαιτήσεις Οδήγησης.....	11
3.1.3 Εμπειρία Χρήστη	11
3.2 Μη-λειτουργικές Απαιτήσεις (Non-Functional Requirement).....	12

3.2.1	Απαιτήσεις Ανάπτυξης.....	12
3.2.2	Απαιτήσεις Εκτέλεσης	12
3.2.3	Επικοινωνία με τον Χρήστη	13
3.2.4	Απαιτήσεις Παραμετροποίησης.....	13
Κεφάλαιο 4ο:	Σχεδίαση Συστήματος.....	14
4.1	Αρχές Σχεδίασης	14
4.2	Η Πλατφόρμα Ανάπτυξης Παιχνιδιών Unity.....	14
4.2.1	Εργαλεία.....	14
4.2.2	Υπηρεσίες.....	15
4.3	Ο Μικροελεγκτής ESP32.....	15
4.3.1	Χρήσεις του μικροελεγκτή ESP32.....	15
4.3.2	Ανάπτυξη ηλεκτρονικών κυκλωμάτων με βάση τον μικροελεγκτή ESP32	16
4.3.3	Προγραμματισμός του μικροελεγκτή ESP32	17
4.3.4	Περιφερειακές συσκευές.....	18
4.4	Αρχιτεκτονική Συστήματος	21
4.4.1	Η Unity εφαρμογή VRacer.....	22
4.4.2	Αυτοσχέδιο τιμόνι.....	24
Κεφάλαιο 5ο:	Υλοποίηση	29
5.1	Unity εφαρμογή VRacer	29
5.1.1	Εργαλεία Ανάπτυξης.....	29
5.1.2	Δομή Εφαρμογής	30
5.1.3	Unity Αντικείμενα	31
5.1.4	Ειδικοί Τύποι Δεδομένων.....	39
5.1.5	Κώδικας Εφαρμογής.....	47
5.2	Αυτοσχέδιο Τιμόνι	52
5.2.1	Ηλεκτρονικό Κύκλωμα	52
5.2.2	Κώδικας.....	56
5.2.3	Κατασκευή πρωτοτύπου	59
5.3	Προσομοίωση Λειτουργίας.....	60
5.3.1	Προσομοίωση Ηλεκτρονικού Κυκλώματος Τιμονιού	60
5.3.2	Προσομοίωση Λειτουργίας Παιχνιδιού	65
Κεφάλαιο 6ο:	Εκτέλεση.....	72
6.1	Διάθεση Εφαρμογής σε Εκτελέσιμη Μορφή	72
6.1.1	Προφίλ Έκδοσης.....	72
6.1.2	Παραγωγή Έκδοσης.....	72

6.1.3	Στοιχεία Εγκατάστασης μίας Έκδοσης	73
6.2	Εγκατάσταση Παιχνιδιού.....	73
6.2.1	Προαπαιτούμενα Εγκατάστασης	73
6.2.2	Διαδικασία Εγκατάστασης	73
6.2.3	Προαπαιτούμενα Εκτέλεσης.....	74
6.2.4	Παραμετροποίηση.....	74
6.3	Εκτέλεση Παιχνιδιού	76
6.3.1	Έναρξη Παιχνιδιού	77
6.3.2	Οδήγηση.....	77
6.3.3	Τερματισμός Παιχνιδιού	78
6.4	Σενάρια Εκτέλεσης.....	79
6.4.1	Εκκίνηση Παιχνιδιού	79
6.4.2	Επιτάχυνση	80
6.4.3	Αλλαγή Κατεύθυνσης – Ομαλή Αριστερή Στροφή	81
6.4.4	Αλλαγή Κατεύθυνσης – Απότομη Δεξιά Στροφή.....	82
6.4.5	Εκτός Πίστας – Απότομη Δεξιά Στροφή.....	83
6.4.6	Πλαγιολίσθηση	84
6.4.7	Διάσχιση Γέφυρας.....	85
6.4.8	Πέρασμα από Σαμαράκι.....	87
6.4.9	Πρόσκρουση σε Εμπόδιο	88
Κεφάλαιο 7ο:	Συμπεράσματα και Προτάσεις Βελτίωσης.....	90
7.1	Επίτευξη Σκοπού Π.Ε.....	90
7.2	Επίτευξη Στόχων Π.Ε.	90
7.2.1	Εντρύφηση στις VR Τεχνολογίες	90
7.2.2	Μελέτη των Βασικών Χαρακτηριστικών της Εικονικής Πραγματικότητας	90
7.2.3	Εκτίμηση της Αύξησης της Ευχρηστίας μίας Εφαρμογής με τη Χρήση των VR Τεχνολογιών	91
7.2.4	Εκτίμηση της Θετικής Επίδρασης των VR Τεχνολογιών στην Εμπειρία Χρήστη	91
7.3	Αξιολόγηση Εμπειρίας Χρήστη	91
7.3.1	Ανάλυση αποτελεσμάτων ανά Χαρακτηριστικό	92
7.3.2	Ανάλυση αποτελεσμάτων ανά Ποιότητα	93
7.3.3	Αξιολόγηση αποτελεσμάτων.....	94
7.4	Μελλοντικές Προκλήσεις	95
ΒΙΒΛΙΟΓΡΑΦΙΑ.....		96
ΠΑΡΑΡΤΗΜΑ Α: Κώδικας της κλάσης CarController		101

ΠΑΡΑΡΤΗΜΑ Β: Κώδικας της κλάσης CarXROriginFollow	126
ΠΑΡΑΡΤΗΜΑ Γ: Προγραμματισμός του Ηλεκτρονικού Κυκλώματος του Τιμονιού	128
ΠΑΡΑΡΤΗΜΑ Δ: Κώδικας του Python script VRacer_Arduino_Simulator_Wifi.py.....	159
ΠΑΡΑΡΤΗΜΑ Ε: Κώδικας του Python script VRacer_Arduino_Simulator_Serial_Bridge.py.....	161
ΠΑΡΑΡΤΗΜΑ ΣΤ: Ερωτηματολόγια Εμπειρίας Χρήστη	169

Κατάλογος Σχημάτων

Σχήμα 2.1: Το συνεχές πραγματικότητας-εικονικότητας [16]	4
Σχήμα 2.2: Το πρωτότυπο για οθόνες κεφαλής της Oculus Rift [26]	6
Σχήμα 2.3: Ενσύρματα γάντια με οπτικές ίνες [28]	6
Σχήμα 2.4: Το δεξιό Oculus Χειριστήριο Αφής	6
Σχήμα 2.5: Ένα ενδεικτικό CAVE περιβάλλον [28]	7
Σχήμα 2.6: Αρχιτεκτονική συστήματος προσομοίωσης μάχης [32]	8
Σχήμα 4.7 Διάγραμμα βασικών λειτουργιών ESP32 [51]	16
Σχήμα 4.8 Ενδεικτική πλακέτα ανάπτυξης ηλεκτρονικών κυκλωμάτων	16
Σχήμα 4.9 Ενδεικτικό διάγραμμα ακροδεκτών (pin) του ESP32 [51]	17
Σχήμα 4.10 Προγραμματισμός του ESP32 με το Ολοκληρωμένο Περιβάλλον Ανάπτυξης ESP-IDF [53]	18
Σχήμα 4.11 Ένα τυπικό Γυροσκόπιο οκτώ (8) ακροδεκτών	19
Σχήμα 4.12 Ένα τυπικό Joystick πέντε (5) ακροδεκτών	20
Σχήμα 4.13 Μια τυπική Συσκευή Παραγωγής Δονήσεων	21
Σχήμα 5.14: Η εφαρμογή VRacer_Unity στον Unity Editor	29
Σχήμα 5.15: Η δομή της εφαρμογής VRacer όπως αυτή φαίνεται στον Unity Editor	30
Σχήμα 5.16: Το Αγωνιστικό Αυτοκίνητο της εφαρμογής	31
Σχήμα 5.17: Το Περιβάλλον του παιχνιδιού	33
Σχήμα 5.18: Η πίστα ταχύτητας του παιχνιδιού	34
Σχήμα 5.19: Γέφυρα ανισόπεδου κόμβου	34
Σχήμα 5.20: Τα Σαμαράκια της πίστας του παιχνιδιού	35
Σχήμα 5.21: Τα Σηθαία της πίστας του παιχνιδιού	35
Σχήμα 5.22: Οι Πύλες πάνω από την πίστα του παιχνιδιού	36
Σχήμα 5.23: Χαρακτηριστικά δέντρα του εξωτερικού περιβάλλοντος του παιχνιδιού	37
Σχήμα 5.24: Χαρακτηριστικό Κιόσκι του εξωτερικού περιβάλλοντος του παιχνιδιού	37
Σχήμα 5.25: Χαρακτηριστικές Τέντες του εξωτερικού περιβάλλοντος του παιχνιδιού	38
Σχήμα 5.26: Οι Κερκίδες του εξωτερικού περιβάλλοντος του παιχνιδιού	38
Σχήμα 5.27: Χαρακτηριστικές Διαφημιστικές Πινακίδες	39
Σχήμα 5.28: Χαρακτηριστικός Πυργίσκος παρατήρησης/τηλεόρασης	39
Σχήμα 5.29: Τα συστατικά ενός τρισδιάστατου διανύσματος [60]	40
Σχήμα 5.30: Διάγραμμα ηλεκτρονικού κυκλώματος τιμονιού με Γυροσκόπιο	53
Σχήμα 5.31: Διάγραμμα ηλεκτρονικού κυκλώματος τιμονιού με Joystick	55
Σχήμα 5.32: Το πρωτότυπο ηλεκτρονικό κύκλωμα του τιμονιού με Γυροσκόπιο	59
Σχήμα 5.33: Το αυτοσχέδιο "τιμόνι" του παιχνιδιού VRacer στην τελική του μορφή	60
Σχήμα 5.34: Περιβάλλον προσομοίωσης ηλεκτρονικού κυκλώματος τιμονιού στο Wokwi	61
Σχήμα 5.35: Προσομοίωση ηλεκτρονικού κυκλώματος τιμονιού με Γυροσκόπιο	62
Σχήμα 5.36: Προσομοίωση ηλεκτρονικού κυκλώματος τιμονιού με Joystick	63
Σχήμα 5.37: Σενάριο ελέγχου μετάδοσης κίνησης με Γυροσκόπιο μέσω Σειριακής επικοινωνίας	64
Σχήμα 5.38: Σενάριο ελέγχου ενεργοποίησης σειριακής επικοινωνίας σε περίπτωση αποτυχημένης ενεργοποίησης Wi-Fi επικοινωνίας	65
Σχήμα 5.39: Σενάριο ελέγχου χειρισμού της εφαρμογής VRacer_Unity μέσω Wi-Fi	69
Σχήμα 5.40: Σενάριο ελέγχου χειρισμού της εφαρμογής VRacer_Unity μέσω Σειριακής επικοινωνίας	70
Σχήμα 6.41: Τα περιεχόμενα το φακέλου εγκατάστασης του VRacer	73

Σχήμα 6.42: Ένα τυπικό σετ σύγχρονων VR χειριστηρίων της εταιρίας Meta Quest	74
Σχήμα 6.43: Ο βασική οθόνη εργασίας του PuTTY	75
Σχήμα 6.44: Το αγωνιστικό αυτοκίνητο στην αφετηρία	77
Σχήμα 7.45: Διάγραμμα συγκεντρωτικών αποτελεσμάτων ανά χαρακτηριστικό του παιχνιδιού VRacer	93
Σχήμα 7.46: Διάγραμμα συγκεντρωτικών αποτελεσμάτων ανά ποιότητα του παιχνιδιού VRacer	94

Κατάλογος Πινάκων

Πίνακας 4.1: Προσδιορισμός Συντελεστή x με βάση το Roll	26
Πίνακας 4.2: Προσδιορισμός Συντελεστή y με βάση το Pitch	26
Πίνακας 4.3: Προσδιορισμός Συντελεστή x με βάση τη Συντεταγμένη x του Joystick	27
Πίνακας 4.4: Προσδιορισμός Συντελεστή y με βάση τη Συντεταγμένη y του Joystick	27
Πίνακας 5.5: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Γυροσκοπίου	53
Πίνακας 5.6: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Συσκευής Παραγωγής Δονήσεων	54
Πίνακας 5.7: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Joystick	55
Πίνακας 5.8: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Συσκευής Παραγωγής Δονήσεων	56
Πίνακας 6.9: Το Προφίλ Έκδοσης της εφαρμογής VRacer	72
Πίνακας 7.10: Αναλυτική βαθμολόγηση ανά χαρακτηριστικό του παιχνιδιού VRacer	92
Πίνακας 7.11: Συγκεντρωτικά αποτελέσματα ανά χαρακτηριστικό του παιχνιδιού VRacer	93
Πίνακας 7.12: Συγκεντρωτικά αποτελέσματα ανά ποιότητα του παιχνιδιού VRacer	94

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
VR	Virtual Reality
AR	Augmented Reality
MR	Mixed Reality

Κεφάλαιο 1ο: Εισαγωγή

Γενικά, η Εικονική Πραγματικότητα (VR) είναι η τεχνολογία που επιτρέπει την αντικατάσταση του πραγματικού κόσμου από έναν εικονικό. Περιλαμβάνει ένα σύνολο τεχνολογιών που χρησιμοποιούνται για την δημιουργία εικονικών περιβαλλόντων με τη βοήθεια του ηλεκτρονικού υπολογιστή. Μέσα σ' αυτά, οι χρήστες μπορούν να βιώσουν και να αλληλεπιδράσουν όπως ακριβώς θα έκαναν στην πραγματική ζωή [1].

Τα χαρακτηριστικά της Εικονικής Πραγματικότητας (VR) την καθιστούν μια ιδανική τεχνολογία για την κατασκευή ηλεκτρονικών παιχνιδιών. Οι τεχνολογίες και οι εξελιγμένες δυνατότητες αλληλεπίδρασης που προσφέρει δίνουν την δυνατότητα, σε ένα ηλεκτρονικό παιχνίδι, να επεκταθεί πέρα από μία απλή ψυχαγωγία σε μία καθηλωτική εμπειρία, μέσα σε εξαιρετικά διαδραστικούς εικονικούς κόσμους [1].

1.1 Θέμα Π.Ε.

Η τρέχουσα Π.Ε. αποτελεί, ακριβώς, μία μελέτη για την απόδειξη της παραπάνω θέσης. Βασίζεται λοιπόν σε ένα πραγματικό ηλεκτρονικό παιχνίδι που κατασκευάστηκε για το σκοπό αυτό και ονομάζεται VRacer.

Το θέμα του ηλεκτρονικού παιχνιδιού είναι η οδήγηση ενός αγωνιστικού αυτοκινήτου μέσα σε μία συγκεκριμένη πίστα ταχύτητας. Η επιλογή του θέματος βασίζεται στο γεγονός ότι – όπως είναι γενικά αποδεκτό - η οδήγηση μέσα σε μία ασφαλή πίστα ταχύτητας είναι, από μόνη της, μία καθηλωτική εμπειρία.

Πόσο όμως πιο εύκολος θα γινόταν ο χειρισμός του αγωνιστικού αυτοκινήτου αν η αλληλεπίδραση του χρήστη με αυτό βασίζονταν στις τεχνολογίες της Εικονικής Πραγματικότητας;

Επίσης, πόσο πιο έντονη θα γινόταν η εμπειρία χρήστη αν η αλληλεπίδρασή του με το αγωνιστικό αυτό αυτοκίνητο εμπλουτιζονταν με ερεθίσματα που μόνο η Εικονική Πραγματικότητα μπορεί να προσφέρει (δηλ. κίνηση, ήχου και δόνηση);

1.2 Σκοπός Π.Ε.

Ο σκοπός της τρέχουσας Π.Ε. είναι διττός. Κατά βάση, είναι η ανάπτυξη του ηλεκτρονικού παιχνιδιού, που περιγράφεται στην παραπάνω ενότητα (βλ. Ενότητα 1.1), στην πλατφόρμα ανάπτυξης παιχνιδιών Unity. Ταυτόχρονα όμως, είναι και η αξιοποίηση των VR τεχνολογιών (βλ. Ενότητα 2.1.5) για την αλληλεπίδραση του παιχνιδιού με τον χρήστη.

Έτσι, η ανάπτυξη του συγκεκριμένου ηλεκτρονικού παιχνιδιού αποτελεί, ταυτόχρονα, μία μελέτη της αλληλεπίδρασης μίας VR εφαρμογής - με τους χρήστες - μέσω κίνησης, ήχου και δόνησης. Πρακτικά, αυτό σημαίνει ότι ο χειρισμός του αγωνιστικού αυτοκινήτου, που αποτελεί το επίκεντρο του παιχνιδιού, γίνεται με τη χρήση των γνωστών VR χειριστηρίων (βλ. Ενότητα 2.1.5). Επίσης, παράγονται ειδικά ερεθίσματα – προς τον χρήστη - όπως κίνηση, ήχοι και δονήσεις.

1.3 Στόχοι Π.Ε.

Ο βαθύτερος στόχος της τρέχουσας Π.Ε. δεν είναι απλά η κατασκευή μίας VR εφαρμογής αν και, βέβαια, η είσοδος στον επιστημονικό χώρο της Εικονικής Πραγματικότητας είναι, από μόνη της, ένας σημαντικός στόχος.

Πιο συγκεκριμένα, η Π.Ε. στοχεύει στα ακόλουθα:

- Εντρύφηση στις VR τεχνολογίες μέσα από την κατασκευή ενός απαιτητικού ηλεκτρονικού παιχνιδιού που θα τις αξιοποιεί.
- Μελέτη των ιδιαίτερων στοιχείων που προσδίδει, στην εμπειρία χρήστη μίας εφαρμογής, η Εικονική Πραγματικότητα. Τέτοια στοιχεία, όπως αναφέρεται παρακάτω, είναι η εμβύθιση, η διαδραστικότητα και η τηλεπαρουσία (βλ. Ενότητα 2.1.4).
- Εκτίμηση της αύξησης της ευχρηστίας μίας εφαρμογής με τη χρήση των VR τεχνολογιών.
- Εκτίμηση της θετικής επίδρασης των VR τεχνολογιών στην εμπειρία χρήστη μέσω της αλληλεπίδρασης – μίας VR εφαρμογής – με κίνηση, ήχους και δονήσεις.

Κεφάλαιο 2ο: Ανασκόπηση Βιβλιογραφίας

Το κεφάλαιο αυτό ορίζει το θεωρητικό και ακαδημαϊκό υπόβαθρο της τρέχουσας Π.Ε.. Αναφέρεται, επίσης, σε διάφορα ιστορικά στοιχεία και προσδιορίζει τις ιδιαιτερότητες και τις προκλήσεις που επηρεάζουν την επίτευξη των στόχων της Π.Ε..

2.1 Εικονική Πραγματικότητα (VR)

Η Εικονική Πραγματικότητα (Virtual Reality ή VR) προσφέρει μία προσομοίωση του πραγματικού κόσμου. Μέσα σ' αυτόν τον εικονικό κόσμο, ο χρήστης έχει την αίσθηση ότι κινείται αλληλοεπιδρώντας με διάφορα εικονικά αντικείμενα [2].

Ειδικότερα, τα συστήματα εικονικής πραγματικότητας, βασίζονται στην δημιουργία ρεαλιστικών εικόνων, ήχων, δονήσεων και άλλων ερεθισμάτων που δίνουν την αίσθηση της φυσικής παρουσίας του χρήστη σε ένα εικονικό περιβάλλον. Για να το πετύχουν αυτό, χρησιμοποιούν ακουστικά και ειδικές τρισδιάστατες οθόνες που, συνήθως, βρίσκονται κοντά στα μάτια του χρήστη-θεατή [2].

Γενικά, οι εφαρμογές εικονικής πραγματικότητας περιλαμβάνουν την ψυχαγωγία (κυρίως με τη μορφή παιχνιδιών), την εκπαίδευση (όπως ιατρική, εκπαίδευση ή στρατιωτική εκπαίδευση) και επιχειρήσεις (όπως εικονικές συναντήσεις). Έτσι, η εικονική πραγματικότητα αποτελεί μία από τις βασικές τεχνολογίες μέσω των οποίων ο πραγματικός κόσμος μεταφέρεται στον εικονικό [3].

2.1.1 Ιστορική Αναδρομή

Η εικονική πραγματικότητα, ως κλάδος της βιομηχανίας παραγωγής λογισμικού, εμφανίζεται, ουσιαστικά, κατά την περίοδο 1970 – 1990. Οι πρώτες της εφαρμογές αφορούσαν τη στρατιωτική εκπαίδευση, την προσομοίωση πτήσης και τη σχεδίαση αυτοκινήτων [4].

Κατά την περίοδο 1990 – 2000, καθώς οι τεχνολογίες εικόνας και ήχου το επιτρέπουν, εμφανίζονται τα πρώτα παιχνίδια εικονικής πραγματικότητας [5]. Αργότερα, κατά την περίοδο 2000 – 2010, η εικονική πραγματικότητα αρχίζει να γίνεται πλέον διαθέσιμη στο ευρύτερο καταναλωτικό κοινό, μέσω του εμπορίου [6].

Κατά την περίοδο από το 2010 μέχρι και σήμερα σημειώθηκε αλματώδης εξέλιξη στις συσκευές της εικονικής πραγματικότητας. Συγκεκριμένα, αυξήθηκε το οπτικό πεδίο των οθονών και, ταυτόχρονα, μειώθηκαν οι παραμορφώσεις κατά την παραγωγή της κίνησης [7]. Επίσης, ανακαλύφθηκαν οθόνες που έκαναν δυνατή την εμφάνιση των αντικειμένων χωρίς καθυστέρηση και χωρίς κηλίδες [8].

Ως αποτέλεσμα των παραπάνω εξελίξεων, σχεδόν όλες οι μεγάλες εταιρίες πληροφορικής ανέπτυξαν νέα προϊόντα εικονικής πραγματικότητας και δημιούργησαν αποκλειστικές γραμμές παραγωγής [9] [10].

2.1.2 Εικονικά περιβάλλοντα και εικονικοί κόσμοι

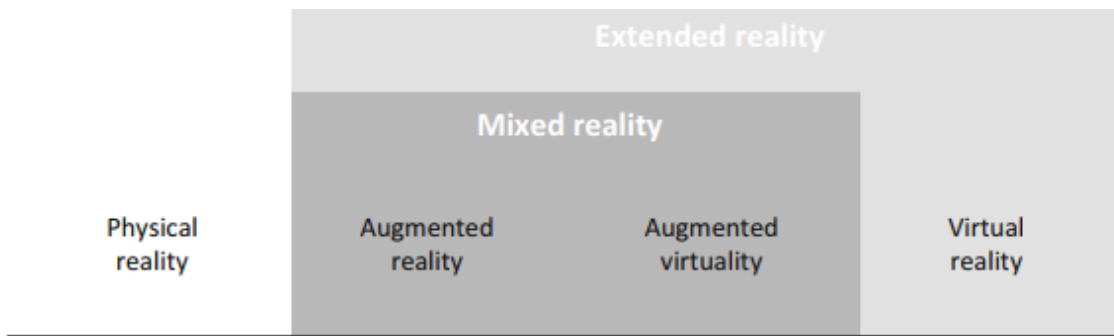
Στο πλαίσιο της εικονικής πραγματικότητας, χρησιμοποιούνται συχνά οι όροι *εικονικό περιβάλλον* (virtual environment) και *εικονικός κόσμος* (virtual world). Ένα εικονικό περιβάλλον δημιουργείται με τη βοήθεια ενός λογισμικού και αποτελείται από αναπαραστάσεις πραγματικών (ή φανταστικών) αντικειμένων και διαδικασιών. Επιπλέον, ο χρήστης του λογισμικού αλληλοεπιδρά με τα παραπάνω αντικείμενα και “συμμετέχει” στις διαδικασίες [11].

Ένας *εικονικός κόσμος* είναι ένας συγκεκριμένος τύπος εικονικού περιβάλλοντος στον οποίο έχουν συνδεθεί πολλοί χρήστες, χρησιμοποιώντας ο καθένας το επιθυμητό *είδωλο* (avatar). Ο εικονικός κόσμος παρέχει κοινόχρηστους, προσομοιωμένους χώρους οι οποίοι κατοικούνται και διαμορφώνονται από τους κατοίκους τους (δηλαδή τα είδωλα των χρηστών) [12].

Γενικά, παρ' όλο που αρκετές εφαρμογές εικονικής πραγματικότητας προσφέρουν την δυνατότητα στους χρήστες τους να δημιουργούν είδωλα, οι εφαρμογές που δημιουργούν εικονικά περιβάλλοντα και εικονικούς κόσμους δε βασίζονται στις τεχνολογίες της εικονικής πραγματικότητας [13].

2.1.3 Εκτεταμένη, μικτή και επαυξημένη πραγματικότητα

Οι συγκεκριμένοι όροι σχετίζονται, κατά μία έννοια, με την εικονική πραγματικότητα αλλά έχουν ξεκάθαρα διαφορετικό νόημα [14] (βλ. Σχήμα 2.1). Πιο συγκεκριμένα, η *εκτεταμένη πραγματικότητα* (extended reality ή XR) περιέχει, ως όρος, όλες εκείνες τις "πραγματικότητες" που ξεπερνούν τα όρια της φυσικής πραγματικότητας (physical reality). Πρόκειται, δηλαδή, για τεχνητές πραγματικότητες που συνήθως δημιουργούνται τους διαφόρους κλάδους και τεχνολογίες πληροφορικής [15].



Σχήμα 2.1: Το συνεχές πραγματικότητας-εικονικότητας [16]

Ειδικότερα, όπως φαίνεται και στο παραπάνω σχήμα η εκτεταμένη πραγματικότητα συμπεριλαμβάνει τη μικτή πραγματικότητα και την εικονική πραγματικότητα. Η *μικτή πραγματικότητα* (mixed reality ή MR), με τη σειρά της, είναι ένας επίσης γενικός όρος που συμπεριλαμβάνει την επαυξημένη πραγματικότητα και την επαυξημένη εικονικότητα [17].

Η *επαυξημένη πραγματικότητα* (augmented reality ή AR) προκύπτει από τον εμπλουτισμό των φυσικών πληροφοριών με ψηφιακές, σε πραγματικό χρόνο. Οι ψηφιακές πληροφορίες παράγονται, με βάση διάφορες τεχνολογίες, από ειδικές συσκευές [18].

Αντιθέτως, η *επαυξημένη εικονικότητα* (augmented virtuality ή AV) προκύπτει από τεχνητούς κόσμους που εμπλουτίζονται (επαυξάνονται) με στοιχεία του φυσικού κόσμου (δηλαδή, πρακτικά, εικόνες ή βίντεο που προέρχονται προερχόμενα από το φυσικό κόσμο) [19].

2.1.4 Βασικά Χαρακτηριστικά Εικονικής Πραγματικότητας

Η εικονική πραγματικότητα προσδίδει στην εμπειρία χρήσης μίας εφαρμογής τρία βασικά χαρακτηριστικά: *εμβύθιση* (immersion), *διαδραστικότητα* (interactivity) και *τηλεπαρουσία* (telepresence) [20].

2.1.4.1 Εμβύθιση (immersion)

Όταν κάποιος διαβάζει ένα μυθιστόρημα και γοητεύεται τόσο πολύ από αυτό αισθάνεται πλήρως απορροφημένος από την πλοκή του. Η εμβύθιση, λοιπόν, περιγράφει μία τέτοια κατάσταση. Κατά μια έννοια, σημαίνει την αναστολή κάθε δυσπιστίας ως προς ένα μέσο [21].

Γενικά, με βάση το βαθμό εμβύθισης, υπάρχουν τρεις κατηγορίες συστημάτων εικονικής πραγματικότητας [22]:

- Συστήματα χωρίς εμβύθιση

Πρόκειται για απλοϊκά συστήματα χαμηλού κόστους.

- Συστήματα πλήρους εμβύθισης

Πρόκειται για συστήματα που προσφέρουν στον χρήστη την απόλυτη εμπειρία ενός εικονικού κόσμου χρησιμοποιώντας υψηλής ποιότητας γραφικά και συσκευές υψηλών επιδόσεων.

- Συστήματα ημι-εμβύθισης

Πρόκειται για συστήματα που βρίσκονται εκτός των δύο παραπάνω κατηγοριών. Χαρακτηριστικό παράδειγμα αυτής της κατηγορίας είναι οι προσομοιωτές πτήσης.

Γενικά, τα συστήματα της κατηγορίας αυτής συνδυάζουν λογισμικό υψηλής απόδοσης με στερεοσκοπική όραση, αυξημένο οπτικό πεδίο και απτική ανατροφοδότηση.

2.1.4.2 Διαδραστικότητα (interactivity)

Η διαδραστικότητα αναφέρεται στην δυνατότητα των χρηστών να αλληλοεπιδρούν με ένα εικονικό περιβάλλον, σε πραγματικό χρόνο [23].

2.1.4.3 Τηλεπαρουσία (telepresence)

Κυριολεκτικά, ο όρος τηλεπαρουσία σημαίνει απομακρυσμένη παρουσία. Δηλαδή, το να νιώθεις ότι βρίσκεσαι σε ένα μέρος το οποίο διαφέρει από τη φυσική σου θέση. Γενικά, αποτελεί ένα από τα βασικότερα χαρακτηριστικά της εικονικής πραγματικότητας και σχετίζεται, ιδιαίτερα, με την εμβάθυνση [24].

2.1.5 Τεχνολογίες Εικονικής Πραγματικότητας

Τα συστήματα εικονικής πραγματικότητας βασίζονται σε συγκεκριμένες συσκευές. Στη συνέχεια, περιγράφονται οι σημαντικότερες από αυτές.

2.1.5.1 Οθόνη Κεφαλής (Head-Mounted Display)

Υπάρχουν πλέον πολλές τέτοιες συσκευές στην αγορά. Οι περισσότερες διαθέτουν στερεοσκοπικές οθόνες και συστήματα παρακολούθησης επιτρέποντας στον χρήστη να βλέπει τρισδιάστατες εικόνες μέσω ενός μεγάλου πεδίου όρασης (βλ. Σχήμα 2.2). Συνήθως, για κάθε μάτι υπάρχει μία ξεχωριστή οθόνη η οποία, μέσω του λογισμικού, τροφοδοτείται από μία εικονική κάμερα. Επιπλέον, γυροσκόπια και επιταχυνσιόμετρα κάνουν δυνατή την αναγνώριση θέσης [25].



Σχήμα 2.2: Το πρωτότυπο για οθόνες κεφαλής της Oculus Rift [26]

2.1.5.2 Ενσύρματα Γάντια

Τα ενσύρματα γάντια είναι ικανά να μετρούν γωνίες, πίεση και απτική ανάδραση. Υπάρχουν τρεις κύριες τεχνολογίες που χρησιμοποιούνται στο ενσύρματο γάντι: οπτικές ίνες (βλ. Σχήμα 2.3), αγωγίμο μελάνι για τη μέτρηση της ηλεκτρικής αντίστασης και μηχανικοί αισθητήρες [27].



Σχήμα 2.3: Ενσύρματα γάντια με οπτικές ίνες [28]

2.1.5.3 Χειριστήρια Αφής

Τα χειριστήρια αφής (touch controller) αποτελούν μία σχετικά πρόσφατη εξέλιξη στις VR τεχνολογίες [29]. Λειτουργούν ασύρματα χωρίς να διαθέτουν κάποιου είδους καλωδίωση. Διακρίνονται σε αριστερό και δεξί ανάλογα με το – ανθρώπινο - χέρι στο οποίο απευθύνονται. Διαθέτουν μία λαβή για να ελέγχονται από το ανθρώπινο χέρι και, επιπλέον, προσφέρουν ένα ειδικό πιλοτήριο (joystick) για τη μετάδοση κίνησης.



Σχήμα 2.4: Το δεξιό Oculus Χειριστήριο Αφής

2.1.5.4 CAVE

Το CAVE (Cave Automatic Virtual Environment) είναι ένα δωμάτιο εικονικής πραγματικότητας. Ειδικοί προβολείς καλύπτουν τους τοίχους του δωματίου με στερεοσκοπικές εικόνες ενώ ο χρήστης,

φορώντας ειδικά γυαλιά (όπως αυτά που χρησιμοποιούνται κατά την παρακολούθηση τρισδιάστατων ταινιών), μπορεί να τις βλέπει. Ταυτόχρονα, διάσπαρτα ηχεία – μέσα στο δωμάτιο – παράγουν ήχους [30].



Σχήμα 2.5: Ένα ενδεικτικό CAVE περιβάλλον [28]

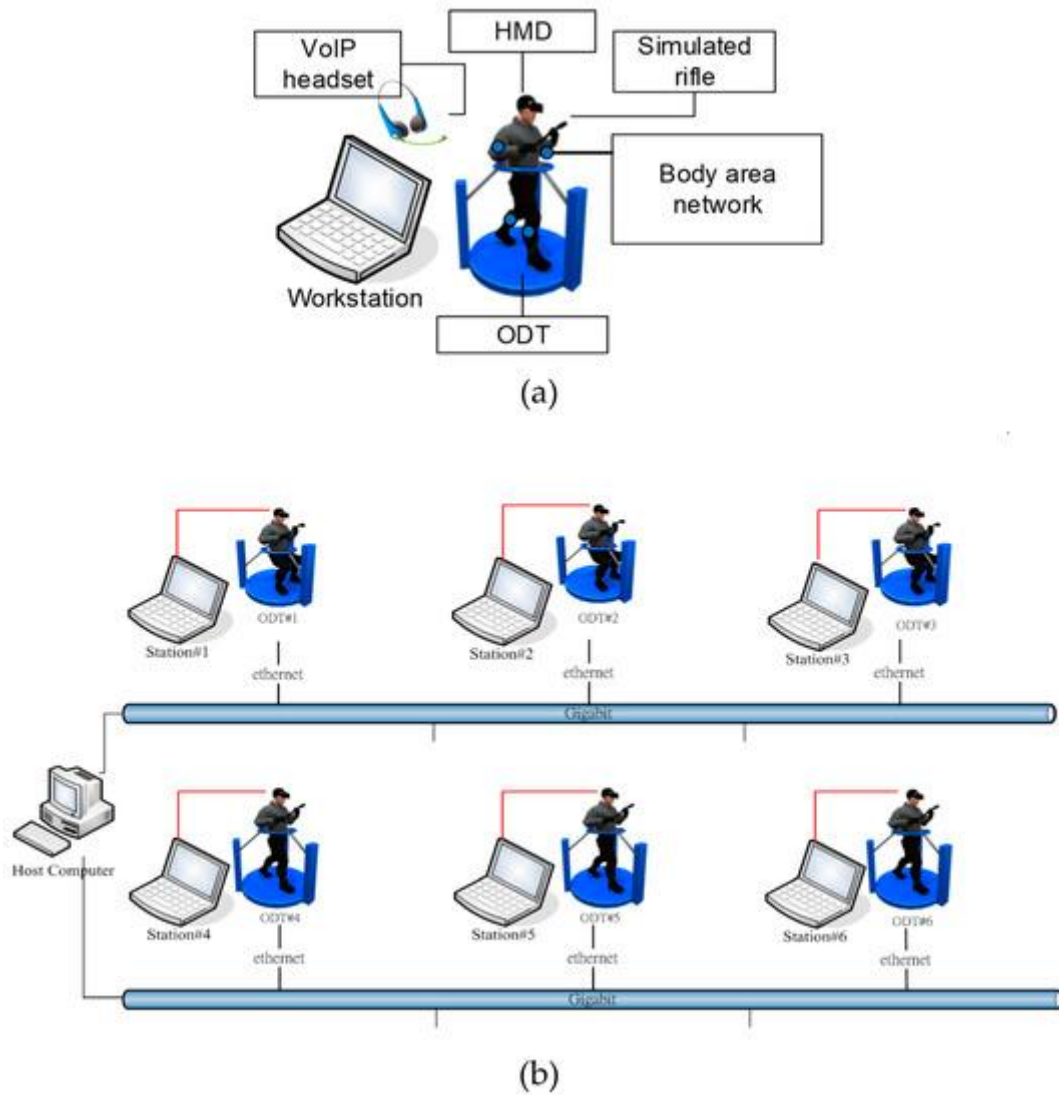
2.1.6 Εφαρμογές Εικονικής Πραγματικότητας

Θεωρητικά, η Εικονική Πραγματικότητα μπορεί να εφαρμοστεί όπου υπάρχει ηλεκτρονικός υπολογιστής. Ωστόσο, οι βασικές περιοχές εφαρμογής της είναι οι ακόλουθες [28].

2.1.6.1 Στρατός

Στο στρατό, η πιο συχνή χρήση της Εικονικής Πραγματικότητας συναντάται στους προσομοιωτές εκπαίδευσης. Πρόκειται για μία μέθοδο εκπαίδευσης που είναι σίγουρα ασφαλέστερη και λιγότερο κοστοβόρα από την πραγματική εκπαίδευση [31]. Ειδικότερα, τα σημαντικότερα πεδία χρήσης είναι οι προσομοιωτές οχημάτων και οι προσομοιωτές πτήσης.

Επιπρόσθετα, η Εικονική Πραγματικότητα χρησιμοποιείται κατά την εκπαίδευση των στρατιωτών με την προσομοίωση μάχης (βλ. Σχήμα 2.6) [32]. Ουσιαστικά, στις εφαρμογές αυτές, η εμπειρία χρήστη μοιάζει πολύ με την εμπειρία χρήστη που προσφέρουν διάφορα δημοφιλή παιχνίδια υπολογιστών.



Σχήμα 2.6: Αρχιτεκτονική συστήματος προσομοίωσης μάχης [32]

2.1.6.2 Ψυχαγωγία

Είναι γεγονός ότι, ακόμα και πριν από την έλευση της Εικονικής Πραγματικότητας, πολλά παιχνίδια υπολογιστών προσπαθούσαν να προσφέρουν – στον χρήστη – τα βασικά χαρακτηριστικά της Εικονικής Πραγματικότητας: εμπύθιση, διαδραστικότητα και τηλεπαρουσία. Ωστόσο, αυτό έγινε πραγματικά εφικτό με τη χρήση των συσκευών που έφερε η Εικονική Πραγματικότητα [33].

2.1.6.3 Εκπαίδευση

Η Εικονική Πραγματικότητα χρησιμοποιείται για εκπαιδευτικούς σκοπούς τόσο στα σχολεία όσο στα πανεπιστήμια και στις εταιρίες [34]. Συνήθως, οι περιοχές στις οποίες εφαρμόζεται σχετίζονται με την ανατομία, τη γεωγραφία, τη φυσική και τη χημεία [35].

2.2 Παιχνίδια και Εικονική Πραγματικότητα

Η Εικονική Πραγματικότητα, εξαιτίας των ιδιαίτερων χαρακτηριστικών της, αποτελεί μία ιδανική τεχνολογία για την κατασκευή παιχνιδιών. Έτσι, η βιομηχανία της ψυχαγωγίας έχει κατασκευάσει πληθώρα εφαρμογών με βάση την Εικονική Πραγματικότητα [36]. Επιπρόσθετα, έχουν κατασκευαστεί

πολλές εφαρμογές που δεν περιορίζονται μόνο στη ψυχαγωγία αλλά επεκτείνονται και σε διαφορετικούς τομείς, κυρίως με σκοπό την εκμάθηση ή την εκπαίδευση μέσω του παιχνιδιού [37].

2.2.1 Πλεονεκτήματα Χρήσης της Εικονικής Πραγματικότητας στην Κατασκευή Παιχνιδιών

Η χρήση της Εικονικής Πραγματικότητας σε ένα παιχνίδι επιτρέπει στους χρήστες να αλληλοεπιδρούν με τους χαρακτήρες και το περιβάλλον του παιχνιδιού. Έτσι, η εμπειρία του παιχνιδιού είναι πιο πλούσια και εξατομικευμένη [38].

Επιπλέον, οι ειδικές συσκευές που χρησιμοποιούν τα παιχνίδια Εικονικής Πραγματικότητας μεταφέρουν στον χρήστη τα ίδια ερεθίσματα που – φαίνεται ότι – δέχεται ο ήρωας του παιχνιδιού. Έτσι, ενώ στα παραδοσιακά παιχνίδια ο χρήστης κατευθύνει τον κεντρικό ήρωα του παιχνιδιού χρησιμοποιώντας κάποιες συσκευές εισόδου (π.χ. το πληκτρολόγιο), με την εικονική πραγματικότητα ο χρήστης βρίσκεται ο ίδιος στη θέση του ήρωα [38].

Με τη βοήθεια της οθόνης-κεφαλής, που προσφέρουν οι τεχνολογίες Εικονικής Πραγματικότητας, ο χρήστης βλέπει και ακούει ότι ο χαρακτήρας του παιχνιδιού. Παράλληλα, χρησιμοποιώντας τα ειδικά χειριστήρια εικονικής πραγματικότητας προσομοιώνει τις λειτουργίες που επιτελεί ο χαρακτήρας στη σκηνή. Έτσι, ο χρήστης έχει καλύτερη αίσθηση του χώρου του παιχνιδιού και αισθάνεται ότι συμμετέχει στη δράση του.

2.2.2 Τεχνικές και μέθοδοι

Κατά την κατασκευή παιχνιδιών με βάση την Εικονική Πραγματικότητα, χρησιμοποιούνται διάφορες τεχνικές προσομοίωσης με σκοπό την δημιουργία εικονικών κόσμων. Έτσι, έχουν αναπτυχθεί διάφοροι αλγόριθμοι για την αυτόματη παραγωγή του περιεχομένου των κόσμων αυτών. Συγκεκριμένα, με τους αλγόριθμους αυτούς κατασκευάζονται διάφορα αντικείμενα όπως: τοπία και βλάστηση, οδικά δίκτυα, κτίρια, ζωντανά όντα (άνθρωποι και πλάσματα) και ιστορίες [39].

Βέβαια, για να μπορέσουν τα παραπάνω αντικείμενα να είναι ρεαλιστικά, οι απαιτήσεις προς τους σχετικούς αλγόριθμους είναι υψηλές. Για το λόγο αυτό, εκπονούνται διάφορες έρευνες που στοχεύουν σε έναν αποδεκτό συμβιβασμό μεταξύ του ρεαλισμού των αντικειμένων και της απόδοσης των αλγορίθμων [39].

2.2.3 Εμπειρία Χρήστη

Ο βασικός στόχος των παιχνιδιών που βασίζονται στην Εικονική Πραγματικότητα είναι να δώσουν την δυνατότητα στους χρήστες τους να αλληλοεπιδρούν, με τους εικονικούς κόσμους που παρέχουν, όσο το δυνατόν πιο φυσικά. Έτσι, οι χρήστες "βυθίζονται" στους κόσμους αυτούς καθώς οι ανθρώπινες αισθήσεις εξαπατώνται με τέτοιο τρόπο ώστε οι αντιληπτικές ενδείξεις, που φτάνουν στον εγκέφαλο, να ευθυγραμμίζονται με μια εναλλακτική πραγματικότητα. Αυτές οι αντιληπτικές ενδείξεις μπορεί να είναι διαφορετικής φύσης, συμπεριλαμβανομένων ακουστικών, οπτικών, απτικών ερεθισμάτων, όσφρησης και κίνησης [40].

Γενικά, ο συνδυασμός ήχου, εικόνας, κίνησης και δόνησης προκαλεί το μεγαλύτερο βαθμό εμπύθισης σε μία εφαρμογή Εικονικής Πραγματικότητας. Ειδικότερα, τα ερεθίσματα της δόνησης και της κίνησης αυξάνουν εκθετικά την αντίληψη της εμπύθισης σε σύγκριση με τον απλό συνδυασμό ήχου και εικόνας [41].

2.2.3.1 Δόνηση

Πιο συγκεκριμένα, η δόνηση αποτελεί μία σημαντική πηγή πληροφορίας, κυρίως κατά τα συμβάντα πρόσκρουσης, ενισχύοντας την εμπειρία χρήστη. Ταυτόχρονα, συνεισφέρει στην δημιουργία μίας πιο ρεαλιστικής ατμόσφαιρας [42] [43]. Ωστόσο, η κίνηση δείχνει να συνεισφέρει σ' αυτό περισσότερο από την δόνηση [41].

2.2.3.2 Κίνηση

Για την δημιουργία της αίσθησης της κίνησης σε ένα εικονικό περιβάλλον απαιτείται, κατά την πάροδο του χρόνου, η αναπαραγωγή όλων εκείνων των πληροφοριών που είναι ικανές να συνθέσουν το περιεχόμενο του περιβάλλοντα χώρου. Στις πρώτες εφαρμογές Εικονικής Πραγματικότητας, κάτι τέτοιο θεωρούνταν ιδιαίτερο δύσκολο να επιτευχθεί. Πρόσφατα όμως η ανάπτυξη συστημάτων παρακολούθησης κίνησης υψηλής ακρίβειας και εξελιγμένων προσομοιωτών κίνησης (όπως, για παράδειγμα, διαδρόμων και πλατφορμών κίνησης) επιτρέπουν πολύ μεγαλύτερο έλεγχο και ευελιξία στην τοποθέτηση του σώματος μέσα στο εικονικό περιβάλλον [44].

Κεφάλαιο 3ο: Ανάλυση Απαιτήσεων

Το κεφάλαιο αυτό περιγράφει τη λειτουργικότητα το παιχνιδιού VRacer αναλύοντας τις απαιτήσεις που καλείται να ικανοποιήσει.

3.1 Λειτουργικές Απαιτήσεις (Functional Requirement)

Οι λειτουργικές απαιτήσεις της εφαρμογής είναι οι ακόλουθες:

3.1.1 Γενικές Απαιτήσεις

- Εκκίνηση παιχνιδιού
 - Η εκκίνηση του παιχνιδιού θα γίνεται, αυτόματα, χωρίς κάποια ειδική προεργασία (από τον χρήστη).
 - Κατά την εκκίνηση του παιχνιδιού, το σύστημα πρέπει να τοποθετεί το όχημα στην αφετηρία της πίστας.
- Εκκίνηση παιχνιδιού
 - Το παιχνίδι πρέπει να τερματίζει με το πάτημα του πλήκτρου Escape.
- Συνθήκες κίνησης
 - Η διατήρηση του οχήματος μέσα στην πίστα, θα είναι ευθύνη του οδηγού (δηλ. του παίκτη-χρήστη).
 - Το σύστημα θα επιτρέπει στο όχημα να κινηθεί εκτός πίστας, όποτε αυτό είναι εφικτό.
 - Το σύστημα θα πρέπει να αντιμετωπίζει κάθε αντικείμενο εκτός της πίστας (π.χ. διαφημιστική πινακίδα) ως εμπόδιο.

3.1.2 Απαιτήσεις Οδήγησης

- Μετάδοση κίνησης
 - Το σύστημα πρέπει να προσφέρει στο χρήστη την δυνατότητα μετακίνησης του αγωνιστικού αυτοκινήτου προς διάφορες κατευθύνσεις (δηλ. μπροστά, πίσω, αριστερά, δεξιά και κάθε δυνατό συνδυασμό αυτών).
- Επιτάχυνση
 - Το σύστημα πρέπει να προσφέρει στο χρήστη την δυνατότητα να αυξάνει την ταχύτητα του αγωνιστικού αυτοκινήτου.
- Επιβράδυνση
 - Το σύστημα πρέπει να προσφέρει στο χρήστη την δυνατότητα να επιβραδύνει (φρενάρει) το αγωνιστικό αυτοκίνητο.

3.1.3 Εμπειρία Χρήστη

- Αίσθηση κίνησης

- Το σύστημα θα πρέπει να προσφέρει στο χρήστη την αίσθηση της κίνησης είτε όταν το όχημα κινείται προς κάποια κατεύθυνση είτε όταν αλλάζει κατεύθυνση.
- Αίσθηση ήχου
 - Το σύστημα θα πρέπει να παράγει ένα συνεχή ήχο που θα προσιδιάζει τον ήχο της λειτουργίας της μηχανής.
 - Το σύστημα θα πρέπει να παράγει ένα χαρακτηριστικό ήχο σε κάθε πρόσκρουση του αυτοκινήτου (είτε σε κάποιο στηθαίο της πίστας είτε σε οποιοδήποτε εμπόδιο).
- Αίσθηση δόνησης
 - Το σύστημα θα πρέπει να παράγει δόνηση σε κάθε πρόσκρουση (είτε σε κάποιο στηθαίο της πίστας είτε σε οποιοδήποτε εμπόδιο).
 - Το σύστημα θα πρέπει να παράγει δόνηση κατά την πλαγιολίσθηση του αυτοκινήτου.
 - Επιπλέον, η ένταση της δόνησης, κατά την πλαγιολίσθηση, θα πρέπει να είναι διαφορετική σε κάθε χέρι, ανάλογα με το είδος της στροφής. Συγκεκριμένα, σε μία αριστερή στροφή, η ένταση της δόνησης θα είναι μεγαλύτερη στο αριστερό χέρι από το δεξί. Το αντίστροφο θα πρέπει να συμβαίνει σε μία δεξιά στροφή.
- Αμεσότητα και ρεαλισμός
 - Η θέση του παίκτη-χρήστη που, ταυτόχρονα, είναι και ο οδηγός του αυτοκινήτου θα είναι “μέσα” στο αυτοκίνητο. Έτσι, ο βαθμός εμπύθισης, που θα προσφέρει η εφαρμογή, θα είναι μεγάλος.

3.2 Μη-λειτουργικές Απαιτήσεις (Non-Functional Requirement)

Οι μη-λειτουργικές απαιτήσεις της εφαρμογής είναι οι ακόλουθες:

3.2.1 Απαιτήσεις Ανάπτυξης

Οι μη-λειτουργικές απαιτήσεις που σχετίζονται με την ανάπτυξη της εφαρμογής είναι οι ακόλουθες:

- Πλατφόρμα Ανάπτυξης
 - Το παιχνίδι θα πρέπει να αναπτυχθεί στην πλατφόρμα ανάπτυξης ηλεκτρονικών παιχνιδιών Unity [45].
- Λειτουργικό Σύστημα
 - Το παιχνίδι θα πρέπει να αναπτυχθεί έτσι ώστε να είναι συμβατό με το Λ/Σ των Windows.

3.2.2 Απαιτήσεις Εκτέλεσης

Οι μη-λειτουργικές απαιτήσεις που σχετίζονται με την εκτέλεση της εφαρμογής είναι οι ακόλουθες:

- Περιβάλλον Εκτέλεσης
 - Το παιχνίδι θα πρέπει να μπορεί να εκτελεστεί ως VR εφαρμογή (βλ. Ενότητα 2.1.6) στα Windows.

3.2.3 Επικοινωνία με τον Χρήστη

Η επικοινωνία με τον χρήστη θα είναι δυνατόν να γίνει με τους ακόλουθους τρόπους:

- Πληκτρολόγιο
 - Ο χειρισμός του αγωνιστικού αυτοκινήτου γίνεται με βάση τα πλήκτρα-βέλη του πληκτρολογίου.
 - Η θέαση γίνεται μέσα από την οθόνη του υπολογιστή.
 - Στην περίπτωση αυτή δε θα είναι δυνατή η αίσθηση της δόνησης.
- Τυπικές VR συσκευές
 - Ο χειρισμός του αγωνιστικού αυτοκινήτου γίνεται με τα VR Χειριστήρια Αφής (βλ. Ενότητα 2.1.5.3).
 - Η θέαση γίνεται μέσα από την VR Οθόνη Κεφαλής (βλ. Ενότητα 2.1.5.1).
 - Η αίσθηση της δόνησης προσφέρεται από τα VR Χειριστήρια Αφής.
- Αυτοσχέδιο τιμόνι
 - Ο χειρισμός του αγωνιστικού αυτοκινήτου γίνεται με ένα ειδικό αυτοσχέδιο "τιμόνι".
 - Η θέαση γίνεται μέσα από την VR Οθόνη Κεφαλής (βλ. Ενότητα 2.1.5.1) ή, απλά, από την οθόνη του υπολογιστή.
 - Η αίσθηση της δόνησης προσφέρεται από το τιμόνι.
 - Το τιμόνι συνδέεται με το μηχανήμα εκτέλεσης του παιχνιδιού είτε ασύρματα (δηλ. μέσω Wi-Fi) είτε ασύρματα (μέσα σειριακής επικοινωνίας).

3.2.4 Απαιτήσεις Παραμετροποίησης

Οι μη-λειτουργικές απαιτήσεις που σχετίζονται με την παραμετροποίηση της εφαρμογής, μετά από την εγκατάστασή της, είναι οι ακόλουθες:

- Ασφάλεια
 - Η εφαρμογή, για λόγους απλότητας, δε θα διαθέτει σύστημα ασφαλείας ούτε θα απαιτεί ταυτοποίηση χρήστη.
- Ρυθμίσεις
 - Η επιλογή του τρόπου χειρισμού του αγωνιστικού αυτοκινήτου θα είναι δυνατόν να καθορίζεται από τον χρήστη, μετά από την εγκατάσταση της εφαρμογής.
 - Τα χαρακτηριστικά κίνησης του βασικού αντικείμενου της εφαρμογής (όπως, για παράδειγμα, της μέγιστης ταχύτητας του αγωνιστικού αυτοκινήτου) θα είναι δυνατόν να καθορίζονται από τον χρήστη, μετά από την εγκατάσταση της εφαρμογής.

Κεφάλαιο 4ο: Σχεδίαση Συστήματος

Το κεφάλαιο αυτό παρουσιάζει τα βασικά στοιχεία σχεδίασης του συστήματος του παιχνιδιού.

4.1 Αρχές Σχεδίασης

Όπως προαναφέρθηκε, η ανάπτυξη του παιχνιδιού VRacer βασίζεται στην πλατφόρμα ανάπτυξης ηλεκτρονικών παιχνιδιών Unity [45]. Αυτό σημαίνει ότι η σχεδίασή του ακολουθεί τις βασικές αρχές σχεδίασης ενός Unity παιχνιδιού.

Επίσης, όσον αφορά την επικοινωνία της παιχνιδιού με τον χρήστη, το ειδικό αυτοσχέδιο "τιμόνι" θα κατασκευαστεί ως ένα ηλεκτρονικό κύκλωμα που θα βασίζεται στον μικροελεγκτή ESP32 [46]. Αυτό σημαίνει ότι θα πρέπει να ακολουθεί τις αρχές σχεδίασης ενός ESP32 κυκλώματος [47].

Στη συνέχεια, περιγράφεται αναλυτικά η αρχιτεκτονική του συστήματος του παιχνιδιού και παρουσιάζονται τα βασικά στοιχεία σχεδίασής του. Πριν όμως από αυτό γίνεται μία συνοπτική παρουσίαση της πλατφόρμας Unity και του μικροελεγκτή ESP32.

4.2 Η Πλατφόρμα Ανάπτυξης Παιχνιδιών Unity

Πρόκειται για μία πλατφόρμα ανάπτυξης ηλεκτρονικών παιχνιδιών που προσφέρει, ταυτόχρονα, την δυνατότητα εμπλουτισμού τους με VR εμπειρίες (όπως, για παράδειγμα, της κίνησης, του ήχου και της δόνησης) [48]. Στην πραγματικότητα, αποτελείται από ένα σύμπλεγμα διαφόρων εργαλείων και υπηρεσιών [49].

4.2.1 Εργαλεία

Τα σημαντικότερα εργαλεία της πλατφόρμας Unity είναι τα ακόλουθα:

- Unity Engine
Αποτελεί τον πυρήνα εκτέλεσης των εφαρμογών-παιχνιδιών.
- Unity Editor
Πρόκειται για το βασικό περιβάλλον εργασίας για την δημιουργία, την οργάνωση και την διαχείριση των διαφόρων εφαρμογών.
- Asset Store
Περιέχει μία βιβλιοθήκη από στοιχεία (όπως υφές, εικόνες, μοντέλα, κινούμενα σχέδια κλπ.) που χρησιμοποιούνται για τη συγκρότηση του περιβάλλοντος των παιχνιδιών. Ταυτόχρονα, επιτρέπει τον εμπλουτισμό της με στοιχεία που μπορεί να προσθέσει ο ίδιος ο χρήστης της πλατφόρμας.
- Scripting Engine
Πρόκειται για ένα μηχανισμό που επιτρέπει την δημιουργία κώδικα με σκοπό την διαχείριση των παραπάνω στοιχείων (όπως, για παράδειγμα, της κίνησής τους, την παραγωγή ήχων κλπ.).

- XR Toolkit

Πρόκειται για ένα σύνολο από εργαλεία που προσφέρουν την δυνατότητα κατασκευής VR και AR εφαρμογών (βλ. Ενότητα 2.1.3).

- Παραγωγή Έκδοσης (Build)

Πρόκειται για το εργαλείο, μέσα από το οποίο, παράγεται μία νέα έκδοση μίας εφαρμογής με σκοπό την διάθεσή της προς τους χρήστες της.

4.2.2 Υπηρεσίες

Οι σημαντικότερες υπηρεσίες της πλατφόρμας Unity, που μπορεί να παράσχει σε μία εφαρμογή, είναι οι ακόλουθες:

- Κίνηση

Πρόκειται για την δυνατότητα παραγωγής κίνησης.

- Ήχος

Πρόκειται για την δυνατότητα παραγωγής ήχων.

- Δόνηση

Πρόκειται για την δυνατότητα παραγωγής δόνησης.

- Φωτισμός

Πρόκειται για την δυνατότητα εμπλουτισμών των σκηνών με ρεαλιστικό φωτισμό.

- Οπτικά Εφέ

Πρόκειται για την δυνατότητα δημιουργίας οπτικών εφέ.

4.3 Ο Μικροελεγκτής ESP32

Ο μικροελεγκτής ESP32 μπορεί να προγραμματιστεί ώστε να αποτελέσει τη βάση πολλών και ποικίλων ηλεκτρονικών κυκλωμάτων. Ταυτόχρονα, μπορεί να επικοινωνήσει με διάφορες άλλες συσκευές είτε ενσύρματα (σειριακή επικοινωνία) είτε ασύρματα (Wi-Fi ή Bluetooth επικοινωνία) [50].

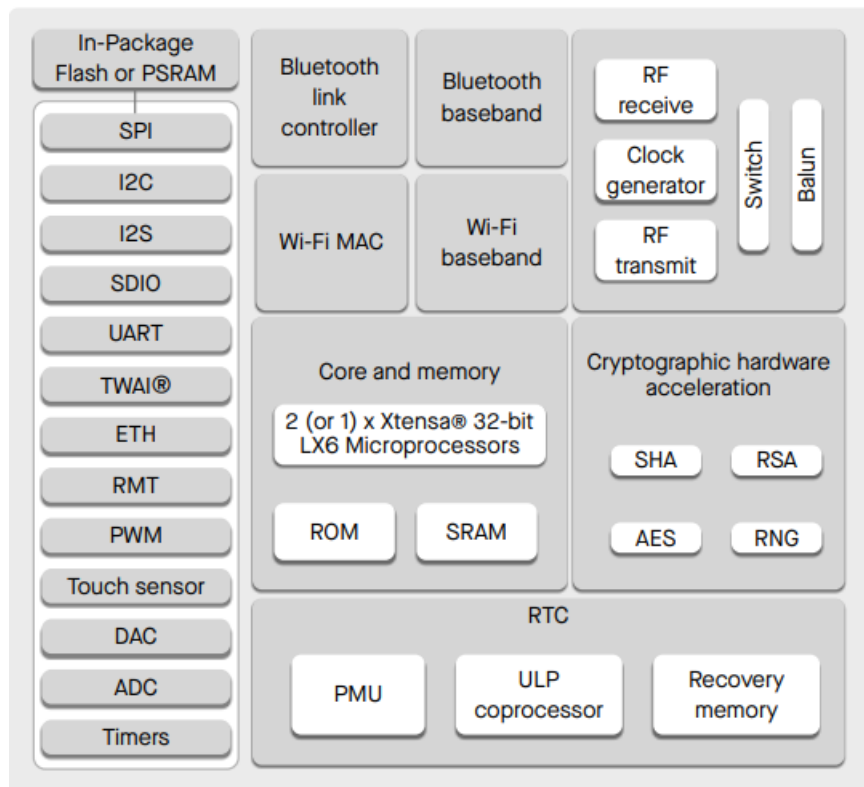
Στην πραγματικότητα, δεν πρόκειται για ένα μικροελεγκτή αλλά για μία οικογένεια από μικροελεγκτές. Κάθε ένας από αυτούς διαθέτει διαφορετική ΚΜΕ (CPU) και δυνατότητες. Όλοι όμως προσφέρουν το ίδιο πακέτο εργαλείων ανάπτυξης κώδικα (SDK) και τις ίδιες βασικές λειτουργίες (βλ. Σχήμα 4.7).

4.3.1 Χρήσεις του μικροελεγκτή ESP32

Εξαιτίας των μικρών απαιτήσεων σε ενέργεια και των πολλών λειτουργιών του, ο μικροελεγκτής ESP32 αποτελεί μία ιδανική λύση στο χώρο των IoT εφαρμογών [51]. Ενδεικτικά, μπορεί να χρησιμοποιηθεί στις ακόλουθες περιοχές:

- Έξυπνα σπίτια
- Βιομηχανικοί αυτοματισμοί
- Έξυπνη γεωργία
- Ρομποτική

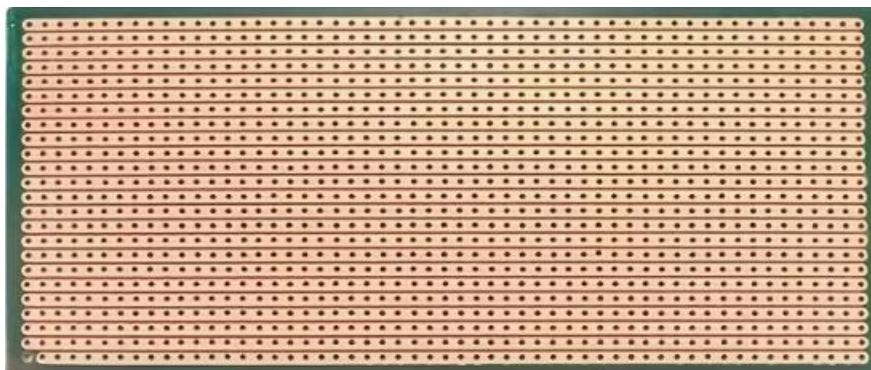
- Αναγνώριση ήχου και εικόνας



Σχήμα 4.7 Διάγραμμα βασικών λειτουργιών ESP32 [51]

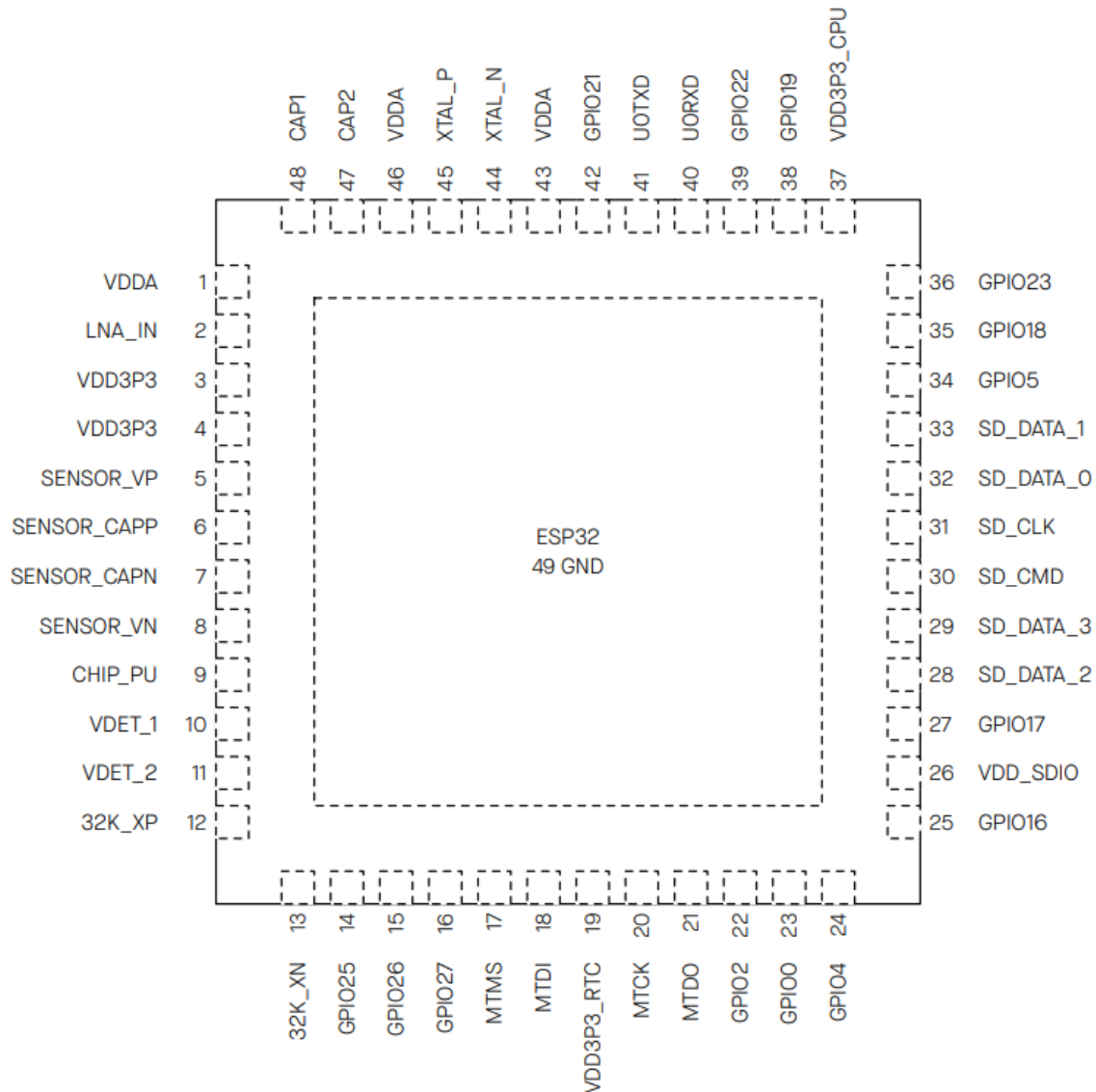
4.3.2 Ανάπτυξη ηλεκτρονικών κυκλωμάτων με βάση τον μικροελεγκτή ESP32

Ένα ηλεκτρονικό κύκλωμα με βάση τον μικροελεγκτή ESP32 αναπτύσσεται πάνω σε ειδικές πλακέτες ανάπτυξης (βλ. Σχήμα 4.8) που επιτρέπουν την ενσύρματη επικοινωνία του με διάφορες περιφερειακές συσκευές.



Σχήμα 4.8 Ενδεικτική πλακέτα ανάπτυξης ηλεκτρονικών κυκλωμάτων

Η επικοινωνία του μικροελεγκτή ESP32 με τις περιφερειακές συσκευές βασίζεται στην πλειάδα από ακροδέκτες (pin) που διαθέτει [51] (βλ. Σχήμα 4.9). Κάποιοι από αυτούς έχουν συγκεκριμένη χρήση (όπως για παράδειγμα την παροχή – στις συσκευές - τροφοδοσίας και γείωσης). Υπάρχουν όμως και αρκετοί γενικής χρήσης αποδέκτες (General Purpose Input/Output) ή (GPIO). Αυτοί, προσφέρουν την δυνατότητα ανταλλαγής χρήσιμων πληροφοριών μεταξύ του ESP32 και των περιφερειακών συσκευών.



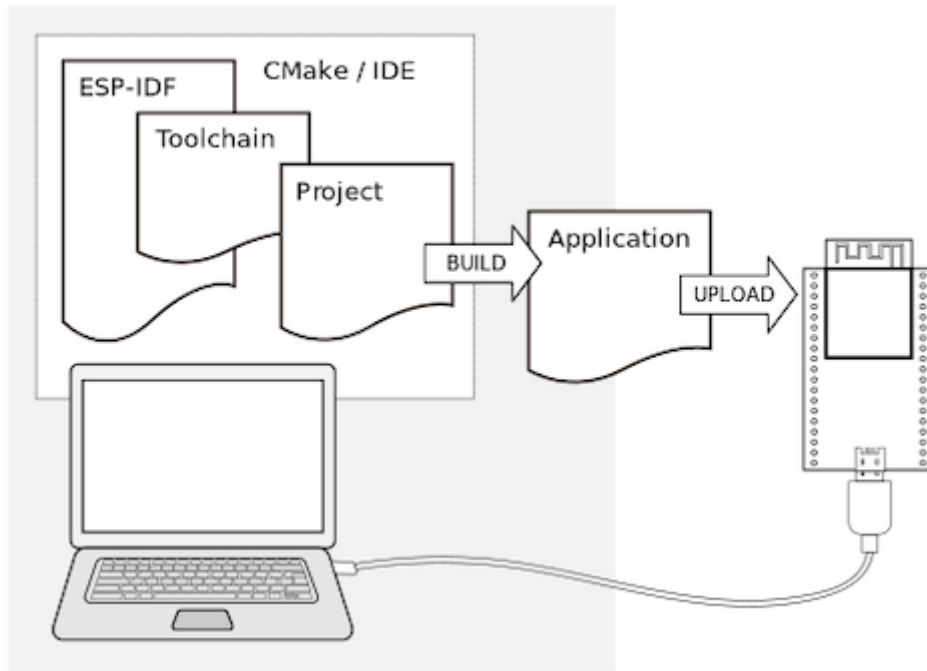
Σχήμα 4.9 Ενδεικτικό διάγραμμα ακροδεκτών (pin) του ESP32 [51]

4.3.3 Προγραμματισμός του μικροελεγκτή ESP32

Από τη στιγμή που ο μικροελεγκτής ESP32 βρίσκεται σε ένα κύκλωμα με διάφορες περιφερειακές συσκευές, απαιτείται ο κατάλληλος προγραμματισμός του για να μπορέσει να επικοινωνήσει με αυτές και να τις αξιοποιήσει [52]. Πρόκειται για τον κλασικό προγραμματιστικό κύκλο ο οποίος συντελείται σε κάποιο Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE) και αποτελείται από τις ακόλουθες φάσεις [53] (βλ. Σχήμα 4.10):

- Συγγραφή κώδικα
Συνήθως, γράφεται σε C++ ή Python.
- Μεταγλώττιση (Build)
- Μεταφορά του τελικού εκτελέσιμου κώδικα στον μικροελεγκτή ESP32 (Upload)

Η συγκεκριμένη μεταφορά γίνεται, συνήθως, μέσω σειριακής επικοινωνίας.



Σχήμα 4.10 Προγραμματισμός του ESP32 με το Ολοκληρωμένο Περιβάλλον Ανάπτυξης ESP-IDF [53]

Μερικά από τα πιο δεδομένα Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης, για τον μικροελεγκτή ESP32, είναι τα ακόλουθα:

- Espressif Integrated Development Framework (ESP-IF) [53] (βλ. Σχήμα 4.10)
- Arduino-ESP32 [54]
- Visual Studio Code

Απαιτείται η προσθήκη της κατάλληλης επέκτασης (extension) που παρέχει το Espressif Integrated Development Framework (ESP-IF) [55].

4.3.4 Περιφερειακές συσκευές

Μερικές από τις βασικότερες περιφερειακές συσκευές ενός ηλεκτρονικού κυκλώματος είναι οι ακόλουθες.

4.3.4.1 Γυροσκόπιο (Gyroscope)

Πρόκειται για μία συσκευή που είναι ικανή να αντιλαμβάνεται κάθε αλλαγή της θέσης στην οποία βρίσκεται, μέσα στο χώρο [56] και να προσφέρει αναλυτικά στοιχεία της θέσης αυτής (βλ. Σχήμα 4.11). Μπορεί λοιπόν να χρησιμοποιηθεί τόσο για τη μετάδοση κίνησης όσο και τον προσδιορισμό της κατεύθυνσης.



Σχήμα 4.11 Ένα τυπικό Γυροσκόπιο οκτώ (8) ακροδεκτών

Στα πλαίσια ενός ηλεκτρονικού κυκλώματος, λειτουργώντας ως συσκευή εισόδου, μπορεί να αποστέλλει διάφορα δεδομένα – θέσης - σε ένα μικροελεγκτή ESP32. Από τα δεδομένα αυτά, όπως αναλύεται παρακάτω, τα σημαντικότερα (για την τρέχουσα Π.Ε.) είναι τα ακόλουθα:

- Γωνία Pitch

Πρόκειται για τη γωνία που σχηματίζει ο διαμήκης άξονας (εμπρός-πίσω) ενός αντικειμένου ως από το οριζόντιο επίπεδο που ορίζει ο εγκάρσιος άξονάς του (από πλευρά σε πλευρά). Για παράδειγμα, για ένα αεροπλάνο, καθορίζει τη γωνία της μύτης του αεροπλάνου ως προς το οριζόντιο επίπεδο που ορίζουν τα φτερά του.

- Γωνία Roll

Πρόκειται για τη γωνία που σχηματίζει ο εγκάρσιος άξονας (από πλευρά σε πλευρά) ενός αντικειμένου ως προς το οριζόντιο επίπεδο που ορίζει ο διαμήκης άξονάς του (εμπρός-πίσω). Για παράδειγμα, για ένα αεροπλάνο, καθορίζει τη γωνία που σχηματίζει ο εγκάρσιος άξονας των φτερών του ως προς το οριζόντιο επίπεδο που ορίζει ο διαμήκης άξονάς του (μύτη-ουρά).

4.3.4.2 Χειριστήριο Πλοήγησης (Joystick)

Πρόκειται για το κλασικό χειριστήριο ηλεκτρονικών παιχνιδιών που μπορεί να μεταδώσει κίνηση προς κάθε κατεύθυνση με τη χρήση ενός λεβιέ [57] (βλ. Σχήμα 4.12).



Σχήμα 4.12 Ένα τυπικό Joystick πέντε (5) ακροδεκτών

Στα πλαίσια ενός ηλεκτρονικού κυκλώματος, και λειτουργώντας ως συσκευή εισόδου, αποστέλλει τις ακόλουθες ενδείξεις σε ένα μικροελεγκτή ESP32:

- Ένδειξη κίνησης ως προς τον κατακόρυφο άξονα (Συντεταγμένη στον άξονα y)

Συνήθως, πρόκειται για μία ακέραια τιμή που ανήκει σε ένα συγκεκριμένο διάστημα τιμών (π.χ. διάστημα 0 – 4000).

Όταν ο λεβιές κινείται προς τα κάτω, παράγεται η μικρότερη τιμή του παραπάνω διαστήματος τιμών (ή τιμές που είναι "κοντά" σ' αυτή).

Αντιθέτως, όταν ο λεβιές κινείται προς τα πάνω, παράγεται η μέγιστη τιμή (ή τιμές που είναι "κοντά" σ' αυτή).

Τέλος, όταν λεβιές βρίσκεται στην αρχική του θέση, ως προς τον κατακόρυφο άξονα, παράγεται η ενδιάμεση τιμή του διαστήματος.

- Ένδειξη κίνησης ως προς τον οριζόντιο άξονα (Συντεταγμένη στον άξονα x)

Συνήθως, πρόκειται για μία ακέραια τιμή που ανήκει σε ένα συγκεκριμένο διάστημα τιμών (π.χ. διάστημα 0 – 4000).

Όταν ο λεβιές κινείται προς τα αριστερά, παράγεται η μικρότερη τιμή του παραπάνω διαστήματος τιμών (ή τιμές που είναι "κοντά" σ' αυτή).

Αντιθέτως, όταν ο λεβιές κινείται προς τα δεξιά, παράγεται η μέγιστη τιμή (ή τιμές που είναι "κοντά" σ' αυτή).

Τέλος, όταν λεβιές βρίσκεται στην αρχική του θέση, ως προς τον οριζόντιο άξονα, παράγεται η ενδιάμεση τιμή του διαστήματος.

4.3.4.3 Συσκευή Παραγωγής Δονήσεων (Vibration Motor)

Πρόκειται για συσκευή που είναι ικανή να δέχεται εντολές δόνησης και να παράγει δονήσεις με βάση τα χαρακτηριστικά των εντολών αυτών [58] (βλ. Σχήμα 4.13). Πιο συγκεκριμένα, κάθε εντολή δόνησης αποτελείται από ένα μόνο στοιχείο που είναι η ένταση της δόνησης. Πρόκειται για ένα ακέραιο αριθμό

που ανήκει στο διάστημα τιμών $[0 - 255]$. Το μηδέν (0) υποδηλώνει απουσία δόνησης ενώ το 255 τη μέγιστη ένταση δόνησης.



Σχήμα 4.13 Μια τυπική Συσκευή Παραγωγής Δονήσεων

Κάθε φορά που η συσκευή δέχεται μία μη-μηδενική εντολή δόνησης, παράγεται μία συνεχής δόνηση, ανάλογης έντασης, που διαρκεί μέχρις ότου ληφθεί μία μηδενική εντολή δόνησης.

4.4 Αρχιτεκτονική Συστήματος

Γενικά, το σύστημα του παιχνιδιού VRacer αποτελείται από δύο βασικά μέρη-υποσυστήματα:

- Το βασικό περιβάλλον εκτέλεσης του παιχνιδιού.

Πρόκειται για μία Unity εφαρμογή, με όνομα *VRacer_Unity*, η οποία φτιάχτηκε στα πλαίσια της τρέχουσας Π.Ε.. Υλοποιεί το αγωνιστικό αυτοκίνητο και το περιβάλλον δράσης του παιχνιδιού.

- Υποσύστημα επικοινωνίας με τον χρήστη.

- Πληκτρολόγιο/Οθόνη

Ως δυνατότητα, παρέχεται αυτόματα από το Unity, όπως συμβαίνει για κάθε Unity εφαρμογή.

- Τυπικές VR συσκευές (VR Χειριστήρια Αφής / VR Οθόνη Κεφαλής)

Ως δυνατότητα, παρέχεται αυτόματα από τις ειδικές βιβλιοθήκες του Unity και το συνοδευτικό λογισμικό των VR συσκευών.

- Αυτοσχέδιο τιμόνι

Πρόκειται για μία σύνθετη συσκευή που φτιάχτηκε στα πλαίσια της τρέχουσας εργασίας και έχει το ρόλο και το σχήμα ενός τιμονιού.

Αποτελείται από ένα ειδικό ηλεκτρονικό κύκλωμα που είναι τοποθετημένο μέσα σε ένα τρισδιάστατο "τιμόνι". Το κύκλωμα βασίζεται στον μικροελεγκτή ESP32.

Στη συνέχεια, γίνεται μία εκτενής αναφορά στην αρχιτεκτονική της Unity εφαρμογής VRacer και του αυτοσχέδιου τιμονιού.

4.4.1 Η Unity εφαρμογή VRacer

Όπως κάθε Unity εφαρμογή, η εφαρμογή VRacer αποτελείται, κατά βάση, από:

- Αντικείμενα (Game Object)

Πρόκειται για το σύνολο όλων των αντικειμένων που στοιχειοθετούν το περιβάλλον εργασίας του παίκτη-χρήστη.

- Κώδικας (Script)

Πρόκειται για c# (ως συνήθως) κώδικα που πλαισιώνει τα παραπάνω αντικείμενα δίνοντάς τους "ζωή".

Ειδικότερα, η αρχιτεκτονική της εφαρμογής προσδοκά να ικανοποιήσει τις συγκεκριμένες απαιτήσεις που έχουν τεθεί – γι' αυτή (βλ. Κεφάλαιο 3).

4.4.1.1 Αντικείμενα (Game Object)

Τα βασικότερα αντικείμενα της εφαρμογής είναι τα ακόλουθα.

4.4.1.1.1 Σκηνή

Το παιχνίδι διαθέτει μία μόνο σκηνή μέσα στην οποία συντελείται όλη η δράση του.

4.4.1.1.2 Αγωνιστικό Αυτοκίνητο

Είναι το βασικό αντικείμενο του παιχνιδιού και το μοναδικό κινούμενο. Είναι σύνθετο και περιέχει, με τη σειρά του, τα ακόλουθα αντικείμενα:

- Σώμα Αυτοκινήτου

Αποτελεί το οπτικό και υλικό μέρος του αυτοκινήτου.

- Κάμερα

Μία κάμερα ακολουθεί διαρκώς το αγωνιστικό αυτοκίνητο προσφέροντας την δυνατότητα της θέασης και, ταυτόχρονα, την αίσθηση της κίνησης.

4.4.1.1.3 Περιβάλλον

Πρόκειται για το εξωτερικό περιβάλλον μέσα στο οποίο κινείται το αυτοκίνητο. Είναι σύνθετο και αποτελείται, με τη σειρά του, από τα ακόλουθα αντικείμενα:

- Έδαφος

Αντιπροσωπεύει το έδαφος πάνω στο οποίο βρίσκονται η πίστα ταχύτητας και τα διάφορα διακοσμητικά στοιχεία.

- Πίστα Ταχύτητας

Πρόκειται για την πίστα μέσα στην οποία – θα πρέπει να – κινείται το αυτοκίνητο.

Εκτός από ευθείες και στροφές διαφόρων κλίσεων, περιέχει και κάποια ειδικά αντικείμενα με σκοπό την εντονότερη εμπειρία οδήγησης:

- Γέφυρα

Πρόκειται για μία γέφυρα που, ουσιαστικά, δημιουργεί ένα ανισόπεδο κόμβο.

- Σαμαράκια

Πρόκειται για υπερυψωμένα τμήματα του οδοστρώματος, σε διάφορες εκδοχές.

- Στηθαία

Πρόκειται για “προστατευτικά” στηθαία που μπαίνουν, και στις δυο πλευρές του οδοστρώματος, σε κλειστές ή μεγάλες στροφές.

- Διακοσμητικά Στοιχεία

Είναι ένα σύνολο από τρισδιάστατα οπτικά στοιχεία που περιβάλλουν την πίστα και έχουν διακοσμητικό ρόλο.

Επιπρόσθετα, κάνουν ακόμα πιο έντονη την αίσθηση της κίνησης (καθώς το αυτοκίνητο διέρχεται ανάμεσά τους) και του ήχου (καθώς, ταυτόχρονα, λειτουργούν και ως εμπόδια).

Πιο συγκεκριμένα, είναι τα ακόλουθα:

- Πύλες
- Δέντρα
- Κιόσκια
- Τέντες
- Κερκίδες θεατών
- Διαφημιστικές πινακίδες
- Πυργίσκοι παρατήρησης/τηλεόρασης

4.4.1.2 Ελεγκτές (Controller)

Πρόκειται για κώδικα που είναι υπεύθυνος για τη ροή εκτέλεσης του παιχνιδιού.

4.4.1.2.1 Ελεγκτής Αυτοκινήτου (CarController)

Είναι μία c# κλάση που περιέχει τον κώδικα που ορίζει τη συμπεριφορά του αυτοκινήτου (δηλ. την παραγωγή κίνησης, ήχων και δονήσεων). Ειδικότερα, οι λειτουργίες που παρέχει είναι οι ακόλουθες:

- Κίνηση

Στην πραγματικότητα πρόκειται για μία σύνθετη λειτουργία που συμπεριλαμβάνει τα ακόλουθα:

- Τοποθέτηση του αυτοκινήτου, κατά την εκκίνηση του παιχνιδιού, στην αφετηρία.
- Μετάδοση κίνησης προς κάθε κατεύθυνση (steering) διατηρώντας, ταυτόχρονα, το αυτοκίνητο κολλημένο στο έδαφος.
- Επιτάχυνση/επιβράδυνση αυτοκινήτου.
- Πλαγιολίσθηση αυτοκινήτου (drifting).

▪ Ήχοι

Οι ήχοι που παράγονται από την εφαρμογή είναι οι ακόλουθοι:

- Ήχος της μηχανής του αυτοκινήτου.

Παράγεται επαναληπτικά, από την εκκίνηση του παιχνιδιού και μετά.

Παράλληλα, η έντασή του αυξομειώνεται ανάλογα με την ταχύτητα του αυτοκινήτου.

- Ήχος πρόσκρουσης

Παράγεται κατά την πρόσκρουση του αυτοκινήτου σε διάφορα εμπόδια (όπως, για παράδειγμα, στα στηθαία της πίστας ή στα διάφορα αντικείμενα του εξωτερικού περιβάλλοντος).

▪ Δόνηση

Τα γεγονότα που παράγουν δόνηση είναι τα ακόλουθα:

- Πρόσκρουση σε εμπόδια (όπως, για παράδειγμα, στα στηθαία της πίστας ή στα διάφορα αντικείμενα του εξωτερικού περιβάλλοντος).
- Πλαγιολίσθηση

4.4.1.2.2 Διαχείριση Κάμερας

Πρόκειται για τον κώδικα που διαχειρίζεται τη θέση της κάμερας με τέτοιο τρόπο ώστε, να φαίνεται ότι, βρίσκεται "μέσα" στο αυτοκίνητο και "δείχνει" πάντα μπροστά.

4.4.1.3 Παραμετροποίηση Λειτουργίας

Η λειτουργία του αγωνιστικού αυτοκινήτου και η επικοινωνία – του παιχνιδιού - με τον χρήστη (όπως αυτή περιγράφεται στην Ενότητα 3.2.3) ρυθμίζονται από παραμέτρους. Οι παράμετροι αυτές βρίσκονται αποθηκευμένες σε ένα JSON αρχείο και είναι, πιο συγκεκριμένα, οι ακόλουθες:

▪ Παράμετροι επικοινωνίας χρήστη

Περιέχει όλες τις παραμέτρους που σχετίζονται με την επιλογή του επιθυμητού τρόπου χειρισμού του αγωνιστικού αυτοκινήτου (μεταξύ των διαθέσιμων τρόπων που προαναφέρθηκαν) και των ρυθμίσεων αυτού.

▪ Παράμετροι λειτουργίας αγωνιστικού αυτοκινήτου

Περιέχει όλες τις παραμέτρους που σχετίζονται με τη συμπεριφορά του αγωνιστικού αυτοκινήτου.

4.4.2 Αυτοσχέδιο τιμόνι

Πρόκειται για μία σύνθετη κατασκευή συσκευή που έχει το ρόλο του τιμονιού. Συνδέεται ενσύρματα ή ασύρματα με τη Unity εφαρμογή VRacer και σκοπό έχει να προσφέρει - στον χρήστη - την δυνατότητα χειρισμού του αγωνιστικού αυτοκινήτου. Ταυτόχρονα, προσφέρει την αίσθηση της δόνησης.

Τα βασικά συστατικά του τιμονιού είναι τα ακόλουθα:

▪ Ηλεκτρονικό κύκλωμα

- Πλακέτα ανάπτυξης

Πρόκειται για την πλακέτα πάνω στην οποία αναπτύσσεται το ηλεκτρονικό κύκλωμα.

- Στοιχεία κυκλώματος (Module)

Πρόκειται για το σύνολο από τις ηλεκτρονικές συσκευές που συγκροτούν το ηλεκτρονικό κύκλωμα. Το επίκεντρο του κυκλώματος είναι φυσικά ο μικροελεγκτής ESP32.

- Κώδικας

Πρόκειται για C++ κώδικα που σκοπό έχει τον προγραμματισμό του μικροελεγκτή ESP32.

- Εξωτερικό περίβλημα

Πρόκειται για ένα αντικείμενο που έχει το σχήμα τιμονιού και διαθέτει κατάλληλη κοιλότητα έτσι ώστε να τοποθετείται μέσα σ' αυτή το παραπάνω ηλεκτρονικό κύκλωμα.

4.4.2.1 Στοιχεία ηλεκτρονικού κυκλώματος

Ειδικότερα, τα στοιχεία που συγκροτούν το ηλεκτρονικό κύκλωμα του τιμονιού περιγράφονται, πιο αναλυτικά, στη συνέχεια.

- Μικροελεγκτής ESP32

Αποτελεί το βασικότερο συστατικό του ηλεκτρονικού κυκλώματος και του τιμονιού γενικότερα. Ταυτόχρονα, είναι αυτό που επιτρέπει την επικοινωνία του τιμονιού με την Unity εφαρμογή VRacer. Η επικοινωνία αυτή μπορεί να γίνει είτε ενσύρματα (σειριακή επικοινωνία) είτε ασύρματα (Wi-Fi).

- Γυροσκόπιο

Με βάση τα δεδομένα που προσφέρει, κάθε στιγμή, επιτρέπει τη μετάδοση κίνησης και τον προσδιορισμό της κατεύθυνσης. Πιο συγκεκριμένα, οι γωνίες Pitch και Roll μετατρέπονται, όπως αναλύεται παρακάτω, σε στοιχεία κίνησης και αποστέλλονται στην Unity εφαρμογή VRacer.

- Joystick

Αποτελεί μία εναλλακτική λύση για τη μετάδοση κίνησης και του προσδιορισμού της κατεύθυνσης. Πιο συγκεκριμένα, οι συντεταγμένες (στους άξονες x και y) που παρέχει μετατρέπονται, όπως αναλύεται παρακάτω, σε στοιχεία κίνησης και αποστέλλονται στην Unity εφαρμογή VRacer.

Είναι σημαντικό να τονιστεί ότι δεν έχει νόημα να συνυπάρχει μαζί με ένα γυροσκόπιο στο ίδιο κύκλωμα.

- Συσκευή Παραγωγής Δονήσεων

Χρησιμοποιείται για την παραγωγή των δονήσεων που προκαλούνται, από διάφορα συμβάντα, κατά την κίνηση του αγωνιστικού αυτοκινήτου.

4.4.2.2 Μετάδοση Κίνησης και Καθορισμός Κατεύθυνσης

Σύμφωνα με τους "φυσικούς" κανόνες του Unity, στο οριζόντιο επίπεδο, η ταχύτητα και η κατεύθυνση ενός αντικειμένου προσδιορίζονται από δύο συντελεστές: i) ένας για τον άξονα των x (αριστερά/δεξιά) και ii) ένας για τον άξονα των y (πίσω/εμπρός).

Κάθε ένας από τους παραπάνω συντελεστές είναι ένας πραγματικός αριθμός και δέχεται τιμές που ανήκουν στο διάστημα τιμών $[-1, 1]$. Η τιμή μηδέν (0), ως προς κάποιο άξονα, σημαίνει απουσία κίνησης κατά τον άξονα αυτόν. Οι αρνητικές τιμές σημαίνουν κίνηση προς τα αριστερά (ή πίσω) με ταχύτητα ανάλογη της (απόλυτης) τιμής. Έτσι, η τιμή -1 σημαίνει κίνηση προς τα αριστερά (ή πίσω) με τη μέγιστη δυνατή ταχύτητα. Αντιθέτως, οι θετικές τιμές σημαίνουν κίνηση προς τα δεξιά (ή μπροστά) με ταχύτητα ανάλογη της (απόλυτης) τιμής. Έτσι, η τιμή 1 σημαίνει κίνηση προς τα δεξιά (ή μπροστά) με τη μέγιστη δυνατή ταχύτητα.

Με βάση τα παραπάνω, προκύπτει ότι, για να κινηθεί το αγωνιστικό αυτοκίνητο, αρκεί το τιμόνι να στέλνει τα κατάλληλα σήματα κατεύθυνσης στην Unity εφαρμογή VRacer. Κάθε τέτοιο σήμα είναι ένα ζεύγος τιμών της ακόλουθης μορφής (οι τιμές διαχωρίζονται με κόμμα):

Συντελεστής_ταχύτητας_για_τον_άξονα_x, Συντελεστής_ταχύτητας_για_τον_άξονα_y

Η παραγωγή των παραπάνω σημάτων γίνεται από τον μικροελεγκτή ESP32 με βάση τα δεδομένα που παρέχει είτε το γυροσκόπιο είτε το Joystick.

4.4.2.2.1 Κίνηση με Γυροσκόπιο

Στην περίπτωση που το ηλεκτρονικό κύκλωμα του τιμονιού βασίζεται σε γυροσκόπιο, για την παραγωγή των σημάτων κατεύθυνσης χρησιμοποιούνται οι γωνίες Pitch και Roll – του γυροσκοπίου.

Πιο συγκεκριμένα, ο παρακάτω πίνακας (βλ. Πίνακας 4.1), δείχνει πως προκύπτει ο Συντελεστής x με βάση το Roll. Το ROLL_THRESHOLD είναι μία σταθερά που χρησιμοποιείται έτσι ώστε να μην προκαλούν κίνηση οι αμελητέες μεταβολές θέσης του γυροσκοπίου.

Πίνακας 4.1: Προσδιορισμός Συντελεστή x με βάση το Roll

Συνθήκη	Συντελεστής x	Κατεύθυνση
$\text{Roll} > \text{ROLL_THRESHOLD}$	-1	Αριστερά
$\text{Roll} < -\text{ROLL_THRESHOLD}$	1	Δεξιά
Κάθε άλλη περίπτωση	0	Καμία

Ομοίως, ο παρακάτω πίνακας (βλ. Πίνακας 4.2), δείχνει πως προκύπτει ο Συντελεστής y με βάση το Pitch. Το PITCH_THRESHOLD είναι μία σταθερά που χρησιμοποιείται έτσι ώστε να μην προκαλούν κίνηση οι αμελητέες μεταβολές θέσης του γυροσκοπίου.

Πίνακας 4.2: Προσδιορισμός Συντελεστή y με βάση το Pitch

Συνθήκη	Συντελεστής y	Κατεύθυνση
$\text{Pitch} > \text{PITCH_THRESHOLD}$	-1	Πίσω
$\text{Pitch} < -\text{PITCH_THRESHOLD}$	1	Εμπρός
Κάθε άλλη περίπτωση	0	Καμία

4.4.2.2 Κίνηση με Joystick

Στην περίπτωση που το ηλεκτρονικό κύκλωμα του τιμονιού βασίζεται σε Joystick, για την παραγωγή των σημάτων κατεύθυνσης χρησιμοποιούνται οι συντεταγμένες (ως προς τους άξονες x και y) που παράγει το Joystick.

Πιο συγκεκριμένα, ο παρακάτω πίνακας (βλ. Πίνακας 4.3), δείχνει πως προκύπτει ο Συντελεστής x με βάση τη Συντεταγμένη x του Joystick.

Πίνακας 4.3: Προσδιορισμός Συντελεστή x με βάση τη Συντεταγμένη x του Joystick

Συντεταγμένη x του Joystick	Συντελεστής x	Κατεύθυνση
Μικρότερη δυνατή τιμή (ή κοντινές τιμές)	-1	Αριστερά
Μεγαλύτερη δυνατή τιμή (ή κοντινές τιμές)	1	Δεξιά
Κάθε άλλη περίπτωση	0	Καμία

Ομοίως, ο παρακάτω πίνακας (βλ. Πίνακας 4.4), δείχνει πως προκύπτει ο Συντελεστής y με βάση τη Συντεταγμένη y του Joystick.

Πίνακας 4.4: Προσδιορισμός Συντελεστή y με βάση τη Συντεταγμένη y του Joystick

Συντεταγμένη y του Joystick	Συντελεστής x	Κατεύθυνση
Μικρότερη δυνατή τιμή (ή κοντινές τιμές)	-1	Πίσω
Μεγαλύτερη δυνατή τιμή (ή κοντινές τιμές)	1	Εμπρός
Κάθε άλλη περίπτωση	0	Καμία

4.4.2.3 Παραγωγή Δονήσεων

Σύμφωνα με τις προδιαγραφές λειτουργίας της Συσκευής Παραγωγής Δονήσεων (βλ. Ενότητα 4.3.4.3), για να παραχθεί μία δόνηση, για ένα συγκεκριμένο χρονικό διάστημα, απαιτείται η αποστολή – στη συσκευή – μίας μη-μηδενικής εντολής δόνησης. Στη συνέχεια, μετά το πέρας του χρονικού αυτού διαστήματος, απαιτείται η αποστολή μίας μηδενικής εντολής, για τον τερματισμό της δόνησης.

Η παραπάνω διαδικασία ελέγχεται από τον μικροελεγκτή ESP32 και εκτελείται κάθε φορά που – αυτός – δέχεται, από τη Unity εφαρμογή VRacer, ένα κατάλληλο σήμα δόνησης. Κάθε τέτοιο σήμα είναι ένα ζεύγος τιμών της ακόλουθης μορφής (οι τιμές διαχωρίζονται με κόμμα):

Ένταση_δόνησης, Διάρκεια_δόνησης

Η Ένταση_δόνησης είναι ένας ακέραιος αριθμός που ανήκει στο διάστημα τιμών [0, 255]. Η Διάρκεια_δόνησης είναι χρόνος σε χιλιοστά του δευτερολέπτου.

4.4.2.4 Παραμετροποίηση Λειτουργίας

Η λειτουργία του τιμονιού (όπως αυτή περιγράφεται στην Ενότητα 3.2.3) ρυθμίζεται από τις ακόλουθες παραμέτρους:

- Συσκευή μετάδοσης κίνησης (γυροσκόπιο ή joystick)

Η συγκεκριμένη παράμετρος επιτρέπει στο κύκλωμα, και με βάση τον κατάλληλο προγραμματισμό του, να λειτουργήσει είτε με βάση ένα γυροσκόπιο είτε με ένα joystick (αλλά όχι και τα δύο μαζί).

- Μέθοδος επικοινωνίας με την Unity εφαρμογή VRacer (Σειριακή ή Wi-Fi)
- Στοιχεία λειτουργίας της Wi-Fi επικοινωνίας:
 - Δίκτυο Wi-Fi (SSID)
 - Συνθηματικό (Password)
 - IP του μηχανήματος στο οποίο εκτελείται η Unity εφαρμογή VRacer.

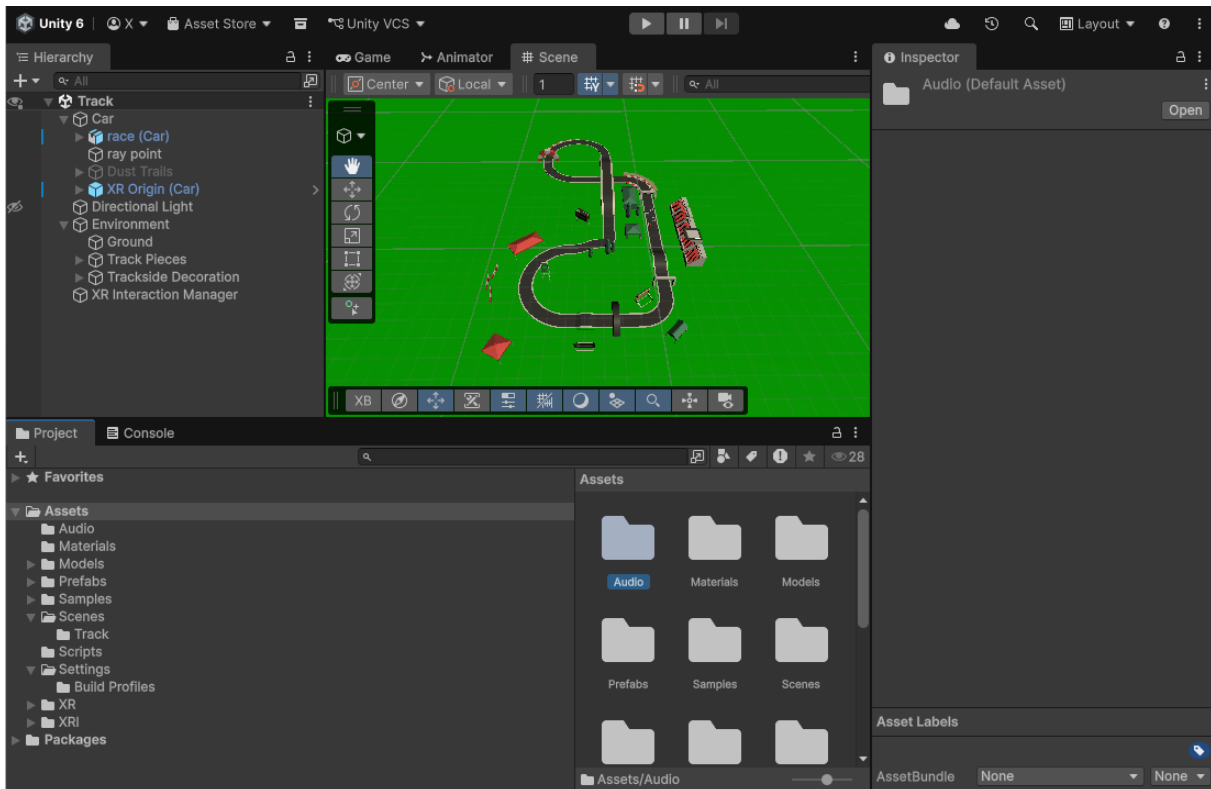
Όλες οι παραπάνω παράμετροι αποθηκεύονται στη μόνιμη μνήμη του μικροελεγκτή ESP32 και ανασύρονται από αυτή κατά την ενεργοποίηση του κυκλώματος. Το κύκλωμα, κατά την ενεργοποίησή του, προσφέρει την δυνατότητα επανακαθορισμού τους (αναμένοντας για μερικά δευτερόλεπτα πριν ξεκινήσει τη λειτουργία του).

Κεφάλαιο 5ο: Υλοποίηση

Το κεφάλαιο αυτό περιγράφει τα βασικά εργαλεία και την διαδικασία ανάπτυξης του παιχνιδιού VRacer.

5.1 Unity εφαρμογή VRacer

Η εφαρμογή VRacer υλοποιήθηκε, εξ' ολοκλήρου, με την πλατφόρμα ανάπτυξης ηλεκτρονικών παιχνιδιών Unity, ως ένα τυπικό Unity "project" με όνομα *VRacer_Unity* (βλ. Σχήμα 5.14).



Σχήμα 5.14: Η εφαρμογή VRacer_Unity στον Unity Editor

5.1.1 Εργαλεία Ανάπτυξης

Τα σημαντικότερα από τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της Unity εφαρμογής VRacer είναι τα ακόλουθα:

- Unity Editor (6η έκδοση)

Στο συγκεκριμένο περιβάλλον, δημιουργήθηκε η ίδια η εφαρμογή. Κατόπιν, κατασκευάστηκαν, μέσα σ' αυτό, όλα τα αντικείμενά της (βλ. Ενότητα 4.4.1.1).

- Microsoft Visual Studio

Στο συγκεκριμένο περιβάλλον προγραμματισμού αναπτύχθηκε ο κώδικας της εφαρμογής.

- Meta Horizon Link

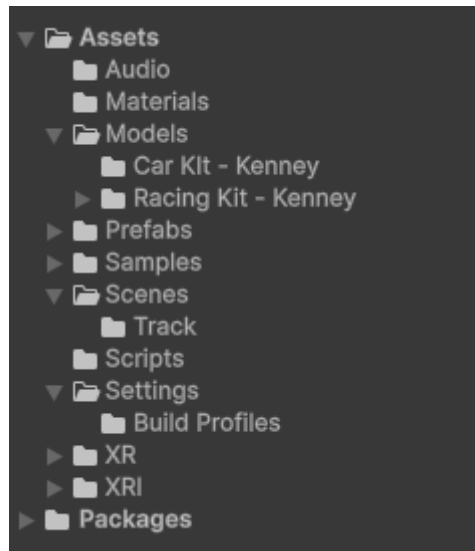
Πρόκειται για μία ενδιάμεση εφαρμογή που επιτρέπει την διασύνδεση των διαφόρων VR χειριστηρίων με τον υπολογιστή.

- Λ/Σ Windows

Όλα τα παραπάνω εργαλεία λογισμικού λειτουργούν στο Λ/Σ των Windows.

5.1.2 Δομή Εφαρμογής

Η Εφαρμογή VRacer διαθέτει την τυπική δομή μίας εφαρμογής του Unity (βλ. Σχήμα 5.15).



Σχήμα 5.15: Η δομή της εφαρμογής VRacer όπως αυτή φαίνεται στον Unity Editor

Ειδικότερα, η εφαρμογή αποτελείται από τα ακόλουθα μέρη:

- Assets

Πρόκειται για το σύνολο των όλων των βασικών συστατικών της εφαρμογής.

- Audio

Είναι ο φάκελος που περιέχει όλα τα αρχεία ήχου που χρησιμοποιούνται στην εφαρμογή.

- Materials

Είναι ο φάκελος που περιέχει όλες τις υφές που χρησιμοποιούνται από διάφορα αντικείμενα της εφαρμογής.

- Models

Είναι ο φάκελος που περιέχει όλα τα μοντέλα από τα οποία χτίζεται το υλικό μέρος της εφαρμογής.

- Scenes

Είναι ο φάκελος που περιέχει όλες τις σκηνές της εφαρμογής. Όπως αναφέρεται και παρακάτω είναι μόνο μία (Track).

- Scripts

Είναι ο φάκελος που περιέχει τα διάφορα αρχεία κώδικα της εφαρμογής.

- Settings

Είναι ο φάκελος που περιέχει διάφορα αρχεία παραμέτρων της εφαρμογής.

- Packages

Γενικά, πρόκειται για το σύνολο των βιβλιοθηκών που είναι απαραίτητες για τη λειτουργία της εφαρμογής.

5.1.3 Unity Αντικείμενα

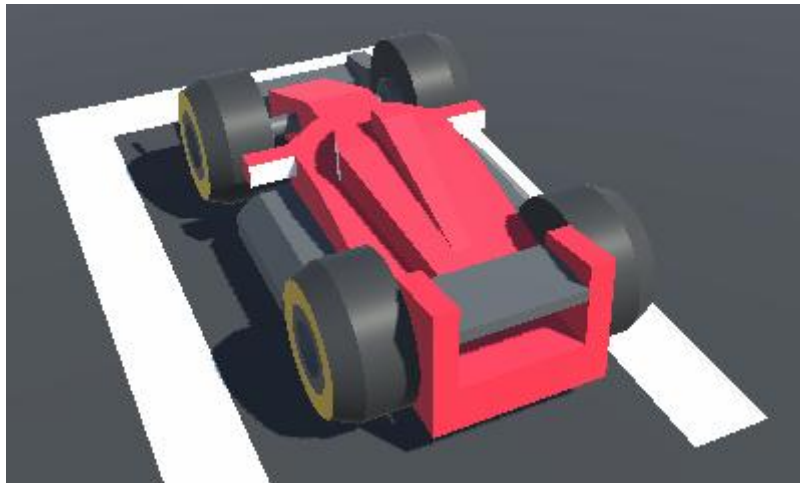
Τα διάφορα αντικείμενα του παιχνιδιού, όπως αυτά προβλέπονται από τη σχεδίαση της εφαρμογής (βλ. Ενότητα 4.4), υλοποιήθηκαν όπως περιγράφεται παρακάτω.

5.1.3.1 Σκηνή

Πρόκειται για ένα σύνθετο αντικείμενο τύπου "Track" που περιέχει, ιεραρχικά, όλα τα υπόλοιπα αντικείμενα του παιχνιδιού.

5.1.3.2 Αγωνιστικό Αυτοκίνητο

Το αγωνιστικό αυτοκίνητο υλοποιείται ως ένα αντικείμενο τύπου "Rigidbody" και διαθέτει το όνομα *Car* (βλ. Σχήμα 5.16).



Σχήμα 5.16: Το Αγωνιστικό Αυτοκίνητο της εφαρμογής

Για την διαχείρισή του, είναι υπεύθυνη η κλάση "CarController" (του αρχείου κώδικα "carController.cs") που περιγράφεται στην Ενότητα 5.1.5.1 παρακάτω.

Για την παραγωγή του ήχου της μηχανής του, συνδέεται, μέσω της ιδιότητας "Audio Source", με το αρχείο ήχου (audio clip) "Engine-loop-1-normalized.wav". Επίσης, για την παραγωγή του ήχου της πρόσκρουσης, συνδέεται, μέσω της ιδιότητας "Collision Clip", με το αρχείο ήχου (audio clip) "Car_spinning.wav". Και τα δύο αυτά αρχεία, διατίθενται δωρεάν από τον οργανισμό παροχής ήχων "OpenGaneArt.org" [59].

Τέλος, όσον αφορά την δομή του, είναι ένα σύνθετο αντικείμενο που περιέχει, ιεραρχικά, τα ακόλουθα αντικείμενα.

5.1.3.2.1 Σώμα Αυτοκινήτου

Πρόκειται για ένα σύνθετο μοντέλο που προσδίδει την τρισδιάστατη εικόνα του αυτοκινήτου. Είναι τύπου "race" της συλλογής μοντέλων "Car Kit – Kenney" και διαθέτει το όνομα *race* (*Car*).

Πρακτικά, αποτελείται από ένα σύνολο από διάφορα μοντέλα (model) της συλλογής "Car Kit – Kenney":

- Σασί
Μοντέλο τύπου "body" της συλλογής "Car Kit – Kenney".
- Πίσω-αριστερός τροχός
Μοντέλο τύπου "weel_backLeft" της συλλογής "Car Kit – Kenney".
- Πίσω-δεξιός τροχός
Μοντέλο τύπου "weel_backRight" της συλλογής "Car Kit – Kenney".
- Εμπρός-αριστερός τροχός
Μοντέλο τύπου "weel_frontLeft" της συλλογής "Car Kit – Kenney".
- Εμπρός-δεξιός τροχός
Μοντέλο τύπου "weel_frontRight" της συλλογής "Car Kit – Kenney".

5.1.3.2.2 Χειριστήρια Οδήγησης

Υλοποιούνται από ένα άυλο αντικείμενο τύπου "XR Origin" με όνομα *XR Origin (Car)*. Ανήκει, ιεραρχικά, στο βασικό αντικείμενο του αυτοκινήτου για να είναι διαρκώς "μέσα" του, κατά την κίνησή του.

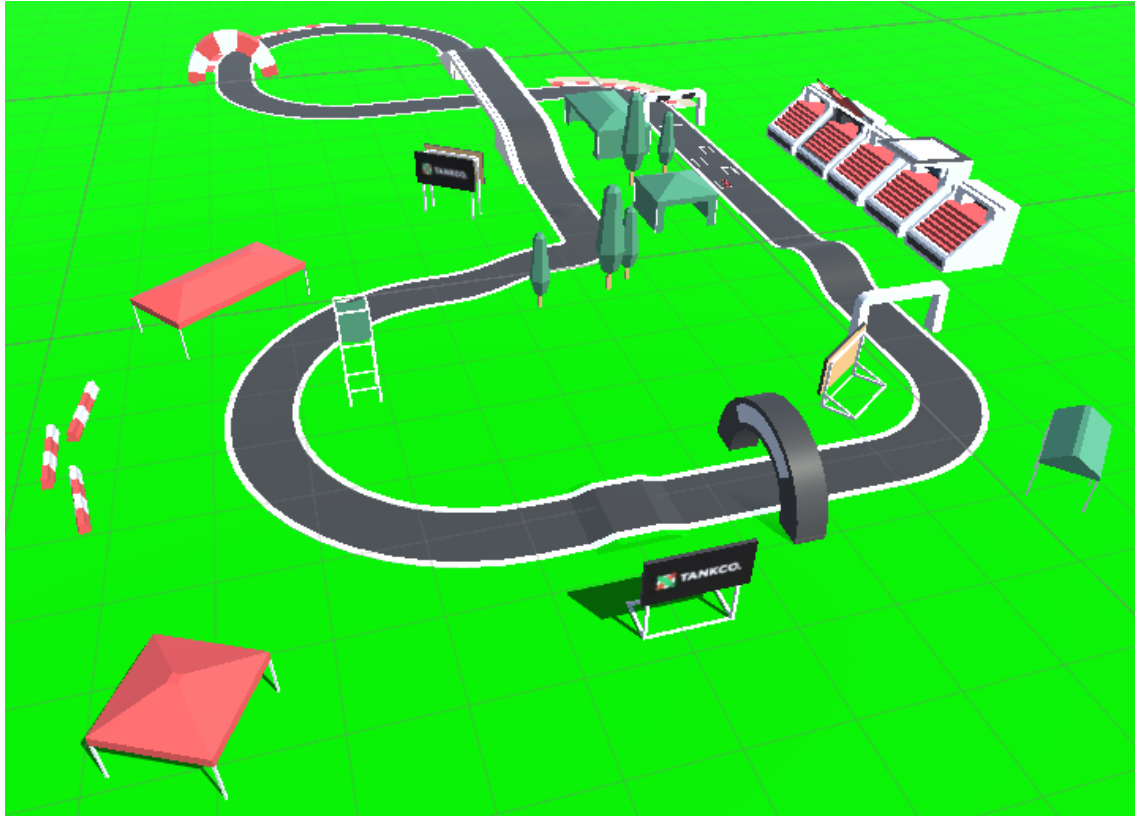
Για την διαχείρισή του, είναι υπεύθυνη η κλάση "CarXROriginFollow" (του αρχείου κώδικα "CarXROriginFollow.cs") που περιγράφεται στην Ενότητα 5.1.5.2 παρακάτω.

Τέλος, όσον αφορά την δομή του, είναι ένα σύνθετο αντικείμενο που περιέχει, ιεραρχικά, τα ακόλουθα αντικείμενα:

- Κάμερα
Υλοποιείται από ένα αντικείμενο τύπου "Camera".
- Αριστερό Χειριστήριο
Υλοποιείται από ένα αντικείμενο τύπου "Controller Input Action Manager", με όνομα *Left Controller*.
Πλαισιώνεται, μεταξύ άλλων, από ένα αντικείμενο τύπου "Haptic Impulse Player" για την παραγωγή δονήσεων.
- Δεξιό Χειριστήριο
Υλοποιείται από ένα αντικείμενο τύπου "Controller Input Action Manager", με όνομα *Right Controller*, και αντιπροσωπεύει το δεξιό χέρι του παίκτη-χρήστη.
Πλαισιώνεται, μεταξύ άλλων, από ένα αντικείμενο τύπου "Haptic Impulse Player" για την παραγωγή δονήσεων.

5.1.3.3 Περιβάλλον

Πρόκειται για ένα σύνθετο αντικείμενο, με όνομα *Environment*, που αποτελεί το περιβάλλον "εργασίας" της εφαρμογής (βλ. Σχήμα 5.17).



Σχήμα 5.17: Το Περιβάλλον του παιχνιδιού

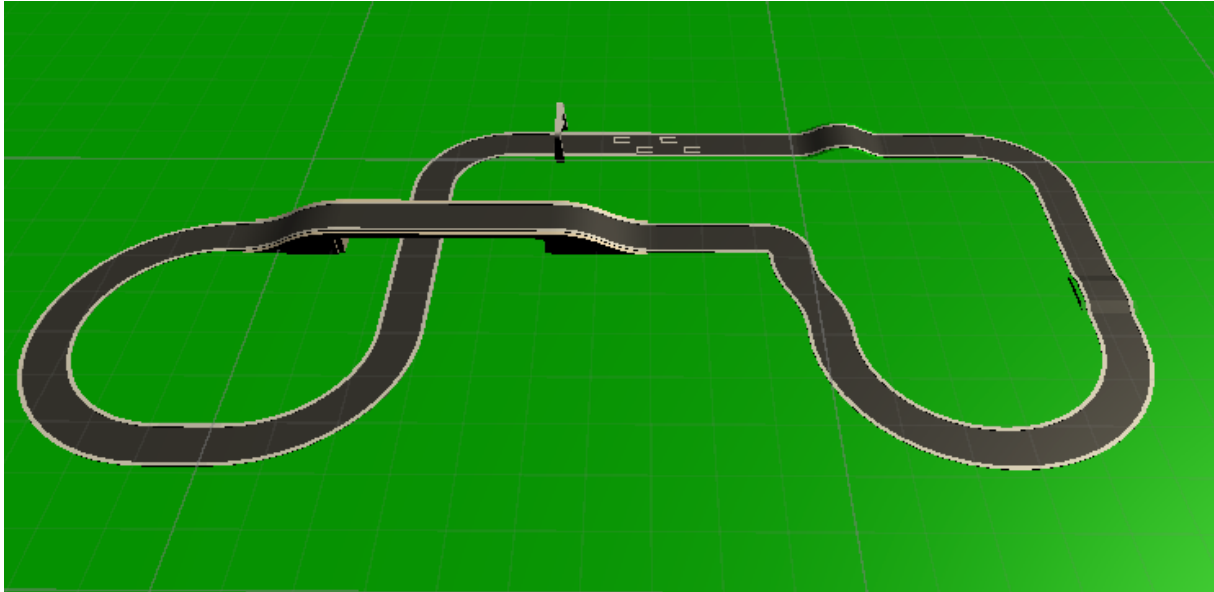
Περιέχει, ιεραρχικά, τα ακόλουθα αντικείμενα.

5.1.3.3.1 Έδαφος

Υλοποιείται από ένα αντικείμενο τύπου "Plane" με όνομα *Ground*.

5.1.3.3.2 Πίστα Ταχύτητας

Υλοποιείται ως ένα σπονδυλωτό αντικείμενο με όνομα *Track Pieces*. Αποτελείται από διάφορα μοντέλα (model) και υφές (material) της συλλογής "Racing Kit – Kenney". Όλα αυτά μαζί, τοποθετημένα το ένα μετά το άλλο, συγκροτούν την πίστα στην οποία πρέπει να κινείται το αυτοκίνητο (βλ. Σχήμα 5.18).

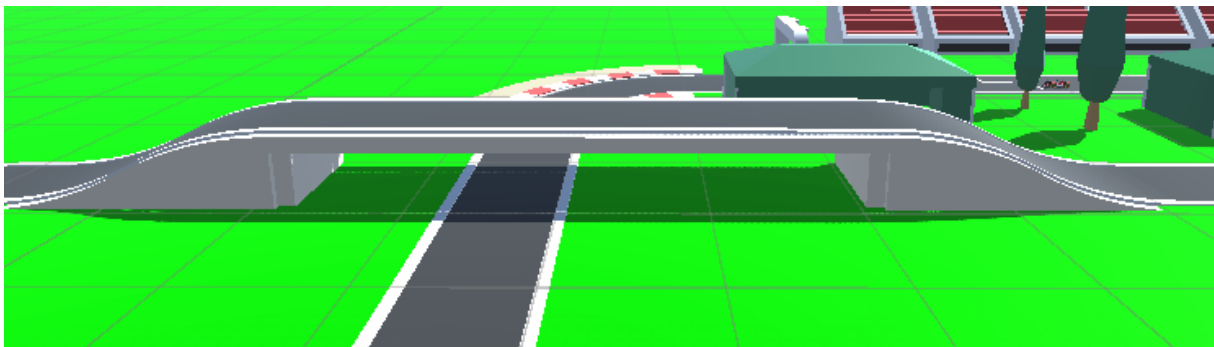


Σχήμα 5.18: Η πίστα ταχύτητας του παιχνιδιού

Πιο συγκεκριμένα, τα μοντέλα από τα οποία αποτελείται η πίστα ταχύτητας είναι τα ακόλουθα:

- Γέφυρα

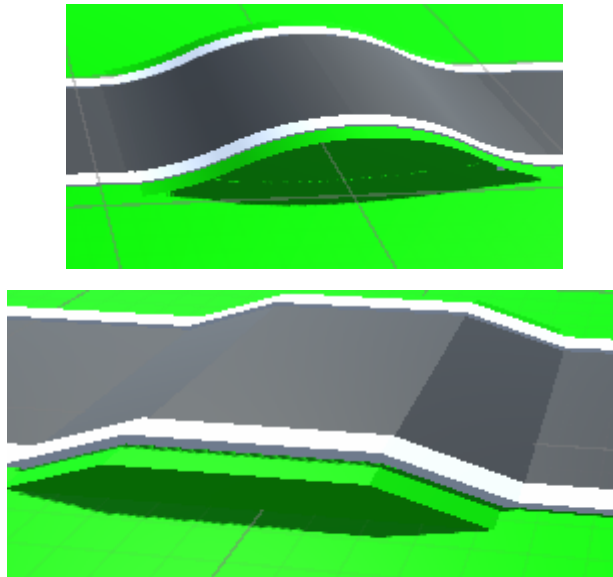
Υλοποιείται ως ένα σπονδυλωτό αντικείμενο που συγκροτείται από διάφορα μοντέλα της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.19).



Σχήμα 5.19: Γέφυρα ανισόπεδου κόμβου

- Σαμαράκια

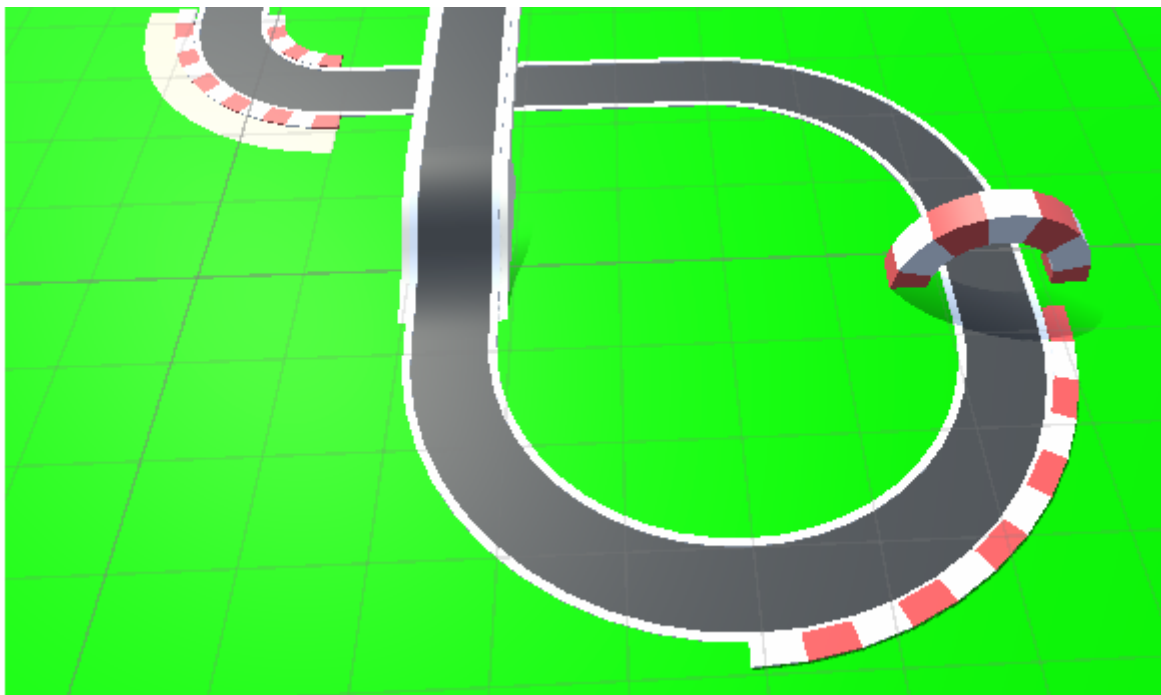
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.20).



Σχήμα 5.20: Τα Σαμαράκια της πίστας του παιχνιδιού

- Στηθαία

Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney".



Σχήμα 5.21: Τα Στηθαία της πίστας του παιχνιδιού

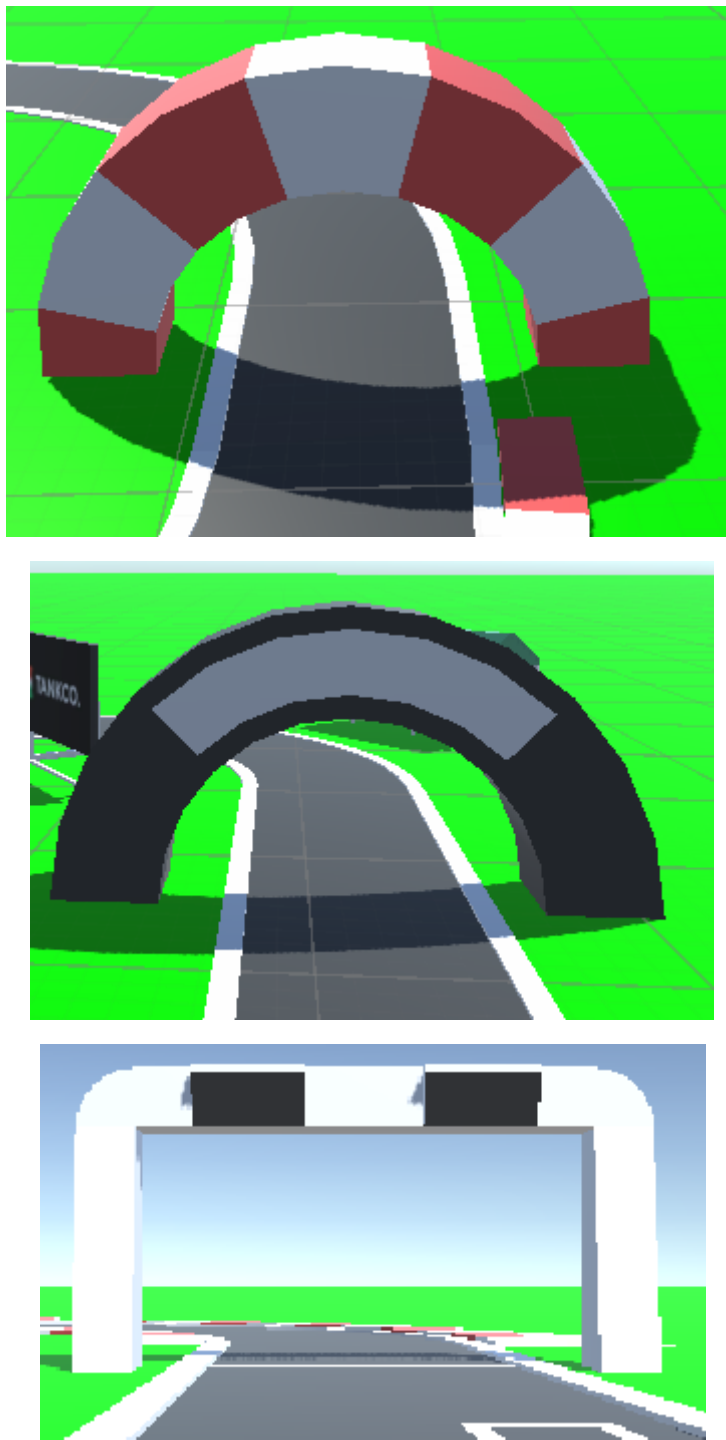
5.1.3.3 Διακοσμητικά Στοιχεία

Υλοποιούνται από ένα σύνθετο αντικείμενο, με όνομα *Trackside Decoration*.

Συγκεκριμένα, αποτελείται από διάφορα μοντέλα (model) της συλλογής "Racing Kit – Kenney" (όπως, για παράδειγμα, δέντρα, διαφημιστικές πινακίδες, στηθαία κλπ.):

- Πύλες

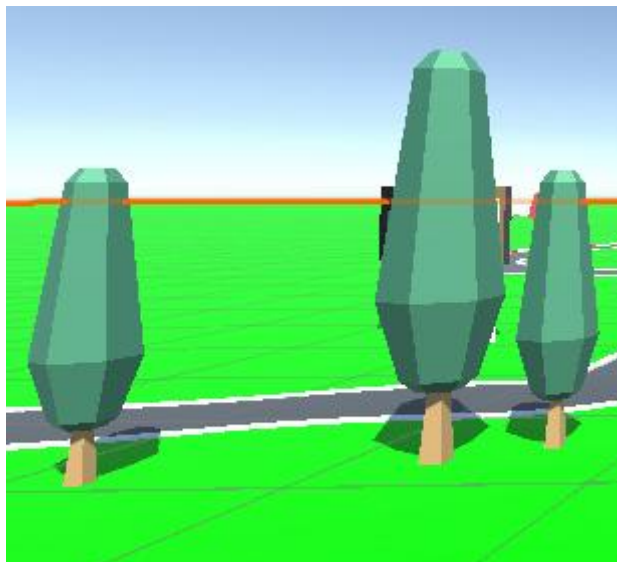
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.22).



Σχήμα 5.22: Οι Πύλες πάνω από την πίστα του παιχνιδιού

- Δέντρα

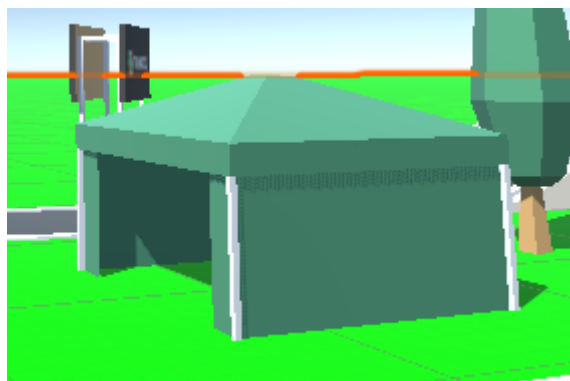
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.23).



Σχήμα 5.23: Χαρακτηριστικά δέντρα του εξωτερικού περιβάλλοντος του παιχνιδιού

- Κιόσκια

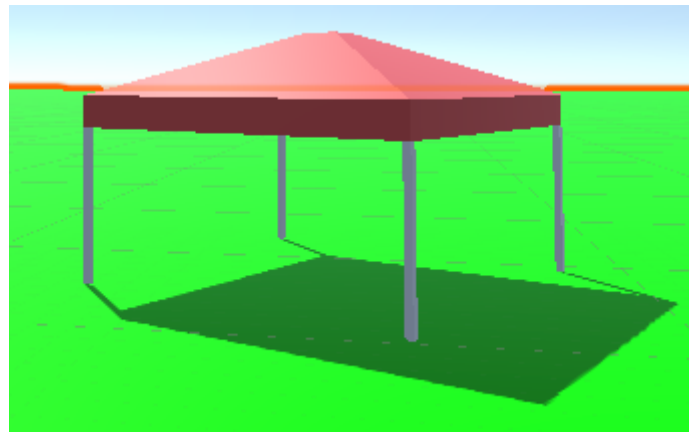
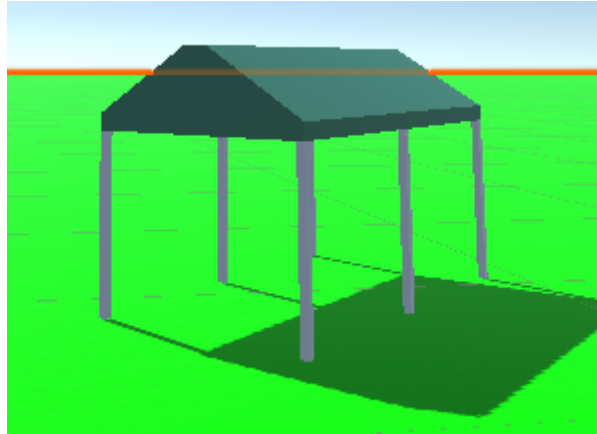
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.24).



Σχήμα 5.24: Χαρακτηριστικό Κιόσκι του εξωτερικού περιβάλλοντος του παιχνιδιού

- Τέντες

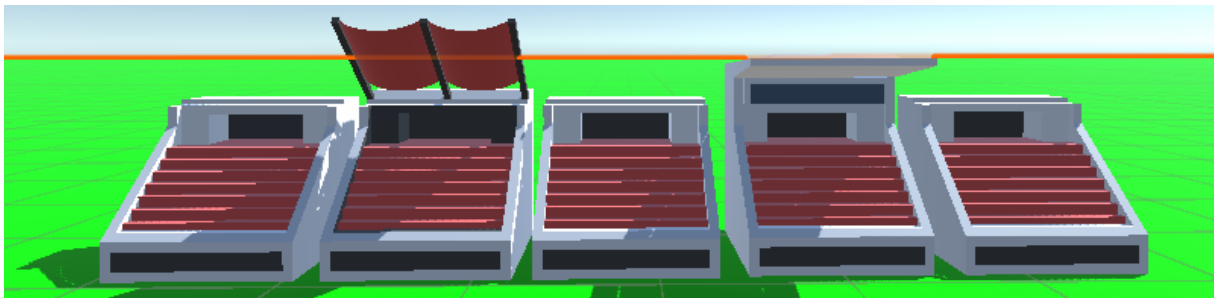
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.25).



Σχήμα 5.25: Χαρακτηριστικές Τέντες του εξωτερικού περιβάλλοντος του παιχνιδιού

- Κερκίδες θεατών

Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.26).



Σχήμα 5.26: Οι Κερκίδες του εξωτερικού περιβάλλοντος του παιχνιδιού

- Διαφημιστικές πινακίδες

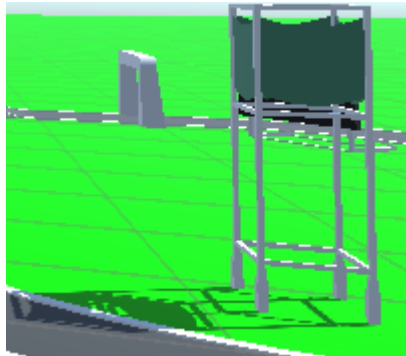
Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.27).



Σχήμα 5.27: Χαρακτηριστικές Διαφημιστικές Πινακίδες

- Πυργίσκοι παρατήρησης/τηλεόρασης

Υλοποιούνται με τη χρήση διαφόρων σχετικών μοντέλων της συλλογής "Racing Kit – Kenney" (βλ. Σχήμα 5.28).



Σχήμα 5.28: Χαρακτηριστικός Πυργίσκος παρατήρησης/τηλεόρασης

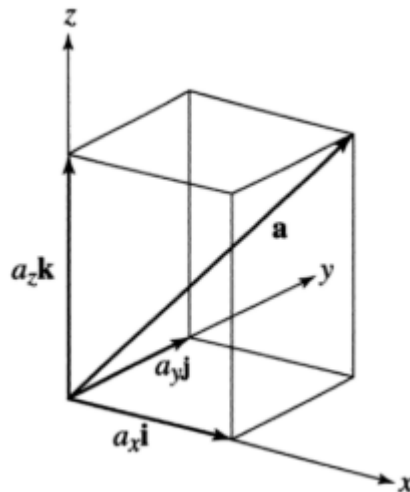
5.1.4 Ειδικοί Τύποι Δεδομένων

Πέρα από τους βασικούς τύπους δεδομένων, που προσφέρουν όλες οι σύγχρονες γλώσσες προγραμματισμού, το Unity βασίζεται επιπλέον στους παρακάτω τύπους.

5.1.4.1 Vector3

Είναι ο τύπος που αντιπροσωπεύει όλα τα τρισδιάστατα διανύσματα [60]. Στο Unity, χρησιμοποιείται εκφράζοντας θέσεις και κατευθύνσεις.

Ένα τρισδιάστατο διάνυσμα (έστω a) αποτελεί μία διατεταγμένη τριπλέτα (βλ. Σχήμα 5.29) από τις συντεταγμένες του a στον άξονα των x (a_x), στον άξονα των y (a_y) και στον άξονα των (a_z).



Σχήμα 5.29: Τα συστατικά ενός τρισδιάστατου διανύσματος [60]

5.1.4.1.1 Βασικές Ιδιότητες

Οι βασικότερες ιδιότητες του τύπου είναι οι ακόλουθες:

- x : Η συντεταγμένη του διανύσματος στον άξονα των x
- y : Η συντεταγμένη του διανύσματος στον άξονα των y
- z : Η συντεταγμένη του διανύσματος στον άξονα των z
- **magnitude**: Το μέτρο του διανύσματος
- **normalized**: Διάνυσμα που έχει την ίδια κατεύθυνση με το τρέχον διάνυσμα αλλά έχει μέτρο ίσο με 1

5.1.4.1.2 Βασικές Σταθερές

Για ευκολία, ο τύπος προσφέρει, ως σταθερές, κάποια έτοιμα διανύσματα::

- **back**: Συντομογραφία του $\text{Vector3}(0, 0, -1)$
- **forward**: Συντομογραφία του $\text{Vector3}(0, 0, 1)$
- **up**: Συντομογραφία του $\text{Vector3}(0, 1, 0)$
- **down**: Συντομογραφία του $\text{Vector3}(0, -1, 0)$
- **left**: Συντομογραφία του $\text{Vector3}(-1, 0, 0)$
- **right**: Συντομογραφία του $\text{Vector3}(1, 0, 0)$
- **zero**: Συντομογραφία του $\text{Vector3}(0, 0, 0)$

5.1.4.1.3 Βασικές Μέθοδοι

Ο τύπος Vector3 προσφέρει μερικές πολύ χρήσιμες μεθόδους για την πραγματοποίηση διαφόρων υπολογισμών μεταξύ των τιμών του. Οι βασικότερες από αυτές είναι οι ακόλουθες:

- $\text{Vector3 Project}(\text{Vector3 vector}, \text{Vector3 onNormal})$

Προβάλλει ένα διάνυσμα σε ένα άλλο και επιστρέφει το διάνυσμα που προκύπτει.

- $\text{Vector3 ProjectOnPlane}(\text{Vector3 vector}, \text{Vector3 planeNormal})$

Προβάλλει ένα διάνυσμα σε ένα αντικείμενο τύπου "Plane" και επιστρέφει το διάνυσμα που προκύπτει.

- **Vector3 ClampMagnitude(Vector3 vector, float maxLength)**

Εφαρμόζει ένα μέγιστο όριο στο μέγεθος ενός διανύσματος και επιστρέφει το διάνυσμα που προκύπτει.

- **Vector3 Lerp(Vector3 a, Vector3 b, float t)**

Επιστρέφει το διάνυσμα που προκύπτει από τη γραμμική παρεμβολή μεταξύ των σημείων a και b. Ειδικότερα:

- Όταν είναι $t \leq 0$, η μέθοδος επιστρέφει το διάνυσμα a.
- Όταν $0 < t < 1$, η μέθοδος επιστρέφει ένα διάνυσμα που δείχνει κατά μήκος της γραμμής μεταξύ a και b. Η απόσταση κατά μήκος της γραμμής αντιστοιχεί στο κλάσμα που αναπαρίσταται από το t.
- Όταν $t > 1$, η μέθοδος επιστρέφει το διάνυσμα b.

5.1.4.1.4 Βασικές Πράξεις

Ο τύπος Vector3 προσφέρει την δυνατότητα πράξεων με διανύσματα:

- **+**: Πρόσθεση δύο διανυσμάτων
- **-**: Αφαίρεση δύο διανυσμάτων
- *****: Πολλαπλασιασμός ενός διανύσματος με έναν πραγματικό αριθμό
- **/**: Διαίρεση ενός διανύσματος με έναν πραγματικό αριθμό

5.1.4.1.5 Ενδεικτικά Παραδείγματα

Στη συνέχεια παρουσιάζονται ορισμένα ενδεικτικά παραδείγματα, σε γλώσσα c#, για τη χρήση των διανυσμάτων:

- Δημιουργία διανυσμάτων και βασικές πράξεις:

```
// Δημιουργία διανυσμάτων
Vector3 a = new Vector3(1f, 2f, 3f);
Vector3 b = new Vector3(0f, 1f, 0f);

// Βασικές Πράξεις
Vector3 sum = a + b;           // πρόσθεση
Vector3 diff = a - b;         // διαφορά
Vector3 scaled = a * 2f;      // κλιμάκωση του a επί 2
```

- Μέτρο

```
float len = a.magnitude;      // μέτρο (sqrt)
```

- Κίνηση και παρεμβολή

```
// δημιουργία διανύσματος με βάση την τρέχουσα θέση
// του κινούμενου αντικειμένου
Vector3 current = transform.position;

// δημιουργία τυχαίου διανύσματος
Vector3 target = new Vector3(10f, 20f, 5f);

// ομαλή κίνηση, από τρέχουσα θέση προς στόχο (target),
// με γραμμική παρεμβολή
transform.position = Vector3.Lerp(current, target, 0.1f);

// κίνηση, από τρέχουσα θέση προς στόχο (target),
// με σταθερή ταχύτητα
transform.position = Vector3.MoveTowards(current, target,
5f*Time.deltaTime);
```

- Προβολή

```
Vector3 v = new Vector3(1, -1, 0);

// προβολή του v πάνω στο up
Vector3 proj = Vector3.Project(v, Vector3.up);

// προβολή του v στο "up" επίπεδο
Vector3 onPlane = Vector3.ProjectOnPlane(v, Vector3.up);
```

5.1.4.2 Quaternion

Είναι ο τύπος που αντιπροσωπεύει γωνίες περιστροφής εκφραζόμενες ως τετραδόνια (quaternion) [61] [62]. Στο Unity, χρησιμοποιείται για τον προσδιορισμό περιστροφών εκφράζοντας προσανατολισμό.

Ένα τετραδόνιο αποτελείται από μία τετράδα πραγματικών αριθμών (x, y, z, w). Τα πρώτα τρία πεδία του τετραδονίου εκφράζουν το τρισδιάστατο διάνυσμα (x, y, z) ενώ το w έχει το νόημα του συντελεστή περιστροφής.

Συγκεκριμένα, για ένα τετραδόνιο που αντιπροσωπεύει μία γωνία περιστροφής θ γύρω από ένα άξονα u:

- $w = \cos(\theta/2)$
- $(x, y, z) = u * \sin(\theta/2)$

Έτσι, με βάση το παραπάνω τετραδόνιο, μπορεί να προκύψουν – αντίστροφα - τα ακόλουθα στοιχεία:

- $\theta = 2 * \arccos(w)$
- $u = (x, y, z) / \sqrt{1 - w^2}$

5.1.4.2.1 Βασικές Ιδιότητες

Οι βασικότερες ιδιότητες του τύπου είναι οι ακόλουθες:

- x: Η συντεταγμένη στον άξονα των x, του διανυσματικού μέρους
- y: Η συντεταγμένη στον άξονα των y, του διανυσματικού μέρους
- z: Η συντεταγμένη στον άξονα των z, του διανυσματικού μέρους

- w: Ο συντελεστής περιστροφής

5.1.4.2.2 Βασικές Μέθοδοι

Ο τύπος Quaternion προσφέρει μερικές πολύ χρήσιμες μεθόδους για την πραγματοποίηση διαφόρων υπολογισμών μεταξύ των τιμών του. Οι βασικότερες από αυτές είναι οι ακόλουθες:

- Quaternion AngleAxis(float angle, Vector3 axis)
Επιστρέφει τη γωνία που προκαλεί περιστροφή angle μοιρών γύρω από το διάνυσμα axis.
- Quaternion Euler(float x, float y, float z)
Επιστρέφει τη γωνία περιστροφής που καθορίζεται από τις γωνίες x, y και z. Η σειρά περιστροφής είναι ZXY.
- Quaternion FromToRotation(Vector3 fromDirection, Vector3 toDirection)
Επιστρέφει τη γωνία περιστροφής που ορίζεται από τα διανύσματα fromDirection και toDirection.
- Quaternion Slerp(Quaternion a, Quaternion b, float t)
Επιστρέφει τη γωνία περιστροφής από τη γραμμική παρεμβολή μεταξύ των a και b.
- Quaternion LookRotation(Vector3 forward, Vector3 upwards = Vector3.up)
Δημιουργεί μια περιστροφή με τα διανύσματα forward και upwards.

5.1.4.2.3 Ενδεικτικά Παραδείγματα

Στη συνέχεια παρουσιάζονται ορισμένα ενδεικτικά παραδείγματα, σε γλώσσα c#, για τη χρήση των τετραδονίων:

- Δημιουργία τετραδονίων:

```
// Δημιουργία περιστροφής 90° γύρω από τον άξονα Y
Quaternion qFromEuler = Quaternion.Euler(0f, 90f, 0f);

// Δημιουργία περιστροφής 45° γύρω από τον άξονα Z
Quaternion qFromAxis = Quaternion.AngleAxis(45f, Vector3.forward);
```

- Περιστροφή:

```
// Περιστροφή ώστε το αντικείμενο να "κοιτάει" προς target
Vector3 target = new Vector3(10f, 0f, 5f);
Quaternion look = Quaternion.LookRotation(
    target - transform.position, Vector3.up);

// Εφαρμογή περιστροφής σε διάνυσμα
Vector3 forward = Vector3.forward;
Vector3 rotated = qEuler * forward; // αποτέλεσμα: (≈ Vector3.right)

// Σύνθεση περιστροφών: πρώτα q1, μετά q2
Quaternion q1 = Quaternion.Euler(0f, 30f, 0f);
Quaternion q2 = Quaternion.Euler(10f, 0f, 0f);
Quaternion combined = q2 * q1;
transform.rotation = combined;
```

- Παρεμβολή:

```
// Slerp (ομαλή παρεμβολή)
Quaternion a = Quaternion.Euler(0f, 0f, 0f);
Quaternion b = Quaternion.Euler(0f, 180f, 0f);

Quaternion mid = Quaternion.Slerp(a, b, 0.5f);
```

5.1.4.3 Rigidbody

Η κλάση Rigidbody αντιπροσωπεύει όλα εκείνα τα αντικείμενα που μπορούν να επικοινωνήσουν με το υποσύστημα φυσικής του Unity Engine (physics) [63]. Έτσι, δίνει την δυνατότητα, σε μία εφαρμογή, να ελέγχει την δυναμική συμπεριφορά ενός αντικειμένου αυτής της κλάσης (όπως, για παράδειγμα, την κίνησή του, την ταχύτητά του, τη ροπή του κλπ.).

Ειδικότερα, για το VRacer, έχει τεράστια σημασία καθώς το αγωνιστικό αυτοκίνητο είναι ένα αντικείμενο της κλάσης αυτής (βλ. Ενότητα 5.1.3.2).

5.1.4.3.1 Βασικές Ιδιότητες

Οι βασικότερες ιδιότητες της κλάσης είναι οι ακόλουθες:

- Κίνηση
 - linearVelocity (Vector3): Γραμμική ταχύτητα
 - angularVelocity (Vector3): Γωνιακή ταχύτητα
 - position (Vector3): Θέση
 - rotation (Quaternion): Περιστροφή
 - maxLinearVelocity (float): Η μέγιστη γραμμική ταχύτητα του αντικειμένου (σε μέτρα ανά δευτερόλεπτο)
 - maxAngularVelocity (float): Η μέγιστη γωνιακή ταχύτητα του αντικειμένου (σε μέτρα ανά δευτερόλεπτο)
- Μάζα
 - mass (float): Μάζα
- Βαρύτητα / κίνηση
 - useGravity (bool): Καθορίζει αν το αντικείμενο επηρεάζεται από τη βαρύτητα
 - isKinematic (bool): Καθορίζει αν η φυσική επηρεάζει το αντικείμενο
- Οπτική ομαλότητα / ανίχνευση συγκρούσεων
 - interpolation (None / Interpolate / Extrapolate): Καθορίζει τον τρόπο διαχείρισης της εμφάνισης του αντικειμένου κατά τη ροή εκτέλεσης του παιχνιδιού
 - collisionDetectionMode (Discrete / Continuous / ContinuousDynamic / ContinuousSpeculative): Καθορίζει τη μέθοδο ανίχνευσης συγκρούσεων

- Περιορισμοί / ανίχνευση
 - constraints (RigidbodyConstraints): Θέτει περιορισμούς ως προς την περιστροφή ανά άξονα.
 - detectCollisions (bool): Καθορίζει αν θα είναι ενεργή η ανίχνευση συγκρούσεων

5.1.4.3.2 Βασικές Μέθοδοι

Οι βασικότερες μέθοδοι της κλάσης είναι οι ακόλουθες:

- Εφαρμογή δυνάμεων/ροπών
 - AddForce(Vector3 force, ForceMode mode = ForceMode.Force)
Εφαρμόζει συγκεκριμένου μεγέθους δύναμη, συνολικά, στο αντικείμενο, για την πρόσδοση ορμής.
 - AddRelativeForce(...)
Εφαρμόζει δύναμη στο αντικείμενο σχετικά με την τρέχουσα κατάστασή του.
 - AddForceAtPosition(Vector3 force, Vector3 position, ForceMode mode)
Εφαρμόζει δύναμη σε ένα σημείο του αντικειμένου, παράγοντας ροπή.
 - AddTorque(Vector3 torque, ForceMode mode)
Προσθέτει στο αντικείμενο ροπή.
- Κίνηση
 - MovePosition(Vector3 position)
Μετακινεί το αντικείμενο σε συγκεκριμένη θέση.
 - MoveRotation(Quaternion rot)
Περιστρέφει το αντικείμενο σε συγκεκριμένη γωνία.

5.1.4.3.3 Βασικά Μηνύματα

Το Unity Engine αποστέλλει διάφορα μηνύματα στα αντικείμενα της κλάσης Rigidbody. Τα βασικότερα από αυτά είναι τα ακόλουθα:

- Μηνύματα σύγκρουσης
 - OnCollisionEnter
Αποστέλλεται όταν το αντικείμενο έχει έρθει σε επαφή με ένα άλλο αντικείμενο.
 - OnCollisionExit
Αποστέλλεται όταν το αντικείμενο σταματά να είναι σε επαφή με άλλο αντικείμενο.

5.1.4.3.4 Ενδεικτικά Παραδείγματα

Στη συνέχεια παρουσιάζονται ορισμένα ενδεικτικά παραδείγματα, σε γλώσσα c#, για την διαχείριση ενός αντικειμένου τύπου "Rigidbody":

- Εφαρμογή δύναμης σε αντικείμενο Rigidbody (rb):

Στον παρακάτω κώδικα, χρησιμοποιείται η μέθοδος Rigidbody.AddForce() για να κινήσει το αντικείμενο προσθέτοντας φυσική δύναμη, αντί να αλλάζει απευθείας τη θέση του. Έτσι, η κίνηση είναι πιο ρεαλιστική.

```
// Διάβασμα δεδομένων κίνησης (από παίκτη)
float inputX = Input.GetAxis("Horizontal"); // Κίνηση μπρος-πίσω
float inputZ = Input.GetAxis("Vertical");    // Κίνηση δεξιά-αριστερά

// Ταχύτητα της κίνησης (πολλαπλασιαστής)
float forceFactor = 10f;

// Δημιουργία ενός διανύσματος δύναμης ανάλογα με τα δεδομένα κίνησης
Vector3 force = new Vector3(inputX, 0f, inputZ) * forceFactor;

// Εφαρμογή της δύναμης στο Rigidbody αντικείμενο
rb.AddForce(force);
```

- Ομαλή μετακίνηση ενός αντικειμένου Rigidbody (rb) με σταθερή ταχύτητα:

Στον παρακάτω κώδικα, μετακινείται το αντικείμενο με σταθερή ταχύτητα (velocity), χωρίς να εφαρμόζονται δυνάμεις.

```
// Διάβασμα δεδομένων κίνησης (από παίκτη)
float inputX = Input.GetAxis("Horizontal");
float inputZ = Input.GetAxis("Vertical");

// Κανονικοποίηση τους διανύσματος για απόδοση σταθερής ταχύτητας
// προς οποιαδήποτε κατεύθυνση
Vector3 direction = new Vector3(inputX, 0f, inputZ);
Vector3 directionNorm = direction.normalized;

// Απευθείας ορισμός της ταχύτητας του Rigidbody αντικειμένου
float speed = 10f;

rb.velocity = directionNorm * speed;
```

- Περιστροφή ενός αντικειμένου Rigidbody (rb) με φυσική ροπή:

Στον παρακάτω κώδικα, εφαρμόζεται ροπή περιστροφής ώστε το αντικείμενο να στρέφεται φυσικά αντί να αλλάζει άμεσα το transform.rotation.

```
float torque = 50f;

// Εφαρμογή ροπής για περιστροφή γύρω από τον άξονα Y,
// προς τα αριστερά
rb.AddTorque(Vector3.up * -torque);

// Εφαρμογή ροπής για περιστροφή γύρω από τον άξονα Y,
// προς τα δεξιά
rb.AddTorque(Vector3.up * torque);
```

- Στροφή ενός κινούμενου αντικειμένου Rigidbody (rb):

Στον παρακάτω κώδικα, το αντικείμενο στρίβει ελεγχόμενο από τη μηχανή φυσικής του Unity Engine.

```
// Διάβασμα δεδομένων κίνησης από χρήστη
float inputX = Input.GetAxis("Horizontal");

// Υπολογισμός του μεγέθους της περιστροφής για το τρέχον frame
float rotationSpeed = 100f; // Η ταχύτητα περιστροφής
float turnAmount = inputX * rotationSpeed * Time.fixedDeltaTime;

// Δημιουργία ενός Quaternion που αντιπροσωπεύει τη νέα περιστροφή
Quaternion newRotation = Quaternion.Euler(0f, turnAmount, 0f);

// Εφαρμογή νέας περιστροφής πάνω στην ήδη υπάρχουσα
rb.MoveRotation(rb.rotation * newRotation);
```

5.1.5 Κώδικας Εφαρμογής

Ο κώδικας της εφαρμογής βρίσκεται μέσα σε αρχεία κώδικα που, με τη σειρά τους, περιέχονται μέσα στο φάκελο "scripts" της εφαρμογής. Είναι δομημένος σε κλάσεις (class) της γλώσσας προγραμματισμού c#, έτσι ώστε κάθε αρχείο κώδικα να περιέχει μία ομώνυμη κλάση.

5.1.5.1 Ο Ελεγκτής CarController

Η κλάση CarController έχει ως ρόλο τον έλεγχο του αγωνιστικού αυτοκινήτου και, ταυτόχρονα, ρυθμίζει την επικοινωνία με τον χρήστη. Είναι αποκλειστικά συνδεδεμένη με το αντικείμενο του αυτοκινήτου (Car), προσδιορίζει τη συμπεριφορά του και, γενικά, είναι υπεύθυνη για όλες τις λειτουργίες του.

Βασίζεται στην κλάση "MonoBehaviour" του Unity Engine και, ως επακόλουθο, η δομή της απορρέει από τη λογική της κλάσης "MonoBehaviour". Ο κώδικας της περιέχεται στο αρχείο κώδικα "carController.cs" και προσφέρεται από το ΠΑΡΑΡΤΗΜΑ Α.

Ειδικότερα, τα βασικά της μέρη είναι τα ακόλουθα.

5.1.5.1.1 Ιδιότητες (Properties)

Πρόκειται για ένα σύνολο από ιδιότητες που, κατά κύριο λόγο, αποτελούν ρυθμίσεις για την επικοινωνία με τον χρήστη και μεταβλητές τον υπολογισμό των διαφόρων φυσικών μεγεθών (π.χ. η τρέχουσα ταχύτητα του αυτοκινήτου, η επιτάχυνσή του, η ένταση των διαφόρων δονήσεων κλπ.).

Οι βασικότερες από αυτές, είναι οι ακόλουθες:

- Ρυθμίσεις Σειριακής επικοινωνίας
 - portName: Όνομα σειριακής θύρας
Αν έχει τιμή και, ταυτόχρονα, η ιδιότητα udpRemoteIp είναι κενή τότε ενεργοποιείται η σειριακή επικοινωνία με το αυτοσχέδιο τιμόνι.
 - baudRate: Ρυθμός μετάδοσης δεδομένων σε baud

- Ρυθμίσεις Wi-Fi
 - udpRemoteIp: Διεύθυνση IP του μικροελεγκτή ESP32 του αυτοσχέδιου τιμονιού
 Αν έχει τιμή και, ταυτόχρονα, η ιδιότητα portName είναι κενή τότε ενεργοποιείται η ασύρματη επικοινωνία (μέσω Wi-Fi) με το αυτοσχέδιο τιμόνι.
 - udpRemotePort: Θύρα UDP για την αποστολή σημάτων (δόνησης) στο αυτοσχέδιο τιμόνι.
 - udpLocalPort: Θύρα UDP για τη λήψη σημάτων (κατεύθυνσης) από το αυτοσχέδιο τιμόνι.
- Ρυθμίσεις Αυτοκινήτου
 - maxSpeed: Μέγιστη ταχύτητα κίνησης
 - acceleration: Επιτάχυνση προς τα εμπρός/πίσω
 - brakeSmooth: Ομαλότητα φρεναρίσματος όταν δεν υπάρχει input
 - turnSpeed: Ταχύτητα στροφής του οχήματος
 - driftFactor: Πόσο γλιστράει το αυτοκίνητο κατά την πλαγιολίσθηση (drift) (0 = καθόλου, 1 = πλήρης πλαγιολίσθηση)
 - rayLength: Μήκος ακτίνας για έλεγχο εδάφους
 - minSteerFactor: Ελάχιστος συντελεστής τιμονιού σε χαμηλή ταχύτητα
 - driftTurnMultiplier: Ενίσχυση στροφής κατά την πλαγιολίσθηση (drift)
- Δόνηση Πλαγιολίσθησης (Haptics)
 - hapticAmplitude: Ένταση δόνησης κατά την πλαγιολίσθηση (drift)
 - hapticPulseDuration: Διάρκεια κάθε παλμού δόνησης
 - hapticRepeatInterval: Χρονικό διάστημα μεταξύ παλμών
- Δόνηση Σύγκρουσης (Haptics)
 - collisionMaxAmplitude: Μέγιστη ένταση δόνησης σε σύγκρουση
 - collisionPulseDuration: Διάρκεια δόνησης σύγκρουσης
 - collisionVelocityThreshold: Ελάχιστη ταχύτητα για να θεωρηθεί έγκυρη σύγκρουση
- Ήχος Σύγκρουσης
 - collisionClip: Audio clip σύγκρουσης
 - collisionClipVolume: Συνολική ένταση του ήχου σύγκρουσης
 - collisionPitchVariance: Τυχαία μεταβολή τόνου για ήχο σύγκρουσης

- Ήχος Μηχανής

- engineAudioSource: Audio Source που αναπαράγει τον ήχο της μηχανής (Αν είναι *null*, χρησιμοποιείται το Audio Source που έχει δηλωθεί στο αυτοκίνητο)
- engineMinVolume: Ελάχιστη ένταση ήχου μηχανής (σε μηδενική ταχύτητα)
- engineMaxVolume: Μέγιστη ένταση ήχου μηχανής (στη μέγιστη ταχύτητα)
- engineMinPitch: Ελάχιστος τόνος ήχου μηχανής (σε μηδενική ταχύτητα)
- engineMaxPitch: Μέγιστος τόνος ήχου μηχανής (στη μέγιστη ταχύτητα)
- engineUseThrottleForPitch: Εάν ο τόνος του ήχου μηχανής θα επηρεάζεται από το πόσο ο χρήστης πατάει το γκάζι (inputVertical) εκτός από την ταχύτητα

5.1.5.1.2 Μηνύματα (Messages)

Τα μηνύματα που δέχεται, και στα οποία "αντιδρά", ο ελεγκτής είναι τα ακόλουθα:

- Awake

Γενικά, πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο, κατά την δημιουργία του.

Κατά τη λήψη του συγκεκριμένου μηνύματος, εκτελούνται οι ακόλουθες βασικές αρχικοποιήσεις:

- Φορτώνεται το αρχείο παραμέτρων (βλ. Ενότητα 5.1.5.3, παρακάτω) και καθορίζονται οι παραπάνω ιδιότητες.
- Κλειδώνεται η περιστροφή στους άξονες X και Z ώστε το αυτοκίνητο να μη γέρνει.
- Τίθεται η μέθοδος παρεμβολής (interpolation) του αυτοκινήτου έτσι ώστε η ανανέωση της θέσης του να γίνεται ομαλά.
- Τίθεται η μέθοδος ανίχνευσης συγκρούσεων του αυτοκινήτου (collisionDetectionMode) έτσι ώστε αυτή – η ανίχνευση συγκρούσεων – να είναι συνεχής.
- Τοποθετείται το αυτοκίνητο πάνω στο έδαφος για την εκκίνηση.
- Μηδενίζεται η αρχική ταχύτητα του αυτοκινήτου.
- Ενεργοποιείται ο ήχος της μηχανής (αν έχει καθοριστεί)

- Start

Γενικά, πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο μίας σκηνής, κατά την αρχικοποίηση αυτής.

Κατά τη λήψη του συγκεκριμένου μηνύματος, ενεργοποιείται το κατάλληλο νήμα (thread) ανάγνωσης σημάτων κατεύθυνσης από το αυτοσχέδιο τιμόνι. Συγκεκριμένα, στην περίπτωση που η ιδιότητα udpRemoteIp δεν είναι κενή και η ιδιότητα portName είναι κενή, ενεργοποιείται η ανάγνωση μέσω Wi-Fi. Αλλιώς, στην περίπτωση που η ιδιότητα udpRemoteIp είναι κενή και η ιδιότητα portName όχι, ενεργοποιείται η ανάγνωση μέσω Σειριακής επικοινωνίας.

Να σημειωθεί ότι, στην περίπτωση που οι παραπάνω ιδιότητες είναι και οι δύο κενές, η κατεύθυνση του αυτοκινήτου καθορίζεται από το ίδιο το Unity Engine (καθώς αυτό λαμβάνει σήματα κατεύθυνσης είτε από το πληκτρολόγιο είτε από το VR χειριστήριο).

▪ Update

Γενικά, πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο, σε κάθε αλλαγή frame (κατά τη ροή εκτέλεσης του παιχνιδιού).

Κατά τη λήψη του συγκεκριμένου μηνύματος προσδιορίζεται η νέα θέση του αυτοκινήτου και προκαλούνται διάφορα ερεθίσματα (όπως ήχοι, δόνηση κλπ.), ανάλογα με την τρέχουσα κατάσταση του αυτοκινήτου.

Πιο αναλυτικά, εκτελούνται οι ακόλουθες λειτουργίες:

- Διαβάζονται τα δεδομένα κίνησης του αυτοκινήτου ανάλογα με την πηγή που έχει ενεργοποιηθεί κατά το Start (δηλ. Wi-Fi, Σειριακή θύρα ή Unity Engine).
- Παράγονται δονήσεις πλαγιολίσθησης.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `UpdateDriftHaptics`.

Η ένταση της δόνησης κατανέμεται στα VR χειριστήρια (αν ο χειρισμός του αυτοκινήτου γίνεται με αυτά) ανάλογα με την κατεύθυνση της στροφής.

Έτσι, σε μία δεξιά στροφή, παράγεται περισσότερη δόνηση στο δεξί χειριστήριο.

Αντιθέτως, σε μία αριστερή στροφή, παράγεται περισσότερη δόνηση στο αριστερό χειριστήριο.

- Παράγονται δονήσεις σύγκρουσης, για όσο διαρκεί ο χρονομετρητής της σύγκρουσης.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `UpdateCollisionHaptics`.

- Προσαρμόζεται η ένταση του ήχου της μηχανής ανάλογα με την ταχύτητα του αυτοκινήτου.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `UpdateEngineAudio`.

- Τέλος, τερματίζεται το παιχνίδι, κατά το πάτημα του πλήκτρου `Escape`.

▪ FixedUpdate

Γενικά, πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο, σε τακτά χρονικά διαστήματα, για την αναθεώρηση της τρέχουσας θέσης και κατάστασης του αντικειμένου (physics).

Κατά τη λήψη του συγκεκριμένου μηνύματος, εκτελούνται τα ακόλουθα:

- Διατηρείται το αυτοκίνητο στο έδαφος (κατά την κίνησή του).

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `StickToGroud`.

Αρχικά, η μέθοδος προσαρμόζει κατάλληλα τη θέση του αυτοκινήτου ως προς τη μορφολογία του εδάφους. Στη συνέχεια, προσαρμόζει κατάλληλα τη γραμμική του ταχύτητα.

- Διαχειρίζεται η επιτάχυνση και η επιβράδυνση του αυτοκινήτου, εξασφαλίζοντας ότι η κίνηση γίνεται πάνω στο επίπεδο του εδάφους.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `HandleMovement`.

Αρχικά, η μέθοδος υπολογίζει την κατεύθυνση του αυτοκινήτου. Στη συνέχεια, αν υπάρχει "input" (δηλ. ο οδηγός πατάει γκάζι), αυξάνει την ορμή του αυτοκινήτου.

Αντιθέτως – κάτι που υποδεικνύει ότι γίνεται φρενάρισμα - μειώνει ομαλά την ταχύτητα του αυτοκινήτου.

- Διαχειρίζεται τις στροφές που πρέπει να πάρει το αυτοκίνητο.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `HandleSteering`.

Αρχικά, η μέθοδος υπολογίζει τη γωνία της στροφής. Στη συνέχεια, περιστρέφει το αυτοκίνητο ως προς την παραπάνω γωνία.

- Πλαγιολισθαίνει το αυτοκίνητο, όταν οι τρέχουσες συνθήκες το επιβάλλουν.

Η συγκεκριμένη λειτουργία υλοποιείται, μέσα στην κλάση, από τη μέθοδο `ApplyDrift`.

Αρχικά, η μέθοδος υπολογίζει την πλευρική συνιστώσα της ταχύτητας. Αν υπάρχει ταυτόχρονα οριζόντιο και κάθετο "input" τότε θεωρεί ότι γίνεται πλαγιολίσθηση και μειώνει ελάχιστα την πλευρική ταχύτητα.

Αλλιώς (όταν δε γίνεται πλαγιολίσθηση), μειώνει αισθητά την πλευρική ταχύτητα.

▪ `OnCollisionEnter`

Γενικά, πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο που συμμετέχει σε σύγκρουση.

Κατά τη λήψη του συγκεκριμένου μηνύματος εκτελούνται τα ακόλουθα:

- Ρυθμίζεται η ένταση του ερεθίσματος της δόνησης και τίθεται ένας χρονομετρητής που καθορίζει πόσο θα διαρκέσει το ερέθισμα αυτό.
- Ρυθμίζεται η ένταση του ήχου της δόνησης και παράγεται άμεσα ο ήχος της.

5.1.5.2 Η Κλάση `CarXROriginFollow`

Η κλάση `CarXROriginFollow` είναι αποκλειστικά συνδεδεμένη με το αντικείμενο "XR Origin (Car)" (δηλ., ουσιαστικά, με την κάμερα) και είναι υπεύθυνη για την διατήρηση της κάμερας μέσα στο αυτοκίνητο.

Βασίζεται στην κλάση “MonoBehaviour” του Unity Engine. Ο κώδικάς της περιέχεται στο αρχείο κώδικα “CarXROriginFollow.cs” και προσφέρεται από το ΠΑΡΑΡΤΗΜΑ Β.

Η δομή της κλάσης απορρέει από τη λογική της κλάσης “MonoBehaviour” στην οποία βασίζεται. Ειδικότερα, τα βασικά της μέρη είναι τα ακόλουθα.

5.1.5.2.1 Μηνύματα (Messages)

Τα μηνύματα στα οποία “αντιδρά” η κλάση είναι τα ακόλουθα:

- LateUpdate

Πρόκειται για το μήνυμα που αποστέλλει το Unity Engine σε κάθε αντικείμενο, για κάθε αλλαγή frame (κατά τη ροή εκτέλεσης του παιχνιδιού), μετά από την ενημέρωση όλων των αντικειμένων. Έτσι, είναι εφικτή η εκτέλεση διαφόρων ενεργειών που πρέπει να γίνουν μετά την ανανέωση της θέσης του αυτοκινήτου.

Κατά τη λήψη του συγκεκριμένου μηνύματος, εκτελούνται τα ακόλουθα:

- Συσχετίζεται η θέση της κάμερας με τη θέση του αυτοκινήτου.
- Περιστρέφεται η κάμερα προς την κατεύθυνση του αυτοκινήτου.

5.1.5.3 Διαχείριση Παραμέτρων

Όλες οι βασικές ιδιότητες του ελεγκτή CarController (βλ. Ενότητα 5.1.5.1.1) μπορούν να καθοριστούν είτε από τον προγραμματιστή (κατά την ανάπτυξη της εφαρμογής VRacer) είτε από τον χρήστη (μετά από την εγκατάσταση του παιχνιδιού). Αυτό επιτρέπει την αλλαγή της συμπεριφοράς του αυτοκινήτου και του τρόπου χειρισμού του.

Συγκεκριμένα, για κάθε μία ιδιότητα, η εφαρμογή διαθέτει μία παράμετρο που την καθορίζει. Όλες οι παράμετροι είναι αποθηκευμένες σε ένα αρχείο παραμέτρων που ονομάζεται car_config.json. Όταν η εφαρμογή εκτελείται μέσα από τον Unity Editor, το αρχείο παραμέτρων βρίσκεται μέσα στο φάκελο Application.persistentDataPath του Unity. Αντιθέτως, όταν η εφαρμογή εκτελείται μετά από την εγκατάστασή της βρίσκεται στο φάκελο VRacer_Data\Managed της εγκατάστασης.

Κατά την εκκίνηση της εφαρμογής (Awake), φορτώνεται το αρχείο παραμέτρων και καθορίζονται οι ιδιότητες του ελεγκτή με βάση τις αντίστοιχες παραμέτρους.

Στην περίπτωση που το αρχείο παραμέτρων δε βρεθεί στο δίσκο τότε, οι ιδιότητες αποκτούν προεπιλεγμένες τιμές. Ταυτόχρονα, δημιουργείται το αρχείο παραμέτρων στο δίσκο χρησιμοποιώντας τις προεπιλεγμένες αυτές τιμές.

5.2 Αυτοσχέδιο Τιμόνι

Το αυτοσχέδιο τιμόνι, όπως προαναφέρθηκε στην Ενότητα 4.4.2, είναι μία σύνθετη κατασκευή που βασίζεται σε ένα ηλεκτρονικό κύκλωμα με βάση τον μικροελεγκτή ESP32. Συνεπώς, η κατασκευή του έχει δύο διαστάσεις: του hardware και του software.

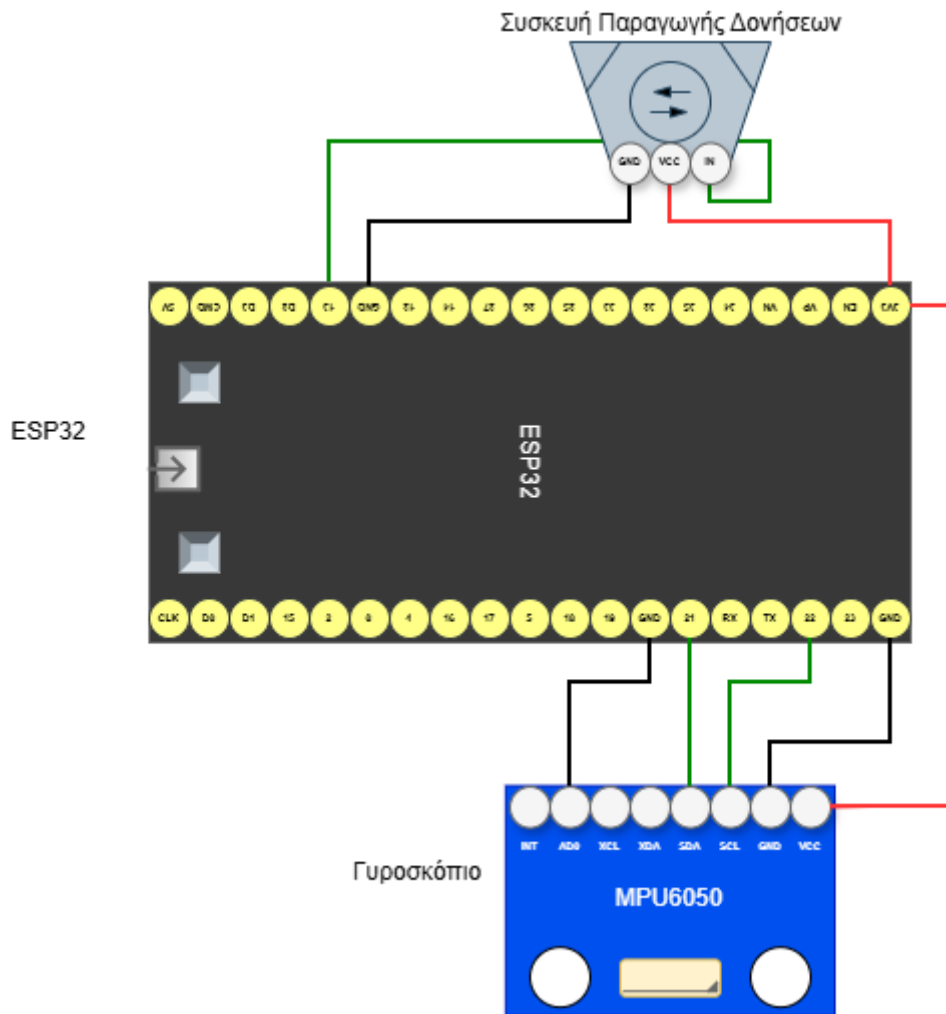
5.2.1 Ηλεκτρονικό Κύκλωμα

Με βάση την αρχιτεκτονική του τιμονιού (βλ. Ενότητα 4.4.2), το ηλεκτρονικό κύκλωμα του τιμονιού έχει δύο εκδοχές, ανάλογα με την επιθυμητή συσκευή μετάδοσης κίνησης.

5.2.1.1 Μετάδοση Κίνησης με Γυροσκόπιο

Το συγκεκριμένο κύκλωμα, του οποίου το διάγραμμα φαίνεται παρακάτω (βλ. Σχήμα 5.30), αποτελείται από τις ακόλουθες συσκευές:

- ESP32 των 38 ακροδεκτών (στη βασική του έκδοχή).
- Γυροσκόπιο MPU6050
- Συσκευή Παραγωγής Δονήσεων



Σχήμα 5.30: Διάγραμμα ηλεκτρονικού κυκλώματος τιμονιού με Γυροσκόπιο

Όπως προκύπτει από το παραπάνω διάγραμμα, οι συνδέσεις μεταξύ των συσκευών του κυκλώματος είναι οι ακόλουθες:

- Γυροσκόπιο - ESP32 (βλ. Πίνακας 5.5)

Πίνακας 5.5: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Γυροσκοπίου

Γυροσκόπιο	ESP32
AD0	GND
SDA	21
SCL	22
GND	GND

VCC	3V3
-----	-----

- Συσκευή Παραγωγής Δονήσεων - ESP32 (βλ. Πίνακας 5.6)

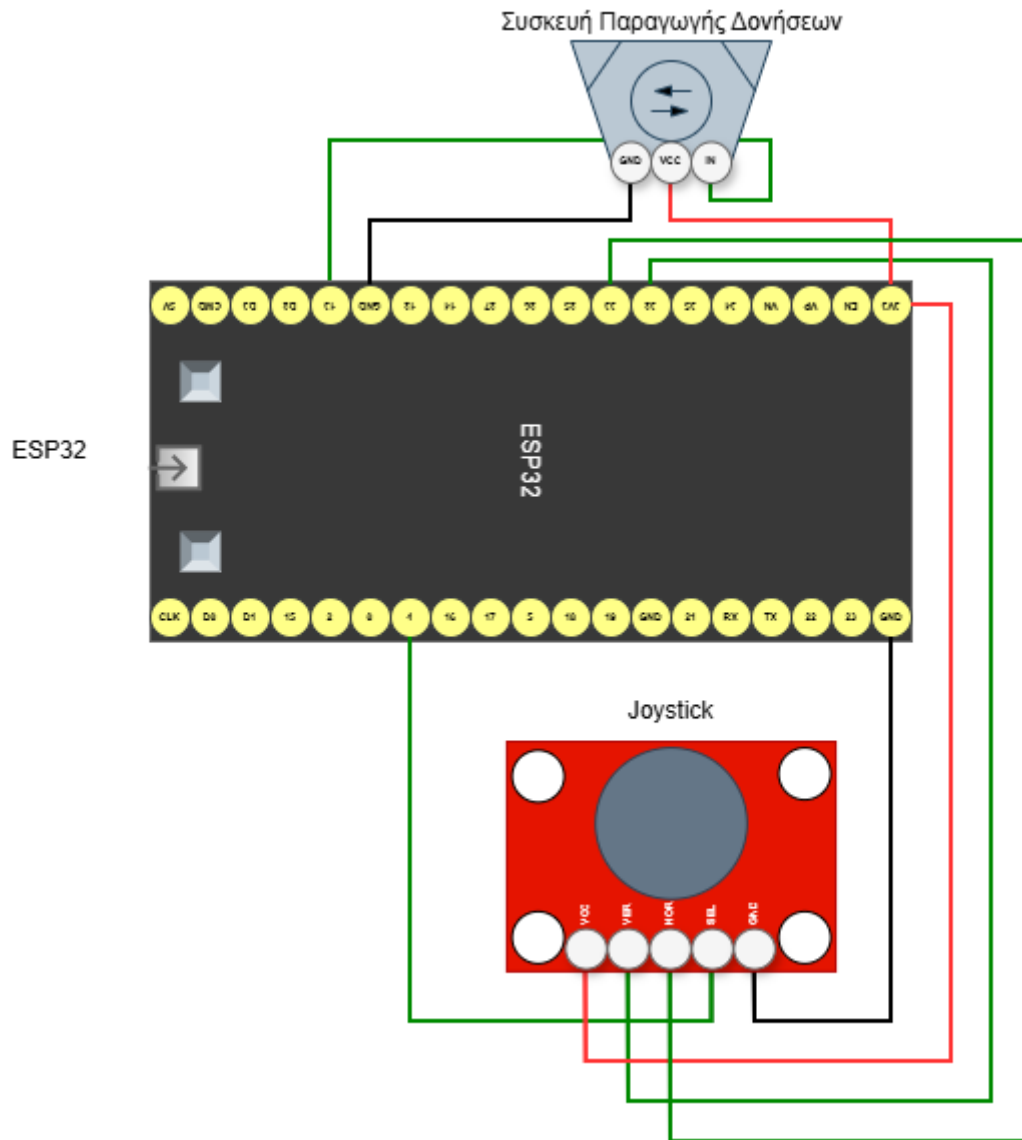
Πίνακας 5.6: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Συσκευής Παραγωγής Δονήσεων

Συσκευής Παραγωγής Δονήσεων	ESP32
GND	GND
VCC	3V3
IN	13

5.2.1.2 Μετάδοση Κίνησης με Joystick

Το συγκεκριμένο κύκλωμα, του οποίου το διάγραμμα φαίνεται παρακάτω (βλ. Σχήμα 5.31), αποτελείται από τις ακόλουθες συσκευές:

- ESP32 των 38 ακροδεκτών (στη βασική του εκδοχή).
- Joystick
- Συσκευή Παραγωγής Δονήσεων



Σχήμα 5.31: Διάγραμμα ηλεκτρονικού κυκλώματος τιμονιού με Joystick

Όπως προκύπτει από το παραπάνω διάγραμμα, οι συνδέσεις μεταξύ των συσκευών του κυκλώματος είναι οι ακόλουθες:

- Joystick - ESP32 (βλ. Πίνακας 5.7)

Πίνακας 5.7: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Joystick

Joystick	ESP32
VCC	3V3
VER	32
HOR	33
SEL	4
GND	GND

- Συσσκευή Παραγωγής Δονήσεων - ESP32 (βλ. Πίνακας 5.8)

Πίνακας 5.8: Συνδέσεις (ακροδεκτών) μεταξύ ESP32 και Συσσκευής Παραγωγής Δονήσεων

Συσσκευής Παραγωγής Δονήσεων	ESP32
GND	GND
VCC	3V3
IN	13

5.2.2 Κώδικας

Για να λειτουργήσει το ηλεκτρονικό κύκλωμα του τιμονιού, απαιτείται ο κατάλληλος προγραμματισμός του μικροελεγκτή ESP32. Αυτό πρακτικά σημαίνει ότι πρέπει να κατασκευαστεί ένα κατάλληλο πρόγραμμα (sketch) και το οποίο πρέπει να φορτωθεί (upload) στον μικροεπεξεργαστή ESP32.

5.2.2.1 Εργαλεία Ανάπτυξης

Όπως προαναφέρθηκε (βλ. Ενότητα 4.3.3), για τη συγγραφή του απαιτούμενου προγράμματος και τη φόρτωσή του στον μικροελεγκτή ESP32, απαιτείται η χρήση ενός κατάλληλου περιβάλλοντος προγραμματισμού. Για την τρέχουσα εργασία, χρησιμοποιήθηκε το περιβάλλον προγραμματισμού του Arduino με βάση το "ESP32 Dev Module" board.

5.2.2.2 Sketch

Σύμφωνα με το Arduino, ο κώδικας για τον προγραμματισμό συσκευών τοποθετείται μέσα σε αρχεία κώδικα που ονομάζονται *sketch*. Γενικά, κάθε sketch περιέχει κώδικα γραμμένο στη γλώσσα προγραμματισμού C++ και έχει την τυπική δομή που προβλέπει το ESP32.

Ο πλήρης κώδικας του τιμονιού παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Γ. Είναι κοινός και για τις δύο εκδοχές του ηλεκτρονικού κυκλώματος του τιμονιού (είτε με Γυροσκόπιο είτε με Joystick). Απλά, κατά την ενεργοποίηση του ηλεκτρονικού κυκλώματος, ο χρήστης καθορίζει αν, ως συσκευή μετάδοσης κίνησης, χρησιμοποιείται Γυροσκόπιο ή Joystick.

Ειδικότερα, τα βασικά στοιχεία του κώδικα είναι τα ακόλουθα.

5.2.2.2.1 Ιδιότητες (Properties)

Πρόκειται για ένα σύνολο από ιδιότητες που, κατά κύριο λόγο, αποτελούν ρυθμίσεις για την επικοινωνία με την Unity εφαρμογή VRacer και μεταβλητές για τη λειτουργία των διαφόρων συσκευών.

Οι βασικότερες από αυτές, είναι οι ακόλουθες:

- Γενικές Ιδιότητες
 - useGyro: Επιλογή εισόδου (ΑΛΗΘΗΣ = Γυροσκόπιο, ΨΕΥΔΗΣ = Joystick)
 - useWiFi: Επιλογή επικοινωνίας (ΑΛΗΘΗΣ = Wi-Fi, ΨΕΥΔΗΣ = Σειριακή)
 - udp: Αντικείμενο UDP για αποστολή και λήψη δεδομένων
 - wifiSSID: Περιέχει το δυναμικά καθορισμένο SSID του Wi-Fi
 - wifiPassword: Περιέχει τον δυναμικά καθορισμένο κωδικό του Wi-Fi
 - udpRemoteIP: Περιέχει την δυναμικά καθορισμένη IP διεύθυνση του μηχανήματος στο οποίο εκτελείται Unity εφαρμογή VRacer.

- Ιδιότητες Γυροσκοπίου
 - MPU: Αντικείμενο MPU6050 για την ανάγνωση δεδομένων από το γυροσκόπιο
- Ιδιότητες Joystick
 - valueX: Συντεταγμένη ως προς τον άξονα των X
 - valueY: Συντεταγμένη ως προς τον άξονα των Y
 - direction: Τρέχουσα κατεύθυνση
- Ιδιότητες Joystick
 - vibCmdBuf: Buffer για την αποθήκευση των χαρακτήρων ενός σήματος δόνησης (κατά την παραλαβή του)
 - vibCmdPos: Τρέχουσα θέση στο vibCmdBuf για τον επόμενο εισερχόμενο χαρακτήρα
 - vibActive: ΑΛΗΘΗΣ αν η συσκευή δόνησης είναι ενεργή (δηλ., έχει λάβει σήμα δόνησης που δεν έχει λήξει)
 - vibEndMs: Χρόνος (σε ms) κατά τον οποίο πρέπει να απενεργοποιηθεί η συσκευή δόνησης (στην περίπτωση που αυτή, προηγουμένως, έχει ενεργοποιηθεί)

5.2.2.2.2 Μέθοδοι (Methods)

Οι βασικές μέθοδοι του sketch είναι οι ακόλουθες:

- setup

Γενικά, πρόκειται για τη μέθοδο που ορίζει το ESP32 για την τοποθέτηση κώδικα που σχετίζεται με διάφορες ρυθμίσεις. Στο συγκεκριμένο sketch, αρχικοποιούνται οι διάφορες συσκευές υλικού και ρυθμίζεται η επικοινωνία με την Unity εφαρμογή VRacer.

Πιο αναλυτικά, εκτελούνται οι ακόλουθες λειτουργίες:

- Διαμορφώνεται η Συσκευή Δόνησης χρησιμοποιώντας το API LEDC του ESP32.
- Αρχικοποιείται η Συσκευή δόνησης και τίθεται εκτός λειτουργίας.
- Διαμορφώνεται το Γυροσκόπιο.
- Ενεργοποιείται η μόνιμη μνήμη του ESP32 (NVS) για αποθήκευση ρυθμίσεων.
- Φορτώνονται ή διαμορφώνονται οι διάφορες παράμετροι (εισόδου / επικοινωνίας).

Η παραπάνω λειτουργία εκτελείται από τη μέθοδο `configureCurrentOrNewSetup` και περιγράφεται, πιο αναλυτικά, στην παρακάτω Ενότητα..

- Αρχικοποιείται το WiFi (αν έχει επιλεγθεί ως μέθοδος επικοινωνίας).

- Εκκινείται η UDP επικοινωνία (αν το Wi-Fi συνδέθηκε επιτυχώς). Σε περίπτωση αποτυχίας σύνδεσης Wi-Fi, ενεργοποιείται η σειριακή επικοινωνία.
- Ενεργοποιείται η Σειριακή επικοινωνία (αν δεν επιλέχθηκε το Wi-Fi ως μέθοδος επικοινωνίας).

- loop

Γενικά, πρόκειται για τον κύριο βρόχο εκτέλεσης ενός sketch, όπως ορίζεται από το ESP32.

Πιο αναλυτικά, εκτελούνται οι ακόλουθες λειτουργίες:

- Λαμβάνεται (αν υπάρχει) κάποιο εισερχόμενο σήμα δόνησης και, στην περίπτωση που έχει ληφθεί τέτοιο, ενεργοποιείται η Συσκευή Παραγωγής Δονήσεων.

Η παραπάνω λειτουργία εκτελείται από τη μέθοδο `processVibrationCommands`.

- Αποστέλλονται σήματα κατεύθυνσης, προς την Unity εφαρμογή VRacer, όπως αυτά έχουν παραχθεί από την επιλεγμένη πηγή εισόδου (Γυροσκόπιο ή Joystick).

Η παραπάνω λειτουργία εκτελείται είτε από τη μέθοδο `sendGyroDirectionOutput` (για Γυροσκόπιο) είτε τη μέθοδο `sendJoystickDirectionOutput` (για Joystick).

- Ολοκληρώνεται η παραγωγή δόνησης, αν έχει ξεκινήσει κάποια τέτοια, και απενεργοποιείται η Συσκευή Παραγωγή Δόνησης.

5.2.2.3 Διαχείριση Παραμέτρων

Όλες οι βασικές ιδιότητες του παραπάνω sketch μπορούν να καθοριστούν από τον χρήστη, πριν από την πρώτη χρήση του τιμονιού. Αυτό επιτρέπει την αλλαγή της λειτουργίας του τιμονιού και του τρόπου επικοινωνίας με την Unity εφαρμογή VRacer.

Πιο συγκεκριμένα, για κάθε μία από τις ιδιότητες υπάρχει μία σχετική παράμετρος η οποία αποθηκεύεται στη μόνιμη μνήμη του ESP32 (NVS):

- Συσκευή μετάδοσης κίνησης (Γυροσκόπιο ή Joystick)
- Μέθοδος επικοινωνίας (Wi-Fi ή Σειριακή)
- SSID του Wi-Fi
- Κωδικός του Wi-Fi
- IP διεύθυνση του μηχανήματος στο οποίο εκτελείται Unity εφαρμογή VRacer.

Η διαχείριση των παραπάνω παραμέτρων γίνεται από τη μέθοδο `configureCurrentOrNewSetup` η οποία εκτελείται κατά την αρχικοποίηση (setup) του ηλεκτρονικού κυκλώματος. Μέσα της εκτελούνται οι ακόλουθες λειτουργίες:

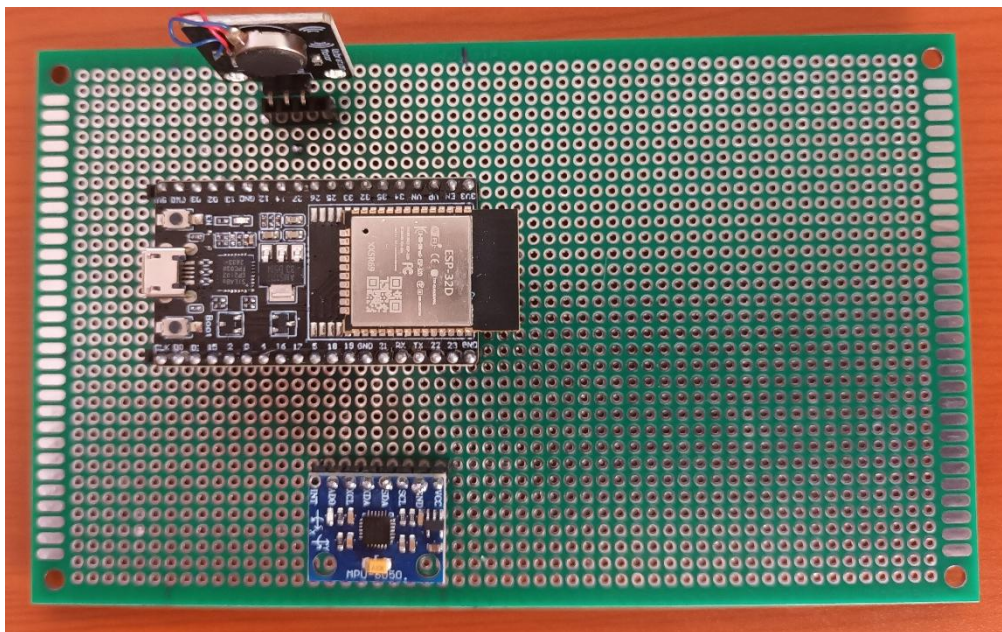
- Ελέγχεται αν υπάρχουν αποθηκευμένες τιμές, για τις παραμέτρους, στο NVS.
- Αν οι παράμετροι έχουν ήδη αποθηκευτεί:

- Εμφανίζονται οι τρέχουσες τιμές των παραμέτρων.
- Δίνεται η δυνατότητα – στον χρήστη – να τροποποιήσει τις παραμέτρους.
- Αν ο χρήστης επιλέξει να διατηρήσει τις ρυθμίσεις:
 - Η διαδικασία ολοκληρώνεται χωρίς να γίνει κάποια αλλαγή.
- Αν δεν έχει επιλεγεί νέα διαμόρφωση ή αν δεν υπάρχουν αποθηκευμένες ρυθμίσεις:
 - Λαμβάνονται οι νέες τιμές των παραμέτρων.
 - Αποθηκεύονται οι νέες τιμές στο NVS για μελλοντική χρήση.

5.2.3 Κατασκευή πρωτοτύπου

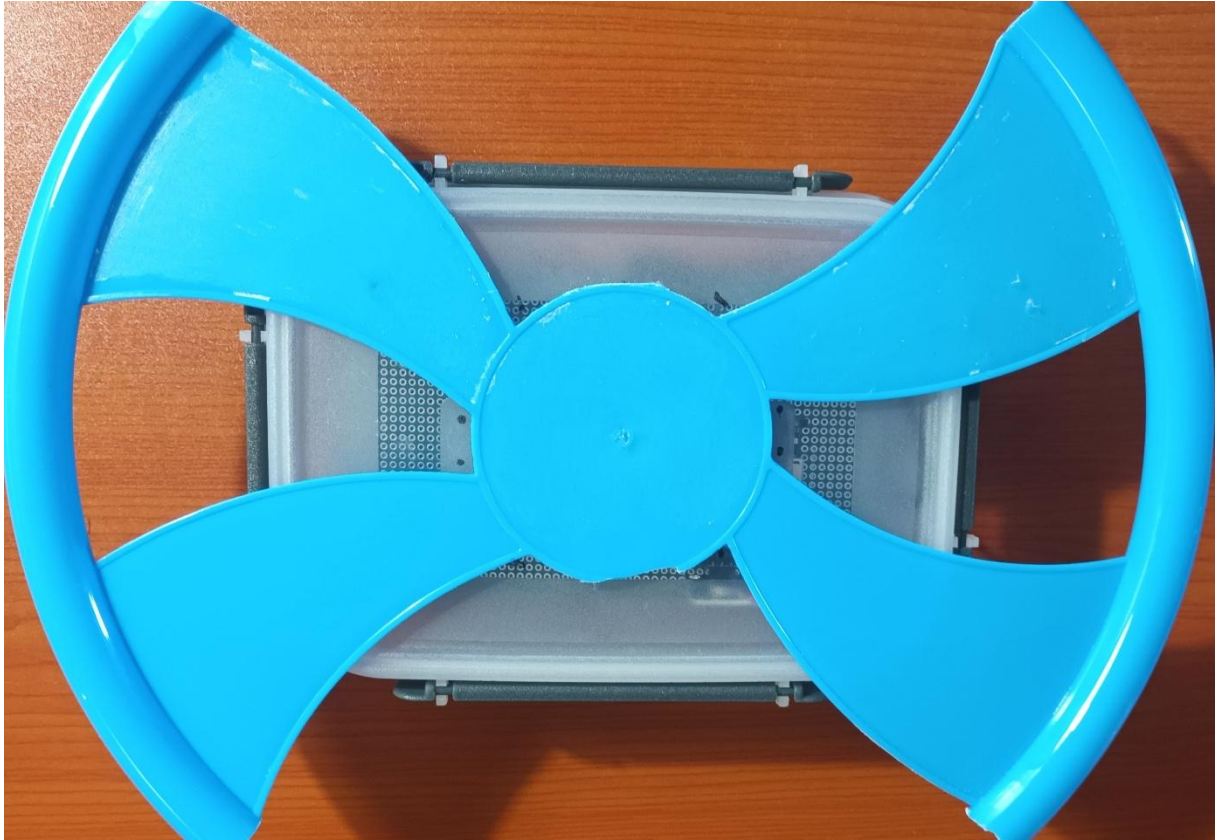
Στα πλαίσια της τρέχουσας Π.Ε., κατασκευάστηκε μία πρωτότυπη εκδοχή του τιμονιού, με βάση ένα ηλεκτρονικό κύκλωμα που χρησιμοποιεί Γυροσκόπιο για τη μετάδοση κίνησης (βλ. Ενότητα 5.2.2.1).

Καταρχάς, το ηλεκτρονικό κύκλωμα αναπτύχθηκε πάνω σε μία πλακέτα ανάπτυξης διπλής όψης (βλ. Σχήμα 5.32). Όλες οι συσκευές είναι τοποθετημένες με σταθερές συνδέσεις πάνω στη μία όψη της πλακέτας ενώ η καλωδίωση, που τις συνδέει, αναπτύχθηκε στην άλλη όψη της πλακέτας.



Σχήμα 5.32: Το πρωτότυπο ηλεκτρονικό κύκλωμα του τιμονιού με Γυροσκόπιο

Το παραπάνω κύκλωμα, περιέχεται σε ένα κουτί που είναι σταθερά προσαρτημένο στο κέντρο ενός πλαστικού αυτοσχέδιου "τιμονιού" (βλ. Σχήμα 5.33). Για την τροφοδοσία του με ηλεκτρικό ρεύμα – το κύκλωμα - συνοδεύεται από ένα power bank που είναι συνδεδεμένο με τον μικροελεγκτή ESP32 μέσω της USB θύρας του. Το power bank είναι επίσης τοποθετημένο στο ίδιο κουτί με τον μικροελεγκτή ESP32.



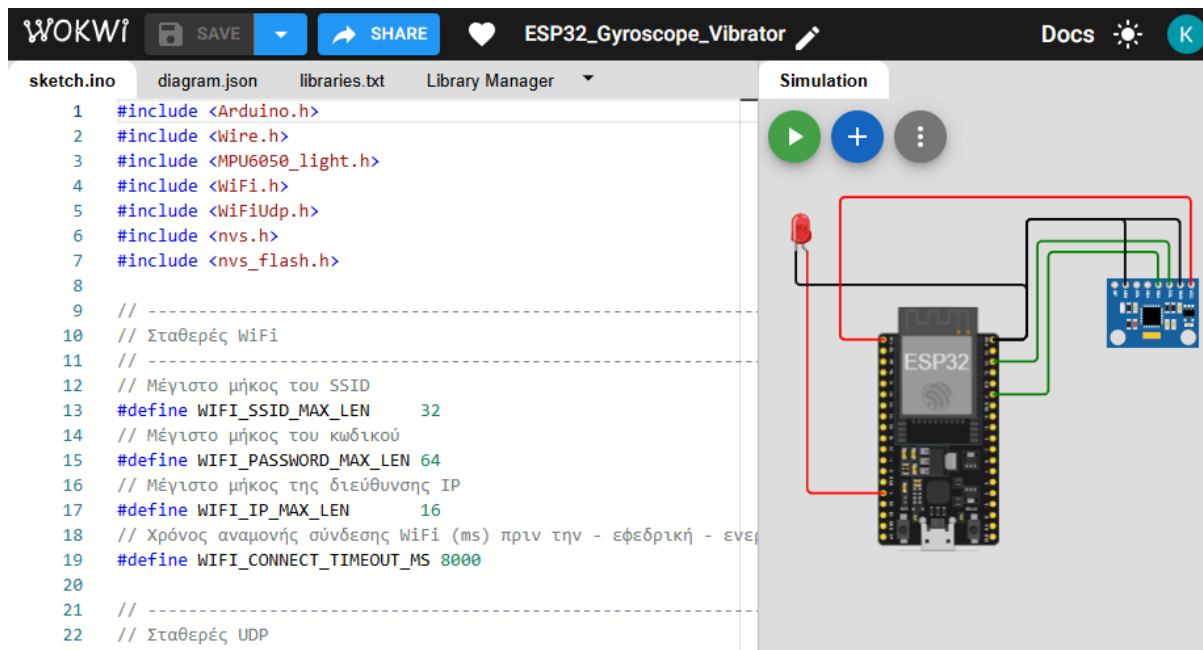
Σχήμα 5.33: Το αυτοσχέδιο "τιμόνι" του παιχνιδιού VRacer στην τελική του μορφή

5.3 Προσομοίωση Λειτουργίας

Το παιχνίδι VRacer αποτελεί σίγουρα μία σύνθετη κατασκευή. Για το λόγο αυτό, για κάθε συνθετικό της, είναι σημαντικό να ελεγχθεί ότι λειτουργεί σωστά. Αυτό επιτυγχάνεται με κατάλληλες μεθόδους προσομοίωσης. Με τη βοήθειά τους εντοπίζονται τα σφάλματα και επιβεβαιώνεται η ορθή λειτουργία τους.

5.3.1 Προσομοίωση Ηλεκτρονικού Κυκλώματος Τιμονιού

Ένα από τα πιο διαδεδομένα περιβάλλοντα προσομοίωσης ηλεκτρονικών κυκλωμάτων, με βάση τον μικροελεγκτή ESP32, είναι το Wokwi [64]. Το συγκεκριμένο περιβάλλον προσφέρει, καταρχάς, την δυνατότητα σχεδίασης ενός ηλεκτρονικού κυκλώματος και, στη συνέχεια, την προσομοίωση της λειτουργίας του (βλ. Σχήμα 5.34).



Σχήμα 5.34: Περιβάλλον προσομοίωσης ηλεκτρονικού κυκλώματος τιμονιού στο Wokwi

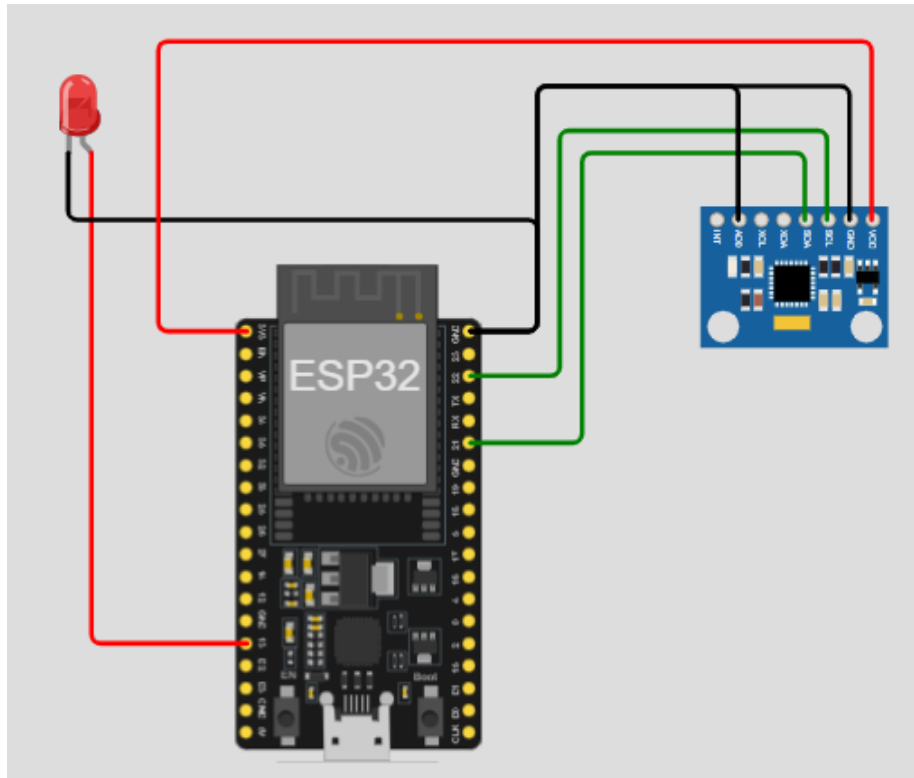
5.3.1.1 Ηλεκτρονικό Κύκλωμα

Όπως προαναφέρθηκε, το ηλεκτρονικό κύκλωμα έχει δύο εκδοχές (βλ. Ενότητα 5.2.1) οι οποίες λειτουργούν με βάση τον ίδιο προγραμματισμό (κώδικα). Συνεπώς, απαιτείται η προσομοίωση λειτουργίας και των δύο εκδοχών.

5.3.1.1.1 Μετάδοση Κίνησης με Γυροσκόπιο

Μέσα στο περιβάλλον του Wokwi, το διάγραμμα του προσομοιωμένου κυκλώματος, που περιγράφεται στην Ενότητα 5.2.1.1, φαίνεται παρακάτω (βλ. Σχήμα 5.35).

Κατά την προσομοίωση του κυκλώματος, αντί για Συσκευή Παραγωγή Δόνησης (που το Wokwi δεν παρέχει), χρησιμοποιείται ένας λαμπτήρας με την ίδια ακριβώς συνδεσμολογία.

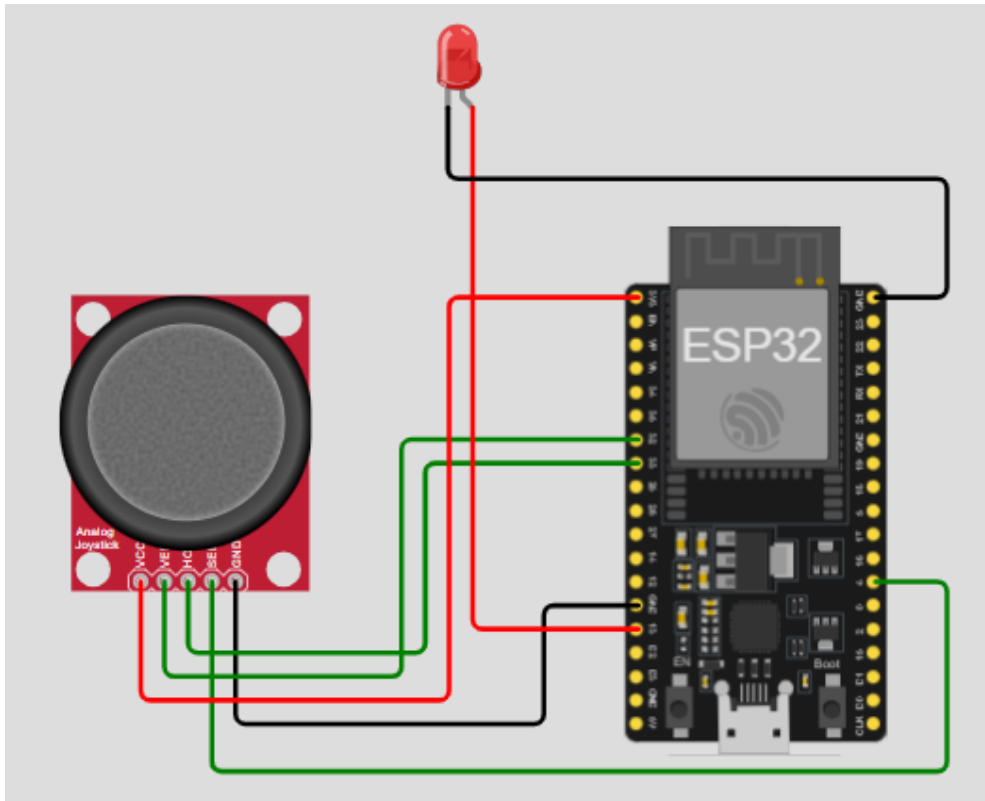


Σχήμα 5.35: Προσομοίωση ηλεκτρονικού κυκλώματος τιμονιού με Γυροσκόπιο

5.3.1.1.2 Μετάδοση Κίνησης με Joystick

Μέσα στο περιβάλλον του Wokwi, το διάγραμμα του προσομοιωμένου κυκλώματος, που περιγράφεται στην Ενότητα 5.2.1.2, φαίνεται παρακάτω (βλ. Σχήμα 5.36).

Κατά την προσομοίωση του κυκλώματος, αντί για Συσκευή Παραγωγή Δόνησης (που το Wokwi δεν παρέχει), χρησιμοποιείται ένας λαμπτήρας με την ίδια ακριβώς συνδεσμολογία.



Σχήμα 5.36: Προσομοίωση ηλεκτρονικού κυκλώματος τιμονιού με Joystick

5.3.1.2 Κώδικας

Οι δύο παραπάνω εκδοχές – του κυκλώματος – βασίζονται στον κώδικα που περιγράφεται στην Ενότητα 5.2.2.2 και παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Γ. Αλλά, κατά την εκτέλεση του κυκλώματος, ο χρήστης καθορίζει τη συσκευή μετάδοσης κίνησης.

5.3.1.3 Εκτέλεση Προσομοίωσης

Κατά την εκτέλεση του κυκλώματος, ο χρήστης έχει την δυνατότητα να προκαλεί κίνηση χρησιμοποιώντας κατάλληλα (με το ποντίκι και το πληκτρολόγιο) είτε το Γυροσκόπιο είτε το Joystick. Σ' αυτή την περίπτωση, τα σήματα κίνησης φαίνονται στην οθόνη ως ζεύγη τιμών της ακόλουθης μορφής (οι τιμές διαχωρίζονται με κόμμα):

Συντελεστής_ταχύτητας_για_τον_άξονα_x, Συντελεστής_ταχύτητας_για_τον_άξονα_y

Επιπρόσθετα, ο χρήστης μπορεί να εισάγει σήματα δόνησης ως ζεύγη τιμών της ακόλουθης μορφής (οι τιμές διαχωρίζονται με κόμμα):

Ένταση_δόνησης, Διάρκεια_δόνησης

Το αποτέλεσμα, κάθε σήματος δόνησης, είναι η ενεργοποίηση του λαμπτήρα με την εισαγομένη ένταση και με διάρκεια ανάλογη του σήματος.

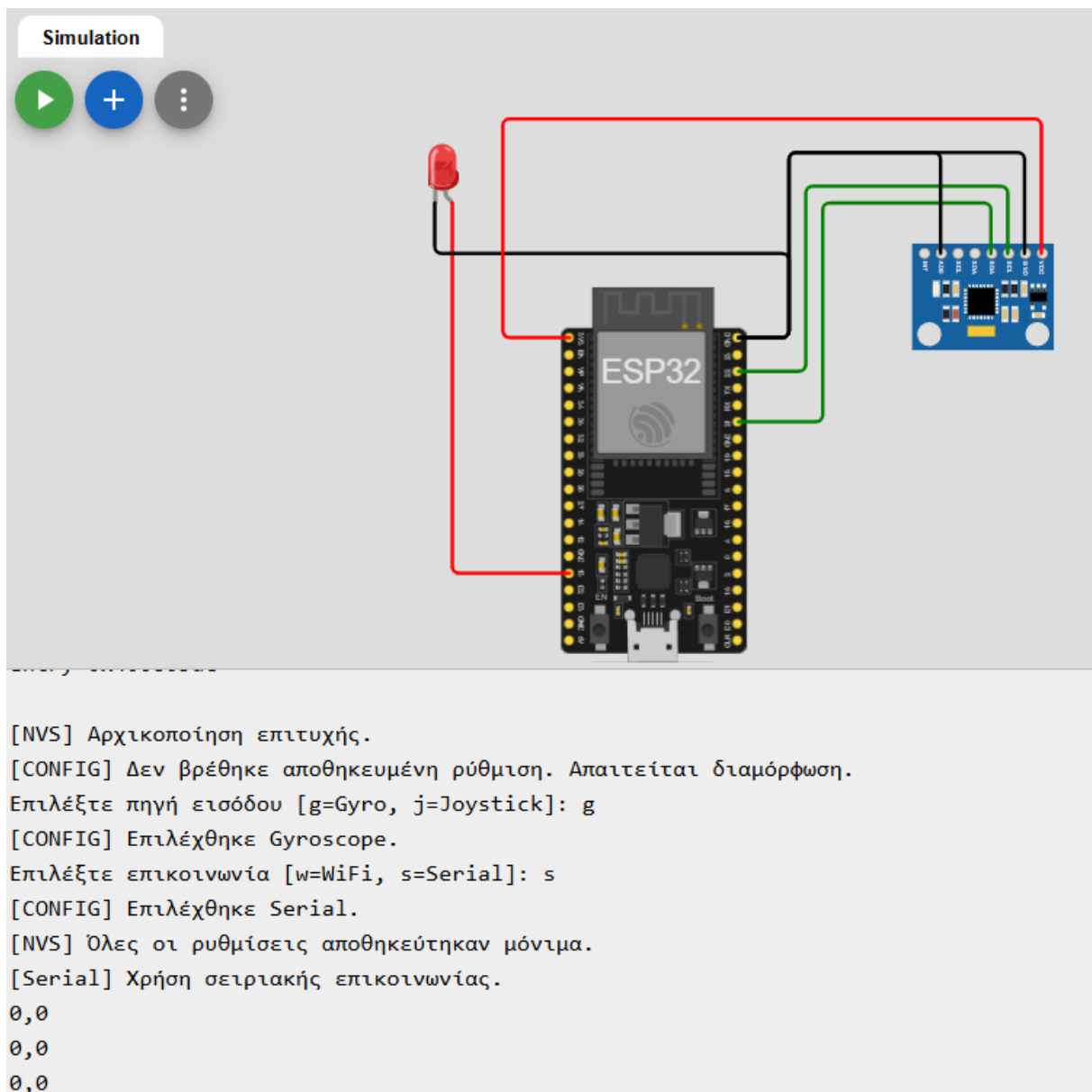
5.3.1.4 Σενάρια Ελέγχου

Κατά την προσομοίωση του ηλεκτρονικού κυκλώματος είναι δυνατόν να εκτελεστούν διάφορα σενάρια ελέγχου με σκοπό τον εντοπισμό σφαλμάτων είτε στο ίδιο το κύκλωμα είτε στον προγραμματισμό του μικροελεγκτή ESP32.

Στη συνέχεια, παρουσιάζονται μερικά ενδεικτικά σενάρια ελέγχου.

5.3.1.4.1 Μετάδοση κίνησης με Γυροσκόπιο μέσω Σειριακής επικοινωνίας

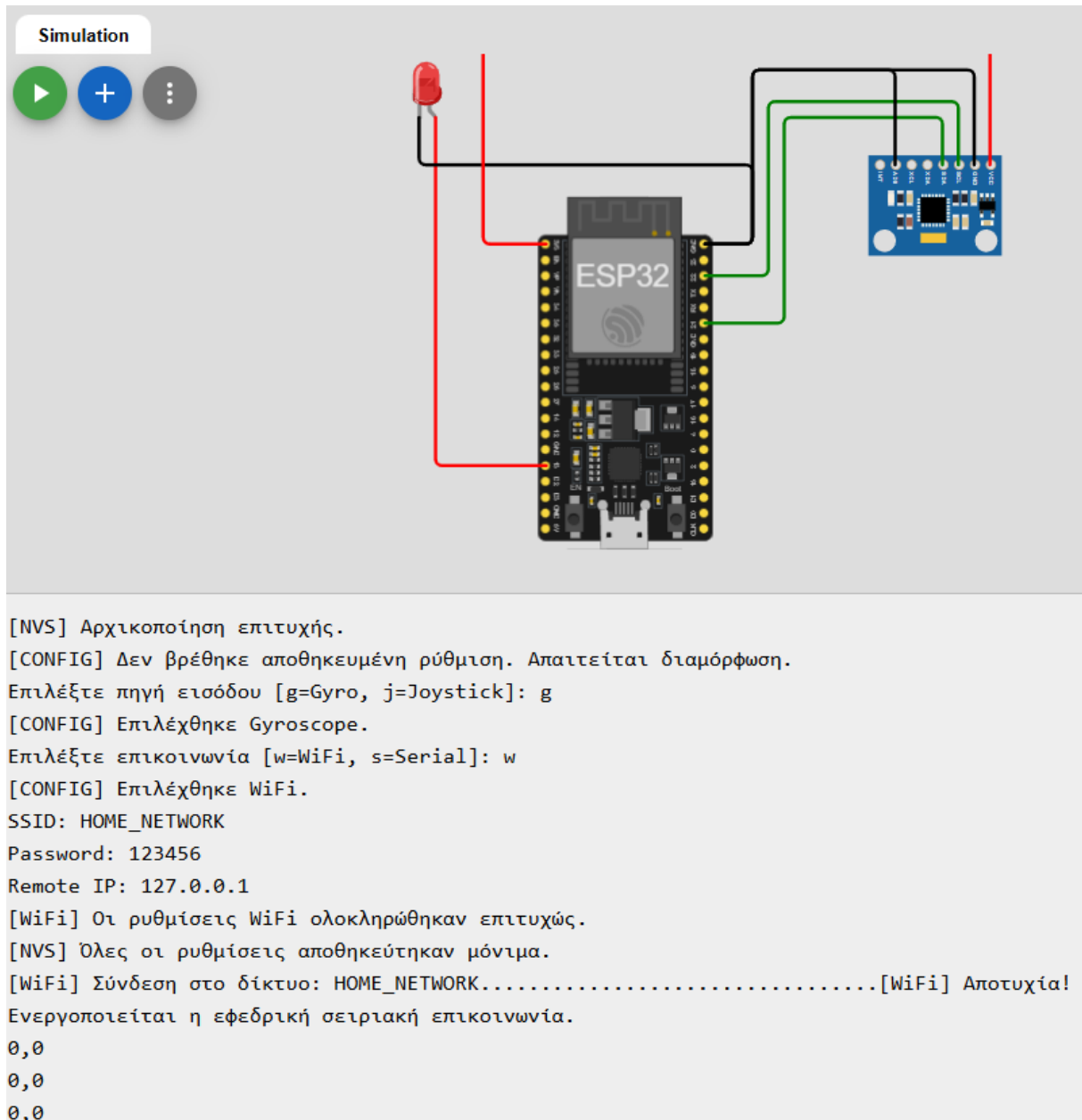
Στο σενάριο ελέγχου που παρουσιάζεται στο παρακάτω σχήμα (βλ. Σχήμα 5.37), ελέγχεται αν το κύκλωμα μεταδίδει σήματα κίνησης, με τη χρήση Γυροσκοπίου, μέσω Σειριακής επικοινωνίας.



Σχήμα 5.37: Σενάριο ελέγχου μετάδοσης κίνησης με Γυροσκόπιο μέσω Σειριακής επικοινωνίας

5.3.1.4.2 Εφεδρική ενεργοποίηση Σειριακής επικοινωνίας

Στο σενάριο ελέγχου που παρουσιάζεται στο παρακάτω σχήμα (βλ. Σχήμα 5.38), ελέγχεται αν έχει υλοποιηθεί σωστά η απαίτηση της - εφεδρικής - ενεργοποίησης της σειριακής επικοινωνίας στην περίπτωση που, αρχικά, είχε επιλεγθεί η επικοινωνία μέσω Wi-Fi.



Σχήμα 5.38: Σενάριο ελέγχου ενεργοποίηση σειριακής επικοινωνίας σε περίπτωση αποτυχημένης ενεργοποίησης Wi-Fi επικοινωνίας

5.3.2 Προσομοίωση Λειτουργίας Παιχνιδιού

Όπως κάθε Unity εφαρμογή έτσι και η VRacer μπορεί να εκτελεστεί με τη χρήση του πληκτρολογίου. Αυτό, αυτομάτως, δίνει την δυνατότητα ελέγχου της βασικής λειτουργικότητας της εφαρμογής. Η απαίτηση όμως χειρισμού του αγωνιστικού αυτοκινήτου μέσω Wi-Fi ή Σειριακής επικοινωνίας επιβάλλει την ανάγκη σχετικών ελέγχων.

Έτσι, για κάθε μέσο επικοινωνίας, έχει αναπτυχθεί ένα ειδικό περιβάλλον προσομοίωσης με βάση τη γλώσσα προγραμματισμού Python και τις άφθονες σχετικές βιβλιοθήκες που προσφέρονται. Πρακτικά, τα περιβάλλοντα αυτά προσομοιώνουν το αυτοσχέδιο τιμόνι και χρησιμοποιούν είτε το Wi-Fi είτε τη Σειριακή επικοινωνία για να επικοινωνήσουν με τη Unity εφαρμογή VRacer.

5.3.2.1 Επικοινωνία μέσω Wi-Fi

Για την προσομοίωση λειτουργίας του παιχνιδιού μέσω Wi-Fi αναπτύχθηκε, στο περιβάλλον προγραμματισμού Visual Studio, ένα Python *solution* με όνομα VRacer_Arduino_Simulator_Wifi.slnx. Στο *solution* αυτό, ο απαιτούμενος κώδικας περιέχεται στο Python script με όνομα VRacer_Arduino_Simulator_Wifi.py (βλ. ΠΑΡΑΡΤΗΜΑ Δ).

5.3.2.1.1 Λειτουργίες

Οι βασικές λειτουργίες του παραπάνω Python script είναι οι ακόλουθες:

- Στέλνει σήματα κατεύθυνσης, στη Unity εφαρμογή VRacer, ως ASCII συμβολοσειρές της μορφής "x,y\r\n". Η αποστολή αυτή γίνεται μέσω Wi-Fi και UDP. Επίσης, τα σήματα παράγονται από τον χρήστη χρησιμοποιώντας τα πλήκτρα-βέλη του πληκτρολογίου ως ακολούθως:
 - LEFT → "-1,0\r\n"
 - RIGHT → "1,0\r\n"
 - FORWARD → "0,1\r\n"
 - BACKWARD → "0,-1\r\n"
 - LEFT + FORWARD → "-1,1\r\n"
 - LEFT + BACKWARD → "-1,-1\r\n"
 - RIGHT + FORWARD → "1,1\r\n"
 - RIGHT + BACKWARD → "1,-1\r\n"
 - (κανένα βέλος) → "0,0\r\n"
- Λαμβάνει σήματα δόνησης από τη Unity εφαρμογή μέσω UDP, τα οποία είναι σε μορφή "<intensity>,<duration_ms>\r\n" (π.χ. "128,500\r\n" για ένταση 128 και διάρκεια 500ms), και εμφανίζει την ένταση και τη διάρκεια στην οθόνη.

5.3.2.1.2 Υλοποίηση

Στο script, για την επικοινωνία μέσω Wi-Fi, χρησιμοποιείται η βιβλιοθήκη socket της Python. Οι βασικές ενέργειες που εκτελούνται από αυτό είναι οι ακόλουθες:

- Εμφανίζεται ένα συνοπτικό μήνυμα οδηγιών – προς τον χρήστη – για τη χρήση των πλήκτρων-βελών και τον τερματισμό της εκτέλεσης (δηλ. με το πάτημα του πλήκτρου ESC).
- Εκκινείται ένα νήμα (thread) για τη λήψη σημάτων δόνησης – μέσω Wi-Fi UDP - από τη Unity εφαρμογή VRacer.
- Εκκινείται ένας listener για την αναγνώριση του πατήματος των πλήκτρων-βελών, μέχρις ότου πατηθεί το πλήκτρο ESC.

5.3.2.1.3 Εκτέλεση Προσομοίωσης

Για την εκτέλεση του script, θα πρέπει, αρχικά, να καθοριστούν οι ακόλουθες παράμετροι:

- UDP_REMOTE_IP: Διεύθυνση IP του μηχανήματος που τρέχει το VRacer_Unity (προτείνεται η 127.0.0.1)

- UDP_REMOTE_PORT: UDP θύρα που ακούει το VRacer_Unity για σήματα κατεύθυνσης
- UDP_LOCAL_PORT: Τοπική UDP θύρα για λήψη σημάτων δόνησης από το VRacer_Unity

Επιπρόσθετα, η Unity εφαρμογή VRacer θα πρέπει να ρυθμιστεί έτσι ώστε να επικοινωνεί με βάση το Wi-Fi. Για να γίνει αυτό, θα πρέπει να καθοριστούν οι ακόλουθες παράμετροι στο αρχείο παραμέτρων της εφαρμογής VRacer_Unity:

- udpRemoteIp = Η Διεύθυνση IP του μηχανήματος που τρέχει το script (προτείνεται η 127.0.0.1)
- udpRemotePort = Η Θύρα UDP για την αποστολή σημάτων (δόνησης) στο script
- udpLocalPort = Η Θύρα UDP για τη λήψη σημάτων (κατεύθυνσης) από το script
- portName = κενό αλφαριθμητικό

Μετά από τις παραπάνω ρυθμίσεις, αρκεί να γίνουν τα ακόλουθα:

- Εκτελείται η εφαρμογή VRacer_Unity μέσα από τον Unity Editor.
- Εκτελείται το script VRacer_Arduino_Simulator_Wifi.py.

Ο χρήστης χρησιμοποιεί τα πλήκτρα-βέλη του πληκτρολογίου για να προσδώσει κίνηση στο αγωνιστικό αυτοκίνητο.

5.3.2.2 Σειριακή Επικοινωνία

Για την προσομοίωση λειτουργίας του παιχνιδιού μέσω Σειριακής Επικοινωνίας αναπτύχθηκε, στο περιβάλλον προγραμματισμού Visual Studio, ένα Python solution με όνομα VRacer_Arduino_Simulator_Serial_Bridge.slnx. Στο solution αυτό, ο απαιτούμενος κώδικας περιέχεται στο Python script με όνομα VRacer_Arduino_Simulator_Serial_Bridge.py (βλ. ΠΑΡΑΡΤΗΜΑ Ε).

Επιπλέον, για την επίτευξη της Σειριακής Επικοινωνίας μεταξύ του παραπάνω Python script και της Unity εφαρμογής VRacer, απαιτείται η ανάπτυξη, στο εσωτερικό του μηχανήματος που εκτελούνται οι δύο εφαρμογές, μίας Γέφυρας Σειριακής επικοινωνίας (Virtual Serial Port Bridge) [65]. Έτσι, δημιουργείται ένα "ζευγάρι" εικονικών Σειριακών Θυρών (π.χ. COM8 και COM9) οι οποίες επικοινωνούν αμφίδρομα. Δηλαδή, ότι γράφεται στην COM8 εμφανίζεται στην COM9 και αντιστρόφως.

Για τις ανάγκες της τρέχουσας προσομοίωσης, το Python script VRacer_Arduino_Simulator_Serial_Bridge.py αποστέλλει σήματα κίνησης στην εικονική θύρα που αποτελεί το ένα άκρο της παραπάνω γέφυρας. Την ίδια στιγμή, η Unity εφαρμογή VRacer λαμβάνει τα σήματα αυτά μέσω της εικονικής θύρας που αποτελεί το άλλο άκρο της γέφυρας. Τα σήματα δόνησης, βέβαια, ακολουθούν την αντίστροφη πορεία.

5.3.2.2.1 Λειτουργίες

Οι βασικές λειτουργίες του παραπάνω Python script είναι οι ακόλουθες:

- Στέλνει σήματα κατεύθυνσης, στη Unity εφαρμογή VRacer, ως ASCII συμβολοσειρές της μορφής "x,y\r\n". Η αποστολή αυτή γίνεται μέσω Σειριακής Επικοινωνίας. Επίσης, τα σήματα παράγονται από τον χρήστη χρησιμοποιώντας τα πλήκτρα-βέλη του πληκτρολογίου ως ακολούθως:

- LEFT → "-1,0\r\n"
 - RIGHT → "1,0\r\n"
 - FORWARD → "0,1\r\n"
 - BACKWARD → "0,-1\r\n"
 - LEFT + FORWARD → "-1,1\r\n"
 - LEFT + BACKWARD → "-1,-1\r\n"
 - RIGHT + FORWARD → "1,1\r\n"
 - RIGHT + BACKWARD → "1,-1\r\n"
 - (κανένα βέλος) → "0,0\r\n"
- Λαμβάνει σήματα δόνησης από τη Unity εφαρμογή μέσω Σειριακής επικοινωνίας, τα οποία είναι σε μορφή "<intensity>,<duration_ms>\r\n" (π.χ. "128,500\r\n" για ένταση 128 και διάρκεια 500ms), και εμφανίζει την ένταση και τη διάρκεια στην οθόνη.

5.3.2.2.2 Υλοποίηση

Καταρχάς, για την υλοποίηση της Γέφυρας Σειριακής Επικοινωνίας, χρησιμοποιείται το ελεύθερο λογισμικό Com0Com [65]. Με τη χρήση του, δημιουργείται – στο μηχανήμα εκτέλεσης του Python script VRacer_Arduino_Simulator_Serial_Bridge.py και της Unity εφαρμογής VRacer - μία εικονική γέφυρα μεταξύ δύο διαθέσιμων Θυρών (π.χ. COM8 και COM9).

Στο Python script VRacer_Arduino_Simulator_Serial_Bridge.py, για την επίτευξη της Σειριακής επικοινωνίας, χρησιμοποιείται η βιβλιοθήκη serial της Python. Οι βασικές ενέργειες που εκτελούνται από αυτό είναι οι ακόλουθες:

- Εμφανίζεται ένα συνοπτικό μήνυμα οδηγιών – προς τον χρήστη – για τη χρήση των πλήκτρων-βελών και τον τερματισμό της εκτέλεσης (δηλ. με το πάτημα του πλήκτρου ESC).
- Εκκινείται ένα νήμα (thread) για τη λήψη σημάτων δόνησης – μέσω Σειριακής Επικοινωνίας - από τη Unity εφαρμογή VRacer.
- Εκκινείται ένας listener για την αναγνώριση του πατήματος των πλήκτρων-βελών, μέχρις ότου πατηθεί το πλήκτρο ESC.

5.3.2.2.3 Εκτέλεση Προσομοίωσης

Για την εκτέλεση του script, θα πρέπει, αρχικά, να καθοριστούν οι ακόλουθες παράμετροι:

- PORT: Η σειριακή θύρα (COM) που έχει "γεφυρωποιηθεί" με τη σειριακή θύρα από την οποία δέχεται σήματα κατεύθυνσης το VRacer_Unity (π.χ. COM9)
- BAUDRATE: Ταχύτητα μετάδοσης κατά τη Σειριακή Επικοινωνία (baud rate)

Επιπρόσθετα, η Unity εφαρμογή VRacer θα πρέπει να ρυθμιστεί έτσι ώστε να επικοινωνεί Σειριακά. Για να γίνει αυτό, θα πρέπει να καθοριστούν οι ακόλουθες παράμετροι στο αρχείο παραμέτρων της εφαρμογής VRacer_Unity:

- portName: Όνομα της σειριακής θύρας μέσω της οποίας δέχεται σήματα κατεύθυνσης το VRacer_Unity
- baudRate: Ρυθμός μετάδοσης δεδομένων σε baud

- udpRemoteIp: κενό αλφαριθμητικό

Μετά από τις παραπάνω ρυθμίσεις, αρκεί να γίνουν τα ακόλουθα:

- Εκτελείται η εφαρμογή VRacer_Unity μέσα από τον Unity Editor.
- Εκτελείται το script VRacer_Arduino_Simulator_Wifi.py.

Ο χρήστης χρησιμοποιεί τα πλήκτρα-βέλη του πληκτρολογίου για να προσδώσει κίνηση στο αγωνιστικό αυτοκίνητο.

5.3.2.3 Σενάρια Ελέγχου

Κατά την προσομοίωση λειτουργίας του παιχνιδιού μέσω Wi-Fi ή Σειριακής επικοινωνίας, είναι δυνατόν να εκτελεστούν διάφορα σενάρια ελέγχου με σκοπό τον εντοπισμό σφαλμάτων ως προς την ανταπόκριση του παιχνιδιού σε κάθε μία από τις επικοινωνίες αυτές. Πιο συγκεκριμένα, ελέγχεται η ορθή παραλαβή σημάτων κίνησης και ορθής αποστολής σημάτων δόνησης. Ταυτόχρονα, βέβαια, ελέγχεται και η γενικότερη λειτουργία του παιχνιδιού.

Στη συνέχεια, παρουσιάζονται μερικά ενδεικτικά σενάρια ελέγχου.

5.3.2.3.1 Μετάδοση κίνησης και αποστολή σημάτων δόνησης μέσω Wi-Fi

Στο σενάριο ελέγχου που παρουσιάζεται στο παρακάτω σχήμα (βλ. Σχήμα 5.39), μεταδίδονται – προς την Unity εφαρμογή VRacer - μία σειρά από σήματα κίνησης που ωθούν το αγωνιστικό αυτοκίνητο προς τα εμπρός. Ως αποτέλεσμα, προκαλείται πρόσκρουση του αγωνιστικού αυτοκινήτου σε εμπόδια η οποία έχει ως αποτέλεσμα την παραγωγή σημάτων δόνησης – από την Unity εφαρμογή VRacer.

```
[INFO] Χρησιμοποιήστε τα πλήκτρα-βέλη για τον έλεγχο. Πατήστε ESC για έξοδο.
DIRECTION -> 0,1
DIRECTION -> 0,1
DIRECTION -> 0,1
DIRECTION -> 0,1
VIBRATION <- intensity=253, duration_ms=74
VIBRATION <- intensity=253, duration_ms=82
VIBRATION <- intensity=253, duration_ms=82
VIBRATION <- intensity=253, duration_ms=32
VIBRATION <- intensity=253, duration_ms=32
VIBRATION <- intensity=253, duration_ms=32
VIBRATION <- intensity=253, duration_ms=32
VIBRATION <- intensity=253, duration_ms=32
VIBRATION <- intensity=253, duration_ms=28
```

Σχήμα 5.39: Σενάριο ελέγχου χειρισμού της εφαρμογής VRacer_Unity μέσω Wi-Fi

Για την εκτέλεση του τρέχοντος σεναρίου ελέγχου, χρησιμοποιήθηκε το VRacer_Arduino_Simulator_Wifi.slnx, όπως αναφέρθηκε παραπάνω. Πιο συγκεκριμένα, δέχθηκε την ακόλουθη παραμετροποίηση:

- UDP_REMOTE_IP: 127.0.0.1
- UDP_REMOTE_PORT: 4210
- UDP_LOCAL_PORT: 4220

Επιπρόσθετα, η Unity εφαρμογή VRacer παραμετροποιήθηκε ως εξής:

- udpRemoteIp = 127.0.0.1
- udpRemotePort = 4220
- udpLocalPort = 4210
- portName = κενό αλφαριθμητικό

Μετά από τις παραπάνω ρυθμίσεις, έγιναν τα ακόλουθα:

- Εκτελέστηκε η εφαρμογή VRacer_Unity μέσα από τον Unity Editor.
- Εκτελέστηκε το script VRacer_Arduino_Simulator_Wifi.py.

Ο χρήστης, χρησιμοποιώντας τα πλήκτρα-βέλη του πληκτρολογίου, προσδίδει κίνηση στο αγωνιστικό αυτοκίνητο.

5.3.2.3.2 Μετάδοση κίνησης και αποστολή σημάτων δόνησης μέσω Σειριακής επικοινωνίας

Στο σενάριο ελέγχου που παρουσιάζεται στο παρακάτω σχήμα (βλ. Σχήμα 5.40), μεταδίδονται – προς την Unity εφαρμογή VRacer - μία σειρά από σήματα κίνησης που ωθούν το αγωνιστικό αυτοκίνητο προς τα εμπρός. Ως αποτέλεσμα, προκαλείται πρόσκρουση του αγωνιστικού αυτοκινήτου σε εμπόδια η οποία έχει ως αποτέλεσμα την παραγωγή σημάτων δόνησης – από την Unity εφαρμογή VRacer.

```
[INFO] Χρησιμοποιήστε τα πλήκτρα-βέλη για τον έλεγχο. Πατήστε ESC για έξοδο.
DIRECTION -> : 0,1
DIRECTION -> : 0,1
DIRECTION -> : 0,1
DIRECTION -> : 0,1
DIRECTION -> : 0,1
DIRECTION -> : 0,1
DIRECTION -> : 0,1
VIBRATION <- intensity=253, duration_ms=33
VIBRATION <- intensity=253, duration_ms=33
VIBRATION <- intensity=253, duration_ms=73
VIBRATION <- intensity=253, duration_ms=73
VIBRATION <- intensity=253, duration_ms=50
VIBRATION <- intensity=253, duration_ms=50
VIBRATION <- intensity=253, duration_ms=46
VIBRATION <- intensity=253, duration_ms=46
VIBRATION <- intensity=253, duration_ms=31
VIBRATION <- intensity=253, duration_ms=31
VIBRATION <- intensity=253, duration_ms=35
VIBRATION <- intensity=253, duration_ms=35
```

Σχήμα 5.40: Σενάριο ελέγχου χειρισμού της εφαρμογής VRacer_Unity μέσω Σειριακής επικοινωνίας

Καταρχάς, για την εκτέλεση του τρέχοντος σεναρίου ελέγχου, αναπτύχθηκε μία Γέφυρα Σειριακής Επικοινωνίας (με τη χρήση του λογισμικού Com0Com) μεταξύ των Θυρών COM8 και COM9.

Επίσης, χρησιμοποιήθηκε το VRacer_Arduino_Simulator_Serial_Bridge.slnx, όπως αναφέρθηκε παραπάνω. Πιο συγκεκριμένα, δέχθηκε την ακόλουθη παραμετροποίηση:

- PORT: COM8
- BAUDRATE: 115200

Επιπρόσθετα, η Unity εφαρμογή VRacer παραμετροποιήθηκε ως εξής:

- portName: *COM9*
- baudRate: *115200*
- udpRemoteIp: κενό αλφαριθμητικό

Μετά από τις παραπάνω ρυθμίσεις, έγιναν τα ακόλουθα:

- Εκτελέστηκε η εφαρμογή VRacer_Unity μέσα από τον Unity Editor.
- Εκτελέστηκε το script VRacer_Arduino_Simulator_Serial_Bridge.py.

Ο χρήστης, χρησιμοποιώντας τα πλήκτρα-βέλη του πληκτρολογίου, προσδίδει κίνηση στο αγωνιστικό αυτοκίνητο.

Κεφάλαιο 6ο: Εκτέλεση

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία εγκατάστασης του παιχνιδιού. Επίσης, καταγράφονται κάποια ενδεικτικά σενάρια εκτέλεσης.

6.1 Διάθεση Εφαρμογής σε Εκτελέσιμη Μορφή

Σύμφωνα με την πλατφόρμα ανάπτυξης παιχνιδιών Unity, η διάθεση ενός παιχνιδιού, με σκοπό την εγκατάστασή του σε κάποιο μηχάνημα, ξεκινά από την δημιουργία ενός κατάλληλου "προφίλ έκδοσης" (build profile).

Με βάση το προφίλ έκδοσης, παράγονται οι νέες εκδόσεις (build) της εφαρμογής οι οποίες, πρακτικά, έχουν τη μορφή ενός συνόλου από αρχεία εγκατάστασης (setup).

Στη συνέχεια, με βάση το παραπάνω υλικό είναι εφικτή η εγκατάσταση της εφαρμογής σε ένα οποιαδήποτε μηχάνημα που πληροί τις σχετικές προϋποθέσεις (βλ. Ενότητα 6.2.1).

6.1.1 Προφίλ Έκδοσης

Το προφίλ της πρώτης έκδοσης του VRacer, ως προς τα βασικά του σημεία, παρουσιάζεται στο ακόλουθο πίνακα (βλ. Πίνακας 6.9).

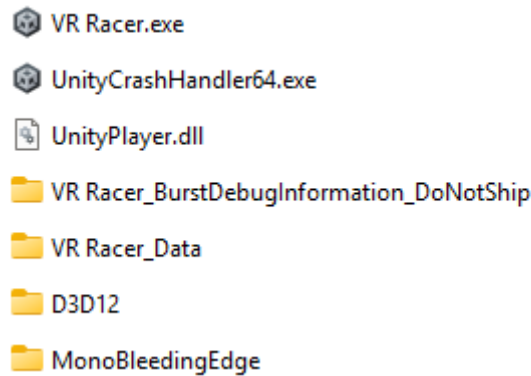
Πίνακας 6.9: Το Προφίλ Έκδοσης της εφαρμογής VRacer

Παράμετρος	Τιμή
Architecture	Intel 64-bit
Build and Run On	Local Machine
Company Name	Kalleas Georgios
Product Name	VRacer
Version	1.0

6.1.2 Παραγωγή Έκδοσης

Με βάση το παραπάνω προφίλ έκδοσης, η παραγωγή μίας νέας έκδοσης γίνεται εύκολα με τη χρήση του αντίστοιχου εργαλείου του Unity (βλ. Ενότητα 4.2.1).

Κατά την διαδικασία αυτή, παράγεται όλο το υλικό που είναι απαραίτητο για την εκτέλεση της εφαρμογής σε ένα μηχάνημα που δεν διαθέτει την πλατφόρμα Unity. Το υλικό αυτό, τελικά, αποθηκεύεται στον φάκελο που υποδεικνύει ο κατασκευαστής της εφαρμογής. Το πλήρες περιεχόμενο του φακέλου φαίνεται στο Σχήμα 6.41 ενώ τα σημαντικότερα στοιχεία του περιγράφονται στην επόμενη ενότητα.



Σχήμα 6.41: Τα περιεχόμενα το φακέλου εγκατάστασης του VRacer

6.1.3 Στοιχεία Εγκατάστασης μίας Έκδοσης

Τα στοιχεία εγκατάστασης της εφαρμογής VRacer, όπως αυτά παράγονται κατά την παραπάνω διαδικασία, είναι τα ακόλουθα:

- VRacer.exe
Πρόκειται για το βασικό εκτελέσιμο αρχείο της εφαρμογής, από την εκτέλεση του οποίου γίνεται η εκκίνηση της εφαρμογής.
- VRacer_Data
Πρόκειται για ένα φάκελο ο οποίος περιέχει τα διάφορα στοιχεία εγκατάστασης που δε σχετίζεται με κώδικα (όπως, για παράδειγμα, το αρχείο παραμέτρων, τα διάφορα μοντέλα, τα αρχεία ήχου κλπ.).
- Αρχείο παραμέτρων (car_config.json)
Το αρχείο παραμέτρων βρίσκεται μέσα στο φάκελο VRacer_Data\Managed,

6.2 Εγκατάσταση Παιχνιδιού

Εφόσον έχουν παραχθεί τα στοιχεία εγκατάστασης μίας έκδοσης, μπορεί να γίνει εύκολα η εγκατάστασή της, σε ένα οποιαδήποτε μηχάνημα που πληροί τα παρακάτω προαπαιτούμενα.

6.2.1 Προαπαιτούμενα Εγκατάστασης

Οι προϋποθέσεις που πρέπει να πληροί ένα μηχάνημα για την εγκατάσταση του VRacer είναι οι ακόλουθες:

- ✓ Λ/Σ: Windows (έκδοσης 10 ή μεταγενέστερη).
- ✓ Αρχιτεκτονική: 64-bit (Intel ή AMD).
- ✓ Ελεύθερος χώρος στο δίσκο: 150 MB.

6.2.2 Διαδικασία Εγκατάστασης

Η διαδικασία εγκατάστασης του VRacer, σε ένα μηχάνημα που πληροί τις παραπάνω προδιαγραφές, είναι πολύ απλή. Αρκεί μόνο να αντιγραφούν τα στοιχεία εγκατάστασης της επιθυμητής έκδοσης της εφαρμογής στο δίσκο του μηχανήματος, σε ένα οποιοδήποτε φάκελο.

6.2.3 Προαπαιτούμενα Εκτέλεσης

Οι προϋποθέσεις που πρέπει να πληρούνται για την εκτέλεση του VRacer, σε ένα μηχάνημα, είναι οι ακόλουθες:

- ✓ Ολοκλήρωση εγκατάστασης εφαρμογής (βλ. Ενότητα 6.2.2, παραπάνω).
- ✓ Ύπαρξη του Meta Horizon Link στο μηχάνημα.

6.2.3.1 Συσκευές

Οι συσκευές που απαιτούνται για την εκτέλεση της εφαρμογής (είτε κατά τη φάση της ανάπτυξης είτε αργότερα) είναι οι ακόλουθες (βλ. Σχήμα 6.42):

- VR Οθόνη Κεφαλής (Headset)
- Τυπικά VR Χειριστήρια (στην περίπτωση που δε χρησιμοποιείται το αυτοσχέδιο τιμόνι):
 - Αριστερό VR Χειριστήριο Αφής (VR Touch Controller)
Πρόκειται για το αριστερό χειριστήριο VR εφαρμογών.
 - Δεξιό VR Χειριστήριο Αφής (VR Touch Controller)
Πρόκειται για το δεξιό χειριστήριο VR εφαρμογών.
- Αυτοσχέδιο τιμόνι (ως εναλλακτική επιλογή αντί των VR Χειριστηρίων)



Σχήμα 6.42: Ένα τυπικό σετ σύγχρονων VR χειριστηρίων της εταιρίας Meta Quest

6.2.4 Παραμετροποίηση

Αμέσως μετά την εγκατάσταση του παιχνιδιού, πρέπει να γίνουν οι απαραίτητες ρυθμίσεις τόσο στην Unity εφαρμογή VRacer όσο και στο αυτοσχέδιο τιμόνι.

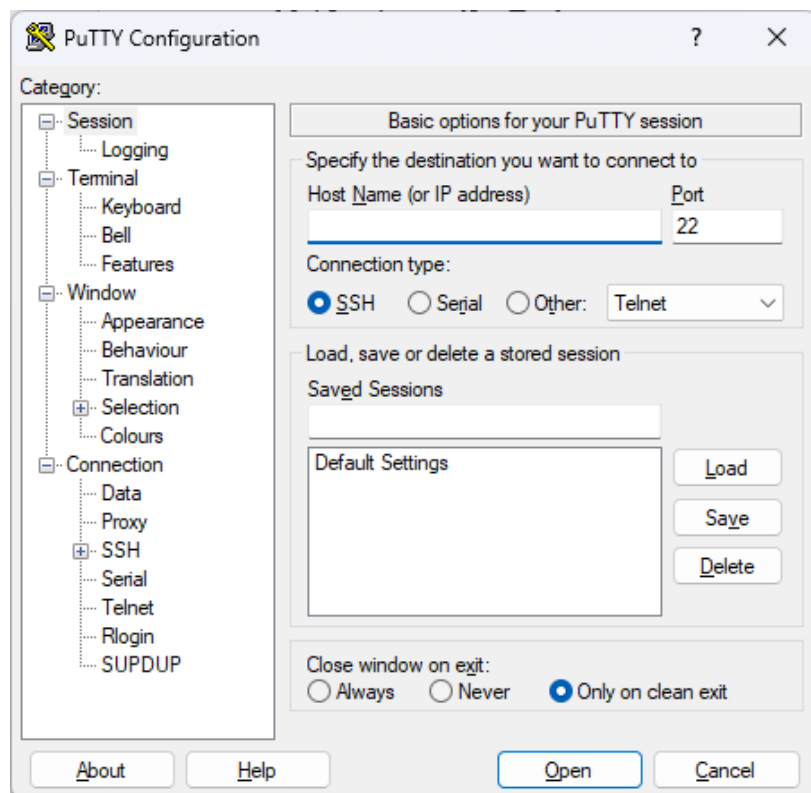
6.2.4.1 Αυτοσχέδιο Τιμόνι

Στην περίπτωση που επιλεγθεί το αυτοσχέδιο τιμόνι, ως εργαλείο χειρισμού του αγωνιστικού αυτοκινήτου, θα πρέπει πρώτα να γίνουν οι κατάλληλες ρυθμίσεις σ' αυτό. Οι δυνατότητες παραμετροποίησης του αυτοσχεδίου τιμονιού περιγράφονται στην Ενότητα 4.4.2.4. Πιο συγκεκριμένα, πρόκειται για τις ακόλουθες παραμέτρους:

- Επιλογή συσκευής μετάδοσης κίνησης (γυροσκόπιο ή joystick)
- Επιλογή μεθόδου επικοινωνίας με την Unity εφαρμογή VRacer (Σειριακή ή Wi-Fi)
- Καθορισμός στοιχείων λειτουργίας της Wi-Fi επικοινωνίας:
 - Δίκτυο Wi-Fi (SSID)
 - Συνθηματικό (Password)
 - IP του μηχανήματος στο οποίο εκτελείται η Unity εφαρμογή VRacer.

Κατά την ενεργοποίηση του ηλεκτρονικού κυκλώματος του τιμονιού, ο μικροελεγκτής ESP32 προσφέρει την δυνατότητα καθορισμού των παραπάνω παραμέτρων. Μετά τον καθορισμό τους, οι τιμές των παραμέτρων αποθηκεύονται στη μόνιμη μνήμη του μικροελεγκτή ESP32. Έτσι, αν αυτό έχει ήδη γίνει μία φορά τότε, ο μικροελεγκτής ESP32 προσφέρει μία μικρή διορία για να αποφασιστεί αν οι τιμές των παραμέτρων θα τροποποιηθούν ή όχι (δηλ. το ηλεκτρονικό κύκλωμα να λειτουργήσει με βάση τις αποθηκευμένες τιμές των παραμέτρων).

Η παραπάνω εργασία μπορεί να εκτελεστεί μέσω μίας οποιασδήποτε εφαρμογής που μπορεί να επικοινωνεί σειριακά με διάφορες συσκευές. Μία τέτοια εφαρμογή είναι και το PuTTY [66]. Το PuTTY είναι ένα ελεύθερο λογισμικό που προσφέρει την δυνατότητα επικοινωνίας, με διάφορες συσκευές, μέσω διαφόρων καναλιών επικοινωνίας.



Σχήμα 6.43: Ο βασική οθόνη εργασίας του PuTTY

6.2.4.2 Unity εφαρμογή VRacer

Οι δυνατότητες παραμετροποίησης της Unity εφαρμογή VRacer περιγράφονται στην Ενότητα 4.4.1.3. Όπως προαναφέρθηκε, όλες οι διαθέσιμες παράμετροι της εφαρμογής περιέχονται σε ένα αρχείο παραμέτρων (βλ. Ενότητα 6.1.3). Συνεπώς, ο καθορισμός τους μπορεί να γίνει μέσα από ένα οποιονδήποτε επεξεργαστή κειμένου.

6.2.4.2.1 Επικοινωνία με τον Χρήστη

Με βάση την επιθυμητή μέθοδο επικοινωνίας με τον χρήστη, πρέπει να γίνει η ακόλουθη παραμετροποίηση:

- Λειτουργία με Αυτοσχέδιο Τιμόνι μέσω με Wi-Fi

Θα πρέπει να τεθούν οι παρακάτω παράμετροι:

- udpRemoteIp = Η Διεύθυνση IP του μικροελεγκτή ESP32 του αυτοσχέδιου τιμονιού
- udpRemotePort = Η Θύρα UDP για την αποστολή σημάτων (δόνησης) στο αυτοσχέδιο τιμόνι.
- udpLocalPort = Η Θύρα UDP για τη λήψη σημάτων (κατεύθυνσης) από το αυτοσχέδιο τιμόνι.
- portName = κενό αλφαριθμητικό

- Λειτουργία με Αυτοσχέδιο Τιμόνι μέσω Σειριακής επικοινωνίας

Θα πρέπει να τεθούν οι παρακάτω παράμετροι:

- portName = Το όνομα της σειριακής θύρας που έχει δοθεί – από το μηχάνημα εκτέλεσης του παιχνιδιού – στον μικροελεγκτή ESP32
- baudRate = Ο ρυθμός μετάδοσης δεδομένων σε baud με τον οποίο αποστέλλει δεδομένα ο μικροελεγκτής ESP32
- udpRemoteIp = κενό αλφαριθμητικό

- Λειτουργία με Πληκτρολόγιο ή VR χειριστήρια

Θα πρέπει να τεθούν οι παρακάτω παράμετροι:

- portName = κενό αλφαριθμητικό
- udpRemoteIp = κενό αλφαριθμητικό

6.2.4.2.2 Συμπεριφορά Αυτοκινήτου

Με βάση την επιθυμητή συμπεριφορά του αυτοκινήτου, θα πρέπει να τεθούν αναλόγως οι παράμετροι που σχετίζονται με τις ιδιότητες του Ελεγκτή του αυτοκινήτου (βλ. 5.1.5.1.1)

6.3 Εκτέλεση Παιχνιδιού

Στη συνέχεια, παρουσιάζονται οι βασικές ενέργειες που πρέπει να γίνουν - από τον χρήστη - σε κάθε εκτέλεση του παιχνιδιού.

6.3.1 Έναρξη Παιχνιδιού

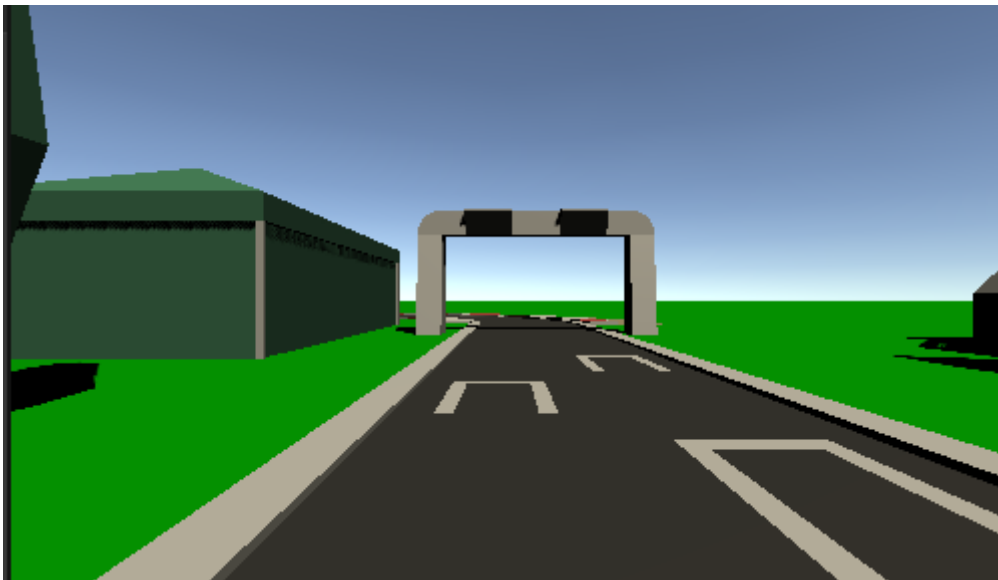
Για να ξεκινήσει το παιχνίδι, απαιτούνται οι ακόλουθες ενέργειες:

- ✓ Εκτέλεση του Meta Horizon Link (αν πρόκειται να χρησιμοποιηθεί κάποια VR συσκευή).
- ✓ Εκτέλεση του VRacer.exe, από το φάκελο που προέκυψε από την εγκατάσταση (βλ. Ενότητα 6.2.2, παραπάνω).

Στην περίπτωση που, κατά την παραμετροποίηση του παιχνιδιού (βλ. Ενότητα 6.2.4), έχει επιλεγθεί η χρήση του αυτοσχέδιου τιμονιού τότε - αυτό - θα πρέπει να είναι ακίνητο και σε οριζόντια θέση (παράλληλο προς το έδαφος).

6.3.2 Οδήγηση

Αμέσως μετά την εκκίνηση του παιχνιδιού, το αυτοκίνητο τοποθετείται στην αφετηρία της πίστας (βλ. Σχήμα 6.44) και είναι σε θέση να κινηθεί!



Σχήμα 6.44: Το αγωνιστικό αυτοκίνητο στην αφετηρία

Με βάση την παραμετροποίηση του παιχνιδιού (βλ. Ενότητα 6.2.4), η οδήγηση το αγωνιστικού αυτοκινήτου μπορεί να γίνει με τους ακόλουθους τρόπους.

6.3.2.1 Οδήγηση με πληκτρολόγιο

Για να γίνει αυτό, χρησιμοποιούνται μόνο τα τέσσερα (4) πλήκτρα-βέλη του πληκτρολογίου. Έτσι, για κάθε μία από τις τέσσερις (4) βασικές κατευθύνσεις (δηλ., πάνω, κάτω, αριστερά ή δεξιά), πρέπει να πατηθούν τα ακόλουθα πλήκτρα-βέλη:

- Κίνηση προς τα εμπρός
Πάνω-βέλος.
- Κίνηση προς τα πίσω
Κάτω-βέλος.
- Κίνηση προς τα δεξιά:
Δεξιό-βέλος.

- Κίνηση προς τα αριστερά:

Αριστερό-βέλος.

Σε κάθε περίπτωση, είναι δυνατόν να πατηθούν δύο πλήκτρα-βέλη ταυτόχρονα (για να επιτευχθεί, για παράδειγμα, πλαγιολίσθηση στις στροφές). Σ' αυτή την περίπτωση, η κίνηση του αυτοκινήτου γίνεται συνδυαστικά.

Τέλος, αν δεν είναι πατημένο κάποιο πλήκτρο-βέλος, το αυτοκίνητο ακινητοποιείται.

6.3.2.2 Οδήγηση με VR Χειριστήρια

Για να γίνει αυτό, χρησιμοποιείται μόνο το πηδάλιο (joystick) του αριστερού VR χειριστηρίου αφής. Βέβαια, ο χρήστης θα πρέπει να κρατά στα χέρια του και τα δύο VR χειριστήρια αφής για να μπορεί να αντιλαμβάνεται τις δονήσεις οι οποίες, σύμφωνα με τις απαιτήσεις της εφαρμογής (βλ. Ενότητα 3.1.3), έχουν διαφορετική ένταση σε κάθε χέρι.

Γενικά, η λειτουργικότητα του πηδαλίου είναι η απολύτως αναμενόμενη για ένα χρήστη που έχει μία βασική εμπειρία χρήσης ηλεκτρονικών παιχνιδιών.

6.3.2.3 Οδήγηση με Αυτοσχέδιο Τιμόνι

Η οδήγηση με το τιμόνι εξαρτάται από το αν, για τη μετάδοση κίνησης, χρησιμοποιείται γυροσκόπιο ή joystick.

6.3.2.3.1 Με Γυροσκόπιο

Για να μπορέσει το αυτοκίνητο να κινηθεί προς μία από τις τέσσερις βασικές κατευθύνσεις (δηλ., πάνω, κάτω, αριστερά ή δεξιά), ο χρήστης πρέπει να πράξει τα ακόλουθα:

- Κίνηση προς τα εμπρός

Το μπροστινό μέρος του τιμονιού στρέφεται προς τα κάτω.

- Κίνηση προς τα πίσω

Το πίσω μέρος του τιμονιού στρέφεται προς τα κάτω.

- Κίνηση προς τα δεξιά:

Το δεξιό μέρος του τιμονιού στρέφεται προς τα κάτω.

- Κίνηση προς τα αριστερά:

Το αριστερό μέρος του τιμονιού στρέφεται προς τα κάτω.

Σε κάθε περίπτωση, είναι δυνατόν να γίνει και συνδυασμός των παραπάνω κινήσεων. Τότε, η κίνηση του αυτοκινήτου προκύπτει ανάλογα.

Τέλος, κάθε φορά που το τιμόνι επανέρχεται σε οριζόντια θέση, το αυτοκίνητο ακινητοποιείται.

6.3.2.3.2 Με Joystick

Στην περίπτωση αυτή, το τιμόνι παραμένει ακίνητο σε οριζόντια θέση. Η κίνηση του αυτοκινήτου γίνεται, με προφανή τρόπο, με τη χρήση του πηδαλίου.

6.3.3 Τερματισμός Παιχνιδιού

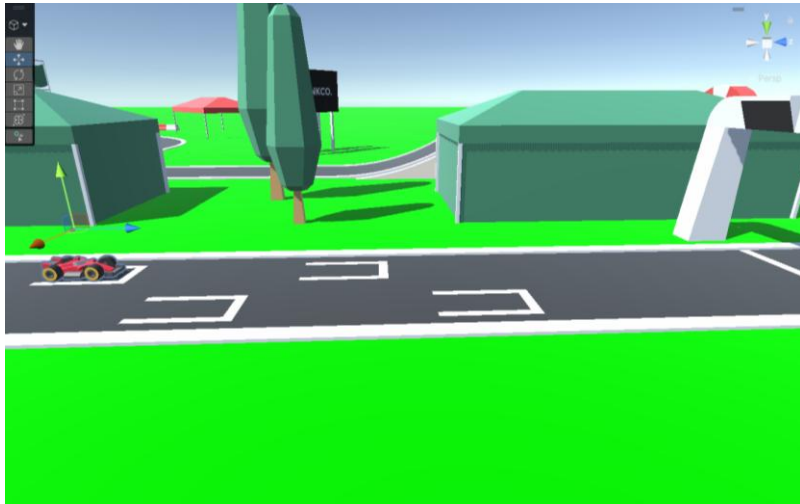
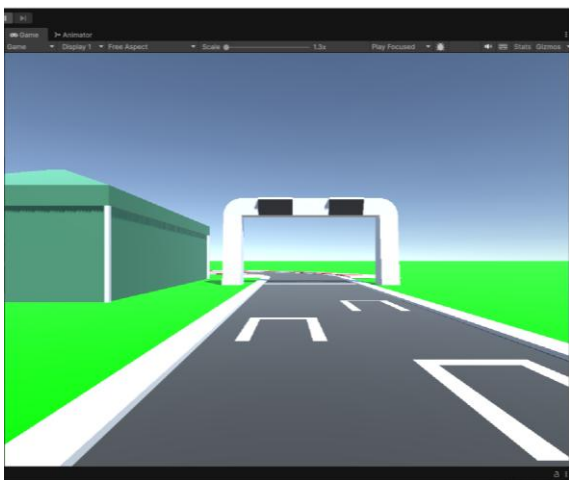
Το παιχνίδι τερματίζει άμεσα με το πάτημα του πλήκτρου Escape.

6.4 Σενάρια Εκτέλεσης

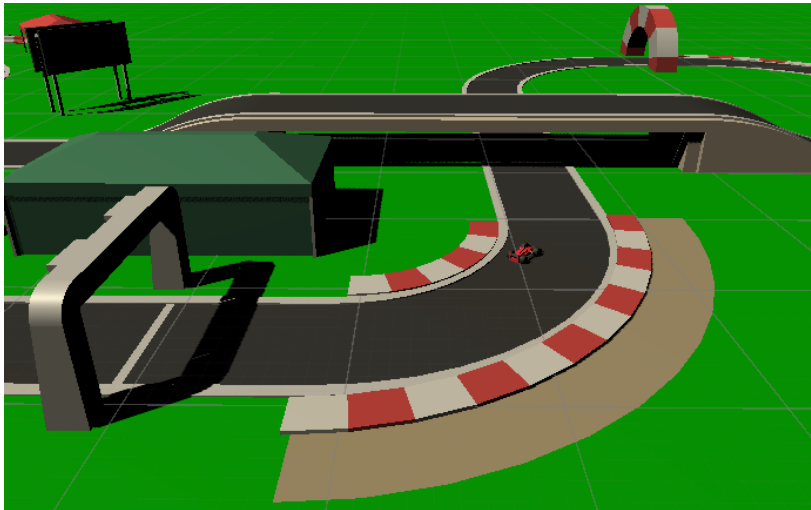
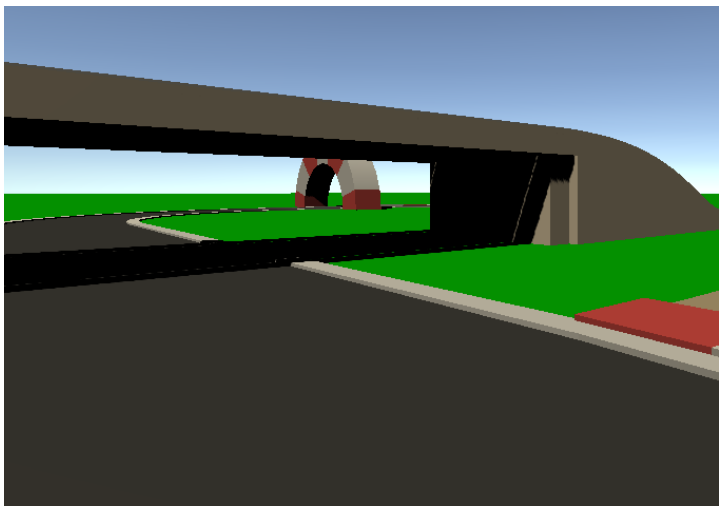
Τα ακόλουθα σενάρια εκτέλεσης επιχειρούν να καλύψουν τα βασικά σημεία της οδήγησης του αυτοκινήτου. Γενικά, βασίζονται στην υπόθεση ότι ο χειρισμός του αυτοκινήτου γίνεται με τα VR χειριστήρια.

Για κάθε βήμα των σεναρίων παρουσιάζονται δύο στιγμιότυπα: ένα πανοραμικό, που δείχνει την τρέχουσα θέση του αυτοκινήτου, και ένα που δείχνει τι ακριβώς βλέπει ο οδηγός.

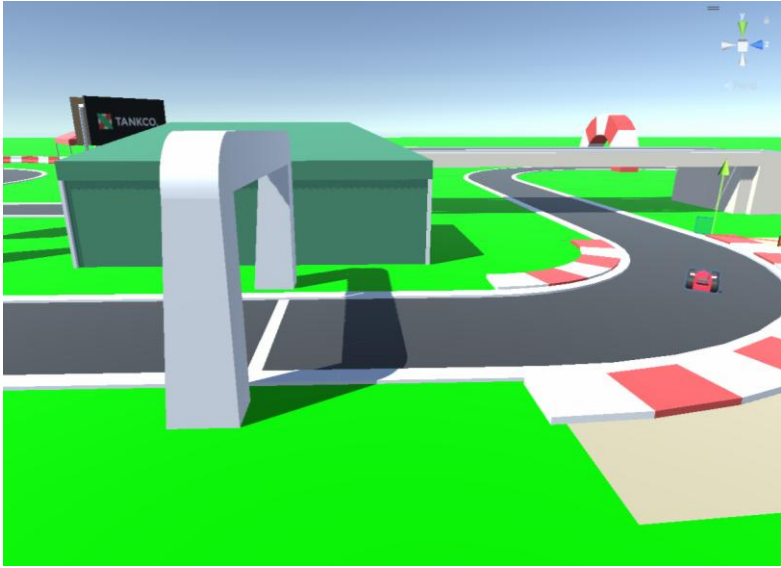
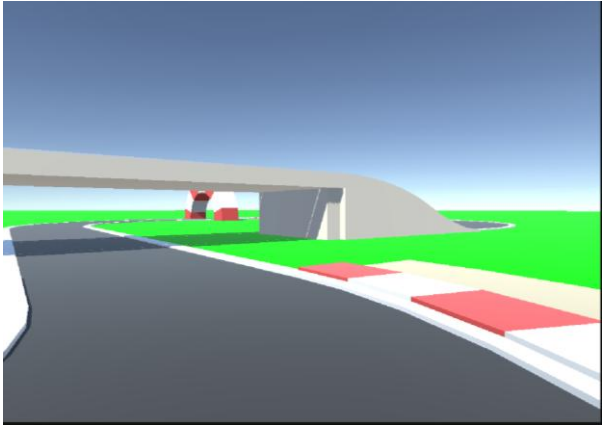
6.4.1 Εκκίνηση Παιχνιδιού

Σενάριο	Εκκίνηση Παιχνιδιού
Προαπαιτούμενα εκτέλεσης	
Ροή εκτέλεσης	Το σύστημα τοποθετεί το αυτοκίνητο στην αφετηρία ενεργοποιώντας τον ήχο μηχανής (σε χαμηλή ένταση):  

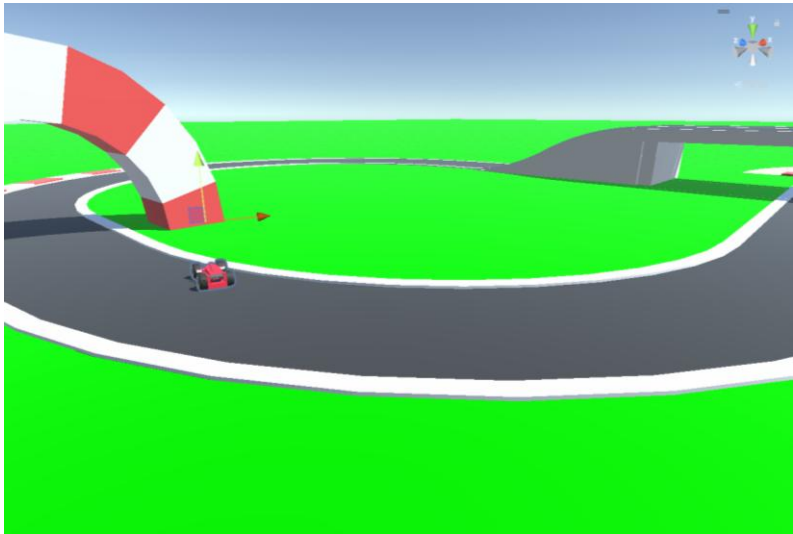
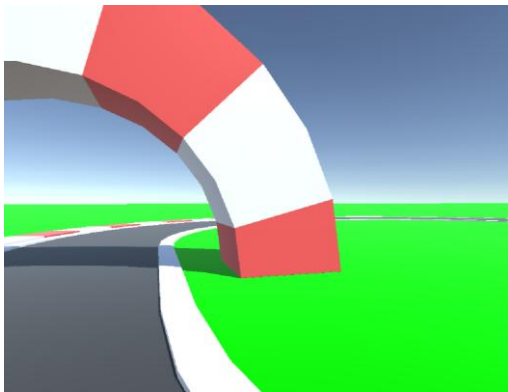
6.4.2 Επιτάχυνση

Σενάριο	Επιτάχυνση
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, αμέσως μετά την εκκίνησή του και μετά την πρώτη μεγάλη αριστερή στροφή, εισέρχεται σε μία ευθεία.
Ροή Εκτέλεσης	Ο χρήστης επαναφέρει ομαλά το πηδάλιο του VR χειριστηρίου προς τα εμπρός και επιταχύνει. Το σύστημα αυξάνει την ταχύτητα του αυτοκινήτου και, ταυτόχρονα, δυναμώνει την ένταση του ήχου μηχανής.  

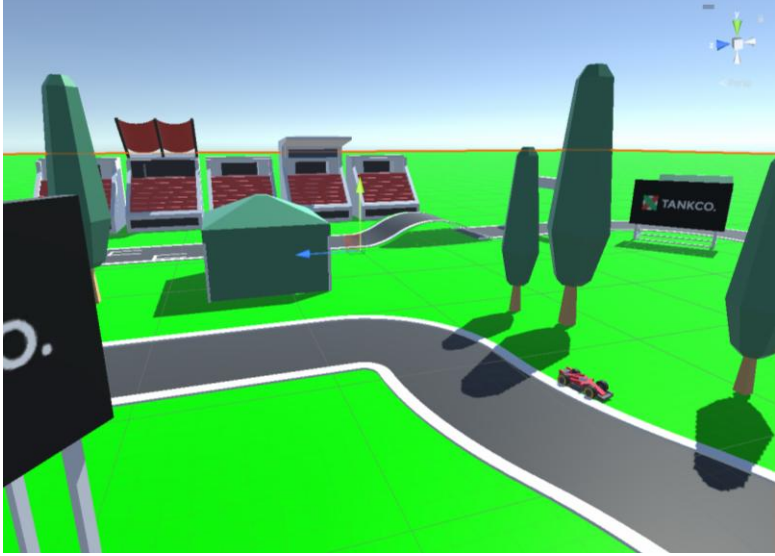
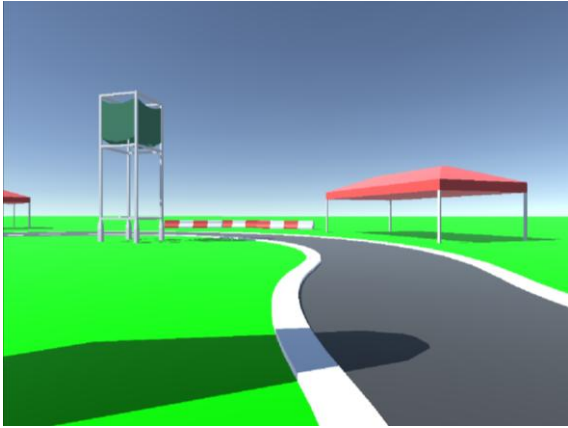
6.4.3 Αλλαγή Κατεύθυνσης – Ομαλή Αριστερή Στροφή

Σενάριο	Αλλαγή Κατεύθυνσης – Ομαλή Αριστερή Στροφή
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, κατά την κίνησή του, εισέρχεται σε αριστερή στροφή.
Ροή Εκτέλεσης	Ο χρήστης στρέφει ελαφρά το πηδάλιο του VR χειριστηρίου αφής προς τα αριστερά. Το σύστημα περιστρέφει το αυτοκίνητο προς τα αριστερά διατηρώντας την ταχύτητά του:  

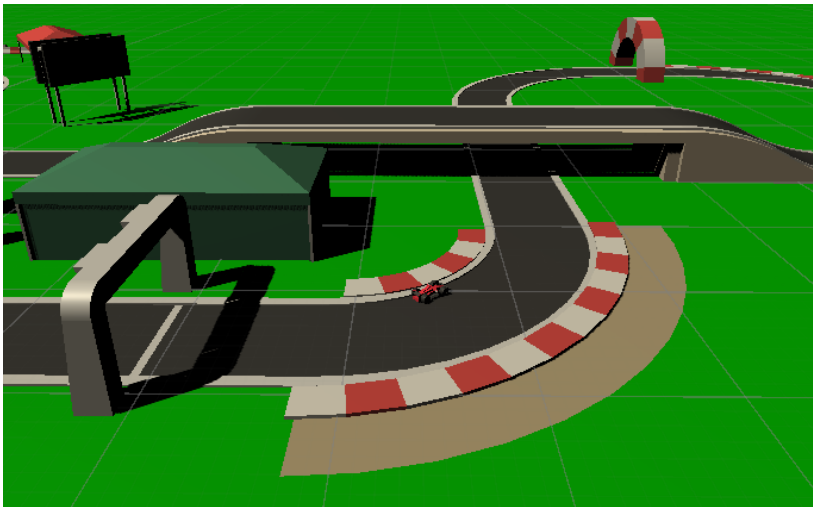
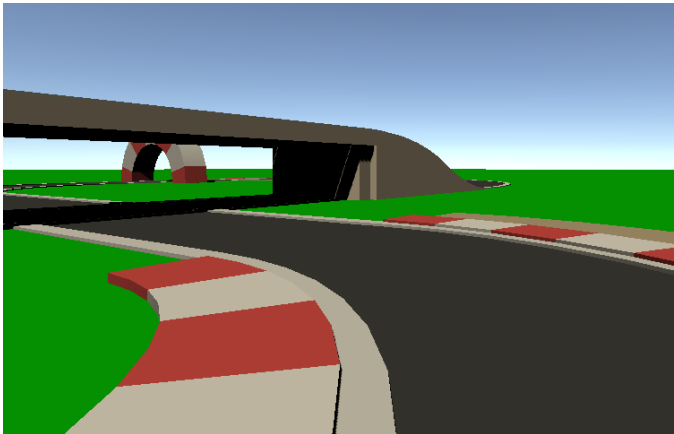
6.4.4 Αλλαγή Κατεύθυνσης – Απότομη Δεξιά Στροφή

Σενάριο	Αλλαγή Κατεύθυνσης – Απότομη Δεξιά Στροφή
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, κατά την κίνησή του, εισέρχεται σε μεγάλη δεξιά στροφή
Ροή Εκτέλεσης	Ο χρήστης στρέφει απότομα το πηδάλιο του VR χειριστηρίου αφής προς τα δεξιά. Το σύστημα περιστρέφει το αυτοκίνητο προς τα δεξιά οδηγώντας το, ουσιαστικά, σε πρόσκρουση:  

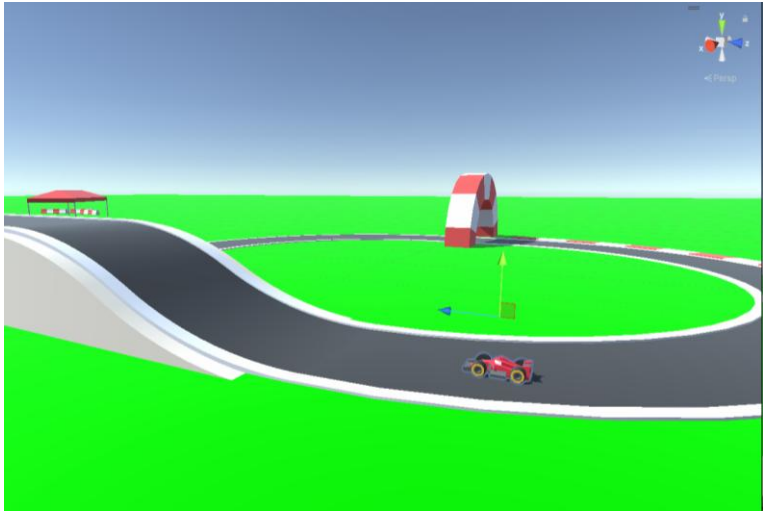
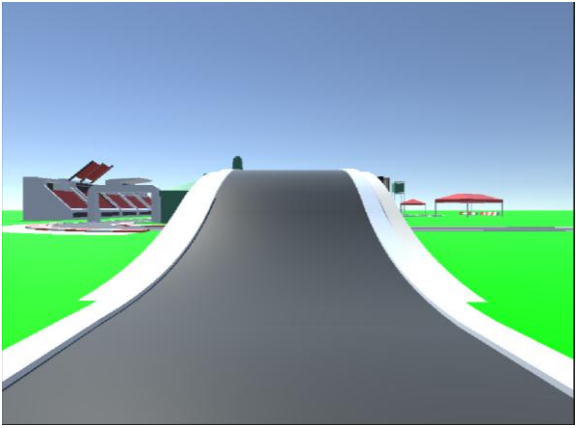
6.4.5 Εκτός Πίστας – Απότομη Δεξιά Στροφή

Σενάριο	Εκτός Πίστας – Απότομη Δεξιά Στροφή
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, κατά την κίνησή του, κατεβαίνει τη γέφυρα, με μεγάλη ταχύτητα, και φτάνει σε μια δεξιά κλειστή στροφή.
Ροή Εκτέλεσης	Ο χρήστης στρέφει απότομα το πηδάλιο του VR χειριστηρίου αφής προς τα δεξιά. Το σύστημα οδηγεί το αυτοκίνητο εκτός πίστας, προσκρούοντας στο στηθαίο. Ταυτόχρονα, παράγει ένα σχετικό ήχο πρόσκρουσης και προκαλεί δονήσεις:  

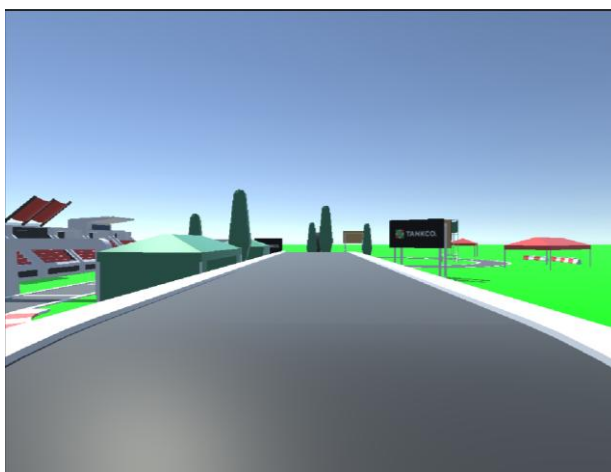
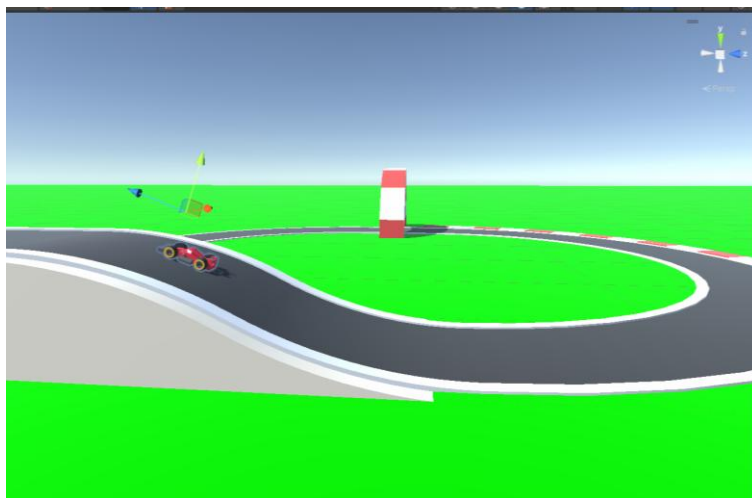
6.4.6 Πλαγιολίσθηση

Σενάριο	Πλαγιολίσθηση
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, αμέσως μετά την αφετηρία, εισέρχεται σε μία αριστερή στροφή 90°.
Ροή Εκτέλεσης	Ο χρήστης στρέφει το πεδάλιο του VR χειριστηρίου αψής αριστερά-εμπρός, έτσι ώστε - αυτό - να παραμείνει στο οδόστρωμα και, παράλληλα, να συνεχίσει να κινείται προς τα εμπρός. Το σύστημα προκαλεί πλαγιολίσθηση. Ταυτόχρονα, παράγει δονήσεις και στα δύο VR χειριστήρια αψής αλλά με διαφορετική ένταση σε κάθε χέρι (η ένταση των δονήσεων είναι μεγαλύτερη στο αριστερό VR χειριστήριο αψής):  

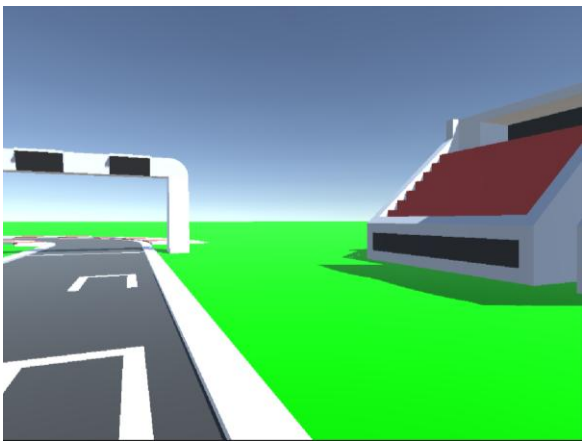
6.4.7 Διάσχιση Γέφυρας

Σενάριο	Διάσχιση Γέφυρας
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, κατά την κίνησή του, φτάνει στη γέφυρα της πίστας.
Ροή Εκτέλεσης	Ο χρήστης διατηρεί το πηδάλιο του VR χειριστηρίου αφής στραμμένο προς τα εμπρός. Το σύστημα οδηγεί το αυτοκίνητο προς τη γέφυρα:  

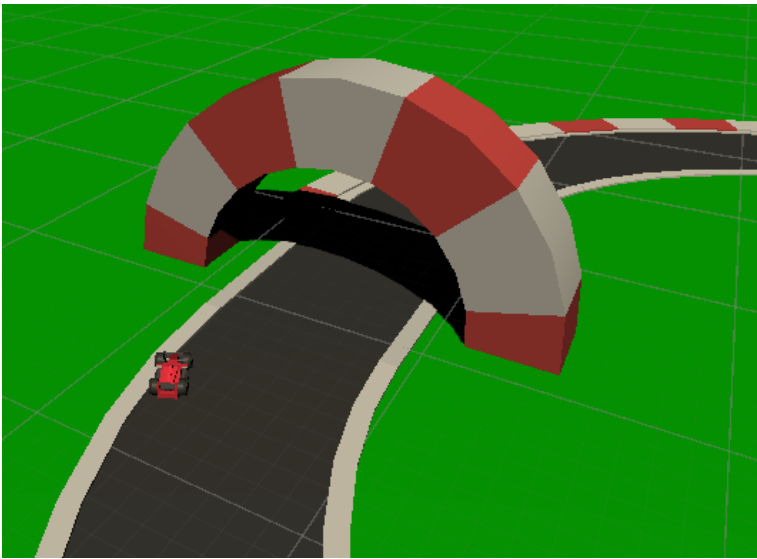
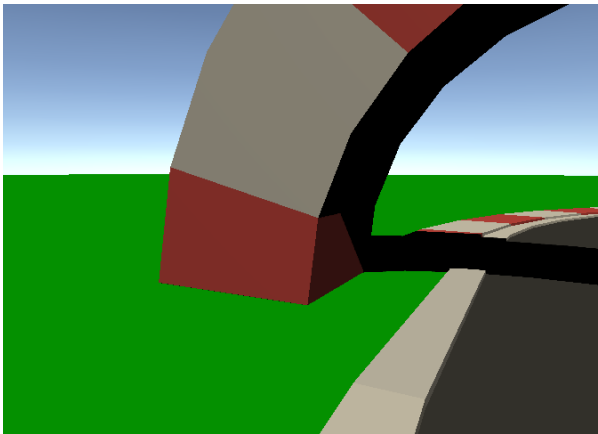
Στη συνέχεια, το σύστημα, καθώς το αυτοκίνητο ανεβαίνει στη γέφυρα, παράγει ένα "μικρής" έντασης ήχο πρόσκρουσης:



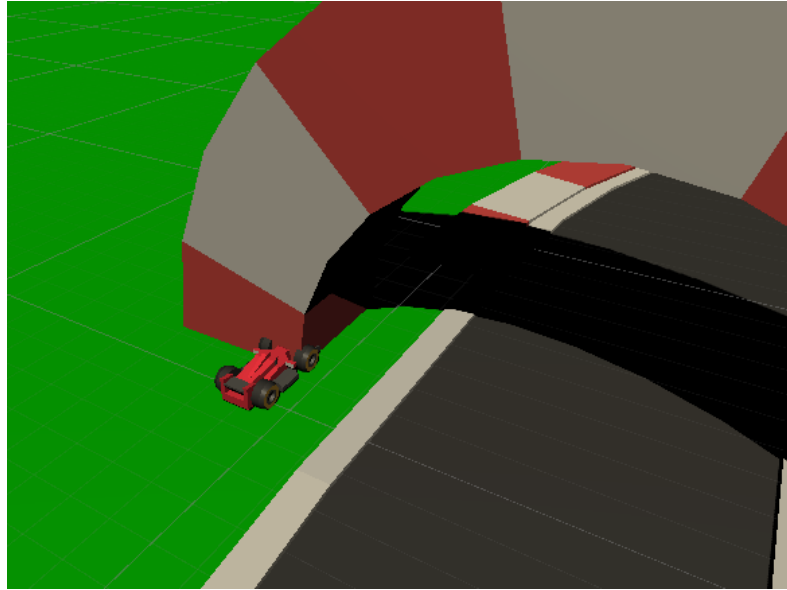
6.4.8 Πέρασμα από Σαμαράκι

Σενάριο	Πέρασμα από Σαμαράκι
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο, κατά την κίνησή του, περνάει με ταχύτητα πάνω από ένα ψηλό σαμαράκι και πραγματοποιεί άλμα.
Ροή Εκτέλεσης	Ο χρήστης διατηρεί το πηδάλιο του VR χειριστηρίου αφής στραμμένο προς τα εμπρός χωρίς να μειώσει την ταχύτητα του αυτοκινήτου. Το σύστημα προσεδαφίζει το αυτοκίνητο στο οδόστρωμα παράγοντας ισχυρές δονήσεις και ήχους πρόσκρουσης:  

6.4.9 Πρόσκρουση σε Εμπόδιο

Σενάριο	Πρόσκρουση σε Εμπόδιο
Προαπαιτούμενα εκτέλεσης	Το αυτοκίνητο εισέρχεται, με μεγάλη ταχύτητα, σε μεγάλη δεξιά στροφή.
Ροή Εκτέλεσης	Ο χρήστης στρέφει το πηδάλιο του VR χειριστηρίου αφής προς τα αριστερά χωρίς να μειώσει την ταχύτητα του αυτοκινήτου. Το σύστημα κατευθύνει το αυτοκίνητο εκτός του οδοστρώματος το οποίο προσκρούει στο στηθαίο. Παράλληλα, το σύστημα παράγει ήχο πρόσκρουσης και, ταυτόχρονα, δονήσεις πρόσκρουσης:  
	Ο χρήστης στρέφει διατηρεί το πηδάλιο του VR χειριστηρίου αφής προς τα αριστερά χωρίς να μειώσει την ταχύτητα του αυτοκινήτου.
	Το σύστημα κατευθύνει το αυτοκίνητο προς την αριστερή βάση της έγχρωμης πύλης.

Παράλληλα, το σύστημα παράγει έντονο ήχο πρόσκρουσης και, ταυτόχρονα, έντονες δονήσεις πρόσκρουσης:



Κεφάλαιο 7ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Στην ενότητα 1.1 παρουσιάστηκε το βασικό θέμα της τρέχουσας Π.Ε.. Έπειτα, στις ενότητες 1.2 και 1.3 παρουσιάστηκαν οι βασικοί της σκοποί και στόχοι, αντίστοιχα. Προφανώς, θα είχε μεγάλη αξία να εξεταστεί κατά πόσο επιτεύχθηκαν οι σκοποί και οι στόχοι αυτοί.

Επιπλέον, θα ήταν σημαντικό να καταγραφούν κάποια ζητήματα τα οποία ενδέχεται στο μέλλον να αποτελέσουν την αφετηρία για περαιτέρω έρευνα και ανάπτυξη.

7.1 Επίτευξη Σκοπού Π.Ε.

Όπως περιγράφεται στην ενότητα 1.2, ο σκοπός της τρέχουσας Π.Ε. είναι διττός. Συνοπτικά, είναι η δημιουργία ενός ηλεκτρονικού παιχνιδιού στην πλατφόρμα Unity και, ταυτόχρονα, η δυνατότητα αξιοποίησης, από αυτό, των VR τεχνολογιών.

Καταρχάς, με βάση τα στοιχεία που παραθέτει η Ενότητα 4.2, προκύπτει καθαρά ότι η πλατφόρμα ανάπτυξης παιχνιδιού Unity αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης παιχνιδιών. Το VRacer αποτελεί μία χαρακτηριστική περίπτωση παιχνιδιού όπου τόσο το περιβάλλον του όσο και η λειτουργικότητά του έχουν υλοποιηθεί εξ' ολοκλήρου στο Unity.

Επιπρόσθετα, με βάση τα σενάρια εκτέλεσης που περιγράφονται στο 5^ο κεφάλαιο, προκύπτει εύλογα το ότι VRacer είναι σε θέση να αξιοποιήσει τις VR τεχνολογίες για την αλληλεπίδρασή του με τον χρήστη. Πρακτικά, δηλαδή, μπορεί να μεταφέρει στο χρήστη τα ερεθίσματα της κίνησης, του ήχου και της δόνησης. Μάλιστα, σ' αυτή του την διάσταση, είναι γεγονός ότι ακόμα και στην πρώτη του έκδοση προσφέρει μία έντονη εμπειρία χρήστη.

Συνεπώς, από τα παραπάνω, προκύπτει το συμπέρασμα ότι ο σκοπός της τρέχουσας Π.Ε. έχει επιτευχθεί και ως προς τις δύο του διαστάσεις.

7.2 Επίτευξη Στόχων Π.Ε.

Στην ενότητα 1.3, παρουσιάζονται αναλυτικά οι βαθύτεροι στόχοι της τρέχουσας Π.Ε. Στη συνέχεια, γίνεται ένας απολογισμός του βαθμού επιτυχίας με την οποία η Π.Ε. προσέγγισε έναν-έναν τους στόχους αυτούς.

7.2.1 Εντρύφηση στις VR Τεχνολογίες

Όπως εύλογα προκύπτει από τη μελέτη του 5^{ου} κεφαλαίου, είναι σχετικά απλό, για ένα παιχνίδι που κατασκευάζεται στο Unity, να περάσει στο χώρο της Εικονικής Πραγματικότητας. Αυτό όμως είναι το πρώτο βήμα!

Η πλήρης αξιοποίηση των δυνατοτήτων της Εικονικής Πραγματικότητας, για την δημιουργία ενός "ρεαλιστικού" εικονικού κόσμου, αποτελεί ένα ευρύ πεδίο μελέτης και έρευνας. Το σίγουρο όμως είναι ότι οι απαιτήσεις που τέθηκαν – προς το VRacer – από την Ενότητα 3.1.3 ήταν το έναυσμα για να γίνει και το δεύτερο βήμα στο χώρο της Εικονικής Πραγματικότητας.

7.2.2 Μελέτη των Βασικών Χαρακτηριστικών της Εικονικής Πραγματικότητας

Όπως αναφέρθηκε στην ενότητα 2.1.4, τα βασικά χαρακτηριστικά της Εικονικής Πραγματικότητας είναι: η εμβύθιση, η διαδραστικότητα και η τηλεπαρουσία.

Το VRacer επιχειρεί να παρέχει και τα τρία. Ειδικότερα:

- Για την επίτευξη της τηλεπαρουσίας, ο παίκτης-χρήστης τοποθετείται μέσα στο "μέσα" στο αυτοκίνητο ως οδηγός του.
- Η εμπύθιση επιτυγχάνεται μέσω των πολλαπλών ερεθισμάτων της κίνησης, των ήχων και των δονήσεων σε συνδυασμό με την εικονική θέση του παίκτη-χρήστη.
- Τέλος, η διαδραστικότητα επιτυγχάνεται κυρίως από το γεγονός ότι ο παίκτης-χρήστης είναι, απλά, ο οδηγός ενός αγωνιστικού αυτοκινήτου.

Για παράδειγμα, το αντίκτυπο κάθε οδηγικού σφάλματος γίνεται πολύ έντονα αντιληπτό, σε πραγματικό χρόνο.

7.2.3 Εκτίμηση της Αύξησης της Ευχρηστίας μίας Εφαρμογής με τη Χρήση των VR Τεχνολογιών

Το παιχνίδι VRacer προσφέρει διάφορους τρόπους επικοινωνίας με το χρήστη. Έτσι, αν και το πηδάλιο (joystick) του αριστερού VR χειριστηρίου δεν είναι και το πιο εύχρηστο μέσο για την οδήγηση του αυτοκινήτου, το αυτοσχέδιο τιμόνι είναι σίγουρα πιο εύχρηστο. Ταυτόχρονα, το ότι προσφέρει την αίσθηση της δόνησης δημιουργεί μία ρεαλιστική εμπειρία οδήγησης.

7.2.4 Εκτίμηση της Θετικής Επίδρασης των VR Τεχνολογιών στην Εμπειρία Χρήστη

Από την εκτέλεση των σεναρίων εκτέλεσης του 6^{ου} κεφαλαίου, γίνεται φανερό ότι οι VR Τεχνολογίες εκτοξεύουν, θετικά, την εμπειρία χρήστη.

Ενδεικτικά:

- Ο παίκτης-χρήστης αισθάνεται ότι "κάθεται", πραγματικά, "μέσα" στο αυτοκίνητο ως οδηγός του.
- Αλληλοεπιδρά με το παιχνίδι έχοντας την δυνατότητα να παρατηρήσει τόσο τον "κόσμο" γύρω του όσο και το ίδιο το αυτοκίνητό του.
- Αντιλαμβάνεται την κίνηση (για παράδειγμα, κάθε αλλαγή κατεύθυνσης, επιτάχυνση κλπ.) με τον καθηλωτικό τρόπο που προσφέρουν οι VR τεχνολογίες.
- Νιώθει έντονα τις δονήσεις που θα παράγονταν, στον πραγματικότητα, σε μία πλαγιολίσθηση ή σε κάποια πρόσκρουση.
- Νιώθει έντονα τις μεταβολές της έντασης και του τόνου των ήχων που παράγονται κατά την κίνηση ή σε κάποια πρόσκρουση.

7.3 Αξιολόγηση Εμπειρίας Χρήστη

Για μία αντικειμενική αξιολόγηση της εμπειρίας χρήστη που προσφέρει το παιχνίδι VRacer, χρησιμοποιήθηκε το Ερωτηματολόγιο Εμπειρίας Χρήστη (UEQ) [67]. Πρόκειται για ένα γενικά αποδεκτό και αξιόπιστο ερωτηματολόγιο με το οποίο μπορεί να μετρηθεί η εμπειρία χρήστη που προσφέρει μία διαδραστική κατασκευή.

Γενικά, διατίθεται σε δύο εκδοχές: την πλήρη και τη σύντομη. Στα πλαίσια της τρέχουσας Π.Ε. χρησιμοποιήθηκε η σύντομη εκδοχή. Σύμφωνα με αυτή, οι συμμετέχοντες καλούνται να βαθμολογήσουν το παιχνίδι ως προς οκτώ (8) χαρακτηριστικά που σχετίζονται με την εμπειρία χρήστη. Τα μισά από αυτά σχετίζονται με την "Πρακτική Ποιότητα" του παιχνιδιού και τα άλλα μισά με την "Ήδονική Ποιότητα".

Πιο συγκεκριμένα, τα χαρακτηριστικά αυτά είναι:

1. Υποστηρικτικότητα
2. Πολυπλοκότητα
3. Αποδοτικότητα
4. Σαφήνεια
5. Συναρπαστικότητα
6. Ενδιαφέρον
7. Ευρηματικότητα
8. Πρωτοπορία

Κάθε χαρακτηριστικό βαθμολογείται από το ένα (1) έως το επτά (7). Το ένα (1) αντιπροσωπεύει το χαμηλότερο (αρνητικό) βαθμό ενώ το επτά (7) το μεγαλύτερο (θετικό). Μετά τη συγκέντρωση όλων των απαντήσεων-βαθμών, οι βαθμοί μετασχηματίζονται από την κλίμακα [0, 7] στην κλίμακα [-3, 3]. Στο τέλος, προκύπτουν τα αποτελέσματα της αξιολόγησης όπως περιγράφεται παρακάτω.

Για την τρέχουσα Π.Ε., η συγκέντρωση των βαθμολογιών και, στη συνέχεια, η στατιστική τους επεξεργασία βασίστηκε στο Data Analysis Tool (έκδοσης 14) της σύντομης εκδοχής του Ερωτηματολογίου Εμπειρίας Χρήστη (UEQ) [68].

Συγκεκριμένα, για το παιχνίδι VRacer, συμπληρώθηκαν 10 ερωτηματολόγια τα οποία παρουσιάζονται αναλυτικά στο ΠΑΡΑΡΤΗΜΑ ΣΤ. Στον παρακάτω πίνακα (βλ. Πίνακας 7.10) παρουσιάζεται ο πίνακας – του Data Analysis Tool – με την αναλυτική βαθμολογία που δόθηκε από κάθε ερωτηματολόγιο. Ειδικότερα, κάθε γραμμή αντιπροσωπεύει ένα ερωτηματολόγιο ενώ κάθε στήλη αντιπροσωπεύει ένα από τα παραπάνω χαρακτηριστικά εμπειρίας χρήστης.

Πίνακας 7.10: Αναλυτική βαθμολόγηση ανά χαρακτηριστικό του παιχνιδιού VRacer

Items							
1	2	3	4	5	6	7	8
7	6	6	5	7	7	7	5
6	7	6	7	7	7	6	7
6	7	7	7	7	7	6	6
7	6	7	7	6	6	7	7
7	6	6	6	5	6	6	6
6	7	7	6	6	7	6	6
6	7	7	7	6	7	7	6
7	6	6	6	5	6	6	6
7	6	6	6	5	6	5	6
7	6	6	6	5	6	6	5

7.3.1 Ανάλυση αποτελεσμάτων ανά Χαρακτηριστικό

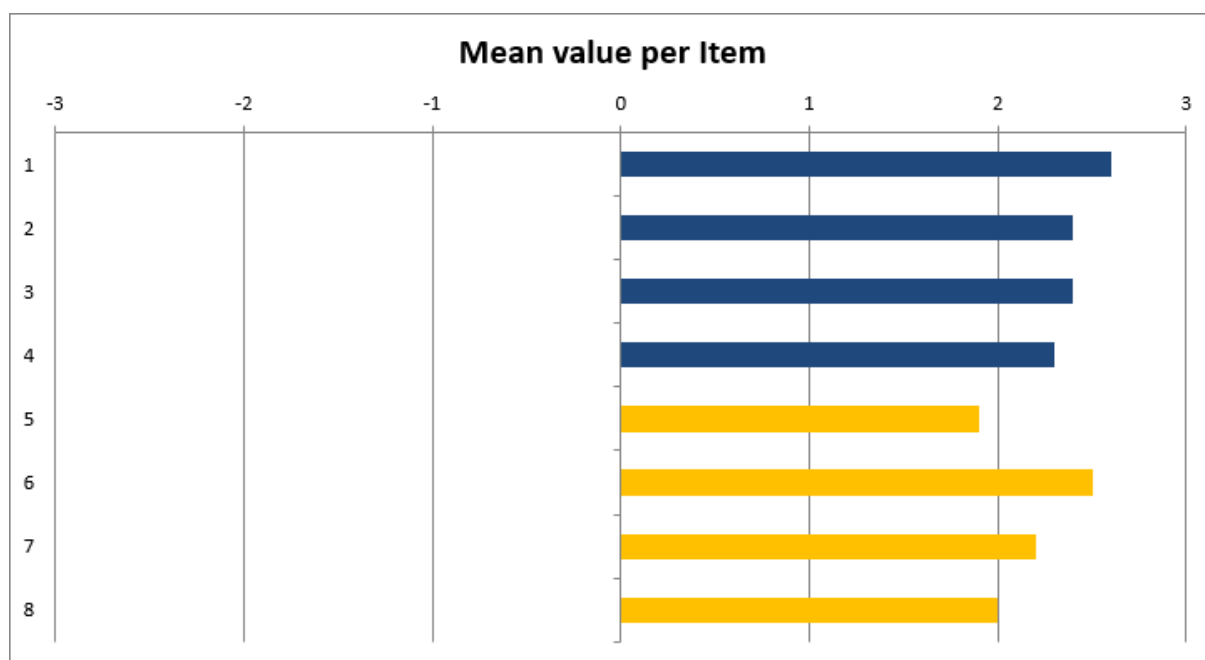
Κατά την ανάλυση αυτή, για κάθε χαρακτηριστικό, υπολογίζονται, με βάση του βαθμούς που έχει δεχθεί αυτό, τα ακόλουθα μεγέθη: μέση τιμή, διακύμανση και τυπική απόκλιση. Οι μέσες τιμές που βρίσκονται μεταξύ του -0,8 και του 0,8 εκφράζουν μια ουδέτερη αξιολόγηση. Οι μέσες τιμές που είναι μεγαλύτερες

του 0,8 εκφράζουν μια θετική αξιολόγηση ενώ αυτές που είναι μικρότερες του -0,8 εκφράζουν μια αρνητική αξιολόγηση.

Για το παιχνίδι VRacer, τα συγκεντρωτικά αποτελέσματα των ερωτηματολογίων, ανά χαρακτηριστικό, προσφέρονται αυτόματα από το Data Analysis Tool. Συγκεκριμένα, ο Πίνακας 7.11 παρουσιάζει τα συγκεντρωτικά αποτελέσματα ενώ, το αντίστοιχο διάγραμμα φαίνεται στο Σχήμα 7.45.

Πίνακας 7.11: Συγκεντρωτικά αποτελέσματα ανά χαρακτηριστικό του παιχνιδιού VRacer

Item	Mean	Variance	Std. Dev.	No.	Negative	Positive	Scale
1	↑ 2,6	0,3	0,5	10	παρεμποδιστικό	υποστηρικτικό	Πρακτική Ποιότητα
2	↑ 2,4	0,3	0,5	10	πολύπλοκο	απλό	Πρακτική Ποιότητα
3	↑ 2,4	0,3	0,5	10	μη αποδοτικό	αποδοτικό	Πρακτική Ποιότητα
4	↑ 2,3	0,5	0,7	10	ασαφές	σαφές	Πρακτική Ποιότητα
5	↑ 1,9	0,8	0,9	10	βαρετό	συναρπαστικό	Ήδονική Ποιότητα
6	↑ 2,5	0,3	0,5	10	αδιάφορο	ενδιαφέρον	Ήδονική Ποιότητα
7	↑ 2,2	0,4	0,6	10	συμβατικό	ευρηματικό	Ήδονική Ποιότητα
8	↑ 2,0	0,4	0,7	10	συνηθισμένο	πρωτοποριακό	Ήδονική Ποιότητα



Σχήμα 7.45: Διάγραμμα συγκεντρωτικών αποτελεσμάτων ανά χαρακτηριστικό του παιχνιδιού VRacer

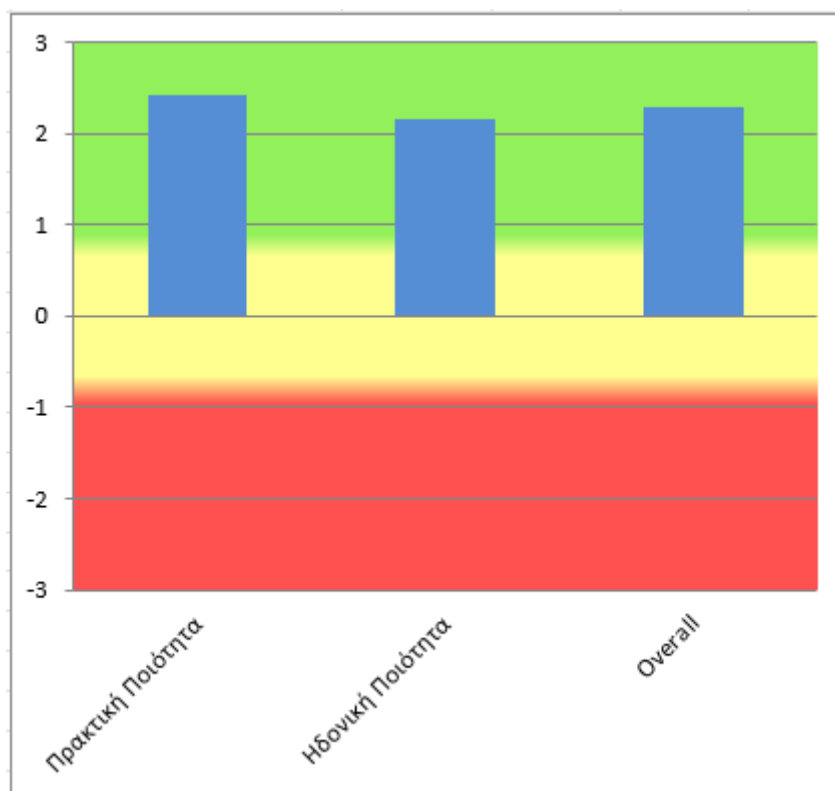
7.3.2 Ανάλυση αποτελεσμάτων ανά Ποιότητα

Κατά την ανάλυση αυτή, για κάθε ερωτηματολόγιο, προκύπτουν δύο μέση όρη: ένας για την "Πρακτική Ποιότητα" (στον οποίο συμμετέχουν οι βαθμοί των τεσσάρων (4) πρώτων χαρακτηριστικών του ερωτηματολογίου) και ένας για την "Ήδονική Ποιότητα" (στον οποίο συμμετέχουν οι βαθμοί των υπολοίπων χαρακτηριστικών). Στη συνέχεια, για κάθε "ποιότητα", υπολογίζεται ο μέσος όρος των παραπάνω μέσων όρων.

Για το παιχνίδι VRacer, τα συγκεντρωτικά αποτελέσματα των ερωτηματολογίων, ανά ποιότητα, προσφέρονται αυτόματα από το Data Analysis Tool. Συγκεκριμένα, ο Πίνακας 7.12 παρουσιάζει τα συγκεντρωτικά αποτελέσματα ενώ, το αντίστοιχο διάγραμμα φαίνεται στο Σχήμα 7.46.

Πίνακας 7.12: Συγκεντρωτικά αποτελέσματα ανά ποιότητα του παιχνιδιού VRacer

Short UEQ Scales	
Πρακτική Ποιότητα	↑ 2,425
Ηδονική Ποιότητα	↑ 2,150
Overall	↑ 2,288



Σχήμα 7.46: Διάγραμμα συγκεντρωτικών αποτελεσμάτων ανά ποιότητα του παιχνιδιού VRacer

7.3.3 Αξιολόγηση αποτελεσμάτων

Με βάση τις παραπάνω αναλύσεις προκύπτει, ως γενική εικόνα, ότι το παιχνίδι VRacer προσφέρει μία εξαιρετική εμπειρία χρήστη. Μάλιστα, τα αποτελέσματα της Πρακτικής και Ηδονικής Ποιότητας είναι πολύ πιο πάνω από το κατώφλι των θετικών αποτελεσμάτων (+0,8).

Ειδικότερα, η Πρακτική Ποιότητα (2,425) δείχνει να είναι η ισχυρότερη διάσταση. Οι χρήστες αξιολογούν το παιχνίδι ως εξαιρετικά υποστηρικτικό, απλό και αποδοτικό. Ταυτόχρονα, η μικρή τυπική απόκλιση (~0.5) δείχνει μεγάλη συμφωνία μεταξύ των ερωτηθέντων. Το "σαφές" έχει ελαφρώς μεγαλύτερη διακύμανση (Std. Dev. = 0.7), που σημαίνει κάποιες διαφωνίες ως προς τη σαφήνεια.

Η Ηδονική Ποιότητα (2,15) είναι επίσης πολύ υψηλή, αλλά με μεγαλύτερη διακύμανση. Αξίζει να σημειωθεί ότι το "ενδιαφέρον" σκόραρε υψηλότερα (2,50) από το "συναρπαστικό" (1,90). Επίσης, το "συναρπαστικό" έχει τη μεγαλύτερη διακύμανση (Std. Dev. = 0,9) ολόκληρης της έρευνας. Πιθανώς αυτό να σημαίνει ότι το παιχνίδι δεν είναι από τη φύση του "εκπληκτικό/ενθουσιώδες", κάτι που είναι φυσιολογικό για αρκετές κατηγορίες εφαρμογών.

7.4 Μελλοντικές Προκλήσεις

Με βάση τον παραπάνω απολογισμό, προκύπτει το γενικό συμπέρασμα ότι οι στόχοι που έθεσε η τρέχουσα Π.Ε. επιτεύχθηκαν σε ένα ικανοποιητικό βαθμό. Σίγουρα όμως είναι αρκετά που θα μπορούσαν να γίνουν ακόμα.

Καταρχάς, το VRacer είναι ένα παιχνίδι. Αυτή τη στιγμή όμως, στην πρώτη του έκδοση, δε θέτει κάποιους ιδιαίτερους στόχους στον παίκτη-χρήστη. Καταρχάς, ο παίκτης-χρήστης δεν αισθάνεται την ανάγκη να οδηγήσει γρήγορα, έχοντας ως "αντίπαλό" του το χρόνο. Αυτό συμβαίνει γιατί, πολύ απλά, λείπει το κίνητρο για κάτι τέτοιο. Σίγουρα, λοιπόν, η θέσπιση ενός τέτοιου κινήτρου είναι μία πολύ καλή μελλοντική πρόκληση.

Επιπλέον, μία βελτίωση που θα έδινε ένα επιπρόσθετο κίνητρο στον παίκτη-χρήστη είναι η ύπαρξη πολλών διαθέσιμων πιστών ταχύτητας με αυξανόμενο δείκτη δυσκολίας. Ταυτόχρονα, με αυτόν τον τρόπο, η εμπειρία χρήστη θα γινόταν ακόμα πιο πλούσια.

Τέλος, ένα ενδιαφέρον πεδίο έρευνας και ανάπτυξης, για τις μελλοντικές εκδόσεις του παιχνιδιού είναι μία πιο εξειδικευμένη διαχείριση των ήχων και της δόνησης. Ειδικότερα, θα μπορούσε να γίνει μία αναλυτικότερη μελέτη των διαφόρων πηγών ήχου και δόνησης. Έτσι, θα παράγονταν μία μεγαλύτερη ποικιλία ερεθισμάτων κάνοντας ακόμα πιο πλούσια την εμπειρία χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] C. F. M. & P. C. Cruz-Neira, «Virtual Reality and Games,» *Multimodal Technologies and Interaction*, 2018. [Ηλεκτρονικό]. Available: <https://www.mdpi.com/2414-4088/2/1/8>.
- [2] J. M. Zheng, K. W. Chan, I. Gibson, «Virtual reality,» σε *IEEE Potentials*, IEEE, 1998.
- [3] Alan Wexelblat, *Virtual Reality: Applications and Explorations*, Academic Press, 2014.
- [4] «National Center for Supercomputing Applications: History,» σε *The Board of Trustees of the University of Illinois*, University of Illinois, 2015.
- [5] Engler, Craig E., *Affordable VR by 1994*, Computer Gaming World, 1992.
- [6] Google Street View in 3D: More Than Just an April Fool's Joke, Google, 2010.
- [7] Rubin, Peter, *Oculus Rift*, Wired, 2014.
- [8] James, Paul, *30 Minutes Inside Valve's Prototype Virtual Reality Headset: Owlchemy Labs Share Their Steam Dev Days Experience – Road to VR*, Road to VR, 2014.
- [9] Patel, Nilay, *Apple Vision Pro review: magic, until it's not*, The Verge, 2024.
- [10] *Pilots Are Learning To Fly Helicopters In VR, Thanks To This Swiss Startup*, Forbes, 2024.
- [11] Barfield W, Zeltzer D, Sheridan T, Slater M, «Presence and performance within virtual environments,» σε *Virtual environments and advanced interface design*, Oxford University Press, 1995.
- [12] Carina Girvan, *What is a virtual world? Definition and classification*, Educational Technology Research and Development.
- [13] Ulrike Schultze, *Embodiment and Presence in Virtual Worlds: A Review*, Journal of Information Technology, 2010.
- [14] Mana Farshida, Jeannette Paschena, Theresa Erikssonb, Jan Kietzmann, *Go boldly! Explore augmented reality (AR), virtual reality (VR), and mixed reality (MR) for business*, ELSEVIER, 2018.
- [15] Åsa Fast-Berglund, Liang Gong, Dan Li, *Testing and validating Extended Reality (xR) technologies in manufacturing*, Procedia Manufacturing, 2018.
- [16] David Drassic, Paul Milgram, «Perceptual Issues in Augmented Reality,» σε *Proceedings of SPIE - The International Society for Optical Engineering*, SPIE Volume 2653 Stereoscopic Displays and Virtual Reality Systems III, 1996.
- [17] Paul Milgram, Fumio Kishino, «A Taxonomy of Mixed Reality Visual Displays,» σε *IEICE TRANSACTIONS on Information*, IEICE DIGITAL LIBRARY, 1994.

- [18] Cabero Julio, Julio Barroso, «The educational possibilities of Augmented Reality,» σε *Journal of New Approaches in Educational Research*, Springer, 2016.
- [19] Valente L, Feijo' B, Ribeiro A, Clua E, «The concept of pervasive virtuality and its application in digital entertainment systems,» σε *Proceedings of the 15th international conference on entertainment computing*, 2016.
- [20] Kenneth R. Walsh, Suzanne D. Pawloski, Virtual Reality: A Technology in Need of IS Research, Commun. Assoc. Inf. Syst., 2002.
- [21] Niels Christian Nilsson, Rolf Nordahl, Stefania Serafin, Immersion Revisited: A Review of Existing Definitions of Immersion and Their Relation to Different Theories of Presence, Human Technology, 2016.
- [22] Roy S. Kalawsky, Exploiting Virtual Reality Techniques in Education and Training: Technological Issues, Support Initiative for Multimedia Applications Advisory Group on Computer Graphics, 1996.
- [23] Jonathan Steuer, Defining Virtual Reality: Dimensions Determining Telepresence, Journal of Communication, 2000.
- [24] Maria V Sanchez-Vives, Mel Slater, From presence to consciousness through virtual reality, Nature reviews. Neuroscience, 2005.
- [25] R. Holloway, A. Lastra, Virtual Environments: A Survey of the Technology, SIGGRAPH, 1995.
- [26] Ben Lang, «Oculus Rift S revealed with inside-out tracking, resolution bump, & new ergonomics,» 2019. [Ηλεκτρονικό]. Available: <https://www.roadtovr.com/oculus-rift-s-specs-release-date-announcement-gdc-2019/>.
- [27] D.J. Sturman, D. Zeltzer, A survey of glove-based input, IEEE Computer Graphics and Applications, 1994.
- [28] Y Boas, «Overview of virtual reality technologies,» Interactive Multimedia Conference, 20213.
- [29] V. B. V. d. L. H. Shum LC, «Determining the Accuracy of Oculus Touch Controllers for Motor Rehabilitation Applications Using Quantifiable Upper Limb Kinematics: Validation Study,» 2019. [Ηλεκτρονικό]. Available: <https://biomedeng.jmir.org/2019/1/e12291/>.
- [30] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. Defanti, Robert V. Kenyon, John C. Hart, The cave-audio visual experience virtual environment, SIGGRAPH, 1992.
- [31] J.M. Moshell, Three view of virtual reality: Virtual environments in the U.S. military, IEEE Computer, 1993.
- [32] Yun-Chieh Fan, Chih-Yu Wen, «A Virtual Reality Soldier Simulator with Body Area Networks for Team Training,» σε *Middleware Solutions for Wireless Internet of Things*, MDPI, 2019.
- [33] M. Zyda, From visual simulation to virtual reality to games, Computer, 2005.
- [34] Laura Freina, Michela Ott, A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives, eLearning and Software for Education (eLSE), 2015.

- [35] Abbie Howard Brown, Tim Green, Virtual reality: low-cost tools and resources for the classroom, TechTrends, 2026.
- [36] Vargas González, Andrés N., et al., Exploring the virtuality continuum for complex rule-set education in the context of soccer rule comprehension, Multimodal Technologies and Interaction, 2017.
- [37] Gamito Pedro, et al., Cognitive training on stroke patients via virtual reality-based serious games, Disability and rehabilitation, 2017.
- [38] K. Guo, «Application of virtual reality technology in the development of game industry,» σε *Highlights in Science, Engineering and Technology*, semanticscholar.org, 2022.
- [39] Jonas Freiknecht, Wolfgang Effelsberg, A survey on the procedural generation of virtual worlds, Multimodal Technologies and Interaction, 2017.
- [40] Casas, Sergio, et al., On a first evaluation of romot—A robotic 3D movie theatre—For driving safety awareness, Multimodal Technologies and Interaction, 2017.
- [41] Juan Camilo Gil-Carvajal, Eun Soo Jo, Dong Chul Park, Wookeun Song, Cheol-Ho Jeong, «Driving simulator using virtual reality tools combining sound, vision, vibration, and motion,» σε *Applied Acoustics*, ELSEVIER, 2024.
- [42] A. M. Okamura, M. R. Cutkosky, J. T. Dennerlein, Reality-based models for vibration feedback in virtual environments, IEEE/ASME Transactions on Mechatronics, 2001.
- [43] Xinyi Tao, Keyu Wu, Yujie Yang, «The Effects of Vibration on Assisting Game Play and Improving Player Engagement When Lacking Sound,» σε *Lecture Notes in Computer Science*, SPINGER NATURE, 2021.
- [44] Campos JL, Bühlhoff HH., «Multimodal Integration during Self-Motion in Virtual Reality,» σε *The Neural Bases of Multisensory Processes*, CRC Press/Taylor & Francis, 2012.
- [45] J. Haas, «A history of the unity game engine,» WORCESTER POLYTECHNIC INSTITUTE, 2014.
- [46] A. Kurniawan, Internet of Things Projects with ESP32, Packt, 2019.
- [47] P. F. a. P. S. M. Babiuch, Using the ESP32 Microcontroller for Data Processing, 20th International Carpathian Control Conference (ICCC), 2019.
- [48] J. Linowes, Unity virtual reality projects, Packt Publishing Ltd, 2015.
- [49] «Unity,» Unity Technologies, [Ηλεκτρονικό]. Available: <https://unity.com/>.
- [50] N. Cameron, ESP32 Formats and Communication, Apress, 2023.
- [51] ESPRESSIF, «ESP32 Series - Datasheet,» ESPRESSIF, [Ηλεκτρονικό]. Available: https://documentation.espressif.com/esp32_datasheet_en.pdf.
- [52] «IoT Development Framework,» Espressif, 2023. [Ηλεκτρονικό]. Available: <https://www.espressif.com/en/products/sdks/esp-idf>.

- [53] «ESP-IDF Programming Guide,» Espressif, [Ηλεκτρονικό]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>.
- [54] «Arduino core for the ESP32,» Espressif, 2023. [Ηλεκτρονικό]. Available: <https://github.com/espressif/arduino-esp32>.
- [55] «Getting Started with VS Code IDE – ESP32 – ESP-IDF Programming Guide,» Espressif, 2021. [Ηλεκτρονικό]. Available: <https://www.espressif.com/en/support/documents/technical-documents>.
- [56] Vittorio M. N. Passaro, Antonello Cuccovillo, Lorenzo Vaiani, Martino De Carlo and Carlo Edoardo Campanella, «Gyroscope Technology and Applications: A Review in the Industrial Perspective,» *Sensors*, ap. MDPI, 2017.
- [57] B. Edwards, «Video Games Turn Forty,» Vintage Computing and Gaming, 2017. [Ηλεκτρονικό]. Available: <https://www.vintagecomputing.com/index.php/archives/1107/video-games-turn-forty>.
- [58] C. T. Lab, «Using a Vibration Motor,» 2026. [Ηλεκτρονικό]. Available: <https://lab.arts.ac.uk/books/physical-computing/page/using-a-vibration-motor>.
- [59] «OpenGameArt.org,» [Ηλεκτρονικό]. Available: <https://opengameart.org/>.
- [60] M. E. Mortenson, Mathematics for computer graphics applications, Industrial Press Inc., 1999.
- [61] R. Mukundan, «"Quaternions." Advanced methods in computer graphics,» Springer, 2012.
- [62] J. Baumgarte, «Anleitung zum praktischen Gebrauch von Quaternionen,» Technische Universität Braunschweig, 1978.
- [63] Unity, «Rigidbody,» [Ηλεκτρονικό]. Available: <https://docs.unity3d.com/6000.3/Documentation/ScriptReference/Rigidbody.html>.
- [64] Risal, A., Fadhilatunisa, D., Ashadi, N. R., Nasrullah, A. H., & Dewi, A. C., «Basic Training of Arduino Microcontroller Based on Wokwi for Beginner Student Groups,» Jurnal Sipakatau: Inovasi Pengabdian Masyarakat, 2024. [Ηλεκτρονικό]. Available: <https://journal.globresco.com/index.php/Sipakatau/article/view/330>.
- [65] «Null-modem Emulator,» Com0Com, [Ηλεκτρονικό]. Available: <https://com0com.sourceforge.net/>.
- [66] «PuTTY,» PuTTY Software, [Ηλεκτρονικό]. Available: <https://putty.sourceforge.net/>.
- [67] Bettina Laugwitz, Theo Held, Martin Schrepp, «Construction and evaluation of a user experience questionnaire,» USAB, 2008.
- [68] «User Experience Questionnaire,» UEQ, [Ηλεκτρονικό]. Available: <https://www.ueq-online.org/>.
- [69] "Anaconda," 2016. [Online]. Available: <https://www.anaconda.com/>.
- [70] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, GPU Computing, Proceedings of the IEEE, 2008.

[71] «Unity Editor,» Unity, [Ηλεκτρονικό]. Available: <https://docs.unity3d.com/Manual/unity-editor.html>.

ΠΑΡΑΡΤΗΜΑ Α: Κώδικας της κλάσης CarController

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO.Ports;
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit.Inputs.Haptics;

// -----
// -----
// Κλάση που περιέχει όλες τις ιδιότητες του αυτοκινήτου (ρυθμίσεις εισόδου,
// φυσικής, δονήσεων, ήχου)
// που αποθηκεύονται/φορτώνονται προς/από το αρχείο ρυθμίσεων car_config.json.
// -----
// -----

[Serializable]
public class CarConfig
{
    [Header("Ρυθμίσεις Εισόδου")]
    // Όνομα σειριακής θύρας
    public string portName = "COM9";
    // Ρυθμός μετάδοσης δεδομένων σε baud
    public int baudRate = 115200;

    [Header("Ρυθμίσεις WiFi UDP")]
    // Διεύθυνση IP απομακρυσμένης συσκευής UDP
    public string udpRemoteIp = "192.168.2.18";
    // Θύρα αποστολής UDP
    public int udpRemotePort = 4220;
    // Θύρα λήψης UDP
    public int udpLocalPort = 4210;

    [Header("Ρυθμίσεις Αυτοκινήτου")]

```

```

// Μέγιστη ταχύτητα οχήματος
public float maxSpeed = 15f;
// Δύναμη επιτάχυνσης
public float acceleration = 50f;
// Συντελεστής απόσβεσης πέδησης
public float brakeSmooth = 8f;
// Ταχύτητα στροφής τιμονιού (μοίρες/δευτερόλεπτο)
public float turnSpeed = 40f;
// Συντελεστής διατήρησης εγκάρσιας ταχύτητας κατά το drift
public float driftFactor = 0.95f;
// Μήκος ακτίνας ανίχνευσης εδάφους
public float rayLength = 1.5f;
// Ελάχιστος συντελεστής τιμονιού σε χαμηλή ταχύτητα
public float minSteerFactor = 0.35f;
// Πολλαπλασιαστής στροφής κατά τη διάρκεια drift
public float driftTurnMultiplier = 1.4f;

```

```

[Header("Δόνηση Πλαγιολίσθησης")]

```

```

// Ένταση δόνησης ολίσθησης
[Range(0f, 1f)] public float hapticAmplitude = 0.6f;
// Διάρκεια παλμού δόνησης ολίσθησης (δευτερόλεπτα)
public float hapticPulseDuration = 0.08f;
// Διάστημα επανάληψης παλμών δόνησης (δευτερόλεπτα)
public float hapticRepeatInterval = 0.12f;

```

```

[Header("Δόνηση Σύγκρουσης")]

```

```

// Μέγιστη ένταση δόνησης σύγκρουσης
[Range(0f, 1f)] public float collisionMaxAmplitude = 1f;
// Διάρκεια παλμού δόνησης σύγκρουσης (δευτερόλεπτα)
public float collisionPulseDuration = 0.15f;
// Κατώφλι ταχύτητας για ενεργοποίηση δόνησης σύγκρουσης
public float collisionVelocityThreshold = 7f;

```

```

[Header("Ήχος Σύγκρουσης")]

```

```

// Ένταση ήχου σύγκρουσης
[Range(0f, 1f)] public float collisionClipVolume = 1f;

```

```

// Τυχαία διακύμανση τόνου ήχου σύγκρουσης
public float collisionPitchVariance = 0.05f;

[Header("Ήχος Μηχανής")]
// Ελάχιστη ένταση ήχου κινητήρα
[Range(0f, 1f)] public float engineMinVolume = 0.05f;
// Μέγιστη ένταση ήχου κινητήρα
[Range(0f, 1f)] public float engineMaxVolume = 1f;
// Ελάχιστος τόνος ήχου κινητήρα
public float engineMinPitch = 0.9f;
// Μέγιστος τόνος ήχου κινητήρα
public float engineMaxPitch = 1.6f;
// Σημαία επιρροής γκαζιού στον τόνο κινητήρα
public bool engineUseThrottleForPitch = true;
}

// -----
// -----
// Κλάση Ελεγκτή αυτοκινήτου (carController)
// -----
// -----

[RequireComponent(typeof(Rigidbody))]
public class carController : MonoBehaviour
{
    [Header("Ήχος Σύγκρουσης")]
    [Tooltip("Ήχος σύγκρουσης που θα αναπαράγεται στη θέση του AudioListener (κάμερα).")]
    public AudioClip collisionClip; // Κλιπ ήχου που αναπαράγεται κατά τη σύγκρουση

    [Header("Ήχος Μηχανής")]
    public AudioSource engineAudioSource; // Πηγή ήχου κινητήρα

    // Διαδρομή αποθήκευσης αρχείου ρυθμίσεων.
    private static string ConfigDirectory
    {
        get
        {

```

```

#if UNITY_EDITOR

    return Application.persistentDataPath; //
Κατά την εκτέλεση από τον Editor, χρήση του persistentDataPath
#else

    return Path.GetDirectoryName(
        System.Reflection.Assembly.GetExecutingAssembly().Location); //
Κατά την εκτέλεση εκτός Editor, χρήση του φακέλου της εγκατάστασης
#endif

    }

}

// Πλήρης διαδρομή προς το αρχείο ρυθμίσεων car_config.json.
// Συνένωση καταλόγου και ονόματος αρχείου
private static string ConfigPath =>
    Path.Combine(ConfigDirectory, "car_config.json");

// Αντικείμενο ρυθμίσεων με προεπιλεγμένες τιμές
private CarConfig _cfg = new CarConfig();

// Αναφορά στο συστατικό Rigidbody του οχήματος
private Rigidbody _rb;
// Κατακόρυφη είσοδος (γκάζι/φρένο)
private float _inputVertical;
// Οριζόντια είσοδος (τιμόνι)
private float _inputHorizontal;
// Κάθετο διάνυσμα επιφάνειας εδάφους
private Vector3 _groundNormal;

// Χρονόμετρο επανάληψης παλμών δόνησης drift
private float _driftHapticTimer;
// Χρονόμετρο διάρκειας δόνησης σύγκρουσης
private float _collisionHapticTimer;
// Εκκρεμής ένταση δόνησης από την τελευταία σύγκρουση
private float _pendingCollisionAmplitude;

// Σημαία ελέγχου νημάτων επικοινωνίας
private bool _running = true;

```

```

// Αντικείμενο σειριακής θύρας
private SerialPort _port;
// Νήμα ανάγνωσης σειριακής θύρας
private Thread _portReadThread;

// Αντικείμενο UDP σύνδεσης
private UdpClient _udpClient;
// Νήμα ανάγνωσης UDP πακέτων
private Thread _udpReadThread;
// Διεύθυνση και θύρα απομακρυσμένης συσκευής UDP
private IPEndPoint _udpRemoteEndPoint;

// Τελευταία ληφθείσα είσοδος από θύρα/UDP
private Vector2 _portInput = Vector2.zero;
// Αντικείμενο κλειδώματος για ασφαλή πρόσβαση μεταξύ νημάτων
private readonly object _portDataLock = new object();

// Βοηθητική ιδιότητα που καθορίζει αν θα χρησιμοποιηθεί WiFi (UDP) ή σειριακή
// θύρα για την είσοδο.
private bool UseWifi =>
    string.IsNullOrEmpty(_cfg.portName) &&
    !string.IsNullOrEmpty(_cfg.udpRemoteIp); // True αν δεν έχει οριστεί θύρα αλλά έχει
    οριστεί IP

//
=====
// Αποθήκευση/Φόρτωση ρυθμίσεων
//
=====

/// <summary>
/// Φόρτωση ρυθμίσεων από το αρχείο car_config.json.
/// </summary>
/// <remarks>
/// Φορτώνει τις ρυθμίσεις από το αρχείο car_config.json. Σε περίπτωση που το
// αρχείο λείπει,
/// το δημιουργεί και αποθηκεύει τις προεπιλεγμένες τιμές.

```

```

/// </remarks>
public void LoadConfig()
{
    try
    {
        // Έλεγχος αν υπάρχει το αρχείο ρυθμίσεων
        if (File.Exists(ConfigPath))
        {
            // Ανάγνωση περιεχομένου αρχείου JSON
            string json = File.ReadAllText(ConfigPath);
            // Αποσειριοποίηση JSON σε αντικείμενο CarConfig
            _cfg = JsonUtility.FromJson<CarConfig>(json) ?? new CarConfig();
            // Καταγραφή επιτυχούς φόρτωσης
            Debug.Log($"[CarConfig] Loaded from {ConfigPath}");
        }
        else
        {
            // Δημιουργία αντικειμένου με προεπιλεγμένες τιμές
            _cfg = new CarConfig();
            // Αποθήκευση προεπιλογών ώστε να είναι επεξεργάσιμες
            SaveConfig();
            // Καταγραφή δημιουργίας προεπιλεγμένου αρχείου
            Debug.Log($"[CarConfig] Default config written to {ConfigPath}");
        }
    }
    catch (Exception e)
    {
        // Καταγραφή σφάλματος φόρτωσης
        Debug.LogWarning($"[CarConfig] Load failed - using defaults. ({e.Message})");
        // Επαναφορά στις προεπιλεγμένες τιμές μετά από σφάλμα
        _cfg = new CarConfig();
    }
}

/// <summary>
/// Αποθήκευση ρυθμίσεων στο αρχείο car_config.json.

```

```

    /// </summary>
    /// <remarks>
    /// Σειριοποιεί το αντικείμενο ρυθμίσεων σε JSON και το αποθηκεύει στο αρχείο
    /// car_config.json. Καταγράφει προειδοποίηση στην περίπτωση αποτυχίας
    /// εγγραφής.
    /// </remarks>
    public void SaveConfig()
    {
        try
        {
            // Σειριοποίηση αντικειμένου σε μορφοποιημένο JSON
            string json = JsonUtility.ToJson(_cfg, prettyPrint: true);
            // Εγγραφή JSON στο αρχείο ρυθμίσεων
            File.WriteAllText(ConfigPath, json);
            // Καταγραφή επιτυχούς αποθήκευσης
            Debug.Log($"[CarConfig] Saved to {ConfigPath}");
        }
        catch (Exception e)
        {
            Debug.LogWarning($"[CarConfig] Save failed. ({e.Message})"); //
            Καταγραφή σφάλματος αποθήκευσης
        }
    }

    /// <summary>
    /// Πρόσβαση στο αντικείμενο ρυθμίσεων κατά την εκτέλεση.
    /// </summary>
    /// <remarks>
    /// Παρέχει το αντικείμενο CarConfig για ανάγνωση και τροποποίηση κατά την
    /// εκτέλεση εκτός του Unity Editor.
    /// </remarks>
    public CarConfig Config => _cfg; // Αντικείμενο πρόσβασης των ρυθμίσεων

    //
    =====
    // Κύκλος ζωής Unity

```

```

//
=====

/// <summary>
/// Αρχικοποίηση αντικειμένου κατά την εκκίνηση (Awake).
/// </summary>
/// <remarks>
/// Φορτώνει τις ρυθμίσεις, αρχικοποιεί το Rigidbody, επανατοποθετεί το όχημα
/// στο έδαφος μέσω raycast και ξεκινά την πηγή ήχου κινητήρα.
/// </remarks>
void Awake()
{
    // Φόρτωση ρυθμίσεων από αρχείο
    LoadConfig();

    // Λήψη αναφοράς στο Rigidbody του αντικειμένου
    _rb = GetComponent<Rigidbody>();
    // Κλείδωμα περιστροφής στους άξονες X και Z
    _rb.constraints = RigidbodyConstraints.FreezeRotationX |
RigidbodyConstraints.FreezeRotationZ;
    // Ενεργοποίηση παρεμβολής για ομαλή κίνηση
    _rb.interpolation = RigidbodyInterpolation.Interpolate;
    // Συνεχής ανίχνευση συγκρούσεων για αποφυγή διέλευσης
    _rb.collisionDetectionMode = CollisionDetectionMode.Continuous;

    // Δημιουργία ακτίνας από ψηλά προς τα κάτω
    Ray ray = new Ray(transform.position + Vector3.up * 5f, Vector3.down);
    // Εκτόξευση ακτίνας για εύρεση εδάφους
    if (Physics.Raycast(ray, out RaycastHit hit, 10f))
        // Τοποθέτηση οχήματος ακριβώς πάνω από το έδαφος
        transform.position = hit.point + Vector3.up * 0.5f;
    // Μηδενισμός αρχικής ταχύτητας
    _rb.linearVelocity = Vector3.zero;

    if (engineAudioSource == null)
        // Εύρεση πηγής ήχου αν δεν έχει οριστεί στον Inspector

```

```

        engineAudioSource = GetComponent<AudioSource>();
    if (engineAudioSource != null)
    {
        // Ενεργοποίηση επανάληψης ήχου κινητήρα
        engineAudioSource.loop = true;
        if (!engineAudioSource.isPlaying)
            // Εκκίνηση αναπαραγωγής αν δεν παίζει ήδη
            engineAudioSource.Play();
    }
}

/// <summary>
/// Εκκίνηση νημάτων επικοινωνίας κατά την αρχικοποίηση σκηνής (Start).
/// </summary>
/// <remarks>
/// Ανοίγει τη σύνδεση UDP ή τη σειριακή θύρα ανάλογα με τη διαμόρφωση και
/// εκκινεί το αντίστοιχο νήμα ανάγνωσης στο παρασκήνιο.
/// </remarks>
void Start()
{
    // Έλεγχος αν η διαμόρφωση χρησιμοποιεί WiFi UDP
    if (UseWifi)
    {
        // Δημιουργία UDP client που ακούει στην τοπική θύρα
        _udpClient = new UdpClient(_cfg.udpLocalPort);
        // Ορισμός απομακρυσμένης διεύθυνσης και θύρας
        _udpRemoteEndPoint = new IPEndPoint(IPAddress.Parse(_cfg.udpRemoteIp),
        _cfg.udpRemotePort);
        // Δημιουργία νήματος ανάγνωσης UDP στο παρασκήνιο
        _udpReadThread = new Thread(UdpReadLoop) { IsBackground = true };
        // Εκκίνηση νήματος ανάγνωσης UDP
        _udpReadThread.Start();
        // Καταγραφή παραμέτρων σύνδεσης UDP
        Debug.Log($"[UDP] Listening on port {_cfg.udpLocalPort}, sending to
        {_cfg.udpRemoteIp}:{_cfg.udpRemotePort}");
    }
    // Έλεγχος αν έχει οριστεί σειριακή θύρα

```

```

else if (!string.IsNullOrEmpty(_cfg.portName))
{
    // Δημιουργία σειριακής θύρας με timeout ανάγνωσης
    _port = new SerialPort(_cfg.portName, _cfg.baudRate) { ReadTimeout =
100 };

    // Άνοιγμα σειριακής θύρας
    _port.Open();
    // Δημιουργία νήματος ανάγνωσης σειριακής θύρας στο παρασκήνιο
    _portReadThread = new Thread(PortReadLoop) { IsBackground = true };
    // Εκκίνηση νήματος ανάγνωσης σειριακής θύρας
    _portReadThread.Start();
}
}

/// <summary>
/// Ενημέρωση εισόδου, δονήσεων και ήχου κινητήρα για το τρέχον καρέ (Update).
/// </summary>
/// <remarks>
/// Διαβάζει την είσοδο από τη σειριακή θύρα/UDP ή από το πληκτρολόγιο,
ενημερώνει
/// τους χρονομετρητές δονήσεων και προσαρμόζει τον ήχο κινητήρα.
/// </remarks>
void Update()
{
    // Έξοδος από την εφαρμογή με το πλήκτρο Escape
    if (Input.GetKey("escape"))
        Application.Quit();

    // Έλεγχος αν χρησιμοποιείται εξωτερική είσοδος (UDP ή σειριακή)
    if (UseWifi || !string.IsNullOrEmpty(_cfg.portName))
    {
        // Κλείδωμα για ασφαλή ανάγνωση από το νήμα επικοινωνίας
        lock (_portDataLock)
        {
            // Ανάθεση κατακόρυφης εισόδου από εξωτερική πηγή
            _inputVertical = _portInput.y;
            // Ανάθεση οριζόντιας εισόδου από εξωτερική πηγή

```

```

        _inputHorizontal = _portInput.x;
    }
}
else
{
    // Ανάγνωση κατακόρυφης εισόδου από πληκτρολόγιο/χειριστήριο
    _inputVertical = Input.GetAxis("Vertical");
    // Ανάγνωση οριζόντιας εισόδου από πληκτρολόγιο/χειριστήριο
    _inputHorizontal = Input.GetAxis("Horizontal");
}

// Ενημέρωση δόνησης πλαγιολίσθησης
UpdateDriftHaptics();
// Ενημέρωση δόνησης σύγκρουσης
UpdateCollisionHaptics();
// Ενημέρωση ήχου κινητήρα
UpdateEngineAudio();
}

/// <summary>
/// Ενημέρωση φυσικής κίνησης και κατεύθυνσης σε σταθερά χρονικά διαστήματα
(FixedUpdate).
/// </summary>
/// <remarks>
/// Καλεί διαδοχικά τις μεθόδους εδάφους, κίνησης, τιμονιού και drift
/// με σταθερό βήμα χρόνου φυσικής.
/// </remarks>
void FixedUpdate()
{
    // Ευθυγράμμιση οχήματος με την επιφάνεια εδάφους
    StickToGround();
    // Εφαρμογή δυνάμεων επιτάχυνσης και πέδησης
    HandleMovement();
    // Εφαρμογή περιστροφής τιμονιού
    HandleSteering();
    // Εφαρμογή εφέ πλαγιολίσθησης
    ApplyDrift();
}

```

```

}

/// <summary>
/// Ενημέρωση δόνησης και αναπαραγωγή ήχου κατά το συμβάν σύγκρουσης
(OnCollisionEnter).
/// </summary>
/// <remarks>
/// Υπολογίζει την ένταση σύγκρουσης βάσει της σχετικής ταχύτητας. Αν
υπερβαίνει το
/// κατώφλι, ενεργοποιεί δόνηση χειριστηρίου και αναπαράγει τον ήχο σύγκρουσης
με
/// τυχαία διακύμανση τόνου.
/// </remarks>
/// <param name="collision">Στοιχείο σύγκρουσης (Collision)</param>
void OnCollisionEnter(Collision collision)
{
    // Υπολογισμός μεγέθους σχετικής ταχύτητας σύγκρουσης
    float impactStrength = collision.relativeVelocity.magnitude;
    // Αγνόηση αδύναμων συγκρούσεων κάτω από το κατώφλι
    if (impactStrength <= _cfg.collisionVelocityThreshold) return;

    // Κανονικοποίηση έντασης σύγκρουσης στο [0,1]
    float normalizedImpact = Mathf.Clamp01(impactStrength / _cfg.maxSpeed);
    // Υπολογισμός έντασης δόνησης ανάλογα με την ένταση σύγκρουσης
    _pendingCollisionAmplitude = _cfg.collisionMaxAmplitude *
normalizedImpact;
    // Εκκίνηση χρονομέτρου δόνησης σύγκρουσης
    _collisionHapticTimer = _cfg.collisionPulseDuration;

    if (collisionClip != null) // Έλεγχος αν έχει οριστεί κλιπ ήχου σύγκρουσης
    {
        // Εύρεση του AudioListener στη σκηνή
        var listener = AudioListener.FindFirstObjectByType<AudioListener>();
        Vector3 playPosition = listener != null
            ? listener.transform.position
            : Camera.main != null ? Camera.main.transform.position :
transform.position; // Επιλογή θέσης αναπαραγωγής: AudioListener > κύρια κάμερα >
όχημα

```

```

// Επιλογή θέσης αναπαραγωγής: AudioListener > κύρια κάμερα > όχημα

// Υπολογισμός έντασης ήχου ανάλογα με την ένταση σύγκρουσης
float volume = Mathf.Clamp01(normalizedImpact) *
_cfg.collisionClipVolume;

// Έλεγχος αν η διακύμανση τόνου είναι μηδενική
if (Mathf.Approximately(_cfg.collisionPitchVariance, 0f))
{
    // Αναπαραγωγή ήχου χωρίς διακύμανση τόνου
    AudioSource.PlayClipAtPoint(collisionClip, playPosition, volume);
}
else
{
    // Δημιουργία προσωρινού αντικειμένου για αναπαραγωγή ήχου
    GameObject tmp = new GameObject("TempCollisionSFX");
    // Τοποθέτηση στη θέση αναπαραγωγής
    tmp.transform.position = playPosition;
    // Προσθήκη πηγής ήχου στο προσωρινό αντικείμενο
    var src = tmp.AddComponent<AudioSource>();
    // Ανάθεση κλιπ ήχου σύγκρουσης
    src.clip = collisionClip;
    // Ορισμός έντασης ήχου
    src.volume = volume;
    // Ορισμός ως 2D ήχος (χωρίς χωρική εξασθένηση)
    src.spatialBlend = 0f;
    // Τυχαία διακύμανση τόνου εντός του ορισμένου εύρους
    src.pitch = 1f + UnityEngine.Random.Range(
        -_cfg.collisionPitchVariance,
        _cfg.collisionPitchVariance);
    // Εκκίνηση αναπαραγωγής ήχου
    src.Play();
    // Καταστροφή προσωρινού αντικειμένου μετά το τέλος αναπαραγωγής
    Destroy(tmp, collisionClip.length / Mathf.Abs(src.pitch) + 0.1f);
}
}
}

```

```

    /// <summary>
    /// Εκτέλεση εργασιών καθαρισμού πόρων κατά την καταστροφή του αντικειμένου
    (OnDestroy).
    /// </summary>
    /// <remarks>
    /// Τερματίζει τα νήματα ανάγνωσης UDP και σειριακής θύρας και κλείνει
    /// τις αντίστοιχες συνδέσεις πριν αποδεσμευτεί το αντικείμενο.
    /// </remarks>
    void OnDestroy()
    {
        // Σήμα διακοπής σε όλα τα νήματα επικοινωνίας
        _running = false;
        // Αναμονή τερματισμού νήματος UDP (έως 500ms)
        _udpReadThread?.Join(500);
        // Κλείσιμο σύνδεσης UDP
        _udpClient?.Close();
        // Έλεγχος αν υπήρχε ενεργή σειριακή θύρα
        if (!string.IsNullOrEmpty(_cfg.portName))
        {
            // Αναμονή τερματισμού νήματος σειριακής θύρας (έως 500ms)
            _portReadThread?.Join(500);
            // Κλείσιμο σειριακής θύρας
            _port?.Close();
        }
    }

    //
    =====
    // Haptics
    //
    =====

    /// <summary>
    /// Ενημέρωση δόνησης χειριστηρίου βάσει της τρέχουσας κατάστασης ολίσθησης.
    /// </summary>
    /// <remarks>

```

```

/// Εκπέμπει παλμούς δόνησης ανάλογα με την πλευρά στροφής όταν το όχημα
/// βρίσκεται σε drift. Η ένταση κλιμακώνεται με βάση την εγκάρσια ταχύτητα
/// και την είσοδο τιμονιού.
/// </remarks>
private void UpdateDriftHaptics()
{
    // Εκχώρηση εγκάρσιας συνιστώσας ταχύτητας
    Vector3 lateralVel = Vector3.Project(_rb.linearVelocity, transform.right);
    // Ανίχνευση κατάστασης drift βάσει εισόδου και εγκάρσιας ταχύτητας
    bool isDrifting = _inputHorizontal != 0 && _inputVertical != 0 &&
        lateralVel.magnitude > 1f;

    // Μείωση χρονομέτρου δόνησης drift
    _driftHapticTimer -= Time.deltaTime;
    // Έλεγχος αν είναι ώρα για νέο παλμό δόνησης
    if (isDrifting && _driftHapticTimer <= 0f)
    {
        // Επαναφορά χρονομέτρου για τον επόμενο παλμό
        _driftHapticTimer = _cfg.hapticRepeatInterval;
        // Υπολογισμός απόλυτης τιμής εισόδου τιμονιού
        float steerAmount = Mathf.Abs(_inputHorizontal);
        // Αρχικοποίηση εντάσεων δόνησης αριστερού και δεξιού χειριστηρίου
        float leftAmp = 0f, rightAmp = 0f;

        if (_inputHorizontal < 0f) // Στροφή αριστερά
        {
            leftAmp = _cfg.hapticAmplitude * steerAmount; // Μεγαλύτερη δόνηση
            στο αριστερό χειριστήριο
            rightAmp = _cfg.hapticAmplitude * 0.3f; // Μικρότερη δόνηση
            στο δεξί χειριστήριο
        }
        else if (_inputHorizontal > 0f) // Στροφή δεξιά
        {
            rightAmp = _cfg.hapticAmplitude * steerAmount; // Μεγαλύτερη δόνηση
            στο δεξί χειριστήριο
            leftAmp = _cfg.hapticAmplitude * 0.3f; // Μικρότερη δόνηση
            στο αριστερό χειριστήριο
        }
    }
}

```

```

        if (leftAmp > 0f) SendHaptic(leftAmp, _cfg.hapticPulseDuration,
HapticsUtility.Controller.Left); // Αποστολή παλμού δόνησης στο αριστερό
χειριστήριο

        if (rightAmp > 0f) SendHaptic(rightAmp, _cfg.hapticPulseDuration,
HapticsUtility.Controller.Right); // Αποστολή παλμού δόνησης στο δεξί χειριστήριο
    }
}

/// <summary>
/// Ενημέρωση δόνησης χειριστηρίου βάσει της τρέχουσας κατάστασης σύγκρουσης.
/// </summary>
/// <remarks>
/// Εκπέμπει συνεχή παλμό δόνησης και στα δύο χειριστήρια για όσο διάστημα
/// ο χρονομετρητής σύγκρουσης είναι ενεργός μετά από πρόσκρουση.
/// </remarks>
private void UpdateCollisionHaptics()
{
    // Έλεγχος αν είναι ενεργός ο χρονομετρητής σύγκρουσης
    if (_collisionHapticTimer > 0f)
    {
        // Μείωση χρονομέτρου δόνησης σύγκρουσης
        _collisionHapticTimer -= Time.deltaTime;
        // Αποστολή δόνησης στο αριστερό χειριστήριο
        SendHaptic(_pendingCollisionAmplitude, Time.deltaTime,
HapticsUtility.Controller.Left);
        // Αποστολή δόνησης στο δεξί χειριστήριο
        SendHaptic(_pendingCollisionAmplitude, Time.deltaTime,
HapticsUtility.Controller.Right);
    }
}

/// <summary>
/// Αποστολή εντολής δόνησης μέσω WiFi (UDP), σειριακής θύρας ή XR SDK.
/// </summary>
/// <remarks>
/// Επιλέγει αυτόματα το κανάλι αποστολής βάσει της διαμόρφωσης. Στις
/// περιπτώσεις UDP και σειριακής θύρας, κωδικοποιεί την ένταση (0-255) και

```

```

    /// τη διάρκεια σε μια γραμμή κειμένου CSV πριν την αποστολή.
    /// </remarks>
    /// <param name="amplitude">Ένταση δόνησης (0..1)</param>
    /// <param name="duration">Διάρκεια δόνησης σε δευτερόλεπτα</param>
    /// <param name="controller">Δείκτης χειριστηρίου
    (HapticsUtility.Controller)</param>
    private void SendHaptic(float amplitude, float duration,
    HapticsUtility.Controller controller)
    {
        // Έλεγχος έγκυρων τιμών έντασης και διάρκειας
        if (amplitude <= 0f || duration <= 0f) return;

        // Αποστολή μέσω UDP αν η διαμόρφωση χρησιμοποιεί WiFi
        if (UseWifi)
        {
            // Μετατροπή έντασης [0,1] σε ακέραιο [0,255]
            int ampInt = Mathf.Clamp(Mathf.RoundToInt(amplitude * 255f), 0, 255);
            // Μετατροπή διάρκειας σε ακέραιο (x10000 για ακρίβεια)
            int durInt = Mathf.RoundToInt(duration * 10000);
            // Διαμόρφωση μηνύματος CSV
            string message = ampInt + "," + durInt + "\n";
            // Κωδικοποίηση μηνύματος σε bytes ASCII
            byte[] data = Encoding.ASCII.GetBytes(message);
            try { _udpClient.Send(data, data.Length, _udpRemoteEndPoint); }
            catch (Exception e) { Debug.LogWarning($"[UDP] Send error:
            {e.Message}"); } // Καταγραφή σφάλματος αποστολής UDP
        }
        // Αποστολή μέσω σειριακής θύρας αν έχει οριστεί
        else if (!string.IsNullOrEmpty(_cfg.portName))
        {
            // Μετατροπή έντασης [0,1] σε ακέραιο [0,255]
            int ampInt = Mathf.Clamp(Mathf.RoundToInt(amplitude * 255f), 0, 255);
            // Μετατροπή διάρκειας σε ακέραιο (x10000 για ακρίβεια)
            int durInt = Mathf.RoundToInt(duration * 10000);
            // Διαμόρφωση μηνύματος CSV
            string message = ampInt + "," + durInt + "\n";
            lock (_portDataLock)

```

```

        {
            try { _port.Write(message); }           // Αποστολή μηνύματος στη
σειριακή θύρα
            catch { } // Αγνόηση σφαλμάτων σειριακής θύρας
        }
    }
    else
    {
        // Αποστολή δόνησης μέσω XR SDK ως εναλλακτική
        HapticsUtility.SendHapticImpulse(amplitude, duration, controller);
    }
}

//
=====
// Audio
//
=====

/// <summary>
/// Ενημέρωση έντασης και τόνου ήχου κινητήρα βάσει ταχύτητας και γκαζιού.
/// </summary>
/// <remarks>
/// Κλιμακώνει γραμμικά την ένταση και τον τόνο μεταξύ των ελάχιστων και
μέγιστων
/// τιμών ρύθμισης. Αν το engineUseThrottleForPitch είναι ενεργό, ο τόνος
/// επηρεάζεται επιπλέον από την είσοδο γκαζιού.
/// </remarks>
private void UpdateEngineAudio()
{
    // Έξοδος αν δεν υπάρχει πηγή ήχου κινητήρα
    if (engineAudioSource == null) return;

    // Κανονικοποίηση τρέχουσας ταχύτητας στο [0,1]
    float speedNorm = Mathf.Clamp01(_rb.linearVelocity.magnitude /
_cfg.maxSpeed);

```

```

        // Γραμμική παρεμβολή έντασης ήχου βάσει ταχύτητας
        engineAudioSource.volume          =          Mathf.Lerp(_cfg.engineMinVolume,
_cfg.engineMaxVolume, speedNorm);

        // Αρχικός παράγοντας τόνου βάσει ταχύτητας
        float pitchFactor = speedNorm;
        // Πολλαπλασιασμός με κανονικοποιημένη είσοδο γκαζιού αν απαιτείται
        if (_cfg.engineUseThrottleForPitch)
            pitchFactor *= Mathf.Clamp01(Mathf.Abs(_inputVertical));
        // Γραμμική παρεμβολή τόνου ήχου βάσει παράγοντα
        engineAudioSource.pitch            =            Mathf.Lerp(_cfg.engineMinPitch,
_cfg.engineMaxPitch, pitchFactor);
    }

```

```

    //
    =====
    =====

    // Βοηθητικές μέθοδοι φυσικής
    //
    =====
    =====

```

```

    /// <summary>
    /// Ευθυγράμμιση του οχήματος με την επιφάνεια του εδάφους μέσω raycast.
    /// </summary>
    /// <remarks>
    /// Προβάλλει ακτίνα προς τα κάτω και περιστρέφει ομαλά το Rigidbody ώστε ο
    /// άξονας y να ευθυγραμμιστεί με την κάθετο επιφάνειας. Διαχωρίζει επίσης την
    /// ταχύτητα σε εγκάρσια και προς τα εμπρός συνιστώσες για ρεαλιστική κίνηση.
    /// </remarks>
    void StickToGround()
    {
        // Δημιουργία ακτίνας από λίγο πάνω από το κέντρο του οχήματος προς τα κάτω
        Ray ray = new Ray(transform.position + Vector3.up * 0.5f, Vector3.down);
        // Εκτόξευση ακτίνας ανίχνευσης εδάφους
        if (Physics.Raycast(ray, out RaycastHit hit, _cfg.rayLength + 1f))
        {
            // Αποθήκευση κάθετου διανύσματος επιφάνειας εδάφους

```

```

        _groundNormal = hit.normal;
        // Υπολογισμός στόχου περιστροφής για ευθυγράμμιση με εδαφος
        Quaternion targetRot = Quaternion.FromToRotation(transform.up,
_groundNormal) * transform.rotation;
        // Ομαλή παρεμβολή περιστροφής προς τον στόχο
        _rb.MoveRotation(Quaternion.Slerp(_rb.rotation, targetRot, 0.1f));

        // Εκχώρηση εγκάρσιας συνιστώσας ταχύτητας
        Vector3 lateralVel = Vector3.Project(_rb.linearVelocity,
transform.right);
        // Εκχώρηση συνιστώσας ταχύτητας κατά μήκος του εδάφους
        Vector3 forwardVel = Vector3.Project(_rb.linearVelocity,
            Vector3.ProjectOnPlane(transform.forward, _groundNormal));
        // Αναδόμηση ταχύτητας από προς τα εμπρός και εγκάρσια συνιστώσα
        _rb.linearVelocity = forwardVel + lateralVel;
    }
}

/// <summary>
/// Εφαρμογή δύναμης επιτάχυνσης ή πέδησης βάσει της κατακόρυφης εισόδου.
/// </summary>
/// <remarks>
/// Όταν υπάρχει είσοδος γκαζιού, υπολογίζει τη διαφορά ταχύτητας από τον στόχο
/// και εφαρμόζει αντίστοιχη δύναμη. Σε απουσία εισόδου, αποσβένει ομαλά την
/// προς τα εμπρός ταχύτητα προσομοιώνοντας φυσική πέδηση.
/// </remarks>
void HandleMovement()
{
    // Υπολογισμός κατεύθυνσης κίνησης παράλληλης με το έδαφος
    Vector3 forward = Vector3.ProjectOnPlane(transform.forward,
_groundNormal).normalized;

    // Έλεγχος αν υπάρχει ουσιαστική είσοδος γκαζιού/φρένου
    if (Mathf.Abs(_inputVertical) > 0.01f)
    {
        // Υπολογισμός στοχευόμενης ταχύτητας βάσει εισόδου και μέγιστης
ταχύτητας
        Vector3 targetVel = forward * _inputVertical * _cfg.maxSpeed;

```

```

        // Διαφορά τρέχουσας από στοχευόμενη ταχύτητα
        Vector3 velocityDiff = targetVel - Vector3.Project(_rb.linearVelocity,
forward);
        // Εφαρμογή δύναμης επιτάχυνσης περιορισμένης από το μέγιστο
επιτρεπόμενο
        _rb.AddForce(
            Vector3.ClampMagnitude(velocityDiff, _cfg.acceleration *
Time.fixedDeltaTime),
            ForceMode.VelocityChange);
    }
    else
    {
        // Εκχώρηση συνιστώσας ταχύτητας προς τα εμπρός
        Vector3 forwardVel = Vector3.Project(_rb.linearVelocity,
            Vector3.ProjectOnPlane(transform.forward, _groundNormal));
        // Ομαλή απόσβεση προς τα εμπρός ταχύτητας για φυσική πέδηση
        _rb.linearVelocity = Vector3.Lerp(
            _rb.linearVelocity,
            _rb.linearVelocity - forwardVel,
            _cfg.brakeSmooth * Time.fixedDeltaTime);
    }
}

/// <summary>
/// Εφαρμογή περιστροφής τιμονιού κλιμακωμένης βάσει ταχύτητας και drift.
/// </summary>
/// <remarks>
/// Υπολογίζει τον συντελεστή τιμονιού με γραμμική παρεμβολή βάσει
κανονικοποιημένης
/// ταχύτητας. Εφαρμόζει επιπλέον πολλαπλασιαστική drift όταν υπάρχει ταυτόχρονη
/// οριζόντια και κατακόρυφη είσοδος.
/// </remarks>
void HandleSteering()
{
    // Κανονικοποίηση τρέχουσας ταχύτητας στο [0,1]
    float normalizedSpeed = Mathf.Clamp01(_rb.linearVelocity.magnitude /
_cfg.maxSpeed);
    // Παρεμβολή συντελεστή τιμονιού από ελάχιστο έως πλήρες

```

```

        float steerFactor = Mathf.Lerp(_cfg.minSteerFactor, 1f, normalizedSpeed);
        // Εφαρμογή πολλαπλασιαστική drift αν υπάρχει ταυτόχρονη οριζόντια και
        κατακόρυφη είσοδος
        float driftBoost = (_inputHorizontal != 0 && _inputVertical != 0)
            ? _cfg.driftTurnMultiplier : 1f;
        // Υπολογισμός γωνίας στροφής για το τρέχον καρέ
        float turn = _inputHorizontal * _cfg.turnSpeed * steerFactor * driftBoost
        * Time.fixedDeltaTime;
        // Εφαρμογή περιστροφής γύρω από τον κάθετο άξονα του εδάφους
        _rb.MoveRotation(Quaternion.AngleAxis(turn, _groundNormal) *
        _rb.rotation);
    }

    /// <summary>
    /// Μείωση εγκάρσιας ολίσθησης βάσει της κατάστασης drift και των ρυθμίσεων.
    /// </summary>
    /// <remarks>
    /// Κατά τη διάρκεια drift εφαρμόζει μερική απόσβεση της εγκάρσιας ταχύτητας
    /// χρησιμοποιώντας τον συντελεστή driftFactor. Σε κανονική οδήγηση, η εγκάρσια
    /// ταχύτητα αποσβένεται με σταθερό συντελεστή 0.5.
    /// </remarks>
    void ApplyDrift()
    {
        // Εκχώρηση εγκάρσιας συνιστώσας ταχύτητας
        Vector3 lateralVel = Vector3.Project(_rb.linearVelocity, transform.right);
        // Έλεγχος αν το όχημα βρίσκεται σε κατάσταση drift
        if (_inputHorizontal != 0 && _inputVertical != 0)
            // Μερική απόσβεση εγκάρσιας ταχύτητας βάσει συντελεστή drift
            _rb.linearVelocity -= lateralVel * (1f - _cfg.driftFactor);
        else
            // Ισχυρότερη απόσβεση εγκάρσιας ταχύτητας σε κανονική οδήγηση
            _rb.linearVelocity -= lateralVel * 0.5f;
    }

    //
    =====
    =====

    // Νήματα (Thread) επικοινωνίας

```

```

//
=====

/// <summary>
/// Συνεχής ανάγνωση πακέτων UDP στο παρασκήνιο.
/// </summary>
/// <remarks>
/// Εκτελείται σε ξεχωριστό νήμα. Αποκωδικοποιεί κάθε εισερχόμενο πακέτο UDP
/// ως γραμμή ASCII και ενημερώνει το _portInput με τις τελευταίες έγκυρες
/// συντεταγμένες. Σε σφάλμα, μηδενίζει την είσοδο.
/// </remarks>
void UdpReadLoop()
{
    // Δημιουργία endpoint για αποδοχή πακέτων από οποιαδήποτε διεύθυνση
    IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
    // Συνεχής βρόχος μέχρι τη διακοπή του νήματος
    while (_running)
    {
        try
        {
            // Αναμονή και λήψη εισερχόμενου UDP πακέτου
            byte[] data = _udpClient.Receive(ref remoteEP);
            // Αποκωδικοποίηση bytes σε συμβολοσειρά ASCII
            string line = Encoding.ASCII.GetString(data).Trim();
            // Ανάλυση γραμμής σε συντεταγμένες
            if (TryParsePortLine(line, out var coords))
            {
                // Ενημέρωση κοινού buffer εισόδου με ασφαλή κλείδωμα
                lock (_portDataLock) { _portInput = coords; }
            }
        }
        catch
        {
            // Μηδενισμός εισόδου σε περίπτωση σφάλματος
            lock (_portDataLock) { _portInput = Vector2.zero; }
        }
    }
}

```

```

/// <summary>
/// Συνεχής ανάγνωση δεδομένων από τη σειριακή θύρα στο παρασκήνιο.
/// </summary>
/// <remarks>
/// Εκτελείται σε ξεχωριστό νήμα. Διαβάζει γραμμές από τη σειριακή θύρα και
/// ενημερώνει το _portInput με τις τελευταίες έγκυρες συντεταγμένες.
/// Σε σφάλμα ανάγνωσης (π.χ. timeout), μηδενίζει την είσοδο.
/// </remarks>
void PortReadLoop()
{
    // Συνεχής βρόχος μέχρι τη διακοπή του νήματος
    while (_running)
    {
        try
        {
            // Ανάγνωση γραμμής από σειριακή θύρα
            string line = _port.ReadLine();
            // Ανάλυση γραμμής σε συντεταγμένες
            if (TryParsePortLine(line, out var coords))
            {
                // Ενημέρωση κοινού buffer εισόδου με ασφαλή κλείδωμα
                lock (_portDataLock) { _portInput = coords; }
            }
        }
        catch
        {
            // Μηδενισμός εισόδου σε περίπτωση σφάλματος (π.χ. timeout)
            lock (_portDataLock) { _portInput = Vector2.zero; }
        }
    }
}

/// <summary>
/// Ανάλυση γραμμής εισόδου σε ζεύγος συντεταγμένων.
/// </summary>
/// <remarks>
/// Προσπαθεί να αναλύσει μια γραμμή εισόδου σε ζεύγος συντεταγμένων (Vector2).
Αν η ανάλυση αποτύχει,

```

```

    /// επιστρέφει την προεπιλεγμένη τιμή Vector2.zero.
    /// </remarks>
    /// <param name="line">Γραμμή κειμένου εισόδου</param>
    /// <param name="result">Παραγωγή συντεταγμένων (Vector2)</param>
    /// <returns>Λογική τιμή επιτυχίας ανάλυσης</returns>
    bool TryParsePortLine(string line, out Vector2 result)
    {
        // Αρχικοποίηση αποτελέσματος στην προεπιλεγμένη τιμή
        result = default;
        // Αφαίρεση κενών και διαχωρισμός γραμμής βάσει κόμματος
        var parts = line.Trim().Split(',');
        // Έλεγχος αν υπάρχουν ακριβώς δύο τμήματα και αν μπορούν να αναλυθούν ως
float
        if (parts.Length == 2
            && float.TryParse(parts[0], out float x)
            && float.TryParse(parts[1], out float y))
        {
            // Δημιουργία Vector2 από τις αναλυθείσες τιμές
            result = new Vector2(x, y);
            // Επιστροφή επιτυχίας ανάλυσης
            return true;
        }
        // Επιστροφή αποτυχίας ανάλυσης
        return false;
    }
}

```

ΠΑΡΑΡΤΗΜΑ Β: Κώδικας της κλάσης CarXROriginFollow

```
using UnityEngine;

public class CarXROriginFollow : MonoBehaviour
{
    public Transform car;          // Το αυτοκίνητο που ακολουθεί η κάμερα
    public Vector3 localOffset;    // Τοπική μετατόπιση της κάμερας σε σχέση με
    το αυτοκίνητο (π.χ. πίσω και πάνω από το αυτοκίνητο)
    public float followSpeed = 20f; // Ταχύτητα ακολουθίας της κάμερας

    /// <summary>
    /// Εκτελείται στο τέλος του frame. Ενημερώνει τη θέση και περιστροφή της
    κάμερας
    /// ώστε να ακολουθεί ομαλά το όχημα με βάση την τοπική μετατόπιση και την
    κατεύθυνση του αυτοκινήτου.
    /// </summary>
    void LateUpdate()
    {
        if (!car) return;

        // Υπολογισμός της θέσης της κάμερας με βάση τη θέση του αυτοκινήτου και
        την τοπική μετατόπιση
        Vector3 targetPos = car.TransformPoint(localOffset);

        // Συσχέτιση θέσης κάμερας με τη θέση του αυτοκινήτου
        transform.position = Vector3.Lerp(transform.position, targetPos,
        followSpeed * Time.deltaTime);

        // Συσχέτιση περιστροφής κάμερας με την κατεύθυνση του αυτοκινήτου
        Vector3 forward = Vector3.ProjectOnPlane(car.forward, Vector3.up); //
        Περιστροφή μόνο στον άξονα Y (για να μη γέρνει το κεφάλι του παίκτη)
        if (forward.sqrMagnitude > 0.001f) // Αποφυγή περιστροφής όταν το
        αυτοκίνητο είναι σχεδόν ακίνητο
        {
            Quaternion targetRot = Quaternion.LookRotation(forward, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, targetRot,
            followSpeed * Time.deltaTime);
        }
    }
}
```

```
}  
  }  
}
```

ΠΑΡΑΡΤΗΜΑ Γ: Προγραμματισμός του Ηλεκτρονικού Κυκλώματος του Τιμονιού

```
#include <Arduino.h>
#include <Wire.h>
#include <MPU6050_light.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <nvs.h>
#include <nvs_flash.h>

// -----

// Σταθερές WiFi
// -----

// Μέγιστο μήκος του SSID
#define WIFI_SSID_MAX_LEN 32
// Μέγιστο μήκος του κωδικού
#define WIFI_PASSWORD_MAX_LEN 64
// Μέγιστο μήκος της διεύθυνσης IP
#define WIFI_IP_MAX_LEN 16
// Χρόνος αναμονής σύνδεσης WiFi (ms) πριν την - εφεδρική - ενεργοποίηση της
σειριακής επικοινωνίας
#define WIFI_CONNECT_TIMEOUT_MS 8000

// -----

// Σταθερές UDP
// -----

// Η θύρα στην οποία ο δέκτης αναμένει τα σήματα κατεύθυνσης
#define UDP_REMOTE_PORT 4210
// Η θύρα στην οποία το ESP32 δέχεται εισερχόμενες εντολές δόνησης
#define UDP_LOCAL_PORT 4220
// Χαρακτήρας που σηματοδοτεί χρήση σειριακής επικοινωνίας αντί Wi-Fi ('s' για
serial)
#define SERIAL_MODE_CHAR 's'
```

```

// -----
// Σταθερές NVS (Μόνιμη Αποθήκευση)
// -----

// Όνομα χώρου αποθήκευσης για NVS
#define NVS_NAMESPACE    "vracer"
// Κλειδί για SSID
#define NVS_KEY_SSID      "ssid"
// Κλειδί για κωδικό
#define NVS_KEY_PASSWORD "password"
// Κλειδί για διεύθυνση IP δέκτη
#define NVS_KEY_REMOTE_IP "remote_ip"
// Κλειδί για επιλογή εισόδου: 1 = gyroscope, 0 = joystick
#define NVS_KEY_USE_GYRO "use_gyro"
// Κλειδί για επιλογή επικοινωνίας: 1 = wifi, 0 = serial
#define NVS_KEY_USE_WIFI "use_wifi"
// Χρόνος αναμονής για επιλογή επαναρύθμισης ή διατήρησης της τρέχουσας ρύθμισης
#define CONFIG_CHOICE_TIMEOUT_MS 5000

// -----
// Συνδέσεις
// -----

// ESP32 pin GPIO33 <--> VRY pin του joystick (κάθετος άξονας)
#define JOY_VRX_PIN    33
// ESP32 pin GPIO13 <--> VRX pin του joystick (οριζόντιος άξονας)
#define JOY_MOTOR_PIN 13
// ESP32 pin GPIO32 <--> pin συνδεδεμένο με τη συσκευή δόνησης
#define JOY_VRY_PIN    32
// ESP32 pin GPIO21 <--> SDA pin του MPU6050
#define GYRO_SDA_PIN    21
// ESP32 pin GPIO22 <--> SCL pin του MPU6050
#define GYRO_SCL_PIN    22

```

```

// -----
// -----

// Σταθερές Joystick
// -----

// Τιμή κάτω από την οποία θεωρείται ότι το joystick έχει κατεύθυνση προς τα αριστερά
#define JOY_LEFT_THRESHOLD      1000
// Τιμή πάνω από την οποία θεωρείται ότι το joystick έχει κατεύθυνση προς τα δεξιά
#define JOY_RIGHT_THRESHOLD     3000
// Τιμή κάτω από την οποία θεωρείται ότι το joystick έχει κατεύθυνση προς τα πίσω
#define JOY_BACKWARD_THRESHOLD  1000
// Τιμή πάνω από την οποία θεωρείται ότι το joystick έχει κατεύθυνση προς τα μπροστά
#define JOY_FORWARD_THRESHOLD   3000
// Καμία κατεύθυνση
#define JOY_DIRECTION_NO        0x00
// Κατεύθυνση προς τα αριστερά
#define JOY_DIRECTION_LEFT      0x01
// Κατεύθυνση προς τα δεξιά
#define JOY_DIRECTION_RIGHT     0x02
// Κατεύθυνση προς τα μπροστά
#define JOY_DIRECTION_FORWARD   0x04
// Κατεύθυνση προς τα πίσω
#define JOY_DIRECTION_BACKWARD  0x08

// -----
// -----

// Σταθερές Συσκευής Δόνησης
// -----

// Χρήση του καναλιού 0 για το LEDC (PWM) του ESP32
#define PWM_CHANNEL      0
// Συχνότητα PWM 5 kHz, κατάλληλη για συσκευές δόνησης
#define PWM_FREQ         5000
// Ανάλυση 8-bit (0-255) για τον έλεγχο της έντασης της δόνησης
#define PWM_RESOLUTION 8

```

```

// -----
// -----

// Σταθερές Γυροσκοπίου
// -----

// Τιμή πάνω από την οποία θεωρείται ότι το roll έχει κατεύθυνση προς τα αριστερά
ή δεξιά
#define GYRO_ROLL_THRESHOLD 12.0f

// Τιμή πάνω από την οποία θεωρείται ότι το pitch έχει κατεύθυνση προς τα μπροστά
ή πίσω
#define GYRO_PITCH_THRESHOLD 12.0f

// Τιμή κάτω από την οποία οι ενδείξεις θεωρούνται θόρυβος και αγνοούνται (για roll
και pitch)
#define GYRO_DEADZONE 4.0f

// Χρόνος αναμονής για calibration του γυροσκοπίου (μετά την ενεργοποίηση του
συστήματος)
#define GYRO_CALIB_MS 1000

// -----
// -----

// Σταθερές Κατεύθυνσης
// -----

#define LEFT -1
#define RIGHT 1
#define BACKWARD -1
#define FORWARD 1

// -----
// -----

// Αρχικές ρυθμίσεις Joystick
// -----

// Συντεταγμένη ως προς τον άξονα των X
int valueX = 0;

// Συντεταγμένη ως προς τον άξονα των Y
int valueY = 0;

// Τρέχουσα κατεύθυνση
int direction = JOY_DIRECTION_NO;

```

```

// -----
// Αρχικές ρυθμίσεις Συσκευής Δόνησης
// -----

// Μέγιστο μέγεθος ενός σήματος δόνησης (π.χ., "255,1000\n" είναι 10 χαρακτήρες)
const int VIB_CMD_BUF_SIZE = 32;
// Buffer για την αποθήκευση των χαρακτήρων ενός σήματος δόνησης καθώς λαμβάνονται
char          vibCmdBuf[VIB_CMD_BUF_SIZE];
// Τρέχουσα θέση στο vibCmdBuf για τον επόμενο εισερχόμενο χαρακτήρα
int           vibCmdPos  = 0;
// True αν η συσκευή δόνησης είναι ενεργή (δηλ., έχει λάβει σήμα δόνησης που δεν
// έχει λήξει)
bool          vibActive  = false;
// Χρόνος (σε ms) κατά τον οποίο πρέπει να απενεργοποιηθεί η συσκευή δόνησης (μόνο
// έγκυρο αν vibActive == true)
unsigned long vibEndMs   = 0;

// -----
// Αρχικές ρυθμίσεις Γυροσκοπίου
// -----

// Αντικείμενο MPU6050 για την ανάγνωση δεδομένων από το γυροσκόπιο
MPU6050 mpu(Wire);
// Δομή για την αναπαράσταση ενός διανύσματος κατεύθυνσης (x, y) που θα αποστέλλεται
// ως ζεύγος τιμών
struct DirVector { int x; int y; };

// -----
// Κατάσταση WiFi – ορίζεται κατά την εκκίνηση, δεν αλλάζει κατά τη λειτουργία.
// Αν η σύνδεση αποτύχει, όλες οι λειτουργίες υποβαθμίζονται αυτόματα σε σειριακή
// επικοινωνία.
// -----

// True αν το WiFi συνδέθηκε επιτυχώς κατά το setup()
bool    wifiConnected = false;

```

```

// Επιλογή εισόδου: true = gyroscope, false = joystick
bool    useGyro      = true;
// Επιλογή επικοινωνίας: true = wifi, false = serial
bool    useWiFi      = true;
// Αντικείμενο UDP για αποστολή και λήψη δεδομένων
WiFiUDP udp;
// Αποθήκευση του δυναμικά ρυθμισμένου SSID
char    wifiSSID[WIFI_SSID_MAX_LEN];
// Αποθήκευση του δυναμικά ρυθμισμένου κωδικού
char    wifiPassword[WIFI_PASSWORD_MAX_LEN];
// Αποθήκευση της δυναμικής διεύθυνσης IP του δέκτη (π.χ., "192.168.2.8")
char    udpRemoteIP[WIFI_IP_MAX_LEN];

// -----
// Πρωτότυπα συναρτήσεων
// -----

bool    initNVS();
bool    loadWiFiConfiguration();
void    saveWiFiConfiguration();
void    clearStoredCredentials();
bool    initWiFi();
void    configureInputAndCommunication();
void    configureCurrentOrNewSetup();
void    clearSerialBuffer();
char    readConfigOption(const char* prompt, const char* validOptions);
void    sendLine(const char* line);
int     receiveBytes(char* buf, int maxLen);
void    processVibrationCommands();
void    sendJoystickDirectionOutput();
void    sendGyroDirectionOutput();
DirVector classifyGyroDirection(float pitch, float roll);

//
*****

// Η κατεύθυνση από το γυροσκόπιο και το joystick θα αποστέλλεται ως ζεύγος

```

```

// τιμών (x, y) μέσω UDP (κύριο) ή μέσω της σειριακής γραμμής (εφεδρικά).
// x: -1 = LEFT,      0 = none, +1 = RIGHT
// y: -1 = BACKWARD, 0 = none, +1 = FORWARD
//
// Εισερχόμενες εντολές δόνησης (μορφή: "intensity,duration\n") διαβάζονται
// επίσης μέσω UDP ή σειριακής γραμμής αντίστοιχα.
//
*****

/// <summary>
/// Αρχικοποίηση υλικού και επικοινωνιών κατά την εκκίνηση.
/// </summary>
/// <remarks>
/// Διαμορφώνει τη συσκευή δόνησης μέσω LEDC, αρχικοποιεί το γυροσκόπιο MPU6050,
/// φορτώνει ή ζητά ρυθμίσεις WiFi από το NVS και εκκινεί τη σύνδεση WiFi/UDP.
/// Αν η σύνδεση WiFi αποτύχει, το σύστημα υποβαθμίζεται αυτόματα σε Serial
/// επικοινωνία.
/// </remarks>
void setup() {
    Serial.begin(115200);
    delay(2000);

    // Διαμόρφωση Συσκευής Δόνησης χρησιμοποιώντας το API LEDC του ESP32
    ledcAttach(JOY_MOTOR_PIN, PWM_FREQ, PWM_RESOLUTION);
    // Συσκευή δόνησης εκτός λειτουργίας
    ledcWrite(JOY_MOTOR_PIN, 0);

    // Διαμόρφωση Γυροσκοπίου
    Wire.begin(GYRO_SDA_PIN, GYRO_SCL_PIN);
    mpu.begin();
    delay(GYRO_CALIB_MS);
    // Calibration
    mpu.calcOffsets();
    delay(GYRO_CALIB_MS);

    // Ορισμός εξασθένησης ADC σε 11 dB (έως ~3.3V εισόδου)
    analogSetAttenuation(ADC_11db);

```

```

// Αρχικοποίηση NVS για αποθήκευση ρυθμίσεων
Serial.println();
if (!initNVS()) {
    Serial.println("[NVS] Σφάλμα αρχικοποίησης NVS!");
} else {
    Serial.println("[NVS] Αρχικοποίηση επιτυχής.");
}

// Φόρτωση ή διαμόρφωση των μόνιμων ρυθμίσεων εισόδου / επικοινωνίας
configureCurrentOrNewSetup();

// Αρχικοποίηση WiFi μόνο αν επιλέχθηκε WiFi επικοινωνία
if (useWiFi) {
    wifiConnected = initWiFi();
} else {
    wifiConnected = false;
}

// Εκκίνηση UDP μόνο αν το WiFi συνδέθηκε επιτυχώς
if (useWiFi && wifiConnected) {
    udp.begin(UDP_LOCAL_PORT);
    Serial.println();
    Serial.println("[WiFi] Επιτυχής έναρξη επικοινωνίας.");
    Serial.println("[WiFi] Local IP: " + WiFi.localIP().toString() + ", UDP Local
Port: " + String(UDP_LOCAL_PORT));
}

// Σε περίπτωση αποτυχίας σύνδεσης WiFi, υποβάθμιση σε σειριακή επικοινωνία
else if (useWiFi && !wifiConnected) {
    Serial.println("[WiFi] Αποτυχία! Ενεργοποιείται η εφεδρική σειριακή
επικοινωνία.");
}

// Αν δεν επιλέχθηκε WiFi εξ αρχής, χρήση σειριακής επικοινωνίας
else {
    Serial.println("[Serial] Χρήση σειριακής επικοινωνίας.");
}
}

```

```

/// <summary>
/// Κύριος βρόχος εκτέλεσης.
/// </summary>
/// <remarks>
/// Επεξεργάζεται εισερχόμενες εντολές δόνησης, αποστέλλει σήματα κατεύθυνσης
/// από το γυροσκόπιο και το joystick και ολοκληρώνει τη δόνηση χωρίς μπλοκάρισμα
/// όταν λήξει ο χρονομετρητής της.
/// </remarks>
void loop() {
    // Επεξεργασία εισερχόμενων εντολών δόνησης
    processVibrationCommands();
    // Αποστολή σημάτων κατεύθυνσης από την επιλεγμένη πηγή εισόδου
    if (useGyro) {
        // Αποστολή κατεύθυνσης από το γυροσκόπιο
        sendGyroDirectionOutput();
    } else {
        // Αποστολή κατεύθυνσης από το joystick
        sendJoystickDirectionOutput();
    }
    // Ολοκλήρωση δόνησης χωρίς μπλοκάρισμα
    if (vibActive && millis() >= vibEndMs) {
        ledcWrite(JOY_MOTOR_PIN, 0);

        vibActive = false;

        Serial.println("[VIB] Παύση δόνησης");
    }
}

/// <summary>
/// Διαμόρφωση ρυθμίσεων WiFi μέσω Serial.
/// </summary>
/// <remarks>
/// Λαμβάνει διαδοχικά από τη σειριακή γραμμή το SSID, τον κωδικό και τη διεύθυνση
/// IP του δέκτη. Κάθε πεδίο τερματίζεται με '\n'. Σε αποτυχία λήψης οποιουδήποτε

```

```

/// πεδίου, η συνάρτηση επιστρέφει νωρίς αφήνοντας το αντίστοιχο buffer άδειο.
/// </remarks>
void configureWiFiConfiguration() {
    int ssidIndex = 0;
    bool ssidConfigured = false;

    // Καθαρισμός του σειριακού buffer Serial
    clearSerialBuffer();

    // Πρώτο στάδιο: Λήψη SSID
    Serial.print("SSID: ");
    while (true) {
        if (Serial.available() > 0) {
            char c = Serial.read();

            if (c == '\t') {
                if (ssidIndex > 0) {
                    ssidIndex--;
                    // Διαγραφή του τελευταίου χαρακτήρα από την οθόνη
                    Serial.print("\b \b");
                }
                continue;
            }
            // Λήψη ολόκληρης της γραμμής SSID (τερματίζεται με LF)
            if (c == '\n' or c == '\r') {
                if (ssidIndex > 0) {
                    wifiSSID[ssidIndex] = '\0';
                    Serial.println();
                    ssidConfigured = true;
                    break;
                }
            } else if (ssidIndex < WIFI_SSID_MAX_LEN - 1) {
                wifiSSID[ssidIndex++] = c;
                // Echo back the character for user feedback
                Serial.print(c);
            }
        }
    }
}

```

```

    }
    delay(10);
}

if (!ssidConfigured) {
    Serial.println("\n[WiFi] Σφάλμα λήψης SSID!");
    wifiSSID[0] = '\0';
    return;
}

// Δεύτερο στάδιο: Λήψη κωδικού
int passwordIndex = 0;
bool passwordConfigured = false;

Serial.print("Password: ");
clearSerialBuffer();

while (true) {
    if (Serial.available() > 0) {
        char c = Serial.read();

        if (c == '\t') {
            if (passwordIndex > 0) {
                passwordIndex--;
                // Διαγραφή του τελευταίου χαρακτήρα από την οθόνη
                Serial.print("\b \b");
            }
            continue;
        }
        // Λήψη ολόκληρης της γραμμής κωδικού (τερματίζεται με LF)
        if (c == '\n' or c == '\r') {
            if (passwordIndex > 0) {
                wifiPassword[passwordIndex] = '\0';
                Serial.println();
                passwordConfigured = true;
                break;
            }
        }
    }
}

```

```

    }
} else if (passwordIndex < WIFI_PASSWORD_MAX_LEN - 1) {
    wifiPassword[passwordIndex++] = c;
    // Εμφάνιση του χαρακτήρα που πληκτρολογήθηκε
    Serial.print(c);
}
}
delay(10);
}

if (!passwordConfigured) {
    Serial.println("\n[WiFi] Σφάλμα λήψης κωδικού!");
    wifiPassword[0] = '\0';
    return;
}

// Τρίτο στάδιο: Λήψη διεύθυνσης IP δέκτη
int ipIndex = 0;
bool ipConfigured = false;

// Καθαρισμός του σειριακού buffer
clearSerialBuffer();
Serial.print("Remote IP: ");

while (true) {
    if (Serial.available() > 0) {
        char c = Serial.read();

        if (c == '\t') {
            if (ipIndex > 0) {
                ipIndex--;
                // Διαγραφή του τελευταίου χαρακτήρα από την οθόνη
                Serial.print("\b \b");
            }
            continue;
        }
    }
}

```

```

// Λήψη ολόκληρης της γραμμής IP (τερματίζεται με LF)
if (c == '\n' or c == '\r') {
    if (ipIndex > 0) {
        udpRemoteIP[ipIndex] = '\0';
        Serial.println();
        ipConfigured = true;
        break;
    }
} else if (ipIndex < sizeof(udpRemoteIP) - 1) {
    udpRemoteIP[ipIndex++] = c;
    // Εμφάνιση του χαρακτήρα που πληκτρολογήθηκε
    Serial.print(c);
}
}
delay(10);
}

if (!ipConfigured) {
    Serial.println("\n[WiFi] Σφάλμα λήψης διεύθυνσης IP!");
    udpRemoteIP[0] = '\0';
    return;
}

Serial.println("[WiFi] Οι ρυθμίσεις WiFi ολοκληρώθηκαν επιτυχώς.");
}

/// <summary>
/// Καθαρισμός του buffer της σειριακής επικοινωνίας
/// </summary>
/// <remarks>
/// Διαβάζει και απορρίπτει όλους τους διαθέσιμους χαρακτήρες στο Serial buffer
/// μέχρι να αδειάσει, για να διασφαλίσει ότι δεν υπάρχουν παλαιά δεδομένα που
/// θα επηρεάσουν την επόμενη ανάγνωση από τη σειριακή γραμμή.
/// </remarks>
void clearSerialBuffer() {
    while (Serial.available() > 0) {
        Serial.read();
    }
}

```

```

    }
}

/// <summary>
/// Ανάγνωση επιλογής διαμόρφωσης από τη σειριακή γραμμή με έγκυρες επιλογές.
/// </summary>
/// <remarks>
/// Εμφανίζει ένα prompt στον χρήστη και περιμένει για μια έγκυρη επιλογή χαρακτήρα
/// που περιλαμβάνεται στο validOptions. Αγνοεί τα τερματικά χαρακτήρες και
/// μετατρέπει τα κεφαλαία γράμματα σε πεζά για ευκολία χρήσης. Επιστρέφει τον
/// έγκυρο χαρακτήρα επιλογής όταν ληφθεί.
/// </remarks>
/// <param name="prompt">Το μήνυμα που εμφανίζεται στον χρήστη για να ζητήσει την
/// επιλογή</param>
/// <param name="validOptions">Μια συμβολοσειρά που περιέχει όλους τους έγκυρους
/// χαρακτήρες επιλογής (π.χ., "gj" για 'g' ή 'j')</param>
/// <returns>Ο χαρακτήρας της επιλογής που επέλεξε ο χρήστης, σε πεζά</returns>
char readConfigOption(const char* prompt, const char* validOptions) {
    // Καθαρισμός του σειριακού buffer
    clearSerialBuffer();

    Serial.print(prompt);

    while (true) {
        if (Serial.available() > 0) {
            char c = Serial.read();
            if (c == '\r' || c == '\n') {
                continue;
            }
            Serial.println(c);
            if (c >= 'A' && c <= 'Z') {
                c = c - 'A' + 'a';
            }
            if (strchr(validOptions, c) != nullptr) {
                return c;
            }
            Serial.print("Μη έγκυρη επιλογή. ");
        }
    }
}

```

```

        Serial.print(prompt);
    }
    delay(10);
}
}

/// <summary>
/// Φόρτωση αποθηκευμένων ρυθμίσεων εισόδου και επικοινωνίας από το NVS.
/// </summary>
/// <remarks>
/// Ανοίγει το NVS namespace σε λειτουργία ανάγνωσης και ανακτά τις επιλογές χρήσης
/// gyroscope και WiFi, καθώς και τις ρυθμίσεις WiFi (SSID, κωδικό, IP) αν το WiFi
/// είναι ενεργοποιημένο.
/// Επιστρέφει true αν όλες οι απαιτούμενες ρυθμίσεις ήταν αποθηκευμένες και
/// φορτώθηκαν επιτυχώς, false αλλιώς.
/// </remarks>
bool loadStoredConfiguration() {
    nvs_handle_t nvsHandle;

    // Άνοιγμα του NVS namespace σε λειτουργία ανάγνωσης
    esp_err_t err = nvs_open(NVS_NAMESPACE, NVS_READONLY, &nvsHandle);
    if (err != ESP_OK) {
        return false;
    }

    int32_t gyroValue = 1;
    int32_t wifiValue = 1;
    esp_err_t gyroErr = nvs_get_i32(nvsHandle, NVS_KEY_USE_GYRO, &gyroValue);
    esp_err_t wifiErr = nvs_get_i32(nvsHandle, NVS_KEY_USE_WIFI, &wifiValue);
    if (gyroErr != ESP_OK || wifiErr != ESP_OK) {
        nvs_close(nvsHandle);
        return false;
    }

    // Ορισμός των καθολικών μεταβλητών για χρήση γυροσκοπίου και WiFi με βάση τις
    τιμές που φορτώθηκαν
    useGyro = (gyroValue != 0);

```

```

useWiFi = (wifiValue != 0);

// Αν το WiFi είναι ενεργοποιημένο, φόρτωση των ρυθμίσεων WiFi (SSID, κωδικό,
IP)
bool loaded = true;
if (useWiFi) {
    size_t ssidLen      = WIFI_SSID_MAX_LEN;
    size_t passwordLen = WIFI_PASSWORD_MAX_LEN;
    size_t ipLen        = sizeof(udpRemoteIP);

    esp_err_t ssidErr      = nvs_get_str(nvsHandle, NVS_KEY_SSID,      wifiSSID,
&ssidLen);
    esp_err_t passwordErr  = nvs_get_str(nvsHandle, NVS_KEY_PASSWORD,
wifiPassword, &passwordLen);
    esp_err_t ipErr        = nvs_get_str(nvsHandle, NVS_KEY_REMOTE_IP, udpRemoteIP,
&ipLen);
    loaded = (ssidErr == ESP_OK && passwordErr == ESP_OK && ipErr == ESP_OK);
}

nvs_close(nvsHandle);
return loaded;
}

/// <summary>
/// Αποθήκευση ρυθμίσεων εισόδου και επικοινωνίας στο NVS.
/// </summary>
/// <remarks>
/// Ανοίγει το NVS namespace σε λειτουργία ανάγνωσης/εγγραφής και αποθηκεύει τις
επιλογές χρήσης
/// γυροσκοπίου και WiFi, καθώς και τις ρυθμίσεις WiFi (SSID, κωδικό, IP) αν το
WiFi είναι ενεργοποιημένο.
/// Καταγράφει μήνυμα επιτυχίας ή σφάλματος μέσω σειριακού buffer ανάλογα με το
αποτέλεσμα της αποθήκευσης.
/// </remarks>
void saveStoredConfiguration() {
    nvs_handle_t nvsHandle;

    // Άνοιγμα του NVS namespace σε λειτουργία ανάγνωσης/εγγραφής για αποθήκευση των
    ρυθμίσεων

```

```

esp_err_t err = nvs_open(NVS_NAMESPACE, NVS_READWRITE, &nvsHandle);
if (err != ESP_OK) {
    Serial.println("[NVS] Σφάλμα προσπέλασης μόνιμης μνήμης για αποθήκευση!");
    return;
}

// Αποθήκευση των επιλογών χρήσης γυροσκοπίου και WiFi ως ακέραιες τιμές (1 ή 0)
nvs_set_i32(nvsHandle, NVS_KEY_USE_GYRO, useGyro ? 1 : 0);
// Αποθήκευση των επιλογών χρήσης WiFi. Αν το WiFi είναι ενεργοποιημένο,
αποθήκευση των ρυθμίσεων WiFi (SSID, κωδικό, IP)
nvs_set_i32(nvsHandle, NVS_KEY_USE_WIFI, useWiFi ? 1 : 0);
if (useWiFi) {
    nvs_set_str(nvsHandle, NVS_KEY_SSID,      wifiSSID);
    nvs_set_str(nvsHandle, NVS_KEY_PASSWORD,  wifiPassword);
    nvs_set_str(nvsHandle, NVS_KEY_REMOTE_IP,  udpRemoteIP);
}

// Αποθήκευση των αλλαγών στο NVS και κλείσιμο του handle
err = nvs_commit(nvsHandle);
nvs_close(nvsHandle);

// Καταγραφή μηνύματος επιτυχίας ή σφάλματος ανάλογα με το αποτέλεσμα της
αποθήκευσης
if (err == ESP_OK) {
    Serial.println("[NVS] Όλες οι ρυθμίσεις αποθηκεύτηκαν μόνιμα.");
} else {
    Serial.println("[NVS] Σφάλμα αποθήκευσης ρυθμίσεων!");
}
}

/// <summary>
/// Διαμόρφωση επιλογών εισόδου και επικοινωνίας μέσω σειριακής γραμμής.
/// </summary>
/// <remarks>
/// Ζητά από τον χρήστη να επιλέξει την πηγή εισόδου (γυροσκόπιο ή joystick) και
την μέθοδο επικοινωνίας (WiFi ή Serial).
/// Αν επιλεγεί WiFi, καλεί τη συνάρτηση configureWiFiConfiguration() για να λάβει
τις ρυθμίσεις WiFi.

```

```

/// Καταγράφει τις επιλογές που έγιναν μέσω του σειριακού buffer για επιβεβαίωση
από τον χρήστη.
/// Οι επιλογές αποθηκεύονται στις καθολικές μεταβλητές useGyro και useWiFi για
χρήση κατά τη λειτουργία του συστήματος.
/// </remarks>
void configureInputAndCommunication() {
    // Ορισμός της επιλογής χρήσης γυροσκοπίου ή joystick με βάση την επιλογή του
    χρήστη και καταγραφή της επιλογής μέσω του σειριακού buffer
    char option = readConfigOption("Επιλέξτε πηγή εισόδου [g=Gyro, j=Joystick]: ",
    "gj");

    useGyro = (option == 'g');
    Serial.println(useGyro ? "[CONFIG] Επιλέχθηκε Gyroscope." : "[CONFIG] Επιλέχθηκε
    Joystick.");

    // Ορισμός της επιλογής χρήσης WiFi ή σειριακής επικοινωνίας με βάση την επιλογή
    του χρήστη και καταγραφή της επιλογής μέσω του σειριακού buffer
    option = readConfigOption("Επιλέξτε επικοινωνία [w=WiFi, s=Serial]: ", "ws");

    useWiFi = (option == 'w');
    Serial.println(useWiFi ? "[CONFIG] Επιλέχθηκε WiFi." : "[CONFIG] Επιλέχθηκε
    Serial.");

    // Αν επιλέχθηκε WiFi, λήψη των ρυθμίσεων WiFi (SSID, κωδικό, IP) μέσω της
    σειριακής γραμμής
    if (useWiFi) {
        configureWiFiConfiguration();
    }
}

/// <summary>
/// Διαμόρφωση τρέχουσας ή νέας ρύθμισης εισόδου και επικοινωνίας.
/// </summary>
/// <remarks>
/// Ελέγχει αν υπάρχουν αποθηκευμένες ρυθμίσεις στο NVS.
/// Αν υπάρχουν, εμφανίζει τις τρέχουσες ρυθμίσεις και δίνει στον χρήστη την επιλογή
    να τις διατηρήσει ή να ξεκινήσει νέα διαμόρφωση.
/// Αν ο χρήστης επιλέξει να διατηρήσει τις ρυθμίσεις, η συνάρτηση επιστρέφει χωρίς
    να αλλάξει τίποτα.

```

```

/// Αν ο χρήστης επιλέξει νέα διαμόρφωση ή αν δεν υπάρχουν αποθηκευμένες ρυθμίσεις,
/// καλεί τη συνάρτηση configureInputAndCommunication() για να λάβει νέες ρυθμίσεις
/// και
/// στη συνέχεια αποθηκεύει αυτές τις ρυθμίσεις στο NVS για μελλοντική χρήση.
/// </remarks>
void configureCurrentOrNewSetup() {
    // Έλεγχος για αποθηκευμένες ρυθμίσεις στο NVS και φόρτωση τους αν υπάρχουν, με
    εμφάνιση των τρεχουσών ρυθμίσεων και επιλογή διατήρησης ή νέας διαμόρφωσης από τον
    χρήστη μέσω της σειριακής

    bool configLoaded = loadStoredConfiguration();

    // Αν υπάρχουν αποθηκευμένες ρυθμίσεις, εμφάνιση των τρεχουσών ρυθμίσεων και
    επιλογή διατήρησης ή νέας διαμόρφωσης από τον χρήστη μέσω της σειριακής γραμμής
    if (configLoaded) {
        Serial.println("[CONFIG] Βρέθηκε αποθηκευμένη ρύθμιση.");
        Serial.print("[CONFIG] Τρέχουσα είσοδος: ");
        Serial.println(useGyro ? "Gyroscope" : "Joystick");
        Serial.print("[CONFIG] Τρέχουσα επικοινωνία: ");
        Serial.println(useWiFi ? "WiFi" : "Serial");
        if (useWiFi) {
            Serial.print("[CONFIG] SSID: ");
            Serial.println(wifiSSID);
            Serial.print("[CONFIG] Remote IP: ");
            Serial.println(udpRemoteIP);
        }
        Serial.println("[CONFIG] Πατήστε 'n' μέσα στα επόμενα 5 δευτερόλεπτα για νέα
        διαμόρφωση, ή αφήστε για να χρησιμοποιήσετε την τρέχουσα.");

        unsigned long startMs = millis();
        bool newConfig = false;
        while (millis() - startMs < CONFIG_CHOICE_TIMEOUT_MS) {
            if (Serial.available() > 0) {
                char c = Serial.read();
                if (c == 'n' || c == 'N') {
                    newConfig = true;
                    break;
                }
            }
        }
    }
}

```

```

    delay(10);
}

if (!newConfig) {
    Serial.println("[CONFIG] Χρήση υπάρχουσας ρύθμισης.");
    return;
}

Serial.println("\n[CONFIG] Νέα διαμόρφωση ξεκίνησε...");
}

// Αν δεν υπάρχουν αποθηκευμένες ρυθμίσεις, ενημέρωση του χρήστη ότι απαιτείται
διαμόρφωση μέσω της σειριακής γραμμής
else {
    Serial.println("[CONFIG] Δεν βρέθηκε αποθηκευμένη ρύθμιση. Απαιτείται
διαμόρφωση.");
}

// Εκκίνηση νέας διαμόρφωσης μέσω της σειριακής γραμμής και αποθήκευση των νέων
ρυθμίσεων στο NVS
configureInputAndCommunication();

// Αποθήκευση των νέων ρυθμίσεων στο NVS για μελλοντική χρήση
saveStoredConfiguration();
}

/// <summary>
/// Αρχικοποίηση Μόνιμης Αποθήκευσης (NVS).
/// </summary>
/// <remarks>
/// Αρχικοποιεί το NVS partition για αποθήκευση ρυθμίσεων WiFi. Αν το partition
/// δεν έχει ελεύθερο χώρο ή έχει νέα έκδοση, το διαγράφει και επανεκκινεί.
/// </remarks>
/// <returns>true αν η αρχικοποίηση ήταν επιτυχής, false αλλιώς</returns>
bool initNVS() {
    esp_err_t err = nvs_flash_init();
    if (err == ESP_ERR_NVS_NO_FREE_PAGES || err == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        // Καθαρισμός του NVS partition και επανεκκίνηση

```

```

    err = nvs_flash_erase();
    if (err != ESP_OK) {
        return false;
    }
    err = nvs_flash_init();
}
return (err == ESP_OK);
}

/// <summary>
/// Φόρτωση αποθηκευμένων ρυθμίσεων WiFi από το NVS.
/// </summary>
/// <remarks>
/// Ανοίγει το NVS namespace σε λειτουργία ανάγνωσης και ανακτά το SSID,
/// τον κωδικό και τη διεύθυνση IP του δέκτη στα αντίστοιχα global buffers.
/// </remarks>
/// <returns>true αν όλα τα πεδία ήταν αποθηκευμένα και φορτώθηκαν επιτυχώς, false
αλλιώς</returns>
bool loadWiFiConfiguration() {
    nvs_handle_t nvsHandle;
    esp_err_t err = nvs_open(NVS_NAMESPACE, NVS_READONLY, &nvsHandle);

    if (err != ESP_OK) {
        // Namespace doesn't exist yet
        return false;
    }

    size_t ssidLen      = WIFI_SSID_MAX_LEN;
    size_t passwordLen = WIFI_PASSWORD_MAX_LEN;
    size_t ipLen         = sizeof(udpRemoteIP);

    esp_err_t ssidErr      = nvs_get_str(nvsHandle, NVS_KEY_SSID,      wifiSSID,
&ssidLen);
    esp_err_t passwordErr = nvs_get_str(nvsHandle, NVS_KEY_PASSWORD,  wifiPassword,
&passwordLen);
    esp_err_t ipErr        = nvs_get_str(nvsHandle, NVS_KEY_REMOTE_IP, udpRemoteIP,
&ipLen);

```

```

nvs_close(nvsHandle);

return (ssidErr == ESP_OK && passwordErr == ESP_OK && ipErr == ESP_OK);
}

/// <summary>
/// Αποθήκευση ρυθμίσεων WiFi στο NVS.
/// </summary>
/// <remarks>
/// Ανοίγει το NVS namespace σε λειτουργία ανάγνωσης/εγγραφής και αποθηκεύει
/// το SSID, τον κωδικό και τη διεύθυνση IP που είναι ήδη φορτωμένα στα global
/// buffers.
/// Καταγράφει μήνυμα επιτυχίας ή σφάλματος μέσω Serial.
/// </remarks>
void saveWiFiConfiguration() {
    nvs_handle_t nvsHandle;
    esp_err_t err = nvs_open(NVS_NAMESPACE, NVS_READWRITE, &nvsHandle);

    if (err != ESP_OK) {
        Serial.println("[NVS] Σφάλμα προσπέλασης μόνιμης μνήμης για αποθήκευση!");
        return;
    }

    nvs_set_str(nvsHandle, NVS_KEY_SSID,      wifiSSID);
    nvs_set_str(nvsHandle, NVS_KEY_PASSWORD,   wifiPassword);
    nvs_set_str(nvsHandle, NVS_KEY_REMOTE_IP,  udpRemoteIP);
    err = nvs_commit(nvsHandle);
    nvs_close(nvsHandle);

    if (err == ESP_OK) {
        Serial.println("[NVS] Ρυθμίσεις WiFi αποθηκεύθηκαν.");
    } else {
        Serial.println("[NVS] Σφάλμα αποθήκευσης ρυθμίσεων!");
    }
}

/// <summary>

```

```

/// Διαγραφή αποθηκευμένων ρυθμίσεων WiFi από το NVS.
/// </summary>
/// <remarks>
/// Ανοίγει το NVS namespace σε λειτουργία ανάγνωσης/εγγραφής και διαγράφει
/// τα κλειδιά SSID, κωδικού και διεύθυνσης IP. Καταγράφει μήνυμα επιτυχίας
/// ή σφάλματος μέσω Serial.
/// </remarks>
void clearStoredCredentials() {
    nvs_handle_t nvsHandle;
    esp_err_t err = nvs_open(NVS_NAMESPACE, NVS_READWRITE, &nvsHandle);

    if (err != ESP_OK) {
        Serial.println("[NVS] Σφάλμα προσπέλασης μόνιμης μνήμης για διαγραφή!");
        return;
    }

    nvs_erase_key(nvsHandle, NVS_KEY_SSID);
    nvs_erase_key(nvsHandle, NVS_KEY_PASSWORD);
    nvs_erase_key(nvsHandle, NVS_KEY_REMOTE_IP);
    err = nvs_commit(nvsHandle);
    nvs_close(nvsHandle);

    if (err == ESP_OK) {
        Serial.println("[NVS] Ρυθμίσεις WiFi διαγράφηκαν.");
    } else {
        Serial.println("[NVS] Σφάλμα διαγραφής!");
    }
}

/// <summary>
/// Αρχικοποίηση σύνδεσης WiFi σε λειτουργία Station.
/// </summary>
/// <remarks>
/// Θέτει το WiFi σε "station mode" και επιχειρεί σύνδεση με τα αποθηκευμένα
/// διαπιστευτήρια. Αναμένει έως WIFI_CONNECT_TIMEOUT_MS milliseconds πριν
/// επιστρέψει αποτυχία. Σε αποτυχία, το σύστημα υποβαθμίζεται αυτόματα σε Serial.

```

```

/// </remarks>
/// <returns>true αν η σύνδεση ολοκληρώθηκε εντός του χρονικού ορίου, false
αλλιώς</returns>
bool initWiFi() {
    Serial.print("[WiFi] Σύνδεση στο δίκτυο: ");
    Serial.print(wifiSSID);
    // Station mode – σύνδεση σε υπάρχον access point
    WiFi.mode(WIFI_STA);
    WiFi.begin(wifiSSID, wifiPassword);
    unsigned long startMs = millis();
    while (WiFi.status() != WL_CONNECTED) {
        // Αποτυχία εντός του χρονικού ορίου
        if (millis() - startMs > WIFI_CONNECT_TIMEOUT_MS) {
            return false;
        }
        delay(250);
        Serial.print(".");
    }
    Serial.println();
    return true;
}

/// <summary>
/// Αποστολή γραμμής κειμένου μέσω UDP ή Serial.
/// </summary>
/// <remarks>
/// Αποτελεί το κεντρικό σημείο αποστολής δεδομένων. Αν το WiFi είναι συνδεδεμένο
/// και έχει οριστεί διεύθυνση IP δέκτη, αποστέλλει UDP datagram. Διαφορετικά,
/// υποβαθμίζεται αυτόματα σε αποστολή μέσω σειριακής γραμμής.
/// </remarks>
/// <param name="line">Συμβολοσειρά προς αποστολή (τερματισμένη με '\0')</param>
void sendLine(const char* line) {
    if (wifiConnected && udpRemoteIP[0] != '\0') {
        // Αποστολή UDP datagram προς τη δυναμικά ρυθμισμένη διεύθυνση IP
        udp.beginPacket(udpRemoteIP, UDP_REMOTE_PORT);
        udp.write((const uint8_t*)line, strlen(line));
        udp.endPacket();
    }
}

```

```

    } else {
        // Εφεδρική επικοινωνία: αποστολή μέσω σειριακής γραμμής
        Serial.println(line);
    }
}

/// <summary>
/// Λήψη bytes από UDP ή Serial.
/// </summary>
/// <remarks>
/// Αν το WiFi είναι συνδεδεμένο, ελέγχει για εισερχόμενο UDP packet και διαβάζει
/// έως maxLen bytes. Αλλιώς, διαβάζει byte-byte από τη σειριακή γραμμή.
/// Το buffer δεν τερματίζεται με '\0' από αυτή τη συνάρτηση.
/// </remarks>
/// <param name="buf">Buffer αποθήκευσης των ληφθέντων bytes</param>
/// <param name="maxLen">Μέγιστος αριθμός bytes προς ανάγνωση</param>
/// <returns>Αριθμός bytes που διαβάστηκαν, ή 0 αν δεν υπάρχουν διαθέσιμα
δεδομένα</returns>
int receiveBytes(char* buf, int maxLen) {
    if (wifiConnected) {
        // Έλεγχος για εισερχόμενο UDP packet
        int packetSize = udp.parsePacket();
        if (packetSize > 0) {
            // Ανάγνωση μέχρι το μέγιστο αριθμό bytes, από το packet
            int n = udp.read(buf, maxLen);
            return (n > 0) ? n : 0;
        }
        // Δεν υπάρχει διαθέσιμο packet
        return 0;
    }
    // Εφεδρική επικοινωνία: λήψη από σειριακή γραμμή, byte-byte
    int count = 0;
    while (Serial.available() > 0 && count < maxLen) {
        buf[count++] = (char)Serial.read();
    }
    return count;
}

```

```

/// <summary>
/// Επεξεργασία εισερχόμενων εντολών δόνησης.
/// </summary>
/// <remarks>
/// Διαβάζει bytes μέσω UDP ή Serial και τα συσσωρεύει σε στατικό buffer έως ότου
/// ληφθεί '\n'. Κατόπιν αναλύει τη μορφή "intensity,duration", περιορίζει τις
/// τιμές
/// στα έγκυρα εύρη και ενεργοποιεί τη συσκευή δόνησης για την καθορισμένη διάρκεια.
/// Το '\r' αγνοείται για συμβατότητα με συστήματα που στέλνουν '\r\n'.
/// </remarks>
void processVibrationCommands() {
    // Ενδιάμεσος buffer λήψης – μέγιστο μέγεθος ενός UDP packet ή Serial burst
    char rxBuf[VIB_CMD_BUF_SIZE];
    int rxLen = receiveBytes(rxBuf, sizeof(rxBuf));

    // Επεξεργασία κάθε byte που ελήφθη
    for (int i = 0; i < rxLen; i++) {
        char c = rxBuf[i];

        // Αγνόηση του '\r' (CR) για συμβατότητα με συστήματα που στέλνουν '\r\n'.
        if (c == '\r')
            continue;

        // Το διάβασμα του χαρακτήρα τέλους γραμμής υποδηλώνει την ολοκλήρωση
        // ενός σήματος δόνησης.
        if (c == '\n') {
            // Τοποθέτηση χαρακτήρα τερματισμού του buffer
            vibCmdBuf[vibCmdPos] = '\0';
            vibCmdPos = 0;

            // Αναζήτηση του διαχωριστικού ',' και ανάλυση των αριθμών με δείκτες.
            char* line = vibCmdBuf;
            char* sep = strchr(line, ',');
            if (sep) {
                // Διάσπαση της γραμμής σε δύο μέρη: intensity και duration.
                *sep = '\0';
            }
        }
    }
}

```

```

char* intensityStr = line;
char* durationStr = sep + 1;

// Έλεγχος της εγκυρότητας των στοιχείων του σήματος δόνησης.
char* endPtr = nullptr;
long intensity = strtol(intensityStr, &endPtr, 10);
bool validIntensity = (endPtr != intensityStr && *endPtr == '\0');
endPtr = nullptr;
long duration = strtol(durationStr, &endPtr, 10);
bool validDuration = (endPtr != durationStr && *endPtr == '\0');

if (validIntensity && validDuration) {
    intensity = constrain(intensity, 0, 255);
    if (duration < 0)
        duration = 0;

    // Ενεργοποίηση της συσκευής δόνησης με την καθορισμένη ένταση και
    διάρκεια.

    if (duration > 0) {
        Serial.print("[VIB] Εντολή δόνησης: ένταση=");
        Serial.print(intensity);
        Serial.print(" διάρκεια=");
        Serial.println(duration);

        ledcWrite(JOY_MOTOR_PIN, (int)intensity);

        vibEndMs = millis() + (unsigned long)duration;

        vibActive = true;
    } else {
        Serial.println("[VIB] Παύση δόνησης");

        ledcWrite(JOY_MOTOR_PIN, 0);

        vibActive = false;
    }
}

```

```

    }
}
continue;
}

// Προσθήκη στον buffer (με προστασία overflow).
if (vibCmdPos < VIB_CMD_BUF_SIZE - 1) {
    vibCmdBuf[vibCmdPos++] = c;
} else {
    // Σφάλμα: γραμμή πολύ μεγάλη, επαναφορά buffer.
    vibCmdPos = 0;
}
}
}

/// <summary>
/// Ανάγνωση joystick και αποστολή σήματος κατεύθυνσης.
/// </summary>
/// <remarks>
/// Διαβάζει τις αναλογικές τιμές των αξόνων X και Y, τις συγκρίνει με τα
/// καθορισμένα thresholds και αποστέλλει ζεύγος (x, y) μέσω UDP ή Serial
/// μόνο όταν ανιχνευτεί έγκυρη κατεύθυνση (μη μηδενική).
/// </remarks>
void sendJoystickDirectionOutput() {
    // Διάβασμα τιμών X και Y από το joystick
    valueX = analogRead(JOY_VRX_PIN);
    valueY = analogRead(JOY_VRY_PIN);

    // Προσδιορισμός κατεύθυνσης με βάση τις εισαγόμενες τιμές ως προς X και Y και,
    // ταυτόχρονα, έλεγχος εγκυρότητας των εισαγόμενων τιμών.
    // Κάθε φορά, η κατεύθυνση μπορεί να είναι: Καμία, Απλή (1) ή Σύνθετη (2)
    direction = JOY_DIRECTION_NO;
    int x = 0;
    int y = 0;

    // Έλεγχος για αριστερά/δεξιά ως προς τα σχετικά Threshold

```

```

if (valueX < JOY_LEFT_THRESHOLD) {
    direction = direction | JOY_DIRECTION_LEFT;
    x = LEFT;
} else if (valueX > JOY_RIGHT_THRESHOLD) {
    direction = direction | JOY_DIRECTION_RIGHT;
    x = RIGHT;
}

// Έλεγχος για πάνω/κάτω ως προς τα σχετικά Threshold
if (valueY < JOY_BACKWARD_THRESHOLD) {
    direction = direction | JOY_DIRECTION_BACKWARD;
    y = BACKWARD;
} else if (valueY > JOY_FORWARD_THRESHOLD) {
    direction = direction | JOY_DIRECTION_FORWARD;
    y = FORWARD;
}

// Αποστολή των τιμών X και Y (ως ζεύγος) μέσω UDP ή Serial,
// εφόσον έχουν δοθεί έγκυρες τιμές από το joystick.
if (direction) {
    // Δημιουργία γραμμής "x,y" σε τοπικό buffer πριν την αποστολή
    char outBuf[16];
    snprintf(outBuf, sizeof(outBuf), "%d,%d", x, y);
    sendLine(outBuf);
}
}

/// <summary>
/// Ανάγνωση γυροσκοπίου και αποστολή σήματος κατεύθυνσης.
/// </summary>
/// <remarks>
/// Ενημερώνει τις μετρήσεις του MPU6050, μετατρέπει τις τιμές roll και pitch
/// σε διανυσματική κατεύθυνση (x, y) μέσω classifyGyroDirection() και αποστέλλει
/// το ζεύγος τιμών μέσω UDP ή Serial.
/// </remarks>
void sendGyroDirectionOutput() {

```

```

// Ενημέρωση των εσωτερικών μετρήσεων του γυροσκοπίου
mpu.update();

// Roll → οριζόντιος άξονας (-1 = LEFT, 0 = none, +1 = RIGHT)
// Pitch → κάθετος άξονας (-1 = BACKWARD, 0 = none, +1 = FORWARD)

// Το roll καθορίζει την οριζόντια κατεύθυνση (LEFT/RIGHT)
float roll = mpu.getAngleX();
// Το pitch καθορίζει την κάθετη κατεύθυνση (BACKWARD/FORWARD)
float pitch = mpu.getAngleY();
// Μετατροπή των τιμών pitch και roll σε κατευθύνσεις (x, y)
DirVector dir = classifyGyroDirection(pitch, roll);

// Αποστολή των τιμών X και Y (ως ζεύγος) μέσω UDP ή Serial
char outBuf[16];
snprintf(outBuf, sizeof(outBuf), "%d,%d", dir.x, dir.y);
sendLine(outBuf);
}

/// <summary>
/// Μετατροπή τιμών pitch και roll σε διανυσματική κατεύθυνση.
/// </summary>
/// <remarks>
/// Εφαρμόζει deadzone φίλτρο και στους δύο άξονες. Συγκρίνει τις φιλτραρισμένες
/// τιμές με τα thresholds για να παράγει ακέραιες τιμές κατεύθυνσης στο {-1, 0,
/// +1}
/// για κάθε άξονα ανεξάρτητα.
/// </remarks>
/// <param name="pitch">Γωνία pitch από το MPU6050 (κάθετος άξονας)</param>
/// <param name="roll">Γωνία roll από το MPU6050 (οριζόντιος άξονας)</param>
/// <returns>DirVector με  $x \in \{-1, 0, +1\}$  και  $y \in \{-1, 0, +1\}$ </returns>
DirVector classifyGyroDirection(float pitch, float roll) {
    // Έλεγχος ως προς DEADZONE για το pitch
    float p = (fabsf(pitch) > GYRO_DEADZONE) ? pitch : 0.0f;
    // Έλεγχος ως προς DEADZONE για το roll
    float r = (fabsf(roll) > GYRO_DEADZONE) ? roll : 0.0f;

```

```

// Καθορισμός οριζόντιας κατεύθυνσης με βάση το roll:
// r > ROLL_THRESHOLD → LEFT
// r < -ROLL_THRESHOLD → RIGHT
// διαφορετικά, x = 0 (καμία οριζόντια κατεύθυνση)
int x = (r > GYRO_ROLL_THRESHOLD) ? LEFT :
        (r < -GYRO_ROLL_THRESHOLD) ? RIGHT : 0;

// Καθορισμός κάθετης κατεύθυνσης με βάση το pitch:
// p > PITCH_THRESHOLD → BACKWARD
// p < -PITCH_THRESHOLD → FORWARD
// διαφορετικά, y = 0 (καμία κάθετη κατεύθυνση)
int y = (p > GYRO_PITCH_THRESHOLD) ? BACKWARD :
        (p < -GYRO_PITCH_THRESHOLD) ? FORWARD : 0;

return { x, y };

```

ΠΑΡΑΡΤΗΜΑ Δ: Κώδικας του Python script VRacer_Arduino_Simulator_Wifi.py

```
using UnityEngine;

public class CarXROriginFollow : MonoBehaviour
{
    public Transform car;          // Το αυτοκίνητο που ακολουθεί η κάμερα
    public Vector3 localOffset;    // Τοπική μετατόπιση της κάμερας σε σχέση με
    το αυτοκίνητο (π.χ. πίσω και πάνω από το αυτοκίνητο)
    public float followSpeed = 20f; // Ταχύτητα ακολουθίας της κάμερας

    /// <summary>
    /// Εκτελείται στο τέλος του frame. Ενημερώνει τη θέση και περιστροφή της
    κάμερας
    /// ώστε να ακολουθεί ομαλά το όχημα με βάση την τοπική μετατόπιση και την
    κατεύθυνση του αυτοκινήτου.
    /// </summary>
    void LateUpdate()
    {
        if (!car) return;

        // Υπολογισμός της θέσης της κάμερας με βάση τη θέση του αυτοκινήτου και
        την τοπική μετατόπιση
        Vector3 targetPos = car.TransformPoint(localOffset);

        // Συσχέτιση θέσης κάμερας με τη θέση του αυτοκινήτου
        transform.position = Vector3.Lerp(transform.position, targetPos,
        followSpeed * Time.deltaTime);

        // Συσχέτιση περιστροφής κάμερας με την κατεύθυνση του αυτοκινήτου
        Vector3 forward = Vector3.ProjectOnPlane(car.forward, Vector3.up); //
        Περιστροφή μόνο στον άξονα Y (για να μη γέρνει το κεφάλι του παίκτη)
        if (forward.sqrMagnitude > 0.001f) // Αποφυγή περιστροφής όταν το
        αυτοκίνητο είναι σχεδόν ακίνητο
        {
            Quaternion targetRot = Quaternion.LookRotation(forward, Vector3.up);
```

```
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRot,
followSpeed * Time.deltaTime);
    }
}
}
```

ΠΑΡΑΡΤΗΜΑ Ε: Κώδικας του Python script VRacer_Arduino_Simulator_Serial_Bridge.py

"""

Περιγραφή λειτουργίας:

1) Στέλνει σήματα κατεύθυνσης, ως ASCII συμβολοσειρές της μορφής "x,y\r\n", στη Unity εφαρμογή (VRacer_Unity)

μέσω της COM8 και με βάση τα πλήκτρα-βέλη του πληκτρολογίου.

Συνδυασμοί βελών → Σειριακή COM8 προς Unity:

LEFT	→	"-1,0\r\n"	
RIGHT	→	"1,0\r\n"	
FORWARD	→	"0,1\r\n"	
BACKWARD	→	"0,-1\r\n"	
LEFT + FORWARD	→	"-1,1\r\n"	
LEFT + BACKWARD	→	"-1,-1\r\n"	
RIGHT + FORWARD	→	"1,1\r\n"	
RIGHT + BACKWARD	→	"1,-1\r\n"	
(κανένα βέλος)	→	"0,0\r\n"	← σήμα διακοπής κίνησης

2) Διαβάζει σήματα δόνησης από τη Unity εφαρμογή μέσω της COM8, τα οποία είναι

σε μορφή "<intensity>,<duration_ms>\r\n" (π.χ. "128,500\r\n" για ένταση 128

και διάρκεια 500ms), και εμφανίζει την ένταση και την διάρκεια στην οθόνη.

"""

Εισαγωγή της βιβλιοθήκης pynput για χειρισμό πληκτρολογίου

from pynput import keyboard

Εισαγωγή της βιβλιοθήκης serial για σειριακή επικοινωνία

from serial import Serial

Εισαγωγή της βιβλιοθήκης threading για multithreading

import threading

Εισαγωγή της βιβλιοθήκης time για κατακράτηση χρόνου

import time

```
# Η σειριακή θύρα (COM) που έχει "γεφυρωποιηθεί" με τη σειριακή θύρα από την
οποία δέχεται σήματα κατεύθυνσης το VRacer_Unity (π.χ. COM9)
```

```
PORT      = "COM8"
```

```
# Ταχύτητα μετάδοσης (baud rate)
```

```
BAUDRATE = 115200
```

```
# Αποθήκευση των τρεχόντων πατημένων πλήκτρων
```

```
pressed_keys = set()
```

```
# Άνοιγμα της σειριακής θύρας
```

```
ser = Serial(PORT, BAUDRATE, bytesize=8, parity='N', stopbits=1, timeout=1)
```

```
#
=====
=====
```

```
# Διαχείριση σημάτων δόνησης
```

```
#
=====
=====
```

```
# Event τερματισμού του thread του διαβάσματος δονήσεων, όταν ο listener
τερματίσει
```

```
_stop_reader = threading.Event()
```

```
"""
```

Ανάλυση γραμμής που περιέχει ένταση και διάρκεια διαχωρισμένα με κόμμα.

Παράμετροι:

line (str) – Η γραμμή που διαβάστηκε από τη σειριακή θύρα.

Περιγραφή:

Επιστρέφει ένα ζεύγος (intensity, duration_ms) αν η γραμμή είναι της μορφής
"<intensity>,<duration_ms>" και οι τιμές είναι εντός επιτρεπτού εύρους.

```
"""
```

```
def _parse_vibration_line(line):
```

```
    # Προσπάθεια ανάλυσης της γραμμής
```

```
    try:
```

```

# Διαχωρισμός της γραμμής με βάση το κόμμα
parts = line.strip().split(',')
# Έλεγχος αν υπάρχουν ακριβώς 2 μέρη (ένταση και διάρκεια)
if len(parts) != 2:
    # Επιστροφή None αν δεν υπάρχουν 2 μέρη
    return None

# Μετατροπή του πρώτου μέρους σε ακέραιο (ένταση)
intensity = int(parts[0])
# Μετατροπή του δεύτερου μέρους σε ακέραιο (διάρκεια)
duration = int(parts[1])
# Έλεγχος αν οι τιμές είναι εντός επιτρεπτού εύρους
if not (0 <= intensity <= 255) or duration < 0:
    # Επιστροφή None αν οι τιμές είναι εκτός εύρους
    return None

# Επιστροφή του ζεύγους (ένταση, διάρκεια)
return intensity, duration

# Σφάλμα κατά την ανάλυση της γραμμής
except Exception:
    # Επιστροφή None σε περίπτωση σφάλματος
    return None

"""
Χειρισμός του σήματος δόνησης που προέρχεται από τη σειριακή θύρα.

Παράμετροι:
    intensity (int)    - Ένταση δόνησης, 0..255.
    duration_ms (int)  - Διάρκεια σε milliseconds.
"""

def _handle_vibration_signal(intensity, duration_ms):
    # Εμφάνιση - στον οθόνη - της έντασης και της διάρκειας του σήματος δόνησης
    print(f"VIBRATION <- intensity={intensity}, duration_ms={duration_ms}")

"""
Επανάληψη που διαβάζει γραμμές από τη σειριακή θύρα και επεξεργάζεται
τα σήματα δόνησης που προέρχονται από το VRacer_Unity.

```

Περιγραφή:

Εκτελείται σε ξεχωριστό thread. Διαβάζει γραμμές σε μορφή "<intensity>,<duration_ms>" (κάθε γραμμή είναι και ένα σήμα δόνησης).

Τερματίζει όταν το event `\_stop\_reader` τεθεί.

"""

def _serial_vibration_reader():

Βρόχος που τρέχει όσο δεν έχει τεθεί το event _stop_reader

while not _stop_reader.is_set():

Προσπάθεια ανάγνωσης δεδομένων από τη σειριακή θύρα

try:

Διάβασμα γραμμής από τη σειριακή θύρα

line_bytes = ser.readline().strip()

Έλεγχος αν η γραμμή είναι κενή

if not line_bytes:

Συνέχεια στην επόμενη επανάληψη

continue

χαρακτήρων # Μετατροπή των bytes σε string (ASCII) και αφαίρεση περιττών

line = line_bytes.decode("ascii").strip()

Ανάλυση της γραμμής σε ένταση και διάρκεια (του σήματος δόνησης)

vibration_signal = _parse_vibration_line(line)

Έλεγχος αν η ανάλυση ήταν επιτυχής

if vibration_signal is not None:

Ανάλυση του σήματος στα συστατικά του (ένταση, διάρκεια)

intensity, duration_ms = vibration_signal

Χειρισμός του σήματος δόνησης (εμφάνιση στην οθόνη)

_handle_vibration_signal(intensity, duration_ms)

else:

Αγνόηση μη συμβατών γραμμών

pass

```

        # Σφάλμα κατά την ανάγνωση ή επεξεργασία της γραμμής
    except Exception:
        # Συνέχεια στην επόμενη επανάληψη
        continue

#
=====
=====

# Διαχείριση αποστολής σημάτων κατεύθυνσης
#
=====
=====

"""Κωδικοποίηση και αποστολή σήματος κατεύθυνσης μέσω UDP."""
def _send_direction(x, y):
    # Δημιουργία του μηνύματος σε μορφή "x,y\r\n"
    msg = f"{x},{y}\r\n"
    # Αποστολή του μηνύματος στη σειριακή θύρα ως ASCII bytes
    ser.write(msg.encode("ascii"))

    # Εμφάνιση του μηνύματος που στάλθηκε στη σειριακή θύρα (για debugging)
    print(f"DIRECTION -> : {msg.strip()}")

# Τα πλήκτρα-βέλη που ενδιαφέρουν
# Σύνολο που περιέχει τα πλήκτρα πλοήγησης (Left, Right, Up, Down)
_NAV_KEYS = {keyboard.Key.left, keyboard.Key.right, keyboard.Key.up,
keyboard.Key.down}

"""
Χειρισμός γεγονότος πατήματος πλήκτρου.

```

Παράμετροι:

key (pygame.keyboard.Key) – Το πλήκτρο που πατήθηκε.

Περιγραφή:

Προσθέτει το πατημένο πλήκτρο στο σύνολο `pressed\_keys`, υπολογίζει

```

τον κατάλληλο συνδυασμό (x, y) για τα βελάκια και στέλνει το μήνυμα
στη σειριακή θύρα στη μορφή "x,y\r\n".
"""
def on_press(key):
    # Προσθήκη του πατημένου πλήκτρου στο σύνολο pressed_keys
    pressed_keys.add(key)

    # Έλεγχος αν τα βέλη Right και Up είναι πατημένα ταυτόχρονα
    if keyboard.Key.right in pressed_keys and keyboard.Key.up in pressed_keys:
        # RIGHT + FORWARD
        x, y = 1, 1

    # Έλεγχος αν τα βέλη Right και Down είναι πατημένα ταυτόχρονα
    elif keyboard.Key.right in pressed_keys and keyboard.Key.down in
pressed_keys:
        # RIGHT + BACKWARD
        x, y = 1, -1

    # Έλεγχος αν τα βέλη Left και Up είναι πατημένα ταυτόχρονα
    elif keyboard.Key.left in pressed_keys and keyboard.Key.up in pressed_keys:
        # LEFT + FORWARD
        x, y = -1, 1

    # Έλεγχος αν τα βέλη Left και Down είναι πατημένα ταυτόχρονα
    elif keyboard.Key.left in pressed_keys and keyboard.Key.down in pressed_keys:
        # LEFT + BACKWARD
        x, y = -1, -1

    # Έλεγχος αν το βέλος Right είναι πατημένο
    elif keyboard.Key.right in pressed_keys:
        # RIGHT
        x, y = 1, 0

    # Έλεγχος αν το βέλος Left είναι πατημένο
    elif keyboard.Key.left in pressed_keys:
        # LEFT
        x, y = -1, 0

    # Έλεγχος αν το βέλος Up είναι πατημένο
    elif keyboard.Key.up in pressed_keys:
        # FORWARD
        x, y = 0, 1

```

```

# Έλεγχος αν το βέλος Down είναι πατημένο
elif keyboard.Key.down in pressed_keys:
    # BACKWARD
    x, y = 0, -1
else:
    # Κανένα σχετικό πλήκτρο δεν είναι πατημένο
    return

# Αποστολή του κατάλληλου μηνύματος στη σειριακή θύρα
_send_direction(x, y)

"""
Χειρισμός γεγονότος απελευθέρωσης πλήκτρου.

Παράμετροι:
    key (pygame.keyboard.Key) – Το πλήκτρο που απελευθερώθηκε.

Περιγραφή:
    Αφαιρεί το πλήκτρο από το `pressed_keys`. Αν το πλήκτρο είναι ESC, τερματίζει
    τον listener.
"""
def on_release(key):
    # Έλεγχος αν το πλήκτρο είναι στο σύνολο pressed_keys
    if key in pressed_keys:
        # Αφαίρεση του πλήκτρου από το σύνολο
        pressed_keys.remove(key)

    # Έλεγχος αν το πλήκτρο που απελευθερώθηκε είναι ESC
    if key == keyboard.Key.esc:
        # Έξοδος
        return False

#
=====
=====

# Εκκίνηση του listener για τα πλήκτρα και του thread για τη λήψη σημάτων δόνησης

```

```

#
=====

# Έλεγχος αν το ονόμα του σκριπτ είναι __main__ (εκκίνηση απευθείας, όχι ως
εισαγωγή)
if __name__ == "__main__":
    # Εμφάνιση μηνύματος οδηγίων στον χρήστη
    print("[INFO] Χρησιμοποιήστε τα πλήκτρα-βέλη για τον έλεγχο. Πατήστε ESC για
έξοδο.\n")

    # Εκκίνηση του background reader thread για vibration signals πριν από τον
listener των πλήκτρων πλοήγησης
    _reader_thread = threading.Thread(target=_serial_vibration_reader,
name="SerialVibrationReader", daemon=True)

    # Εκκίνηση του thread
    _reader_thread.start()

    # Κύρια λειτουργία: Εκκίνηση του listener για τα πληκτρα πλοήγησης.
    with keyboard.Listener(on_press=on_press, on_release=on_release) as listener:
        # Αναμονή του listener να τερματίσει
        listener.join()

    # Τερματισμός και του thread διαβάσματος σημάτων δόνησης, μετά τον τερματισμό
του listener για τα πλήκτρα πλοήγησης
    _stop_reader.set()

    # Αναμονή του thread να τερματίσει με timeout 2 δευτερολέπτων
    _reader_thread.join(timeout=2.0)

    # Προαιρετικό κλείσιμο της σειριακής θύρας κατά τον τερματισμό
    try:
        # Κλείσιμο της σειριακής σύνδεσης
        ser.close()
    except Exception:
        # Αγνόηση σφαλμάτων κατά το κλείσιμο
        pass

```

ΠΑΡΑΡΤΗΜΑ ΣΤ: Ερωτηματολόγια Εμπειρίας Χρήστη

1.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο ο	σαφές
βαρετό	ο ο ο ο ο ο ●	συναρπαστικό
αδιάφορο	ο ο ο ο ο ο ●	ενδιαφέρον
συμβατικό	ο ο ο ο ο ο ●	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο ο	πρωτοπόρο

2.

παρελκυστικό	ο ο ο ο ο ο ● ο	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ο ●	εύκολο
ανεπαρκές	ο ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ο ●	σαφές
βαρετό	ο ο ο ο ο ο ●	συναρπαστικό
αδιάφορο	ο ο ο ο ο ο ●	ενδιαφέρον
συμβατικό	ο ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ο ●	πρωτοπόρο

3.

παρελκυστικό	ο ο ο ο ο ο ● ο	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ο ●	εύκολο
ανεπαρκές	ο ο ο ο ο ο ●	επαρκές
μπερδεμένο	ο ο ο ο ο ο ●	σαφές
βαρετό	ο ο ο ο ο ο ●	συναρπαστικό
αδιάφορο	ο ο ο ο ο ο ●	ενδιαφέρον
συμβατικό	ο ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ο ● ο	πρωτοπόρο

4.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ο ●	επαρκές
μπερδεμένο	ο ο ο ο ο ο ●	σαφές
βαρετό	ο ο ο ο ο ● ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ● ο	ενδιαφέρον
συμβατικό	ο ο ο ο ο ο ●	εφευρετικό
συνηθισμένο	ο ο ο ο ο ο ●	Πρωτοπόρο

5.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο	σαφές
βαρετό	ο ο ο ο ● ο ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ● ο	ενδιαφέρον
συμβατικό	ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο	πρωτοπόρο

6.

παρελκυστικό	ο ο ο ο ο ● ο	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ο ●	εύκολο
ανεπαρκές	ο ο ο ο ο ο ●	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο	σαφές
βαρετό	ο ο ο ο ο ● ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ο ●	ενδιαφέρον
συμβατικό	ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο	πρωτοπόρο

7.

παρελκυστικό	ο ο ο ο ο ● ο	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ο ●	εύκολο
ανεπαρκές	ο ο ο ο ο ο ●	επαρκές
μπερδεμένο	ο ο ο ο ο ο ●	σαφές
βαρετό	ο ο ο ο ο ● ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ο ●	ενδιαφέρον
συμβατικό	ο ο ο ο ο ο ●	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο	πρωτοπόρο

8.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο	σαφές
βαρετό	ο ο ο ο ● ο ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ● ο	ενδιαφέρον
συμβατικό	ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο	πρωτοπόρο

9.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο	σαφές
βαρετό	ο ο ο ο ● ο ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ● ο	ενδιαφέρον
συμβατικό	ο ο ο ο ● ο ο	εφευρετικό
συνηθισμένο	ο ο ο ο ο ● ο	πρωτοπόρο

10.

παρελκυστικό	ο ο ο ο ο ο ●	υποστηρικτικό
περίπλοκο	ο ο ο ο ο ● ο	εύκολο
ανεπαρκές	ο ο ο ο ο ● ο	επαρκές
μπερδεμένο	ο ο ο ο ο ● ο	σαφές
βαρετό	ο ο ο ο ● ο ο	συναρπαστικό
αδιάφορο	ο ο ο ο ο ● ο	ενδιαφέρον
συμβατικό	ο ο ο ο ο ● ο	εφευρετικό
συνηθισμένο	ο ο ο ο ● ο ο	πρωτοπόρο

