



ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

**ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη προσομοίωσης ρύθμισης τονικότητας
μουσικών οργάνων

Development of a simulation for the tuning of
musical instruments

Στυλιανίδης Φαίδων

AM:159300

Επιβλέπων Καθηγητής: Ζαφειρόπουλος Βασίλειος

Πάτρα, Σεπτέμβριος 2026

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη προσομοίωσης ρύθμισης τονικότητας
μουσικών οργάνων

Στυλιανίδης Φαίδων

Επιτροπή Επίβλεψης Πτυχιακής Εργασίας

Επιβλέπων Καθηγητής:

Βασίλης Ζαφειρόπουλος

ΣΕΠ – ΕΑΠ

Συν-Επιβλέπουσα Καθηγήτρια:

Γεωργία Γκαράνη

Καθηγήτρια, Πανεπιστήμιο

Θεσσαλίας

Συν-Επιβλέπων Καθηγητής:

Ευθύμιος Ταμπούρης

Καθηγητής, Πανεπιστήμιο

Μακεδονίας

Πάτρα, Σεπτέμβριος 2026

*Η παρούσα εργασία αφιερώνεται σε όλους τους λάτρεις
της μουσικής, των επιστημών, των ψυχαγωγικών
δραστηριοτήτων σε υπολογιστές.*

Περίληψη

Στο πλαίσιο της σύγχρονης ψηφιακής εκπαίδευσης, οι εικονικές προσομοιώσεις έχουν αναδειχθεί ως ισχυρά εργαλεία για τη δημιουργία ρεαλιστικών, ασφαλών και διαθέσιμων περιβαλλόντων μάθησης. Βασισμένες σε προηγμένες τεχνολογίες υπολογιστικών γραφικών, μοντελοποίησης φυσικής και διαδραστικότητας, επιτρέπουν στους χρήστες να πειραματίζονται με πολύπλοκα συστήματα και ευαίσθητες διαδικασίες που θα ήταν δαπανηρές, επικίνδυνες ή απλά αδύνατο να αναπαραχθούν στον πραγματικό κόσμο. Αυτό το τεχνολογικό υπόβαθρο καθιστά τις προσομοιώσεις ιδανικές για την εξερεύνηση πεδίων που απαιτούν πρακτική εξάσκηση, όπως είναι και η τέχνη της χόρδισης και φροντίδας μουσικών οργάνων.

Συγκεκριμένα, αυτή η διπλωματική εργασία αξιοποιεί τις δυνατότητες των προσομοιώσεων για να αναπτύξει ένα εικονικό εργαστήριο μουσικών οργάνων. Το επίκεντρο είναι η προγραμματιστική δημιουργία μιας αλληλεπιδραστικής τρισδιάστατης προσομοίωσης, όπου ο χρήστης μπορεί να εξασκηθεί στις διαδικασίες της ρύθμισης τονικότητας μερικών βασικών οργάνων, έγχορδων, πνευστών και κρουστών. Μέσα από ρεαλιστική αλληλεπίδραση με 3d μοντέλα, αντίγραφα πραγματικών οργάνων και με κριτήριο ανατροφοδότησης τον ακριβή παραγόμενο ήχο, ο χρήστης μαθαίνει τις αιτιολογικές σχέσεις μεταξύ της μηχανικής δράσης και του ακουστικού αποτελέσματος του κουρδίσματός τους, καθώς και μερικές απλές τεχνικές κουρδίσματός τους.

Η υλοποίηση γίνεται στο περιβάλλον ανάπτυξης εφαρμογών Unity, το οποίο προσφέρει τα ιδανικά εργαλεία για να συνδυάσει τα βασικά συστατικά του προγράμματος: την τρισδιάστατη απεικόνιση και το animation των οργάνων, την προσομοίωση της φυσικής συμπεριφοράς τους, τη διαχείριση του ακριβή ήχου και, κυρίως, τη διαδραστική λογική που μετατρέπει τον χρήστη από απλό θεατή σε ενεργό μέρος της διαδικασίας.

Η παρούσα προσομοίωση ως εκπαιδευτικό εργαλείο παρέχει ένα απολύτως ασφαλές περιβάλλον για πειραματισμό, αφαιρώντας τον φόβο της ζημιάς. Επιπλέον, καθιστά προσβάσιμη τη γνωριμία με όργανα και διαδικασίες που συχνά παραμένουν άγνωστες για το ευρύ κοινό ή δύσκολα προσβάσιμες. Με αυτόν τον τρόπο, καλλιεργεί

το ενδιαφέρον και βελτιώνει τη μουσική κατανόηση, λειτουργώντας ως συμπληρωματικό βοήθημα τόσο για αυτοδιδασκαλία όσο και για τυπική μουσική εκπαίδευση.

Η εργασία δεν στοχεύει μόνο στη δημιουργία ενός μεμονωμένου εργαλείου, αλλά και στην επέκταση των ορίων των εκπαιδευτικών προσομοιώσεων όπως π.χ. των εικονικών εργαστηρίων φυσικών επιστημών του ΕΑΠ, Onlabs (<http://onlabs.eap.gr/>). Ενσωματώνοντας συνιστώσες υψηλού βαθμού ρεαλισμού και διαδραστικότητας, θέτει τις βάσεις για μελλοντικές εξελίξεις σε ένα θέμα που δεν έχει αξιοποιηθεί αρκετά προγραμματιστικά όπως είναι η ρεαλιστική μοντελοποίηση και διαδραστική χρήση μουσικών οργάνων και αποτελεί ένα πρώτο βήμα για εφαρμογές με μεγαλύτερη λεπτομέρεια καθώς και με περισσότερα και πιο πολύπλοκα όργανα.

Λέξεις – Κλειδιά

Εκπαιδευτική εφαρμογή προσομοίωσης,

Unity,

Onlabs,

Ρύθμιση τονικού ύψους,

Μουσικά όργανα

Abstract

In the context of modern digital education, virtual simulations have emerged as powerful tools for creating realistic, safe, and accessible learning environments. Based on advanced technologies such as computer graphics, physics simulation, and interactivity, they allow users to experiment with complex systems and sensitive processes that would be costly, dangerous, or simply impossible to replicate in the real world. This technological foundation makes simulations ideal for exploring fields that require practical training, such as the art of tuning and maintaining musical instruments.

Specifically, this thesis leverages the potential of simulations to develop a virtual musical instrument laboratory. The focus is on the programmatic creation of an interactive three-dimensional simulation, where the user can practice the processes of tuning of some basic string, wind, and percussion instruments. Through realistic interaction with 3D models, replicas of real instruments, and using the accurate sound produced as feedback, the user learns the causal relationships between mechanical action and the acoustic result of their tuning, as well as some simple tuning techniques.

The implementation is carried out in the Unity application development environment, which provides the ideal tools to combine the essential components of the program: the three-dimensional visualization and animation of the instruments, the simulation of their physical behavior, the management of accurate sound, and, most importantly, the interactive logic that transforms the user from a simple viewer into an active agent.

As an educational tool, this simulation provides an absolutely safe environment for experimentation, eliminating the fear of damage. Furthermore, it makes accessible the familiarity with instruments and processes that often remain unknown to the general public or difficult to access. In this way, it cultivates interest and improves musical understanding, serving as a complementary aid for both self-instruction and formal music education.

The work aims not only at the creation of a standalone tool but also at expanding the boundaries of educational simulations, such as the virtual science laboratories of the Hellenic Open University, Onlabs (<http://onlabs.eap.gr/>). By incorporating components

of a high degree of realism and interactivity, it lays the groundwork for future developments in a theme that has not been programmatically exploited enough before: the realistic modeling and interactive use of musical instruments. It constitutes the first step towards applications with greater detail and involving more complex instruments.

Keywords

Educational simulation application,
Unity,
Onlabs,
Tuning,
Musical instruments

Περιεχόμενα

1	Εισαγωγή.....	15
1.1	Η Προσομοίωση και η Σημασία της στην Εκπαίδευση.....	15
1.1.1	Τι είναι Προσομοίωση	15
1.1.2	Εξέλιξη στο χρόνο	16
1.1.3	Οφέλη στην Εκπαιδευτική Διαδικασία.....	18
1.2	Η Μηχανή Παιχνιδιών Unity	20
1.3	Το εικονικό εργαστήριο Onlabs.....	22
2	Στοιχεία θεωρίας χόρδισης μουσικών οργάνων	24
2.1	Γενικά.....	24
2.2	Στοιχεία θεωρίας τονικών συστημάτων.....	25
2.2.1	Συχνότητα και μουσικό ύψος.....	25
2.2.2	Λόγοι Συχνοτήτων και Μουσικά Διαστήματα.....	26
2.2.3	Το Ισοσυγκερασμένο Τονικό Σύστημα (12-TET)	27
2.2.4	Απόκλιση Συχνότητας και Μονάδα Cents.....	27
2.3	Στοιχεία Θεωρίας Κουρδίσματος Έγχορδων Οργάνων	28
2.3.1	Φυσική της Ταλαντούμενης Χορδής	28
2.3.1.1	Θεμελιώδης τρόπος ταλάντωσης	28
2.3.2	Το Αρπιχόρδιο και το κούρδισμά του.....	29
2.3.2.1	Γενικά Χαρακτηριστικά και Μηχανισμός Παραγωγής Ήχου.....	30
2.3.2.2	Συστήματα Κουρδίσματος στο Αρπιχόρδιο	31
2.3.2.3	Διαδικασία Κουρδίσματος	32
2.3.2.4	Συσχέτιση Τάσης Χορδής και Γωνίας Περιστροφής Κλειδιού.....	34
2.4	Στοιχεία θεωρίας κουρδίσματος πνευστών οργάνων.....	35
2.4.1	Θεμελιώδεις Αρχές Παραγωγής Ήχου στα Πνευστά	35
2.4.2	Η Φλογέρα	37
2.4.2.1	Ρύθμιση Ύψους.....	38
2.5	Στοιχεία θεωρίας κουρδίσματος κρουστών οργάνων	39
2.5.1	Κατηγορίες Κρουστών.....	39
2.5.2	Φυσική Ταλαντούμενης Μεμβράνης	39
2.5.3	Το Ταμπούρο	40
2.5.3.1	Ρύθμιση τάσης μεμβράνης ταμπούρου	41

3	Ανάπτυξη διαδραστικής προσομοίωσης κουρδίσματος μουσικών οργάνων ..	43
3.1	Τρισδιάστατα μοντέλα μουσικών οργάνων	43
3.1.1	Μορφότυποι αρχείων τρισδιάστατων μοντέλων (OBJ και FBX)....	44
3.1.2	Χρήση του Blender για επεξεργασία των μοντέλων	45
3.1.3	Μοντέλο αρπιχορδίου	45
3.1.4	Μοντέλο ταμπούρου	49
3.1.5	Μοντέλο φλογέρας.....	51
3.2	Ηχητικά δείγματα μουσικών οργάνων	53
3.3	Σύνδεση λειτουργιών με τον κώδικα Onlabs.....	54
3.4	Σύστημα πλοήγησης και ελέγχου χρήστη.....	55
3.4.1	Κίνηση του χρήστη	56
3.4.2	Περιστροφή και έλεγχος της οπτικής κατεύθυνσης.....	57
3.4.3	Μεταφορά αντικειμένων και χρήση της ενεργής κάμερας	59
3.5	Σύστημα αλληλεπίδρασης χρήστη και διαχείρισης ποντικιού.....	61
3.5.1	Κατηγοριοποίηση αντικειμένων και τύποι αλληλεπίδρασης.....	62
3.5.2	Εντοπισμός αντικειμένου και έναρξη αλληλεπίδρασης.....	63
3.5.3	Πάτημα, περιστροφή και ολοκλήρωση της αλληλεπίδρασης.....	65
3.5.4	Συλλογή, μεταφορά και συνδυαστική χρήση αντικειμένων	66
3.6	Γενικές κλάσεις διαδραστικών αντικειμένων	68
3.6.1	Βασική κλάση διαδραστικών αντικειμένων	68
3.6.2	Γενική κλάση περιστρεφόμενων αντικειμένων	70
3.6.3	Περιστροφή με μηχανικούς περιορισμούς.....	71
3.6.4	Γενική κλάση μεταφερόμενων αντικειμένων	73
3.6.5	Γενική κλάση πλήκτρων	75
3.6.6	Συμπληρωματικές γενικές κλάσεις του συστήματος	76
3.6.6.1	Γενική κλάση μουσικού οργάνου	76
3.6.6.2	Πλαίσιο ενεργοποίησης ειδικής προβολής	76
3.6.6.3	Εναλλαγή λειτουργίας εργαλείου κουρδίσματος.....	77
3.6.6.4	Πρόσθετη διεπαφή χρήστη	78
3.6.6.5	Αποτέλεσμα συνδυαστικής χρήσης	78
3.7	Υλοποίηση αρπιχορδίου	79

3.7.1	Διαχείριση δεδομένων του αρπιχορδίου	80
3.7.2	Κεντρική διαχείριση του οργάνου	82
3.7.3	Πλήκτρα και παραγωγή ήχου	84
3.7.4	Υλοποίηση του μηχανισμού κουρδίσματος του αρπιχορδίου	90
3.7.4.1	Μεταφορά και προσάρτηση του κλειδιού κουρδίσματος	91
3.7.4.2	Περιστροφή του κλειδιού κουρδίσματος	92
3.7.4.3	Σύνδεση των ορίων περιστροφής με το εύρος κουρδίσματος	95
3.7.4.4	Φυσικό μοντέλο του πείρου και της χορδής	97
3.7.4.5	Υπολογισμός της απόκλισης σε cents.....	98
3.7.4.6	Εφαρμογή του κουρδίσματος στον παραγόμενο ήχο.....	98
3.7.4.7	Ψηφιακή ένδειξη κουρδίσματος	99
3.7.5	Συνολική λειτουργία κλάσεων αρπιχορδίου	101
3.7.6	Ειδικές περιπτώσεις και υποστηρικτικοί έλεγχοι	104
3.8	Υλοποίηση φλογέρας	105
3.8.1	Βάση δεδομένων της φλογέρας	106
3.8.2	Οπές και επιλογή της νότας	108
3.8.3	Παραγωγή του ήχου	109
3.8.4	Κούρδισμα της φλογέρας.....	110
3.8.4.1	Μετακίνηση του άνω τμήματος της φλογέρας	111
3.8.4.2	Φυσικό μοντέλο του κουρδίσματος	112
3.8.4.3	Εφαρμογή της μεταβολής στον ήχο	113
3.8.4.4	Υπολογισμός της απόκλισης και κουρδιστήρι.....	113
3.8.5	Συνολική λειτουργία της υλοποίησης της φλογέρας	116
3.9	Υλοποίηση ταμπούρου	118
3.9.1	Αναπαράσταση και βασική λειτουργία του ταμπούρου	119
3.9.2	Τμήματα της μεμβράνης και παραγωγή ήχου.....	119
3.9.3	Φυσικό μοντέλο της μεμβράνης	120
3.9.4	Κλειδί και κοχλίας κουρδίσματος.....	122
3.9.4.1	Μετατροπή της περιστροφής σε μεταβολή τάσης	124
3.9.4.2	Έλεγχος της διαδικασίας κουρδίσματος	126
3.9.5	Συνολική λειτουργία της υλοποίησης του ταμπούρου	130

3.10	Συμπληρωματικές λειτουργίες της εφαρμογής.....	132
3.10.1	Τυχαία αρχική κατάσταση κουρδίσματος.....	133
3.10.2	Λειτουργίες επίδειξης των οργάνων	133
3.10.3	Μηνύματα και οδηγίες προς τον χρήστη	134
3.10.4	Κεντρικό μενού της εφαρμογής.....	134
3.10.5	Σύνοψη των συμπληρωματικών λειτουργιών	136
4	Μελλοντικές επεκτάσεις	137
4.1.1	Επέκταση σε προσομοίωση εργασιών συντήρησης.....	137
4.1.2	Προσθήκη πιο σύνθετων συστημάτων κουρδίσματος.....	138
4.1.3	Επέκταση σε περισσότερα και πιο σύνθετα μουσικά όργανα	139
4.1.4	Χρήση προηγμένων συστημάτων επεξεργασίας ήχου.....	139
4.1.5	Επέκταση σε περιβάλλον εικονικής πραγματικότητας.....	140
4.1.6	Συνολική προοπτική	141
	Ευχαριστίες	142
	Αποθετήριο πηγαίου κώδικα.....	143
	Βιβλιογραφία	144

Πίνακας εικόνων

Εικόνα 1. Χρήση προσομοιωτή πτήσης (πηγή: aopa.org).	16
Εικόνα 2. Προσομοιωτές που χρησιμοποιήθηκαν στην αποστολή Apollo 13 (πηγή: televue.com).	17
Εικόνα 3. Προοπτική τρίτου προσώπου (πηγή: Silent Hill 2).	18
Εικόνα 4. Προοπτική πρώτου προσώπου (πηγή: Silent Hill 2 remake mod).	18
Εικόνα 5. Ανάπτυξη της εφαρμογής Onlabs σε περιβάλλον Unity.	21
Εικόνα 6. Το μικροσκόπιο του εικονικού εργαστηρίου Onlabs.	23
Εικόνα 7. Πάτημα (push) του διακόπτη (switch) του μικροσκοπίου.	23
Εικόνα 8. Αρπιχόρδιο 16ου αιώνα (πηγή: earlymusicamerica.org).	30
Εικόνα 9. Μηχανισμός κρούσης αρπιχορδίου (πηγή: Wikipedia).	31
Εικόνα 10. Ηλεκτρονικό κουρδιστήρι (πηγή: Wikipedia).	32
Εικόνα 11. Κλειδί κουρδίσματος αρπιχορδίου (πηγή: hpschd.nu).	33
Εικόνα 12. Πείροι κουρδίσματος χορδών αρπιχορδίου (πηγή: harpsichord.com).	33
Εικόνα 13. Κοινή φλογέρα 2 μερών (πηγή: Amazon).	37
Εικόνα 14. Το ταμπούρο και τα συστατικά του μέρη (πηγή: Yamaha.com).	40
Εικόνα 15. Μεταλλικές χορδές ταμπούρου στο κάτω μέρος του (πηγή: thomann.gr).	41
Εικόνα 16. Ρύθμιση τονικού ύψους ταμπούρου περιμετρικά (πηγή: wikiHow).	42
Εικόνα 17. Τρισδιάστατο μοντέλο αρπιχορδίου (© 3d_molier – 2014).	46
Εικόνα 18. Πείροι κουρδίσματος αρπιχορδίου (© 3d_molier – 2014).	47
Εικόνα 19. Εργαλείο κουρδίσματος αρπιχορδίου και επεξεργασία του στο blender. .	48
Εικόνα 20. 3d μοντέλο ταμπούρου (© NRG_3D).	49
Εικόνα 21. Βίδες σύσφιξης μεμβράνης (© NRG_3D).	50
Εικόνα 22. Εργαλείο σύσφιξης βιδών τάσης ταμπούρου.	51
Εικόνα 23. Τρισδιάστατο μοντέλο φλογέρας με ξύλινη υφή (© tgarch3d).	51
Εικόνα 24. Τα 3 τμήματα της φλογέρας με μαύρη υφή (© tgarch3d).	52
Εικόνα 25. Κρατώντας πατημένο ένα πλήκτρο του αρπιχορδίου.	80
Εικόνα 26. Στιγμιότυπο από την κατάσταση κουρδίσματος.	90
Εικόνα 27. Διάγραμμα κληρονομικότητας του αρπιχορδίου.	102
Εικόνα 28. Βασικές σχέσεις μεταξύ των κλάσεων του αρπιχορδίου.	103
Εικόνα 29. Μεταφορά της φλογέρας.	106

Εικόνα 30. Πίνακας δακτυλισμών της φλογέρας.....	108
Εικόνα 31. Η φλογέρα σε κατάσταση κουρδίσματος.	111
Εικόνα 32. Διάγραμμα κληρονομικότητας φλογέρας.....	117
Εικόνα 33. Βασικές σχέσεις μεταξύ των κλάσεων της φλογέρας.	117
Εικόνα 34. Χτύπημα τμήματος του τυμπάνου με το κλειδί τοποθετημένο στον αντίστοιχο κοχλία.	120
Εικόνα 35. Οδηγίες αλγόριθμου κουρδίσματος ταμπούρου.....	126
Εικόνα 36. Διάγραμμα κληρονομικότητας ταμπούρου	131
Εικόνα 37. Βασικές σχέσεις μεταξύ των κλάσεων του ταμπούρου.....	132
Εικόνα 38. Αρχικό μενού εφαρμογής.	135

1 Εισαγωγή

Σε αυτό το κεφάλαιο αναφέρονται οι βασικές έννοιες και τα τεχνολογικά εργαλεία που αποτελούν το υπόβαθρο της παρούσας εργασίας. Αρχικά αναλύεται η έννοια της προσομοίωσης ενώ στη συνέχεια παρουσιάζεται η μηχανή ανάπτυξης παιχνιδιών Unity, η οποία χρησιμοποιείται για τη δημιουργία διαδραστικών τρισδιάστατων εφαρμογών. Τέλος περιγράφεται το εικονικό εργαστήριο Onlabs, το οποίο αποτελεί τη βάση για την ανάπτυξη της εφαρμογής που υλοποιείται στο πλαίσιο της παρούσας διπλωματικής εργασίας.

1.1 Η Προσομοίωση και η Σημασία της στην Εκπαίδευση

Στην ενότητα αυτή παρουσιάζεται η έννοια της προσομοίωσης και ο ρόλος της στη σύγχρονη εκπαιδευτική διαδικασία. Αρχικά ορίζεται η έννοια της προσομοίωσης και τα βασικά χαρακτηριστικά της, στη συνέχεια εξετάζεται η ιστορική της εξέλιξη και η σταδιακή ενσωμάτωσή της σε διάφορους επιστημονικούς τομείς, ενώ τέλος αναλύονται τα σημαντικότερα οφέλη που προσφέρει στην εκπαιδευτική διαδικασία.

1.1.1 Τι είναι Προσομοίωση

Με τον όρο *προσομοίωση* αναφερόμαστε στη σύνθετη διαδικασία της μίμησης ή αναπαράστασης της λειτουργίας, των χαρακτηριστικών και του περιβάλλοντος ενός πραγματικού συστήματος. Βασικός της στόχος είναι η εμβάθυνση της κατανόησης και η αποτελεσματική εκπαίδευση σε συγκεκριμένους τομείς, μέσω της δημιουργίας ενός ασφαλούς και ελεγχόμενου πλαισίου πειραματισμού. Στην ουσία, η προσομοίωση χρησιμοποιεί διάφορες τεχνικές και μεθόδους, όπως ψηφιακές, ανθρώπινες ή φυσικές αναπαραστάσεις, για να αναπτύξει ένα ακριβές μοντέλο πραγματικών καταστάσεων. Αυτές οι αναπαραστάσεις μπορούν να υλοποιηθούν μέσω υπολογιστικών συστημάτων, τα οποία μιμούνται πραγματικές συνθήκες, εξομοιώνοντας πειράματα με τη χρήση μαθηματικών μοντέλων και γραφικού περιβάλλοντος και επιτρέποντας στους χρήστες να χειρίζονται παραμέτρους, προσαρμόζοντας εικονικές δραστηριότητες σε διάφορα επιστημονικά πεδία.

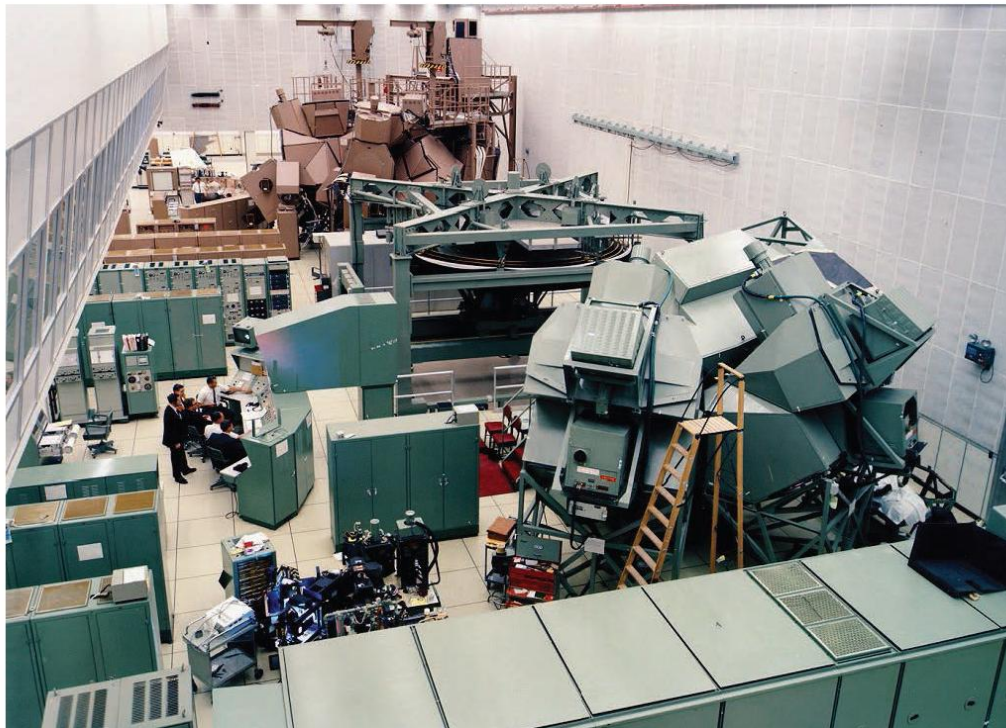
1.1.2 Εξέλιξη στο χρόνο

Η τεχνολογική εξέλιξη έχει υποστηρίξει την ανάπτυξη της προσομοίωσης για πολλές δεκαετίες, με βασικές εφαρμογές σε κρίσιμους τομείς όπως η αεροπορία (Εικόνα 1) και η διαστημική έρευνα. Από τη δεκαετία του '50, οργανισμοί όπως η Ομοσπονδιακή Διοίκηση Αεροπορίας (FAA) των ΗΠΑ τη χρησιμοποιούσαν για την επαναπιστοποίηση των πιλότων, ενώ τη δεκαετία του '70 η NASA την αξιοποίησε για την επίλυση προβλημάτων κατά τη διάσωση της αποστολής Apollo 13 (Εικόνα 2). Στην Ελλάδα, από τα τέλη της δεκαετίας του '90, εγχειρήματα όπως το «ΟΔΥΣΣΕΙΑ» προώθησαν την ανάπτυξη εκπαιδευτικών προγραμμάτων προσομοίωσης, σηματοδοτώντας την ανάγκη για νέες μεθόδους στη διδασκαλία.



Εικόνα 1. Χρήση προσομοιωτή πτήσης (πηγή: aopa.org).

Apollo Simulators installed at NASA - Front - Lunar Module Simulator (green)
Mid- Mission Effects Projector-Lunar surface (green)
Back - Command Module Simulator (tan)



Εικόνα 2. Προσομοιωτές που χρησιμοποιήθηκαν στην αποστολή Apollo 13 (πηγή: televue.com).

Αυτή η εξέλιξη αποτέλεσε τη βάση για τη σύγχρονη αλματώδη ανάπτυξη της εικονικής πραγματικότητας και ειδικότερα των εικονικών εργαστηρίων στην εκπαίδευση. Μια βασική πτυχή για την ενίσχυση του ρεαλισμού και της εμπύθισης (immersion) είναι η *προοπτική πρώτου προσώπου*. Σε αντίθεση με την προοπτική τρίτου προσώπου, η οποία προσφέρει μια εξωτερική θέα όπως σε διάφορα παιχνίδια που χειρίζονται ένα χαρακτήρα που φαίνεται στην οθόνη (Εικόνα 3), η προοπτική πρώτου προσώπου επιτρέπει στους χρήστες να βλέπουν μέσα από τα "μάτια" του εικονικού χαρακτήρα (Εικόνα 4), δημιουργώντας μια πιο άμεση, ενστικτώδη και συναισθηματικά εμπλεκόμενη εμπειρία. Αυτή η προσέγγιση ευθυγραμμίζεται με έρευνες που υποστηρίζουν ότι ενισχύει τη συμμετοχή και τη βαθύτερη κατανόηση σε ψηφιακά περιβάλλοντα (Denisova & Cairns, 2015).



Εικόνα 3. Προοπτική τρίτου προσώπου (πηγή: Silent Hill 2).



Εικόνα 4. Προοπτική πρώτου προσώπου (πηγή: Silent Hill 2 remake mod).

1.1.3 Οφέλη στην Εκπαιδευτική Διαδικασία

Η ενσωμάτωση της προσομοίωσης στην εκπαίδευση προσφέρει ένα πλήθος σημαντικών πλεονεκτημάτων, που την καθιστούν ένα πολύτιμο εργαλείο για τον σύγχρονο εκπαιδευτικό σχεδιασμό:

- **Ενεργητική Συμμετοχή και Εμβάθυνση Κατανόησης:** Οι προσομοιώσεις μετασχηματίζουν τον εκπαιδευόμενο από παθητικό παρατηρητή σε ενεργό συμμετέχοντα. Αυτή η διαδραστική συμμετοχή συνδέεται άμεσα με βαθύτερη κατανόηση και καλύτερη διατήρηση της γνώσης, καθώς οι άνθρωποι μαθαίνουν αλληλοεπιδρώντας με το περιβάλλον τους. Η εικονική πραγματικότητα λειτουργεί ως ένα ισχυρό εργαλείο που επιτρέπει στον

εκπαιδευόμενο να *βιώνει* σενάρια αντί να τα φαντάζεται, θεμελιώνοντας τη γνώση μέσω της άμεσης εμπειρίας.

- **Ασφαλής Πειραματισμός και Εκπαίδευση για Κρίσιμες Καταστάσεις:** Δημιουργεί ένα ασφαλές περιβάλλον όπου οι εκπαιδευόμενοι μπορούν να πειραματιστούν, να κάνουν λάθη και να αναπτύξουν δεξιότητες χωρίς τον κίνδυνο ζημιάς σε ακριβό εξοπλισμό ή τραυματισμού (Mystakidis et al., 2022). Αυτό είναι κρίσιμο σε τομείς υψηλού κινδύνου όπως η μηχανική, η ιατρική και η αεροναυπηγική. Επιπλέον, προετοιμάζει τους συμμετέχοντες να ανταποκριθούν σε ακραίες συνθήκες και καταστάσεις κρίσης, ενισχύοντας την κριτική τους σκέψη και ικανότητα επίλυσης προβλημάτων.
- **Προσβασιμότητα και Ευελιξία:** Οι προσομοιώσεις εργαστηρίων συγκεκριμένα αποτελούν μια λύση για την εξ' αποστάσεως εκπαίδευση, ξεπερνώντας περιορισμούς όπως η έλλειψη φυσικών εργαστηριακών υποδομών, γεωγραφικά εμπόδια ή ειδικές συνθήκες (π.χ. πανδημίες). Οι μαθητές/φοιτητές έχουν πρόσβαση στο εικονικά εργαστήρια από οποιοδήποτε μέρος και σε χρόνο που τους εξυπηρετεί, ενώ εξασφαλίζεται η απρόσκοπτη, επαναληπτική και χωρίς φθορά χρήση των εικονικών οργάνων και υλικών.
- **Κόστος-Αποτελεσματικότητα και Επικαιρότητα:** Μειώνει σημαντικά το κόστος, καθώς δεν απαιτεί έξοδα σε φυσικό εξοπλισμό, χώρους και υλικά. Παράλληλα, η εύκολη ενημέρωση και ενσωμάτωση νέων πληροφοριών στο ψηφιακό περιβάλλον διασφαλίζει την επικαιρότητα του εκπαιδευτικού περιεχομένου.
- **Πολυαισθητηριακή Μάθηση και Ελκυστικότητα:** Σε σύγκριση με τα κείμενα, ένα περιβάλλον προσομοίωσης που συνδυάζει οπτικά γραφικά, διαδραστικά στοιχεία και ακουστική ανατροφοδότηση, είναι πολύ πιο ελκυστικό και αποτελεσματικό για τη νέα γενιά των «ψηφιακών ιθαγενών». Η χρήση πολλαπλών αισθήσεων βοηθά στην καλύτερη αφομοίωση πολύπλοκων εννοιών, ειδικά στις φυσικές επιστήμες.

Συνοψίζοντας, η προσομοίωση αναδεικνύεται ως ουσιώδες συστατικό της σύγχρονης εκπαιδευτικής πρακτικής. Δημιουργεί ένα δυναμικό, συμμετοχικό και ασφαλές περιβάλλον μάθησης που προωθεί την ενεργητική γνώση, την κριτική σκέψη και την καινοτομία, υπηρετώντας τόσο τις ανάγκες της επίσημης εκπαίδευσης όσο και της δια βίου μάθησης.

1.2 Η Μηχανή Παιχνιδιών Unity

Η Unity¹ είναι μια μηχανή παιχνιδιών (game engine) για ανάπτυξη δισδιάστατων και τρισδιάστατων δισδιάστατων εφαρμογών η οποία κυκλοφορεί από το 2005. Η εταιρεία λογισμικού που την αναπτύσσει ιδρύθηκε το 2004 στη Δανία ως Over The Edge Entertainment (OTEE) και μετονομάστηκε σε Unity Technologies το 2007, μεταφέροντας την έδρα της στο Σαν Φρανσίσκο της Καλιφόρνια. Δημιουργήθηκε με στόχο να παρέχει σε ένα ευρύτερο φάσμα προγραμματιστών πρόσβαση σε ισχυρά εργαλεία ανάπτυξης παιχνιδιών και διαδραστικών εφαρμογών, «κρύβοντας» μέρος της τεχνικής πολυπλοκότητας και επιτρέποντας στους δημιουργούς να επικεντρωθούν στο σχεδιασμό, τη λογική και το περιεχόμενο ενός παιχνιδιού (Goldstone, 2009).

Το λογισμικό της Unity υποστηρίζει προγραμματισμό στη γλώσσα C#. Είναι σχεδιασμένο να είναι προσβάσιμο σε όσους έχουν βασικές γνώσεις προγραμματισμού και προσφέρει ένα πλούσιο σύνολο από έτοιμα εργαλεία και υλικά (materials) για τον οπτικό ορισμό των τρισδιάστατων μοντέλων. Ένα βασικό πλεονέκτημά της είναι το Unity Asset Store, μια διαδικτυακή αγορά που παρέχει μια πληθώρα έτοιμων πόρων (Assets), όπως τρισδιάστατα μοντέλα, υφές (textures), ήχους, και scripts, είτε δωρεάν είτε επί πληρωμή. Αυτό διευκολύνει σημαντικά τη διαδικασία ανάπτυξης, καθώς οι δημιουργοί δεν απαιτείται να έχουν ειδικές γνώσεις μοντελοποίησης αντικειμένων.

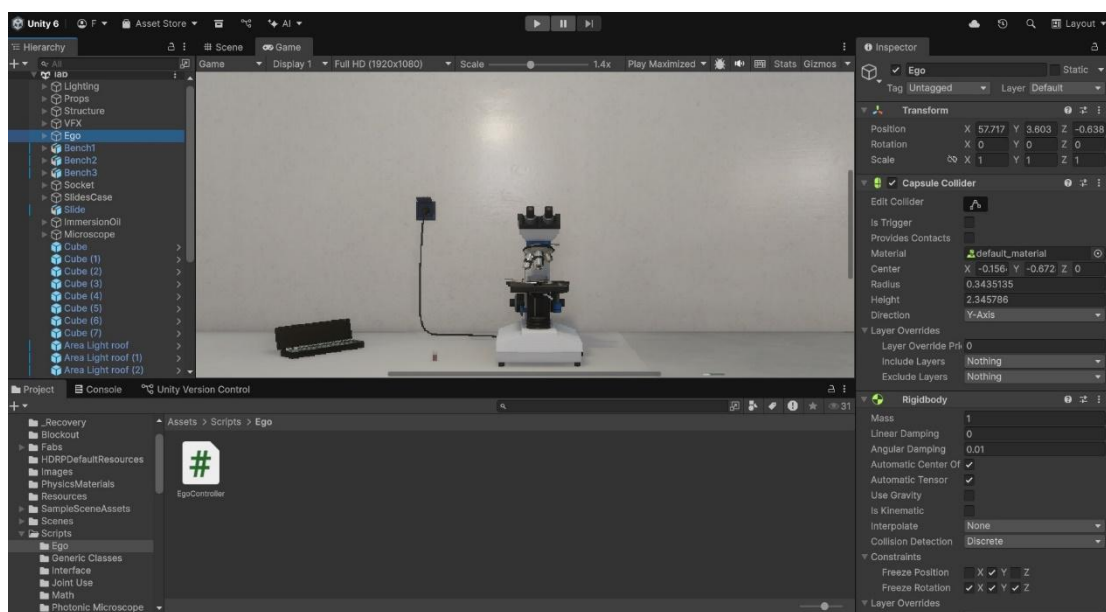
Ένα από τα πιο σημαντικά χαρακτηριστικά της Unity είναι η εκτενής πολυπλατφορμικότητα (multiplatform). Μια εφαρμογή μπορεί να αναπτυχθεί μία φορά και στη συνέχεια να εξαχθεί για πληθώρα πλατφορμών, όπως Android, iOS, Windows, macOS, Linux, κονσόλες (π.χ. PS4, Xbox One), ακόμη και για διαδικτυακούς περιηγητές μέσω HTML5 (Hocking & Hocking, 2022). Οι συχνές ενημερώσεις της πλατφόρμας διασφαλίζουν τη συμβατότητά της με τις πιο πρόσφατες

¹ <https://unity.com/>

τεχνολογίες. Επιπλέον, η δωρεάν διαθεσιμότητα σε συνδυασμό με την ύπαρξη μιας πολύ ενεργής διαδικτυακής κοινότητας και διαφόρων φόρουμ υποστήριξης, την καθιστά μια καλή επιλογή για εκπαιδευτικές και ερευνητικές εφαρμογές.

Το γραφικό περιβάλλον της Unity είναι πολύ απλό στη χρήση όπως φαίνεται στην Εικόνα 5, με μία κεντρική οθόνη που δείχνει την εφαρμογής προς ανάπτυξη, πάνω αριστερά την ιεραρχία των αντικειμένων στη σκηνή, κάτω αριστερά τους φακέλους του project, στα δεξιά τους ένα παράθυρο με τα αρχεία ενός από αυτούς και τέλος, στα δεξιά της οθόνης τον επιθεωρητή (inspector), ο οποίος εμφανίζει όλα τα συστατικά, ιδιότητες και ρυθμίσεις ενός προς επιλογή αντικειμένου (Menard, 2015).

Για τους σκοπούς της παρούσας διπλωματικής εργασίας, η Unity επιλέχθηκε ως το ιδανικό περιβάλλον ανάπτυξης για την αλληλεπιδραστική τρισδιάστατη προσομοίωση ρύθμισης και συντήρησης μουσικών οργάνων. Η ικανότητά της να συνδυάζει ρεαλιστικά γραφικά, ακριβή προσομοίωση φυσικής, διαχείριση ήχου και εύκολη υλοποίηση διαδραστικής λογικής, μαζί με τη δυνατότητα πρόσβασης σε έτοιμους πόρους, την καθιστά την βέλτιστη λύση για τη δημιουργία ενός ρεαλιστικού, ασφαλούς και εκπαιδευτικού εικονικού περιβάλλοντος.



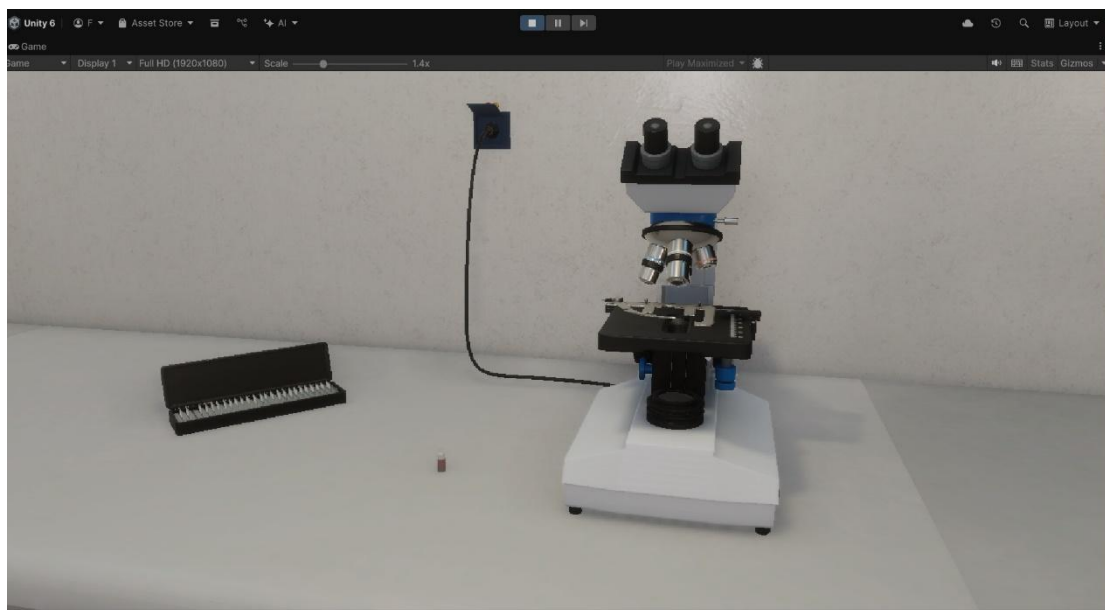
Εικόνα 5. Ανάπτυξη της εφαρμογής Onlabs σε περιβάλλον Unity.

1.3 Το εικονικό εργαστήριο Onlabs

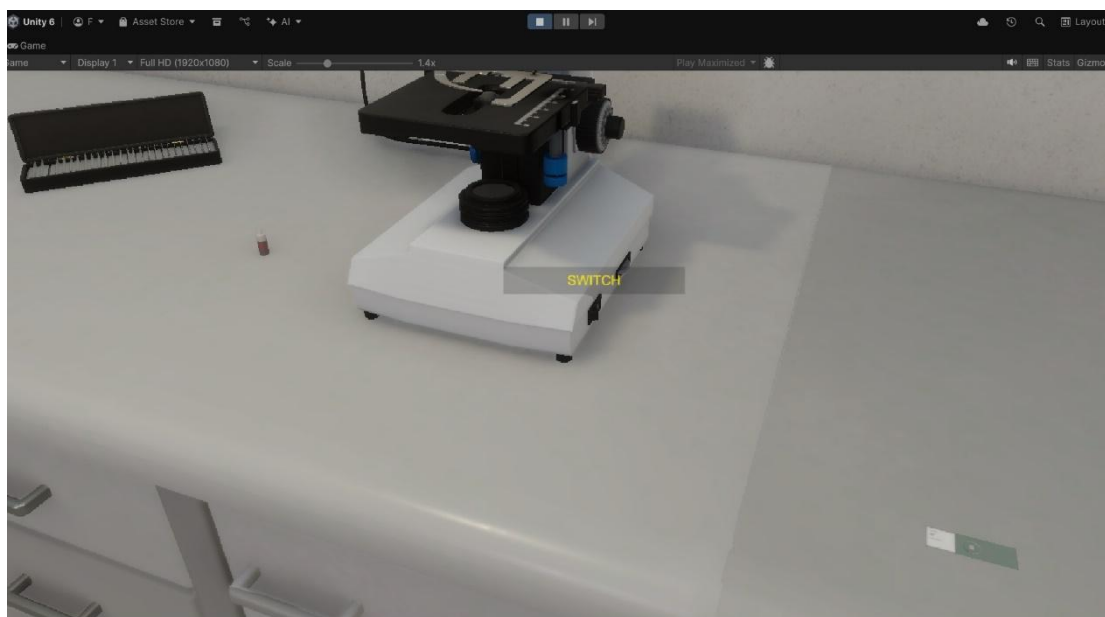
Το Onlabs (Εικόνα 6) είναι ένα εικονικό τρισδιάστατο εργαστήριο επιστημονικών πειραμάτων, αναπτυγμένο στην πλατφόρμα Unity για το ΕΑΠ. Λειτουργεί ως ένα ρεαλιστικό και διαδραστικό εκπαιδευτικό περιβάλλον, που επιτρέπει στους φοιτητές να αλληλεπιδράσουν με τον εξοπλισμό ενός εργαστηρίου με ασφάλεια και ευελιξία, προωθώντας την ενεργητική μάθηση μέσω δοκιμής και σφάλματος.

Αυτό το περιβάλλον αποτελεί τη βάση και την πρωταρχική διεπαφή πρώτου προσώπου (first person interface) για τη δημιουργία της εφαρμογής κουρδίσματος οργάνων της παρούσας εργασίας. Το βασικό σύστημα πλοήγησης και διάδρασης υιοθετεί την κίνηση μέσω πληκτρολογίου και ποντικιού, ενώ προσφέρει ένα εύρος ενεργειών, όπως πάτημα (press) κουμπιών ή πλήκτρων, στρίψιμο (rotate) κλειδιών για κούρδισμα και άλλες, που είναι απαραίτητες για τον ρεαλιστικό χειρισμό αντικειμένων ένα παράδειγμα των οποίων φαίνεται στην Εικόνα 7. Αυτή η διαδραστική προσέγγιση θυμίζει παιχνίδι περιπέτειας (adventure game)², και εξασφαλίζει ρεαλιστική χρήση του εικονικού εξοπλισμού με την απαιτούμενη ακρίβεια.

² Είδος βιντεοπαιχνιδιών, όπου ο παίκτης χειρίζεται πρωταγωνιστή με ικανότητα αλληλεπίδρασης με αντικείμενα του περιβάλλοντός καθώς επίσης και συνομιλίας με άλλους χαρακτήρες, για την επίλυση συγκεκριμένων γρίφων και ολοκλήρωση ενός σεναρίου. Χαρακτηριστικά παραδείγματα αποτελούν το Monkley Island για τρίτου προσώπου και το Myst για πρώτου.



Εικόνα 6. Το μικροσκόπιο του εικονικού εργαστηρίου Onlabs.



Εικόνα 7. Πάτημα (push) του διακόπτη (switch) του μικροσκοπίου.

2 Στοιχεία θεωρίας χόρδισης μουσικών οργάνων

Στο κεφάλαιο αυτό παρουσιάζονται οι βασικές θεωρητικές αρχές που σχετίζονται με τα τονικά συστήματα και τη σχέση συχνότητας-μουσικού ύψους, καθώς και οι φυσικές αρχές που διέπουν το κούρδισμα διαφορετικών κατηγοριών μουσικών οργάνων, όπως τα έγχορδα, τα πνευστά και τα κρουστά. Οι έννοιες αυτές αποτελούν το θεωρητικό υπόβαθρο για την κατανόηση και την υλοποίηση της διαδικασίας κουρδίσματος στο πλαίσιο της παρούσας εργασίας.

2.1 Γενικά

Το κούρδισμα των μουσικών οργάνων αποτελεί μία σύνθετη διαδικασία που συνδυάζει τη μουσική θεωρία, την ακουστική φυσική και τη μαθηματική περιγραφή του ήχου. Για την ανάπτυξη μιας ρεαλιστικής προσομοίωσης κουρδίσματος σε περιβάλλον Unity, είναι απαραίτητη η κατανόηση των θεμελιωδών αρχών που διέπουν τα τονικά συστήματα, τις συχνотικές σχέσεις μεταξύ των φθόγων και τους μηχανισμούς παραγωγής ήχου στα διαφορετικά είδη μουσικών οργάνων.

Αρχικά παρουσιάζονται στοιχεία θεωρίας τονικών συστημάτων. Αναλύεται η έννοια της συχνότητας ως φυσικό μέγεθος που καθορίζει το μουσικό ύψος, η σημασία των λόγων συχνοτήτων στον καθορισμό των μουσικών διαστημάτων και εξετάζεται το ισοσυγκερασμένο τονικό σύστημα (12-TET), το οποίο αποτελεί τη βάση της δυτικής μουσικής πρακτικής και το πρότυπο αναφοράς της παρούσας προσομοίωσης. Παρουσιάζεται η μαθηματική του διατύπωση, ο τρόπος υπολογισμού των συχνοτήτων μέσω εκθετικών σχέσεων, καθώς και η έννοια της απόκλισης σε cents, η οποία χρησιμοποιείται στον υπολογισμό της ακρίβειας κουρδίσματος.

Έπειτα αναπτύσσεται η θεωρία κουρδίσματος εγχόρδων οργάνων. Αρχικά αναλύεται η φυσική της ταλαντούμενης χορδής και η σχέση μεταξύ μήκους, τάσης και γραμμικής πυκνότητας μάζας με τη συχνότητα ταλάντωσης. Στη συνέχεια γίνεται αναφορά στο επιλεγμένο όργανο για αυτήν τη κατηγορία, το αρπικόρδιο, στις ιδιαιτερότητες του και του τρόπου κουρδίσματός του.

Κατόπιν παρουσιάζεται η θεωρία κουρδίσματος πνευστών οργάνων, με έμφαση στη φυσική της στήλης αέρα και στα φαινόμενα στάσιμων κυμάτων. Η ανάλυση

εφαρμόζεται στη φλογέρα, εξετάζοντας τη σχέση μήκους σωλήνα και παραγόμενης συχνότητας.

Τέλος, αναλύεται η περίπτωση των κρουστών οργάνων και ειδικότερα του ταμπούρου, όπου η παραγωγή ήχου βασίζεται στην ταλάντωση μεμβράνης. Παρουσιάζονται οι φυσικές παράμετροι που επηρεάζουν το ύψος και η ιδιαιτερότητα του κουρδίσματος μεμβρανών σε σχέση με τα τονικά όργανα.

Με τον τρόπο αυτό, το κεφάλαιο θεμελιώνει το απαραίτητο θεωρητικό υπόβαθρο για τη μαθηματική μοντελοποίηση και την υλοποίηση της προσομοίωσης κουρδίσματος των επιλεγμένων οργάνων στο ψηφιακό περιβάλλον.

2.2 Στοιχεία θεωρίας τονικών συστημάτων

Στην ενότητα αυτή παρουσιάζονται οι βασικές έννοιες που συνδέουν τη φυσική του ήχου με τη μουσική θεωρία, όπως η συχνότητα και το μουσικό ύψος, οι λόγοι συχνοτήτων που καθορίζουν τα μουσικά διαστήματα, καθώς και το ισοσυγκερασμένο τονικό σύστημα που χρησιμοποιείται ως πρότυπο αναφοράς στη σύγχρονη μουσική πρακτική.

2.2.1 Συχνότητα και μουσικό ύψος

Ο ήχος αποτελεί μηχανικό κύμα που παράγεται από την ταλάντωση ενός σώματος και διαδίδεται μέσω ενός ελαστικού μέσου (Berg & Stork, 2005; Rossing et al., 2002), που στη περίπτωση της μουσικής είναι συνήθως ο αέρας. Το βασικό φυσικό μέγεθος που καθορίζει το μουσικό ύψος (pitch) είναι η συχνότητα της ταλάντωσης f (frequency), η οποία μετράται σε Hertz (Hz) και εκφράζει τον αριθμό των ταλαντώσεων ανά δευτερόλεπτο (Benade, 1990).

Η ανθρώπινη αντίληψη του μουσικού ύψους είναι άμεσα συνδεδεμένη με τη συχνότητα: όσο μεγαλύτερη είναι η συχνότητα ενός ηχητικού περιοδικού σήματος, ως τόσο υψηλότερος γίνεται αντιληπτός αυτός ο ήχος (Rossing et al., 2002). Ωστόσο, η μουσική δεν βασίζεται μόνο στις απόλυτες τιμές συχνοτήτων, αλλά και στις σχέσεις μεταξύ αυτών (Benade, 1990).

2.2.2 Λόγοι Συχνοτήτων και Μουσικά Διαστήματα

Το μουσικό διάστημα μεταξύ δύο φθόγγων καθορίζεται από τον λόγο των συχνοτήτων τους, $\frac{f_2}{f_1}$.

Η σημασία του λόγου αυτού είναι θεμελιώδης, καθώς η ακουστική αντίληψη της συμφωνίας (consonance) συνδέεται με απλές αριθμητικές αναλογίες (Fletcher & Rossing, 2008). Για παράδειγμα, η Οκτάβα έχει λόγο 2:1, η Καθαρή Πέμπτη 3:2 και η Καθαρή Τετάρτη 4:3.

Όταν δύο συχνότητες συνδέονται με απλό αριθμητικό λόγο, οι αρμονικές τους συμπίπτουν σε μεγάλο βαθμό, γεγονός που οδηγεί σε ακουστική σταθερότητα και αρμονική συνοχή (Fletcher & Rossing, 2008; Rossing et al., 2002) και οδηγεί σε ακουστικά ευχάριστο αποτέλεσμα. Αυτή η μαθηματική θεμελίωση των εν λόγω διαστημάτων ανάγεται ήδη στην αρχαιότητα και συνδέεται με το έργο του Πυθαγόρα (Barbour, 2004; Christensen, 2002).

Ωστόσο, η αυστηρή διατήρηση όλων των «καθαρών» λόγων σε ένα πλήρες τονικό σύστημα δημιουργεί μαθηματικές και ηχητικές ασυμβατότητες, γεγονός που οδήγησε στην ανάπτυξη συστημάτων συγκερασμού (Barbour, 2004). Για παράδειγμα ο μεσοτονικός συγκερασμός (meantone temperament), βελτιώνει την καθαρότητα των μεγάλων τρίτων εις βάρος ορισμένων πέμπτων, ή τα λεγόμενα «καλοσυγκερασμένα» συστήματα (well temperaments), τα οποία επιτρέπουν χρήση όλων των τονικοτήτων με άνιση αλλά μουσικά αποδεκτή κατανομή αποκλίσεων (Barbour, 2004; Jorgensen, 1991). Η ιστορική εξέλιξη των συστημάτων συγκερασμού αποκτά ιδιαίτερη καλλιτεχνική και πρακτική έκφραση κατά την περίοδο του Μπαρόκ, μέσα από το έργο του Johann Sebastian Bach και ειδικότερα το *Das Wohltemperierte Klavier* (μτφρ.: Το Καλώς Συγκερασμένο Κλειδοκύμβαλο) το οποίο ανέδειξε τη δυνατότητα χρήσης όλων των τονικοτήτων σε ένα ενιαίο σύστημα κουρδίσματος. Η κορύφωση αυτής της εξέλιξης δεν ήταν άλλη από την καθιέρωση του ισοσυγκερασμένου συστήματος, στο οποίο οι αποκλίσεις κατανέμονται ισότιμα σε όλα τα διαστήματα της οκτάβας, προσφέροντας πλήρη συμμετρία και μαθηματική απλότητα εις βάρος της απόλυτης καθαρότητας ορισμένων συμφωνιών. Έτσι, πρακτικά, ένα μουσικό κομμάτι μπορεί να παιχτεί σε οποιαδήποτε τονικότητα χωρίς να συσσωρεύονται αποκλίσεις που να οδηγούν παραφωνίες κατά την εκτέλεσή του.

2.2.3 Το Ισοσυγκερασμένο Τονικό Σύστημα (12-TET)

Το επικρατέστερο σύστημα στη σύγχρονη δυτική μουσική είναι το ισοσυγκερασμένο σύστημα δώδεκα φθόγγων (12-TET: Twelve Tone Equal Temperament) (Barbour, 2004).

Στο σύστημα αυτό ορίζεται ως *οκτάβα* το μουσικό διάστημα μεταξύ δύο φθόγγων οι οποίοι γίνονται αντιληπτοί ως «ίδιος» φθόγγος σε διαφορετικό τονικό ύψος. Γενικά ισχύουν οι παρακάτω ιδιότητες:

- Η οκτάβα διατηρεί λόγο συχνοτήτων 2:1.
- Η οκτάβα διαιρείται σε 12 ίσα λογαριθμικά διαστήματα (Benade, 1990). Η λογαριθμική διαμέριση επιβάλλεται λόγω του διπλασιασμού της συχνότητας σε κάθε νότα επόμενης οκτάβας.
- Κάθε διαδοχικό ημιτόνιο προκύπτει από πολλαπλασιασμό της προηγούμενης συχνότητας με τον παράγοντα $2^{1/12}$

Επομένως, εάν μία νότα έχει συχνότητα f_0 , τότε η συχνότητα φθόγγου που απέχει n ημιτόνια υπολογίζεται από τη σχέση:

$$f = f_0 \times 2^{n/12} \quad (1)$$

Η εκθετική αυτή σχέση εξασφαλίζει ότι όλα τα ημιτόνια είναι ίσα σε λογαριθμική κλίμακα, επιτρέποντας μετατροπία (modulation) σε οποιαδήποτε τονικότητα χωρίς μεταβολή της δομής των διαστημάτων (Jorgensen, 1991).

Στην πράξη, ως συχνότητα αναφοράς χρησιμοποιείται συνήθως το Λα της τέταρτης οκτάβας (A4), το οποίο έχει οριστεί στα 440 Hz.

Το 12-TET επιλέγεται στην παρούσα εργασία ως πρότυπο αναφοράς, καθώς προσφέρει σαφή και υπολογίσιμη μαθηματική δομή, κατάλληλη για ψηφιακή υλοποίηση.

2.2.4 Απόκλιση Συχνότητας και Μονάδα Cents

Για την αξιολόγηση της ακρίβειας κουρδίσματος στο συγκεκριμένο σύστημα χρησιμοποιείται η μονάδα **cent**, η οποία αποτελεί λογαριθμική υποδιαίρεση του ημιτονίου.

Συγκεκριμένα:

- 1 οκτάβα = 1200 cents
- 1 ημιτόνιο = 100 cents

Η απόκλιση μεταξύ μετρούμενης συχνότητας $f_{measured}$ και επιθυμητής συχνότητας f_{target} σε cents υπολογίζεται ως εξής:

$$c = 1200 \times \log_2 \left(\frac{f_{measured}}{f_{target}} \right) \quad (2)$$

Η χρήση λογαριθμικής κλίμακας ανταποκρίνεται στον τρόπο με τον οποίο το ανθρώπινο αυτί αντιλαμβάνεται τις διαφορές ύψους (Howard & Angus, 2017).

2.3 Στοιχεία Θεωρίας Κουρδίσματος Έγχορδων Οργάνων

Στην ενότητα αυτή παρουσιάζονται οι βασικές αρχές της φυσικής της ταλαντούμενης χορδής και οι παράγοντες που καθορίζουν το μουσικό ύψος των παραγόμενων φθόγγων. Στη συνέχεια εξετάζεται ως παράδειγμα το αρπιχόρδιο που αποτελεί και το όργανο αυτής της κατηγορίας που περιλαμβάνει η υλοποίηση της εργασίας.

2.3.1 Φυσική της Ταλαντούμενης Χορδής

Η παραγωγή ήχου στα έγχορδα μουσικά όργανα βασίζεται στην ταλάντωση μιας τεντωμένης χορδής. Όταν η χορδή εκτραπεί από τη θέση ισορροπίας της και αφεθεί ελεύθερη, δημιουργούνται εγκάρσια κύματα τα οποία διαδίδονται κατά μήκος της και ανακλώνται στα σταθερά άκρα της, σχηματίζοντας στάσιμα κύματα (Benade, 1990; Fletcher & Rossing, 2008).

2.3.1.1 Θεμελιώδης τρόπος ταλάντωσης

Η συχνότητα της θεμελιώδους ταλάντωσης μιας χορδής μήκους L , υπό τάση T , με γραμμική πυκνότητα μάζας μ (μάζα ανά μονάδα μήκους), δίνεται από τη σχέση:

$$f_1 = \frac{1}{2L} \sqrt{\frac{T}{\mu}} \quad (3)$$

όπου f_1 η συχνότητα θεμελιώδους τρόπου ταλάντωσης, L το ενεργό συνολικό μήκος της χορδής, T η τάση της χορδής και μ η γραμμική πυκνότητα μάζας.

Η εξίσωση αυτή προκύπτει από τη λύση της κυματικής εξίσωσης για χορδή με σταθερά άκρα και αποτελεί θεμελιώδη σχέση στη μουσική ακουστική.

Από την παραπάνω σχέση προκύπτει ότι η συχνότητα μιας χορδής εξαρτάται από τρεις βασικούς παράγοντες:

- **Μήκος Χορδής (L):** Η συχνότητα είναι αντιστρόφως ανάλογη του μήκους. Μείωση του μήκους στο μισό οδηγεί σε διπλασιασμό της συχνότητας, δηλαδή σε άνοδο κατά μία οκτάβα.
- **Τάση (T):** Η συχνότητα είναι ανάλογη της τετραγωνικής ρίζας της τάσης. Αύξηση της τάσης αυξάνει το ύψος, αλλά με μη γραμμικό τρόπο.
- **Γραμμική Πυκνότητα (μ):** Όσο μεγαλύτερη είναι η μάζα ανά μονάδα μήκους, τόσο χαμηλότερη η συχνότητα. Παχύτερες ή πυκνότερες χορδές παράγουν χαμηλότερους φθόγγους.

Στην πράξη, το κούρδισμα στα περισσότερα έγχορδα όργανα επιτυγχάνεται μέσω μεταβολής της τάσης της χορδής, διατηρώντας σταθερά το μήκος και τα φυσικά χαρακτηριστικά της.

Πέρα από τη θεμελιώδη συχνότητα, η χορδή μπορεί να ταλαντώνεται και σε ανώτερους τρόπους (αρμονικούς), των οποίων οι συχνότητες είναι ακέραια πολλαπλάσια της θεμελιώδους:

$$f_n = n \cdot f_1 \quad (4)$$

όπου $n = 1, 2, 3, \dots$

Η παρουσία αρμονικών καθορίζει το ηχόχρωμα (timbre) του οργάνου, ενώ η σχετική έντασή τους επηρεάζεται από το σημείο διέγερσης της χορδής, το υλικό κατασκευής και τον τρόπο στήριξης της χορδής.

Στη προσομοίωση μάς ενδιαφέρει μόνο η θεμελιώδης συχνότητα στην οποία κουρδίζεται κάθε χορδή· οι ανώτερες αρμονικές εμφανίζονται μόνο ως το χαρακτηριστικό ηχόχρωμα στα δείγματα (samples) ήχων των οργάνων. Η μεταβολή της τάσης θα χρησιμοποιηθεί ως βασική μεταβλητή ελέγχου, ενώ η υπολογιζόμενη συχνότητα θα συγκρίνεται με τη συχνότητα-στόχο του ισοσυγκερασμένου συστήματος, όπως παρουσιάστηκε στην προηγούμενη ενότητα.

2.3.2 Το Αρπιχόρδιο και το κούρδισμά του

Το αρπιχόρδιο (harpsichord) ή τσέμπαλο (Εικόνα 8) αποτελεί ένα από τα σημαντικότερα πληκτροφόρα όργανα της περιόδου της Αναγέννησης και του Μπαρόκ και είναι το επιλεγμένο έγχορδο όργανο στην παρούσα εργασία. Στην ενότητα αυτή παρουσιάζονται αρχικά τα βασικά χαρακτηριστικά και ο μηχανισμός παραγωγής ήχου του οργάνου, ενώ στη συνέχεια εξετάζονται τα συστήματα κουρδίσματος που

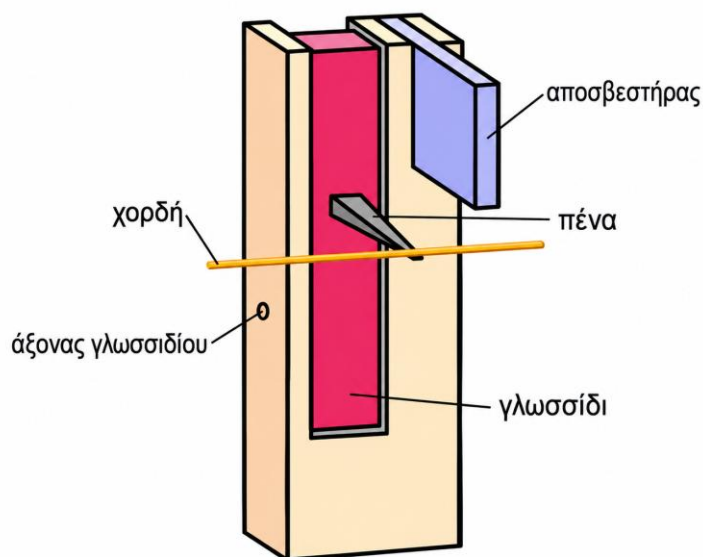
χρησιμοποιήθηκαν ιστορικά στο αρπικόρδιο και η διαδικασία κουρδίσματος που θα ακολουθηθεί στην υλοποίηση, καθώς και το πώς εξαρτάται η τάση της χορδής από τη γωνία περιστροφής της.



Εικόνα 8. Αρπικόρδιο 16ου αιώνα (πηγή: earlymusicamerica.org).

2.3.2.1 Γενικά Χαρακτηριστικά και Μηχανισμός Παραγωγής Ήχου

Το αρπικόρδιο αποτελεί πληκτροφόρο έγχορδο όργανο, στο οποίο ο ήχος παράγεται μέσω νύξης (plucking) των χορδών με τη βοήθεια μηχανισμού πλήκτρων και γλωσσιδίων. Σε αντίθεση με το πιάνο, όπου η διέγερση γίνεται με κρούση σφυριού, στο αρπικόρδιο η χορδή τίθεται σε ταλάντωση μέσω μικρού ρύγχους ή πέννας (plectrum), το οποίο αποσπά στιγμιαία τη χορδή από τη θέση ισορροπίας της (Kottick, 2016). Ο εν λόγω μηχανισμός φαίνεται στην Εικόνα 9.



Εικόνα 9. Μηχανισμός κρούσης αρπιχορδίου (πηγή: Wikipedia).

Στο πάτημα ενός πλήκτρου το γλωσσίδι (tongue) ανεβαίνει και το πτερύγιο κρούει τη χορδή, ενώ όταν αφηθεί το πλήκτρο, λόγω βάρους κατεβαίνει, και το γλωσσίδι υποχωρεί μέσω ενός άξονα περιστροφής ώστε να μην κρούσει τη χορδή ξανά. Μόλις κατέβει τελείως, ο αποσβεστήρας (damper) ακουμπά τη χορδή και σβήνει τον ήχο.

Ο μηχανισμός αυτός έχει ως αποτέλεσμα περιορισμένη δυνατότητα δυναμικής διαφοροποίησης, αλλά σταθερό και σαφώς καθορισμένο φασματικό περιεχόμενο. Η ένταση του ήχου εξαρτάται κυρίως από τον αριθμό των χορδών ανά φθόγγο και όχι από την πίεση του εκτελεστή σε αντίθεση με το πιάνο. Αυτό τον κάνει ιδανικό για τις ανάγκες της προσομοίωσης όπου θέλουμε σταθερή ένταση ήχου που αναλογεί σε κάθε πλήκτρο. Στην εν λόγω εργασία χρησιμοποιούμε μονόσκαλο αρπιχόρδιο (που έχει δηλαδή ένα πληκτρολόγιο) με μονές χορδές προσκείμενες σε κάθε πλήκτρο.

2.3.2.2 Συστήματα Κουρδίσματος στο Αρπιχόρδιο

Ιστορικά, το αρπιχόρδιο δεν κουρδιζόταν με ισοσυγκερασμένο σύστημα, αλλά με διάφορα συστήματα μεσοτονικού ή καλοσυγκερασμένου τύπου (Barbour, 2004; Jorgensen, 1991). Κατά την περίοδο της Αναγέννησης και του Μπαρόκ, ο μεσοτονικός συγκερασμός (meantone temperament) ήταν ιδιαίτερα διαδεδομένος, καθώς παρείχε καθαρές μεγάλες τρίτες, κατάλληλες για τη μουσική πρακτική της εποχής (Kottick, 2016). Κατά τον 18ο αιώνα, οι λεγόμενοι καλοί συγκερασμοί επέτρεπαν την εκτέλεση

σε όλες τις τονικότητες, χωρίς όμως πλήρη ισότητα διαστημάτων. Η πρακτική αυτή συνδέεται με το έργο του Johann Sebastian Bach και ειδικότερα με το Das Wohltemperierte Klavier το οποίο κατέδειξε τη δυνατότητα σύνθεσης και εκτέλεσης σε κάθε τονικότητα (Barbour, 2004). Το πλήρως ισοσυγκερασμένο σύστημα (12-TET) επικράτησε ευρύτερα τον 19ο αιώνα και αποτελεί σήμερα το πρότυπο για σύγχρονες εφαρμογές και ψηφιακές προσομοιώσεις (Jorgensen, 1991).

2.3.2.3 Διαδικασία Κουρδίσματος

Η γενική διαδικασία κουρδίσματος του αρπικορδίου πραγματοποιείται με τις εξής δράσεις:

1. Επιλογή νότας αναφοράς (συνήθως A4 = 440 Hz ή ιστορικά χαμηλότερα πρότυπα) (Denis, 1987).
2. Καθορισμός της μεσαίας οκτάβας με βάση τη νότα αναφοράς.
3. Δημιουργία αλυσίδας διαστημάτων (συνήθως πέμπτων ή τετάρτων) σύμφωνα με το επιλεγμένο σύστημα συγκερασμού.

Η ακρίβεια του κουρδίσματος ελέγχεται μέσω ακουστικής σύγκρισης διαστημάτων ή, σε σύγχρονο πλαίσιο, μέσω ψηφιακής μέτρησης συχνότητας και υπολογισμού απόκλισης σε cents. Στη περίπτωση μας το επιλεγμένο σύστημα συγκερασμού είναι το 12-TET ώστε να διασφαλίζεται μαθηματική συνέπεια και απλότητα υπολογισμών στο περιβάλλον της Unity. Κάθε χορδή μπορεί να κουρδιστεί ανεξάρτητα με βάση ένα ηλεκτρονικό κουρδιστήρι (Εικόνα 10).



Εικόνα 10. Ηλεκτρονικό κουρδιστήρι (πηγή: Wikipedia).

Το κούρδισμα ως μηχανική διαδικασία πραγματοποιείται με την περιστροφή ενός κλειδιού κουρδίσματος (tuning hammer) (Εικόνα 11) που προσάπτεται πάνω στους ξύλινους ή μεταλλικούς πείρους (tuning pins) στους οποίους δένονται οι χορδές (Εικόνα 12).



Εικόνα 11. Κλειδί κουρδίσματος αρπιχορδίου (πηγή: hpschd.nu).



Εικόνα 12. Πείροι κουρδίσματος χορδών αρπιχορδίου (πηγή: harpsichord.com).

2.3.2.4 Συσχέτιση Τάσης Χορδής και Γωνίας Περιστροφής Κλειδιού

Όπως ήδη αναφέραμε στα έγχορδα πληκτροφόρα όργανα όπως το αρπιχόρδιο, η ρύθμιση της τάσης της χορδής επιτυγχάνεται μέσω περιστροφής του μεταλλικού πείρου κουρδίσματος, γύρω από τον οποίο είναι τυλιγμένο το άκρο της χορδής (Kottick, 2016). Η περιστροφή του κλειδιού μεταβάλλει το ενεργό μήκος της χορδής που βρίσκεται υπό τάση, προκαλώντας αντίστοιχη μεταβολή της εφελκυστικής δύναμης (τάσης).

Η τάση T της χορδής συνδέεται με την επιμήκυνσή της σύμφωνα με τον νόμο του Hooke:

$$T = k \cdot \Delta L \quad (5)$$

όπου k η σταθερά ελαστικότητας της χορδής και ΔL η μεταβολή μήκους λόγω τάνυσης (Benade, 1990; Fletcher & Rossing, 2008).

Εφόσον η περιστροφή του κλειδιού προκαλεί περιέλιξη ή αποδέσμευση μικρού τμήματος χορδής γύρω από τον άξονα του πείρου ακτίνας r , η μεταβολή μήκους μπορεί να συσχετιστεί με τη γωνία περιστροφής θ (σε ακτίνια) μέσω της σχέσης:

$$\Delta L = r\Delta\theta \quad (6)$$

Συνεπώς, η τάση γίνεται:

$$T = k \cdot r \cdot \Delta\theta \quad (7)$$

και αντικαθιστώντας στην εξίσωση της συχνότητας παίρνουμε:

$$f = \frac{1}{2L} \sqrt{\frac{k \cdot r \cdot \Delta\theta}{\mu}} \quad (8)$$

Η σχέση αυτή δείχνει ότι η συχνότητα μεταβάλλεται ανάλογα με την τετραγωνική ρίζα της γωνίας περιστροφής του κλειδιού, εφόσον θεωρηθεί γραμμική ελαστική συμπεριφορά της χορδής. Δηλαδή μιας και $f \propto \sqrt{\theta}$ όσο μεγαλώνει η γωνία στριψίματος τόσο πιο αργά αλλάζει η συχνότητα· για παράδειγμα, για διπλασιασμό της συχνότητας χρειάζεται τετραπλασιασμός της γωνίας, ενώ για τριπλασιασμό της συχνότητας απαιτείται εννιαπλασιασμός της γωνίας.

Πρέπει να σημειωθεί ότι στην πραγματικότητα η σχέση τάσης-περιστροφής δεν είναι απολύτως γραμμική, λόγω τριβών, υλικών ιδιοτήτων και μη ιδανικής ελαστικότητας της χορδής (Fletcher & Rossing, 2008). Όσο πλησιάζουμε στο όριο θραύσης της χορδής η συχνότητα αντί να αλλάζει αργά ως τετραγωνική ρίζα της γωνίας, αρχίζει και αλλάζει ως δύναμη του n ($n > 1$), δηλαδή εκθετικά. Το αρπιχόρδιο

κουρδίζεται σωστά πολύ κοντά στο όριο θραύσης κάθε χορδής ώστε να έχει τον χαρακτηριστικό του «τεντωμένο» μεταλλικό ήχο. Ωστόσο, για σκοπούς εκπαιδευτικής και λειτουργικής προσομοίωσης, η γραμμική προσέγγιση της τάσης και η συσχέτιση με την τετραγωνική ρίζα της γωνίας παρέχει επαρκή ακρίβεια με εύρος λίγων ημιτονίων πάνω από τη νότα κουρδίσματος.

Στην προσομοίωση του αρπιχορδίου σε περιβάλλον Unity, η γωνία περιστροφής του κλειδιού μπορεί να οριστεί ως μεταβλητή εισόδου του χρήστη. Έτσι, γυρίζοντας ο χρήστης το εικονικό κλειδί και μεταβάλλοντας τη γωνία περιστροφής του, επιτυγχάνεται η μεταβολή του μήκους της εικονικής χορδής και ως αποτέλεσμα υπολογίζεται η νέα τάση μέσω ελαστικού μοντέλου, ενημερώνεται η συχνότητα βάσει της εξίσωσης ταλαντούμενης χορδής, συγκρίνεται η συχνότητα με την τιμή-στόχο του 12-TET και υπολογίζεται η απόκλιση της σε cents. Με τον τρόπο αυτό επιτυγχάνεται μια φυσικά τεκμηριωμένη και μαθηματικά συνεπής προσομοίωση της διαδικασίας κουρδίσματος.

2.4 Στοιχεία θεωρίας κουρδίσματος πνευστών οργάνων

Στην ενότητα αυτή παρουσιάζονται αρχικά οι θεμελιώδεις φυσικές αρχές παραγωγής ήχου στα πνευστά όργανα και στη συνέχεια εξετάζεται η περίπτωση της φλογέρας, η οποία χρησιμοποιείται ως παράδειγμα εφαρμογής των παραπάνω αρχών στο πλαίσιο της παρούσας προσομοίωσης.

2.4.1 Θεμελιώδεις Αρχές Παραγωγής Ήχου στα Πνευστά

Στα πνευστά όργανα, ο ήχος παράγεται μέσω της ταλάντωσης μιας στήλης αέρα που βρίσκεται εγκλωβισμένη στο εσωτερικό ενός σωλήνα. Η στήλη αυτή τίθεται σε ταλάντωση από τη ροή αέρα που παρέχει ο εκτελεστής και από τον τρόπο με τον οποίο ο αέρας αλληλεπιδρά με συγκεκριμένα δομικά μέρη του οργάνου. Η διέγερση μπορεί να πραγματοποιείται με τους εξής τρόπους:

1. Από ροή αέρα σε αιχμηρή ακμή (τύπου φλογέρας): Στα όργανα αυτά, όπως η φλογέρα και το φλάουτο, ο αέρας κατευθύνεται προς μια αιχμηρή ακμή που βρίσκεται κοντά σε ένα άνοιγμα του σωλήνα. Κατά την πρόσκρουσή του στην ακμή, ο αέρας εκτρέπεται περιοδικά, άλλοτε προς το εσωτερικό και άλλοτε προς το εξωτερικό του οργάνου. Η διαδικασία αυτή προκαλεί μεταβολές της

- πίεσης και συντηρεί την ταλάντωση της στήλης αέρα μέσα στον σωλήνα, με αποτέλεσμα την παραγωγή ήχου.
2. Από μονή ή διπλή γλωσσίδα: Σε όργανα όπως το κλαρινέτο (μονή γλωσσίδα) και το όμποε (διπλή γλωσσίδα), ο ήχος παράγεται με τη βοήθεια ενός ή δύο λεπτών ελαστικών στοιχείων που πάλλονται κατά το φύσημα. Η κίνησή τους μεταβάλλει περιοδικά τη ροή του αέρα που εισέρχεται στο όργανο, προκαλώντας ταλαντώσεις της στήλης αέρα στο εσωτερικό του σωλήνα. Οι ταλαντώσεις αυτές έχουν ως αποτέλεσμα την παραγωγή ήχου.
 3. Από ταλάντωση χειλιών (χάλκινα πνευστά): Στα χάλκινα πνευστά όργανα, όπως η τρομπέτα και το τρομπόνι, ο ήχος παράγεται από την ταλάντωση των χειλιών του εκτελεστή κατά το φύσημα στο επιστόμιο. Τα χείλη ανοίγουν και κλείνουν περιοδικά, προκαλώντας μεταβολές στη ροή του αέρα που εισέρχεται στο όργανο. Οι μεταβολές αυτές διεγείρουν τη στήλη αέρα στο εσωτερικό του σωλήνα, η οποία συντονίζεται σε συγκεκριμένες συχνότητες, συμβάλλοντας στην παραγωγή του χαρακτηριστικού ήχου του οργάνου. ταλάντωση χειλιών .

Η συχνότητα του παραγόμενου ήχου εξαρτάται κυρίως από το ενεργό μήκος της στήλης αέρα και τις οριακές συνθήκες του σωλήνα. Το ενεργό μήκος καθορίζει τις συχνότητες συντονισμού, καθώς η αύξησή του οδηγεί γενικά σε χαμηλότερες συχνότητες, ενώ η μείωσή του σε υψηλότερες. Παράλληλα, η συμπεριφορά του αέρα στα άκρα του σωλήνα επηρεάζει τον σχηματισμό στάσιμων κυμάτων. Σε ένα ανοιχτό άκρο η μεταβολή της ακουστικής πίεσης είναι περίπου μηδενική, ενώ σε ένα κλειστό άκρο η κίνηση του αέρα περιορίζεται. Επομένως, σωλήνες ίδιου μήκους αλλά διαφορετικών οριακών συνθηκών παρουσιάζουν διαφορετικές συχνότητες συντονισμού (Fletcher & Rossing, 2008).

Για σωλήνα ανοικτό και στα δύο άκρα η συχνότητα είναι:

$$f_n = n \frac{v}{2L} \quad (9)$$

ενώ για σωλήνα κλειστό στο ένα άκρο είναι:

$$f_n = (2n - 1) \frac{v}{4L} \quad (10)$$

όπου v η ταχύτητα ήχου στον αέρα, L το ενεργό μήκος σωλήνα και n ο αριθμός αρμονικής (Benade, 1990).

Η μεταβολή του ενεργού μήκους της στήλης αέρα επηρεάζει τη θεμελιώδη συχνότητα του παραγόμενου ήχου. Στα περισσότερα πνευστά όργανα, το ενεργό μήκος μπορεί να μεταβληθεί με διαφορετικούς μηχανισμούς. Στη φλογέρα, το άνοιγμα ή το κλείσιμο των οπών μεταβάλλει το μήκος της στήλης αέρα που συμμετέχει στην ταλάντωση. Στο φλάουτο και στο κλαρινέτο, η ίδια λειτουργία πραγματοποιείται μέσω ενός συστήματος κλειδιών που ανοίγουν και κλείνουν τις οπές του σωλήνα. Στην τρομπέτα, οι βαλβίδες κατευθύνουν τον αέρα μέσα από πρόσθετα τμήματα σωλήνα, μεταβάλλοντας το συνολικό μήκος της διαδρομής του. Αντίστοιχα, στο τρομπόνι, η μετακίνηση του ολισθητήρα αυξομειώνει το μήκος του σωλήνα.

Όταν μια οπή ανοίγει, η στήλη αέρα «τερματίζεται» ουσιαστικά στο σημείο αυτό, μειώνοντας το ενεργό μήκος L .

Επειδή η θεμελιώδης συχνότητα είναι αντιστρόφως ανάλογη του μήκους ($f \propto 1/L$), η μείωση του μήκους οδηγεί σε αύξηση της συχνότητας, ενώ η αύξηση του μήκους οδηγεί σε χαμηλότερο φθόγγο.

2.4.2 Η Φλογέρα

Η φλογέρα (recorder) (Εικόνα 13) αποτελεί απλό αερόφωνο τύπου ανοικτού σωλήνα. Ο ήχος παράγεται όταν η ροή αέρα προσκρούει σε αιχμηρή ακμή, δημιουργώντας περιοδική διαταραχή της πίεσης (Benade, 1990).



Εικόνα 13. Κοινή φλογέρα 2 μερών (πηγή: Amazon).

Οι περισσότερες φλογέρες κοινού τύπου έχουν δύο τμήματα: *επιστόμιο* (κεφαλή) και *κυρίως σώμα*.

Το επιστόμιο περιλαμβάνει τα εξής τρία μέρη: το *κανάλι αέρα* (windway) δηλαδή μια στενή σχισμή που κατευθύνει τον αέρα από το στόμα εντός της φλογέρας, την *ακμή* (labium) που είναι ένα αιχμηρό άκρο στο οποίο προσπίπτει ο αέρας και μετά διαχωρίζεται προς τα έξω και στον θάλαμο εισόδου, της περιοχής σταθεροποίησης της ροής του αέρα.

Το κύριο σώμα της φλογέρας είναι ένας κυλινδρικός σωλήνας, συνήθως από ξύλο ή πλαστικό, ο οποίος καθορίζει το ενεργό μήκος L , την εσωτερική διάμετρο και τη μορφή των ιδιοσυχνοτήτων. Στο σώμα βρίσκονται οι *οπές δακτύλων* (συνήθως 6-8 σε πλήθος), μια πιθανή *οπή αντίχειρα* και το κάτω άκρο εξόδου (ανοιχτό άκρο φλογέρας).

Ο διαχωρισμός της φλογέρας σε δύο τμήματα επιτρέπει τη ρύθμιση του μήκους μέσω της σχετικής μετατόπισης των δύο αυτών τμημάτων. Η εν λόγω δυνατότητα αξιοποιείται για τη λεπτορύθμιση του κουρδίσματος.

Σε επαγγελματικές ή μεγαλύτερες φλογέρες μπορεί να υπάρχουν τρία τμήματα (κεφαλή, μεσαίο σώμα, κάτω τμήμα), ώστε να επιτυγχάνεται καλύτερη ευθυγράμμιση οπών και πιο ακριβής ρύθμιση μήκους.

2.4.2.1 Ρύθμιση Ύψους

Το μουσικό ύψος μεταβάλλεται με τους εξής τρόπους:

1. άνοιγμα/κλείσιμο οπών
2. μεταβολή της πίεσης φυσήματος (υπερφύσημα) με αποτέλεσμα αλλαγή οκτάβας
3. Αλλαγής μήκους μέσω μετατόπισης των 2 τμημάτων

Η μεταβολή του ενεργού μήκους της στήλης αέρα επηρεάζει τη συχνότητα του παραγόμενου ήχου. Στην προσομοίωση, το νέο ενεργό μήκος υπολογίζεται ως:

$$L_{eff} = L_o + \Delta L \quad (11)$$

όπου L_o είναι το ενεργό μήκος στη θέση σωστού κουρδίσματος και ΔL η μεταβολή του μήκους σε σχέση με τη θέση αυτή, ενώ η νέα συχνότητα ως:

$$f = \frac{v}{2L_{eff}} \quad (12)$$

Όταν $\Delta L > 0$, το ενεργό μήκος αυξάνεται και η συχνότητα μειώνεται, ενώ όταν $\Delta L < 0$, το ενεργό μήκος μειώνεται και η συχνότητα αυξάνεται. Για μικρές μεταβολές

του μήκους, από την παραγωγή της σχέσης $f = u/2L$ προκύπτει η προσεγγιστική σχέση:

$$\frac{\Delta f}{f_o} \approx -\frac{\Delta L}{L_o} \quad (13)$$

Επομένως, η σχετική μεταβολή της συχνότητας είναι περίπου ίση σε μέτρο και αντίθετη σε πρόσημο από τη σχετική μεταβολή του ενεργού μήκους της στήλης αέρα (Fletcher & Rossing, 2008).

Σε περιβάλλον Unity, το άνοιγμα οπής ή η διαφοροποίηση του μήκους μεταξύ των μερών μπορεί να προσομοιωθεί μέσω της μεταβολής της παραμέτρου του μήκους και του άμεσου επανυπολογισμού της συχνότητας.

2.5 Στοιχεία θεωρίας κουρδίσματος κρουστών οργάνων

Στην ενότητα αυτή παρουσιάζονται τα βασικά θεωρητικά στοιχεία που σχετίζονται με το κούρδισμα των κρουστών μουσικών οργάνων. Αρχικά εξετάζεται η φυσική της ταλαντούμενης μεμβράνης και δίνεται ιδιαίτερη έμφαση στο ταμπούρο, το οποίο χρησιμοποιείται ως βασικό παράδειγμα μεμβρανόφωνου οργάνου. Στη συνέχεια παρουσιάζεται ο τρόπος ρύθμισης της τάσης της μεμβράνης μέσω των περιμετρικών κοχλιών, που και αποτελεί τη θεωρητική βάση για την προσομοίωση που υλοποιήθηκε σε Unity.

2.5.1 Κατηγορίες Κρουστών

Τα κρουστά διακρίνονται σε *ιδιόφωνα*, στα οποία το ίδιο το σώμα ταλαντώνεται, και *μεμβρανόφωνα*, στα οποία ο ήχος δημιουργείται από την ταλάντωση της μεμβράνης. Στην παρούσα εργασία εστιάζουμε στα μεμβρανόφωνα, και συγκεκριμένα στο ταμπούρο.

2.5.2 Φυσική Ταλαντούμενης Μεμβράνης

Η λύση της κυματικής εξίσωσης οδηγεί σε ιδιομορφές/ιδιοσυναρτήσεις που περιγράφονται από τις ρίζες των συναρτήσεων Bessel (Fletcher & Rossing, 2008). Σε μια πρώτη προσέγγιση, η θεμελιώδης ιδιοσυχνότητα δίνεται από τη σχέση:

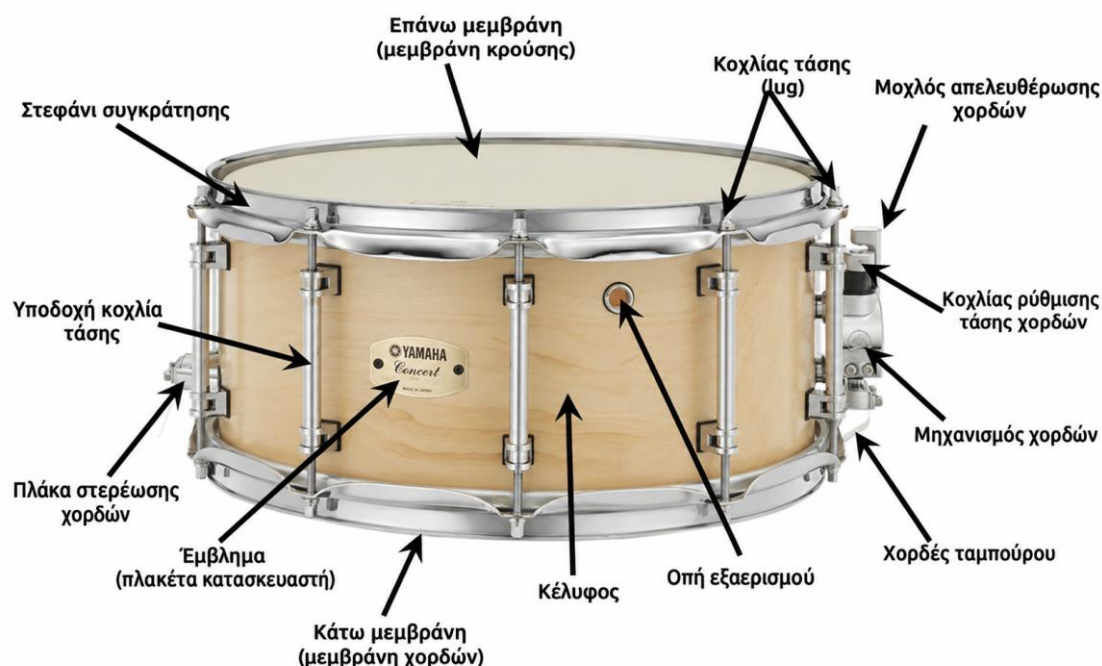
$$f \propto \frac{1}{R} \sqrt{\frac{T}{\sigma}} \quad (14)$$

όπου R η ακτίνα μεμβράνης, T η επιφανειακή τάση και σ η μάζα ανά μονάδα επιφάνειας.

Η μη αρμονική κατανομή των ιδιοσυχνοτήτων εξηγεί γιατί τα περισσότερα τύμπανα δεν παράγουν καθαρό τονικό ύψος· σε αντίθεση με τις χορδές, οι ιδιομορφές ταλάντωσης της μεμβράνης δεν είναι αρμονικά πολλαπλάσιες, γεγονός που προσδίδει στο όργανο μη τονική ακουστική συμπεριφορά και εκλαμβάνεται ως ένα είδος θορύβου ή κρότου.

2.5.3 Το Ταμπούρο

Το ταμπούρο (snare drum) (Εικόνα 14) είναι μεμβρανόφωνο όργανο με δύο δέρματα και μεταλλικές χορδές (snare wires) στο κάτω μέρος (Εικόνα 15), οι οποίες προσδίδουν χαρακτηριστικό ηχόχρωμα κροταλιστού ήχου μέσω επιπλέον τριβών, μικροδονήσεων και σύνθετης απόσβεσης ταλάντωσης.



Εικόνα 14. Το ταμπούρο και τα συστατικά του μέρη (πηγή: Yamaha.com).



Εικόνα 15. Μεταλλικές χορδές ταμπούρου στο κάτω μέρος του (πηγή: thomann.gr).

Το ταμπούρο έχει δύο μεμβράνες, την *πάνω μεμβράνη* ή μεμβράνη κρούσης (batter head), η οποία καθορίζει την αίσθηση κρούσης (rebound) και τον βασικό τόνο, και την *κάτω μεμβράνη* ή μεμβράνη σύρματος (snare side), η οποία είναι πολύ λεπτή και συνήθως κουρδίζεται πολύ πιο ψηλά από την πάνω μεμβράνη για να ανταποκρίνονται άμεσα τα σύρματα που βρίσκονται από κάτω της.

Στην προσομοίωση θα ασχοληθούμε μόνο με το κούρδισμα της πάνω μεμβράνης.

2.5.3.1 Ρύθμιση τάσης μεμβράνης ταμπούρου

Η ρύθμιση της τάσης μεμβράνης ταμπούρου επιτυγχάνεται μέσω κοχλιών περιμετρικά του στεφανιού (Εικόνα 16). Η αύξηση της ροπής σύσφιξης αυξάνει την τάση της μεμβράνης, άρα και τη συχνότητα.

Το κούρδισμα του ταμπούρου επιτυγχάνεται μέσω ομοιόμορφης σύσφιξης των περιμετρικών βιδών, οι οποίες αυξάνουν την τάση της μεμβράνης (Toulson & Toulson, 2021).

Η ροπή σύσφιξης μετατρέπεται σε τάση με το παρακάτω προσεγγιστικό γραμμικό μοντέλο:

$$T = k \cdot \theta \quad (15)$$

όπου k η σταθερά μηχανικής απόκρισης μεμβράνης και θ η γωνία περιστροφής κοχλίου.

Η τάση πρέπει να είναι όσο το δυνατόν πιο ομοιόμορφη σε όλη την περιφέρεια της μεμβράνης. Τυχόν ασύμμετρη τάση προκαλεί παραμόρφωση ιδιομορφών, δημιουργεί «νεκρά σημεία» ακουστικής απόκρισης και μειώνει τη σταθερότητα του ήχου (Benade, 1990).



Εικόνα 16. Ρύθμιση τονικού ύψους ταμπούρου περιμετρικά (πηγή: wikiHow).

Σε περίπτωση ομοιόμορφης κατανομής της τάσης στους κοχλίες η συχνότητα f (Εξίσωση 14), εξαρτάται από τη μέση τάση:

$$T_{avg} = \frac{1}{N} \sum_i^N T_i \quad (16)$$

όπου T_i είναι η τάση σε κάθε σημείο σύσφιξης για N σημεία σύσφιξης συνολικά.

Πρακτικά στη Unity θα εισάγεται μια επιθυμητή μέση συχνότητα και στα περιμετρικά σημεία θα βιδώνονται οι πείροι μέχρι να έχουν όλοι τον ίδιο ήχο στην παρακείμενη μεμβράνη σε αυτή τη συχνότητα.

3 Ανάπτυξη διαδραστικής προσομοίωσης κουρδίσματος μουσικών οργάνων

Το παρόν κεφάλαιο αφορά την ανάπτυξη της διαδραστικής προσομοίωσης κουρδίσματος μουσικών οργάνων που υλοποιήθηκε σε περιβάλλον Unity. Στόχος της εφαρμογής είναι η δημιουργία ενός εικονικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να αλληλεπιδρά με διαφορετικά μουσικά όργανα και να εκτελεί βασικές διαδικασίες κουρδίσματος.

Σε πρώτο στάδιο παρουσιάζονται τα τρισδιάστατα μοντέλα μουσικών οργάνων που επιλέχτηκαν για το σκοπό αυτό, τους λόγους για τους οποίους προτιμήθηκαν καθώς και διάφορες τροποποιήσεις που χρειάστηκαν να γίνουν. Στη συνέχεια αναλύεται η διαδραστικότητα που πρέπει να έχουν για την εφαρμογή μαζί με μέρος τροποποιημένου κώδικα του εικονικού εργαστηρίου Onlabs που υλοποιεί αυτή την διαδραστικότητα. Στο τελευταίο μέρος παρουσιάζεται ο πρωτότυπος κώδικας για την ακουστική προσομοίωση του κουρδίσματος και ο τρόπος μεταβολής του ήχου ανάλογα με τις ενέργειες του χρήστη, ολοκληρώνοντας την χρηστικότητα της εφαρμογής.

3.1 Τρισδιάστατα μοντέλα μουσικών οργάνων

Όπως αναφέρθηκε στην ενότητα 2, η προσομοίωση περιλαμβάνει τρία διαφορετικά μουσικά όργανα, τα οποία αντιπροσωπεύουν διαφορετικές κατηγορίες μηχανισμών κουρδίσματος: το **αρπιχόρδιο** (πληκτροφόρο έγχορδο όργανο), το **ταμπούρο** (μεμβρανόφωνο κρουστό όργανο) και τη **φλογέρα** (πνευστό όργανο).

Η επιλογή των συγκεκριμένων οργάνων επιτρέπει την παρουσίαση διαφορετικών μηχανικών διαδικασιών ρύθμισης της συχνότητας, όπως η μεταβολή της τάσης χορδής, η μεταβολή της τάσης μεμβράνης και η μεταβολή του αποτελεσματικού μήκους σωλήνα.

Για την ανάπτυξη της προσομοίωσης χρησιμοποιήθηκαν έτοιμα τρισδιάστατα μοντέλα υψηλής ποιότητας τα οποία αγοράστηκαν από διαδικτυακές βιβλιοθήκες τρισδιάστατων γραφικών. Τα εν λόγω μοντέλα είναι φωτορεαλιστικά και έχουν σχεδιαστεί αρχικά σε λογισμικά τρισδιάστατης μοντελοποίησης όπως το 3D Max, το Maya και το Blender. Περιλαμβάνουν πλήρη γεωμετρική περιγραφή του αντικειμένου καθώς και υφές υψηλής ανάλυσης που προσδίδουν ρεαλιστική οπτική εμφάνιση. Οι

υφές αυτές φτάνουν σε ανάλυση έως και 4096×4096 pixels, γεγονός που επιτρέπει την καθαρή απεικόνιση των επιφανειών ακόμη και σε κοντινή παρατήρηση.

Τα μοντέλα εισήχθησαν στη Unity μέσω των αρχείων αυτών (.fbx ή .obj) και τοποθετήθηκαν στη σκηνή της εφαρμογής, όπου στη συνέχεια έγινε η προσαρμογή τους προκειμένου να χρησιμοποιηθούν στην ανάπτυξη της διαδραστικής προσομοίωσης.

3.1.1 Μορφότυποι αρχείων τρισδιάστατων μοντέλων (OBJ και FBX)

Για την εισαγωγή τρισδιάστατων μοντέλων στο περιβάλλον ανάπτυξης Unity χρησιμοποιούνται διάφοροι μορφότυποι αρχείων, από τους οποίους οι πιο διαδεδομένοι είναι οι OBJ (Wavefront Object) και FBX (Filmbox). Και οι δύο είναι ιδιαίτερα δημοφιλείς στη βιομηχανία των γραφικών υπολογιστών και των βιντεοπαιχνιδιών για τη μεταφορά γεωμετρικών δεδομένων μεταξύ διαφορετικών εφαρμογών τρισδιάστατης μοντελοποίησης. Στο πλαίσιο της παρούσας εργασίας, τα μοντέλα των μουσικών οργάνων ήταν διαθέσιμα κυρίως σε μία από αυτές τις δύο μορφές και εισήχθησαν στη Unity μέσω της αντίστοιχης διαδικασίας εισαγωγής (import).

Ο μορφότυπος OBJ αποτελεί μια απλή και ανοικτή μορφή αρχείου που αποθηκεύει κυρίως γεωμετρικά χαρακτηριστικά, όπως κορυφές (vertices), επιφάνειες (faces), συντεταγμένες υφών (UV coordinates) και διανύσματα καθέτων (normals). Πλεονεκτεί λόγω της απλότητας και της ευρείας συμβατότητάς του με εφαρμογές όπως το Blender, ωστόσο δεν υποστηρίζει σύνθετα χαρακτηριστικά όπως ιεραρχίες αντικειμένων, σκελετούς κίνησης (rigging), animations ή προηγμένες πληροφορίες υλικών. Αντίθετα, ο μορφότυπος FBX υποστηρίζει εκτός από τη γεωμετρία και πρόσθετα δεδομένα, όπως animations, φωτισμούς και πιο σύνθετες ιδιότητες υλικών, γεγονός που τον καθιστά ιδιαίτερα κατάλληλο για εφαρμογές πραγματικού χρόνου και μηχανές παιχνιδιών όπως η Unity. Το κύριο μειονέκτημά του είναι η μεγαλύτερη πολυπλοκότητα και το αυξημένο μέγεθος των αρχείων. Για τους σκοπούς της παρούσας προσομοίωσης, όπου απαιτείται κυρίως η απεικόνιση στατικών μοντέλων με περιορισμένες κινήσεις, και οι δύο μορφότυποι κρίθηκαν κατάλληλοι, με το FBX να προσφέρει μεγαλύτερη ευελιξία σε περιπτώσεις όπου απαιτούνται επιπλέον πληροφορίες κίνησης ή αλληλεπίδρασης.

3.1.2 Χρήση του Blender για επεξεργασία των μοντέλων

Το Blender είναι ένα λογισμικό ανοιχτού κώδικα για τρισδιάστατη μοντελοποίηση και επεξεργασία γεωμετρίας. Στην παρούσα εργασία χρησιμοποιήθηκε κυρίως για τον διαχωρισμό σύνθετων αντικειμένων σε επιμέρους στοιχεία.

Πολλά από τα μοντέλα που χρησιμοποιήθηκαν είχαν σχεδιαστεί ως ενιαία αντικείμενα, γεγονός που δεν επιτρέπει την ανεξάρτητη αλληλεπίδραση με τα επιμέρους μέρη τους. Για τον λόγο αυτό ήταν απαραίτητο να γίνει επεξεργασία της γεωμετρίας τους ώστε να διαχωριστούν τα στοιχεία που πρέπει να κινούνται ανεξάρτητα. Η διαδικασία αυτή περιλαμβάνει την επιλογή επιμέρους γεωμετρικών τμημάτων ενός μοντέλου, το διαχωρισμό τους σε ξεχωριστά αντικείμενα και την εξαγωγή του μοντέλου σε μορφή κατάλληλη για εισαγωγή στη Unity.

Στις επόμενες ενότητες αναφέρεται αναλυτικά πώς εφαρμόστηκε αυτή η διαδικασία.

3.1.3 Μοντέλο αρπιχορδίου

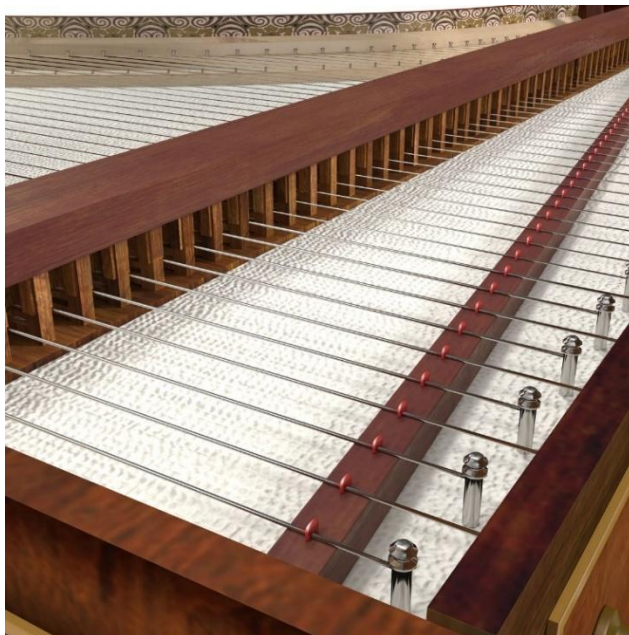
Το αρπιχόρδιο αποτελεί το πρώτο όργανο που ενσωματώθηκε στην προσομοίωση. Η επιλογή του συγκεκριμένου μοντέλου (Εικόνα 17) βασίστηκε κυρίως στη δομή του μηχανισμού κουρδίσματος. Συγκεκριμένα, το μοντέλο που επιλέχθηκε ήταν από τα ελάχιστα διαθέσιμα τρισδιάστατα μοντέλα που περιλαμβάνουν εμφανείς πείρους κουρδίσματος στο τμήμα των χορδών (Εικόνα 18). Η παρουσία των πύρων αυτών είναι απαραίτητη για την προσομοίωση της διαδικασίας κουρδίσματος, καθώς επιτρέπει τη συσχέτιση της περιστροφής τους με τη μεταβολή της τάσης της χορδής.



Εικόνα 17. Τρισδιάστατο μοντέλο αρπιχορδίου (© 3d_molier – 2014).

Ένα επιπλέον πλεονέκτημα του μοντέλου είναι ότι κάθε πλήκτρο αντιστοιχεί σε μονή χορδή. Η διάταξη αυτή απλοποιεί σημαντικά την προσομοίωση, καθώς σε ορισμένα αρπιχόρδια κάθε πλήκτρο μπορεί να αντιστοιχεί σε δύο ή περισσότερες χορδές. Η χρήση μονών χορδών επιτρέπει μια πιο άμεση αντιστοίχιση μεταξύ πλήκτρου, χορδής και πείρου κουρδίσματος.

Παρά τα πλεονεκτήματα αυτά, το μοντέλο παρουσίαζε ορισμένα μειονεκτήματα όσον αφορά τη χρήση του σε διαδραστική εφαρμογή. Συγκεκριμένα, στο αρχικό μοντέλο όλοι οι πείροι κουρδίσματος ήταν ενσωματωμένοι σε ένα ενιαίο αντικείμενο. Η δομή αυτή είναι κατάλληλη για σκοπούς οπτικής απεικόνισης, όμως δεν επιτρέπει την ανεξάρτητη περιστροφή κάθε πείρου μέσα στο περιβάλλον της προσομοίωσης.



Εικόνα 18. Πείροι κουρδίσματος αρπυχορδίου (© 3d_molier – 2014).

Για τον λόγο αυτό, το μοντέλο που ήταν σε μορφή .obj εισήχθη στο Blender, όπου πραγματοποιήθηκε διαχωρισμός των πίων σε ξεχωριστά αντικείμενα. Η διαδικασία αυτή επέτρεψε σε κάθε πίο να μπορεί να περιστρέφεται ανεξάρτητα στην προσομοίωση. Επιπλέον, τα πλήκτρα του πληκτρολογίου διατηρήθηκαν ως ξεχωριστά αντικείμενα ώστε να μπορούν να ενεργοποιούνται ανεξάρτητα από τον χρήστη ενώ ταυτόχρονα άλλαξε η θέση του κέντρου περιστροφής τους από το γεωμετρικό τους κέντρο στο πίσω μέρος που εφαρμόζει στο έπιπλο, ώστε να εμφανίζουν ρεαλιστική κίνηση όταν πιέζονται. Για να διατηρηθούν αυτές οι αλλαγές στο αρχείο χρειάστηκε η εξαγωγή του (export) σε .fbx το οποίο διατηρεί τις αλλαγές σε τέτοια στοιχεία γεωμετρίας τα οποία αφορούν την κίνηση.

Ένα επιπλέον μειονέκτημα του συγκεκριμένου μοντέλου αρπυχορδίου αφορά τον αριθμό των πίων κουρδίσματος. Συγκεκριμένα, το μοντέλο διαθέτει 54 πείρους κουρδίσματος, ενώ το πληκτρολόγιο περιλαμβάνει 60 πλήκτρα. Αυτό σημαίνει ότι δεν υπάρχει αντιστοιχία ενός πείου για κάθε πλήκτρο, όπως θα αναμενόταν σε ένα πλήρως λειτουργικό όργανο. Για τον λόγο αυτό, στο πλαίσιο της προσομοίωσης, μόνο τα πλήκτρα που αντιστοιχούν σε διαθέσιμους πείρους μπορούν να συνδεθούν με πραγματική λειτουργία παραγωγής ήχου και κουρδίσματος. Τα υπόλοιπα 6 πλήκτρα είναι βουβά (mute) και χρησιμοποιούνται αποκλειστικά για οπτικούς/διακοσμητικούς

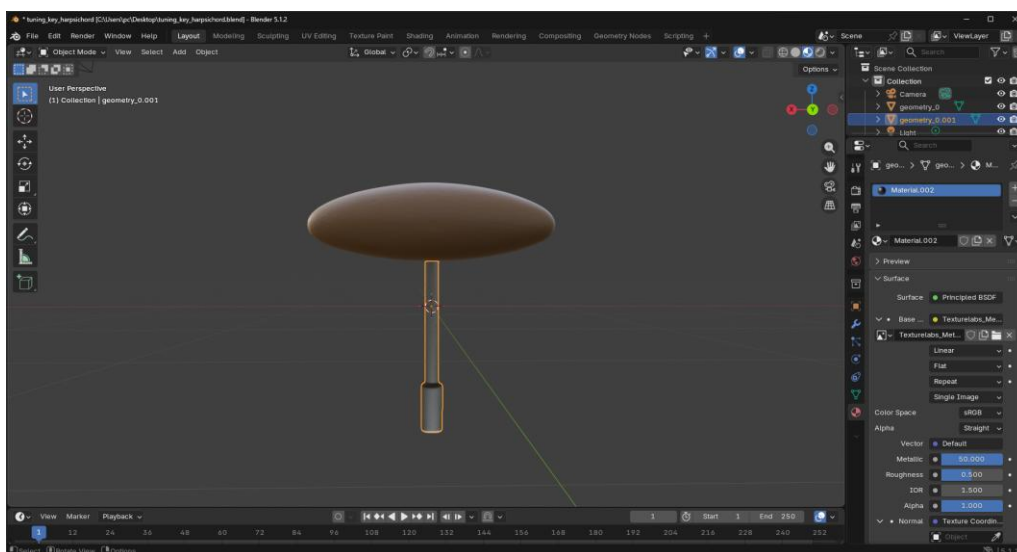
σκοπούς χωρίς να συμμετέχουν στη λειτουργικότητα του οργάνου. Παρότι το συγκεκριμένο στοιχείο αποτελεί ένα περιορισμό του μοντέλου, δεν επηρεάζει σημαντικά τη συνολική λειτουργία της εφαρμογής, καθώς η βασική διαδικασία αλληλεπίδρασης και κουρδίσματος του οργάνου μπορεί να προσομοιωθεί επαρκώς μέσω των υπολοίπων πλήκτρων.

Η βασική λειτουργικότητα που υλοποιείται στο 3d μοντέλο του αρπικόρδιου περιλαμβάνει δύο βασικές ενέργειες του χρήστη:

- το πάτημα των πλήκτρων του πληκτρολογίου
- την περιστροφή των πίων κουρδίσματος με τη βοήθεια εργαλείου κουρδίσματος

Ο χρήστης μπορεί να πιέζει τα πλήκτρα του οργάνου ώστε να ενεργοποιείται η αντίστοιχη χορδή και παράλληλα να χρησιμοποιεί ένα εικονικό εργαλείο για την περιστροφή του αντίστοιχου πείρου. Με τον τρόπο αυτό προσομοιώνεται η βασική διαδικασία κουρδίσματος ενός πληκτροφόρου έγχορδου οργάνου.

Το τρισδιάστατο μοντέλο του κλειδιού κουρδίσματος του αρπικόρδιου δεν ήταν διαθέσιμο σε έτοιμη μορφή και δημιουργήθηκε με τη βοήθεια της διαδικτυακής υπηρεσίας fast3d.io. Ως είσοδος χρησιμοποιήθηκε φωτογραφία πραγματικού εργαλείου κουρδίσματος, από την οποία παράχθηκε αυτόματα ένα αρχικό τρισδιάστατο μοντέλο. Στη συνέχεια εισήχθη στο Blender (Εικόνα 19), όπου πραγματοποιήθηκε περαιτέρω επεξεργασία και βελτιστοποίηση της μορφής του.



Εικόνα 19. Εργαλείο κουρδίσματος αρπικόρδιου και επεξεργασία του στο blender.

3.1.4 Μοντέλο ταμπούρου

Το δεύτερο όργανο που χρησιμοποιήθηκε στην προσομοίωση είναι το ταμπούρο. Η επιλογή του συγκεκριμένου μοντέλου (Εικόνα 20) βασίστηκε κυρίως στη σαφή παρουσία των μηχανισμών σύσφιξης της μεμβράνης. Το μοντέλο περιλαμβάνει δέκα βίδες σύσφιξης (tension rods) (Εικόνα 21) περιμετρικά του στεφανιού, οι οποίες χρησιμοποιούνται στην πραγματικότητα για τη ρύθμιση της τάσης της μεμβράνης.

Η ύπαρξη των δέκα αυτών σημείων σύσφιξης επιτρέπει την προσομοίωση της διαδικασίας κουρδίσματος του ταμπούρου μέσω της περιστροφής των αντίστοιχων κοχλιών. Επιπλέον, το μοντέλο διαθέτει υφές υψηλής ανάλυσης και λεπτομερή γεωμετρική δομή, γεγονός που συμβάλλει στη ρεαλιστική απεικόνιση του οργάνου μέσα στο περιβάλλον της προσομοίωσης.



Εικόνα 20. 3d μοντέλο ταμπούρου (© NRG_3D).

Ωστόσο, όπως και στην περίπτωση του αρπιχορδίου, το μοντέλο δεν ήταν αρχικά σχεδιασμένο για διαδραστική χρήση. Η επάνω μεμβράνη του ταμπούρου ήταν σχεδιασμένη ως ένα ενιαίο γεωμετρικό αντικείμενο, χωρίς διαχωρισμό σε επιμέρους περιοχές που να αντιστοιχούν στα σημεία σύσφιξης.



Εικόνα 21. Βίδες σύσφιξης μεμβράνης (© NRG_3D).

Για να μπορεί κάθε βίδα να επηρεάζει το αντίστοιχο σημείο της μεμβράνης, κρίθηκε απαραίτητο να γίνει επεξεργασία του μοντέλου στο Blender. Συγκεκριμένα, η επιφάνεια της επάνω μεμβράνης χωρίστηκε σε δέκα ίσα κυκλικά τόξα, τα οποία αντιστοιχούν στις θέσεις των δέκα βιδών σύσφιξης. Με τον τρόπο αυτό κάθε κοχλίας μπορεί να συνδεθεί με το αντίστοιχο τμήμα της μεμβράνης.

Η λειτουργικότητα που υλοποιείται στην προσομοίωση του ταμπούρου περιλαμβάνει δύο βασικές ενέργειες:

- το χτύπημα της μεμβράνης με μπαγκέτα σε διαφορετικά σημεία
- την περιστροφή των βιδών σύσφιξης για τη ρύθμιση της τάσης

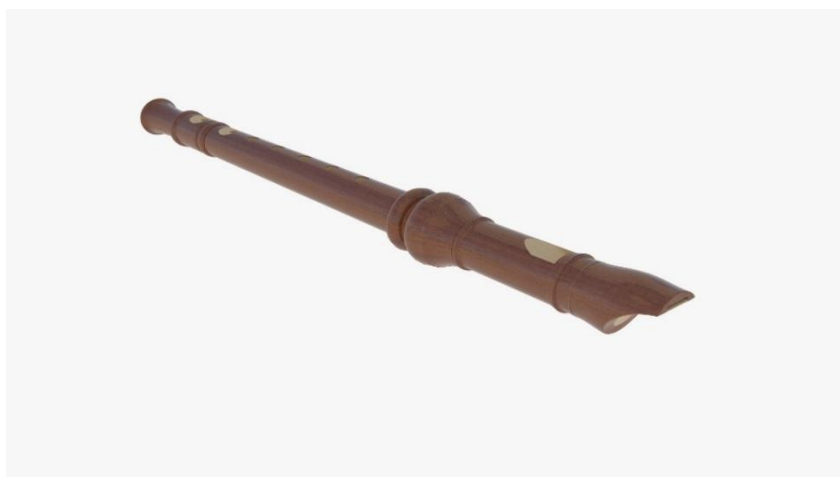
Ο χρήστης μπορεί να χτυπά τη μεμβράνη κοντά σε κάθε ένα από τα δέκα σημεία σύσφιξης ώστε να ελέγχει την απόκριση της μεμβράνης στο συγκεκριμένο σημείο. Στη συνέχεια μπορεί να περιστρέψει την αντίστοιχη βίδα με τη βοήθεια ενός εικονικού εργαλείου, προσομοιώνοντας έτσι τη διαδικασία κουρδίσματος του ταμπούρου. Όπως και στο αρπιχόρδιο, το εργαλείο σύσφιξης των βιδών σχεδιάστηκε με τη βοήθεια της διαδικτυακής υπηρεσίας fast3d.io (Εικόνα 22).



Εικόνα 22. Εργαλείο σύσφιξης βιδών τάσης ταμπούρου.

3.1.5 Μοντέλο φλογέρας

Το τρίτο όργανο που ενσωματώθηκε στην προσομοίωση είναι η φλογέρα. Η επιλογή του συγκεκριμένου μοντέλου (Εικόνα 23) βασίστηκε κυρίως στη δομή του οργάνου. Σε αντίθεση με πολλά άλλα διαθέσιμα μοντέλα, η φλογέρα αυτή ήταν σχεδιασμένη σε τρία ξεχωριστά τμήματα (Εικόνα 24), τα οποία αντιστοιχούν στη δομή πραγματικών φλογερών.



Εικόνα 23. Τρισδιάστατο μοντέλο φλογέρας με ξύλινη υφή (© tgarch3d).



Εικόνα 24. Τα 3 τμήματα της φλογέρας με μαύρη υφή (© tgarch3d).

Η διάταξη αυτή είναι ιδιαίτερα χρήσιμη για την προσομοίωση, καθώς επιτρέπει τη μετακίνηση των επιμέρους τμημάτων του οργάνου και τη μεταβολή του συνολικού μήκους του σωλήνα. Η δυνατότητα αυτή χρησιμοποιείται για την προσομοίωση της διαδικασίας μικρορύθμισης του κουρδίσματος μέσω της μετατόπισης των τμημάτων της φλογέρας. Στη προσομοίωση μας ενδιαφέρουν μόνο δύο από αυτά τα τμήματα, το επιστόμιο και το κυρίως σώμα, οπότε το τρίτο θα παραμένει ακίνητο.

Ωστόσο, το μοντέλο παρουσίαζε ένα μειονέκτημα που σχετίζεται με τις οπές δακτύλων. Στο αρχικό μοντέλο οι οπές αποτελούσαν μέρος της γεωμετρίας του σωλήνα και δεν υπήρχαν ως ξεχωριστά αντικείμενα. Για τις ανάγκες της προσομοίωσης, κρίθηκε απαραίτητο οι οπές να διαχωριστούν σε ανεξάρτητα αντικείμενα ώστε να μπορούν να ενεργοποιούνται ή να απενεργοποιούνται από τον χρήστη. Η διαδικασία αυτή πραγματοποιήθηκε στη Unity, όπου οι περιοχές των οπών επιλέχθηκαν και απέκτησα άλλες ιδιότητες από το υπόλοιπο μοντέλο.

Η λειτουργικότητα που υλοποιείται στη φλογέρα περιλαμβάνει δύο βασικές ενέργειες:

- το άνοιγμα και κλείσιμο των οπών
- τη μετακίνηση των τμημάτων του οργάνου ώστε να μεταβάλλεται το συνολικό μήκος του σωλήνα

Ο χρήστης μπορεί να καλύπτει ή να αποκαλύπτει τις οπές της φλογέρας, μεταβάλλοντας έτσι τη διάταξη των δακτύλων, ενώ παράλληλα μπορεί να μετακινεί τα τμήματα του οργάνου για να επιτύχει λεπτομερή ρύθμιση του μήκους του σωλήνα.

3.2 Ηχητικά δείγματα μουσικών οργάνων

Για την ηχητική αναπαράσταση των τριών μουσικών οργάνων χρησιμοποιήθηκαν προηχογραφημένα ηχητικά δείγματα, τα οποία εισήχθησαν στη Unity και αξιοποιούνται από τα αντίστοιχα AudioSource ως αντικείμενα τύπου AudioClip. Με αυτόν τον τρόπο, η βασική ηχητική αναπαράσταση των οργάνων στηρίζεται σε πραγματικά ηχητικά δείγματα και όχι σε συνθετική παραγωγή ήχου από την εφαρμογή.

Για το αρπικόρδιο χρησιμοποιήθηκαν 60 αρχεία ήχου σε μορφή WAV, τα οποία αντιστοιχούν στις νότες που καλύπτει το τρισδιάστατο πληκτρολόγιο. Τα αρχεία προέρχονται από την πλατφόρμα Freesound και συγκεκριμένα από ηχητικά δείγματα που δημοσιεύθηκαν από τον χρήστη rjcohen. Μετά την εισαγωγή τους στη Unity, τα αρχεία αντιστοιχίστηκαν στις σωστές νότες στο αντίστοιχο script (Harpisichord Database). Κατά τη λειτουργία του κουρδίσματος, το κατάλληλο ηχητικό δείγμα αποτελεί τη βάση για την αναπαραγωγή της νότας, ενώ η συχνότητά του μεταβάλλεται μέσω της ιδιότητας pitch του AudioSource, σύμφωνα με τη συχνότητα που προκύπτει από το φυσικό μοντέλο της χορδής.

Για τη φλογέρα χρησιμοποιήθηκαν αρχεία ήχου σε μορφή MP3, τα οποία λήφθηκαν από τον διαδραστικό πίνακα δακτυλοθεσίας φλογέρας της πλατφόρμας Tomplay. Η συγκεκριμένη σελίδα παρέχει τις διαφορετικές νότες του οργάνου μαζί με τα αντίστοιχα δακτυλοθετήματα. Τα επιλεγμένα αρχεία εισήχθησαν στη Unity ως AudioClip και αντιστοιχίστηκαν στις σωστές νότες της φλογέρας. Κατά το κούρδισμα, το pitch του ηχητικού δείγματος μεταβάλλεται σύμφωνα με τη συχνότητα που υπολογίζεται από το φυσικό μοντέλο της μεταβολής του ενεργού μήκους του αεροσωλήνα.

Τέλος, για το ταμπούρο χρησιμοποιήθηκε ένα ηχητικό δείγμα από τη συλλογή ήχων ταμπούρου της πλατφόρμας Pixabay. Το αρχείο εισάχθηκε στη Unity επίσης ως AudioClip και χρησιμοποιείται για την αναπαραγωγή του ήχου του οργάνου. Όπως και στα υπόλοιπα όργανα, το ηχητικό δείγμα αποτελεί τη βάση της ακουστικής αναπαράστασης, ενώ οι μεταβολές που σχετίζονται με τη διαδικασία κουρδίσματος

εφαρμόζονται προγραμματιστικά σύμφωνα με το φυσικό μοντέλο που υλοποιείται για τη μεμβράνη του ταμπούρου.

3.3 Σύνδεση λειτουργιών με τον κώδικα Onlabs

Για την υλοποίηση των κινήσεων και των αλληλεπιδράσεων χρησιμοποιήθηκαν ορισμένα τμήματα από τον κώδικα του εικονικού εργαστηρίου Onlabs που ήταν ήδη έτοιμα.

Στην παρούσα εργασία αξιοποιούνται τα τμήματα του Onlabs που σχετίζονται με τις παρακάτω λειτουργίες:

- τη πλοήγηση του χρήστη στο χώρο
- την διεπαφή του χρήστη με τα χρησιμοποιούμενα αντικείμενα μέσω του mouse
- τη μετακίνηση αντικειμένων από το χρήστη
- το πάτημα διακοπών (πλήκτρων στην παρούσα εργασία)
- την περιστροφική και μεταφορική κίνηση αντικειμένων
- το συνδυασμό αντικειμένων
- την επιλογή από κατάσταση πλοήγησης σε κατάσταση μεγέθυνσης (ζουμαρίσματος) αντικειμένων
- το άνοιγμα βοηθητικής οθόνης με πληροφορίες

Ο κώδικας αυτός τροποποιήθηκε αρκετά και κατάλληλα ώστε να προσαρμοστεί στις ανάγκες της προσομοίωσης, μέσω προσθήκης κλάσεων και μέσω αλλαγών στην ονοματολογία των αντικειμένων και στη σύνδεσή τους με τα αντίστοιχα στοιχεία των μουσικών οργάνων, για παράδειγμα με κατάλληλες αλλαγές της κλάσης διακόπτη (switch) που εξομοιώνει έναν κλασσικό διακόπτη δύο θέσεων, την προσθήκη μιας κλάσης πλήκτρου (key) που μοιάζει με διακόπτη αλλά επιστρέφει στην αρχική του θέση αφού πατηθεί, κ.λπ.

Είναι σημαντικό να σημειωθεί ότι μέχρι το στάδιο αυτό *δεν υπήρχαν οι λειτουργίες ήχου*, καθώς η ακουστική προσομοίωση είναι πρωτότυπη και δεν συνδέεται με το Onlabs. Επομένως στα πρώτα τμήματα του κώδικα 3.4-3.6 αναλύεται η υλοποίηση των κινήσιων αλληλεπιδράσεων, ο τρόπος δηλαδή με τον οποίο τα αντικείμενα μπορούν να μετακινηθούν, να περιστραφούν ή να ενεργοποιηθούν μέσα στο εικονικό περιβάλλον. Στη συνέχεια αναλύεται η πρωτότυπη υλοποίηση της αρχιτεκτονικής των αντίστοιχων οργάνων με την αντίστοιχη ηχητική επεξεργασία.

Παρακάτω παρουσιάζονται επιλεγμένα αποσπάσματα κώδικα που είναι απαραίτητα για την κατανόηση της λειτουργίας της εφαρμογής. Τα αποσπάσματα δεν αποτελούν το σύνολο του κώδικα, αλλά επιλέχθηκαν επειδή δείχνουν τη βασική λογική της πλοήγησης, της αλληλεπίδρασης με το ποντίκι και της εφαρμογής των μηχανισμών αυτών στα όργανα και στη συνέχεια της αρχιτεκτονικής τους. Το σύνολο των scripts βρίσκεται είναι ωστόσο προσβάσιμο σε δημόσιο αποθετήριο (παρατίθεται στο τέλος της εργασίας), ώστε ο αναγνώστης να έχει τη δυνατότητα, εφόσον το επιθυμεί, να μελετήσει τον πλήρη κώδικα σε συνδυασμό με την ανάλυση που ακολουθεί.

3.4 Σύστημα πλοήγησης και ελέγχου χρήστη

Η κλάση EgoController είναι υπεύθυνη για τη διαχείριση της παρουσίας και της κίνησης του χρήστη στο τρισδιάστατο περιβάλλον της εφαρμογής. Η προσομοίωση χρησιμοποιεί προοπτική πρώτου προσώπου, επιτρέποντας στον χρήστη να περιηγείται στον εικονικό χώρο, να μεταβάλλει την κατεύθυνση παρατήρησης μέσω του ποντικιού και να προσεγγίζει τα μουσικά όργανα και τα επιμέρους διαδραστικά αντικείμενα.

Παράλληλα, η EgoController διαχειρίζεται την τρέχουσα κατάσταση λειτουργίας της εφαρμογής, τη συμπεριφορά του δείκτη του ποντικιού και τη μεταφορά αντικειμένων από τον χρήστη. Για τον σκοπό αυτό ορίζονται οι ακόλουθες καταστάσεις λειτουργίας:

```
public enum Modes
{
    Navigation,
    Harpsichord_Tuning,
    SnareDrum_Tuning,
    Recorder_Tuning
}
```

Η κατάσταση Navigation αντιστοιχεί στην ελεύθερη πλοήγηση του χρήστη στον χώρο, ενώ οι υπόλοιπες καταστάσεις αντιστοιχούν στις λειτουργίες αλληλεπίδρασης με τα τρία μουσικά όργανα. Η τρέχουσα κατάσταση αποθηκεύεται στη μεταβλητή Mode, ενώ η InstrumentOnFocus διατηρεί αναφορά στο μουσικό όργανο με το οποίο αλληλεπιδρά εκείνη τη στιγμή ο χρήστης.

Η αλλαγή κατάστασης πραγματοποιείται μέσω της μεθόδου:

```
public void setMode(  
    Modes _Mode,  
    GameObject _InstrumentOnFocus)  
{  
    CursorFrozen =  
        (_Mode == Modes.Navigation);  
    InstrumentOnFocus =  
        _InstrumentOnFocus;  
  
    Mode =  
        _Mode;  
}
```

Η μέθοδος ενημερώνει τόσο την ενεργή κατάσταση λειτουργίας όσο και το όργανο που βρίσκεται υπό εστίαση. Παράλληλα, προσαρμόζει τη συμπεριφορά του δείκτη του ποντικιού. Στην κατάσταση Navigation η μεταβλητή CursorFrozen λαμβάνει την τιμή true, ώστε το ποντίκι να χρησιμοποιείται για τον έλεγχο της οπτικής κατεύθυνσης. Στις λειτουργίες κουρδίσματος η τιμή γίνεται false, επιτρέποντας την ελεύθερη χρήση του δείκτη για αλληλεπίδραση με τα στοιχεία του αντίστοιχου οργάνου.

3.4.1 Κίνηση του χρήστη

Για τη μετακίνηση στον τρισδιάστατο χώρο χρησιμοποιείται το component CharacterController:

```
public CharacterController controller;
```

Όταν ο χρήστης βρίσκεται σε κατάσταση Navigation, διαβάζονται οι τιμές στους άξονες εισόδου του πληκτρολογίου:

```
move_horizontally =  
    Input.GetAxis("Horizontal");  
  
move_vertically =  
    Input.GetAxis("Vertical");
```

Οι δύο αυτές τιμές χρησιμοποιούνται για τον υπολογισμό του διανύσματος κίνησης:

```
Vector3 move =  
    transform.right * move_horizontally +  
    transform.forward * move_vertically;  
  
controller.Move(  
    move * speed * Time.deltaTime  
);
```

Το `transform.forward` εκφράζει την εμπρόσθια κατεύθυνση του χρήστη, ενώ το `transform.right` την πλευρική του κατεύθυνση. Συνεπώς, η μετακίνηση πραγματοποιείται ως προς τον τρέχοντα προσανατολισμό του χρήστη και όχι ως προς σταθερούς άξονες του καθολικού συστήματος συντεταγμένων.

Η χρήση του `Time.deltaTime` καθιστά τη μετακίνηση ανεξάρτητη από τον ρυθμό ανανέωσης των καρτέ, καθώς η απόσταση που διανύεται προσαρμόζεται στον χρόνο που μεσολάβησε από το προηγούμενο καρτέ.

3.4.2 Περιστροφή και έλεγχος της οπτικής κατεύθυνσης

Η κατεύθυνση παρατήρησης ελέγχεται μέσω της κίνησης του ποντικιού:

```
rotate_horizontally =  
    Input.GetAxis("Mouse X");  
  
rotate_vertically =  
    Input.GetAxis("Mouse Y");
```

Η περιστροφή πραγματοποιείται μόνο όταν το σύστημα `MouseUI` (το οποίο αναλυθεί στην ενότητα 3.5) δεν χρησιμοποιεί ήδη το ποντίκι για την περιστροφή κάποιου διαδραστικού αντικειμένου:

```
if (!MouseUI.Rotating)
{
    float mouseX =
        rotate_horizontally *
        mouseSensitivity *
        Time.deltaTime;

    float mouseY =
        rotate_vertically *
        mouseSensitivity *
        Time.deltaTime;

    xRotation -= mouseY;

    xRotation =
        Mathf.Clamp(
            xRotation,
            -45f,
            45f
        );

    YRotation += mouseX;

    transform.localRotation =
        Quaternion.Euler(
            xRotation,
            YRotation,
            0f
        );
}
```

Η μεταβλητή YRotation ελέγχει την οριζόντια περιστροφή, ενώ η xRotation την κατακόρυφη κατεύθυνση παρατήρησης. Η τελευταία περιορίζεται μεταξύ -45° και 45° μέσω της `Mathf.Clamp()`, ώστε να αποφεύγεται υπερβολική κατακόρυφη περιστροφή.

Η τελική περιστροφή δημιουργείται μέσω της `Quaternion.Euler()`, η οποία μετατρέπει τις συγκεκριμένες γωνίες σε Quaternion και την εφαρμόζει στο `localRotation` του αντικειμένου.

Η συνθήκη `!MouseUI.Rotating` είναι ιδιαίτερα σημαντική για τη συνεργασία των δύο συστημάτων. Όταν ο χρήστης περιστρέφει κάποιο διαδραστικό αντικείμενο, η

κίνηση του ποντικιού αξιοποιείται από το MouseUI και δεν μεταβάλλει ταυτόχρονα την οπτική κατεύθυνση του χρήστη.

3.4.3 Μεταφορά αντικειμένων και χρήση της ενεργής κάμερας

Η κλάση EgoController συμμετέχει στον μηχανισμό μεταφοράς των διαδραστικών αντικειμένων. Καθώς η εφαρμογή μπορεί να χρησιμοποιεί διαφορετικές κάμερες ανάλογα με την τρέχουσα λειτουργία, είναι απαραίτητο το αντικείμενο που μεταφέρει ο χρήστης να συσχετίζεται κάθε φορά με την κάμερα που είναι ενεργή. Για τον σκοπό αυτό χρησιμοποιείται η μέθοδος `getActiveCamera()`:

```
private Camera getActiveCamera()
{
    Camera[] cameras = Camera.allCameras;

    foreach (Camera cam in cameras)
    {
        if (cam.enabled &&
            cam.gameObject.activeInHierarchy)
            return cam;
    }

    return null;
}
```

Η μέθοδος ανακτά τις διαθέσιμες κάμερες μέσω της `Camera.allCameras` και εξετάζει διαδοχικά αν καθεμία είναι ενεργοποιημένη (`enabled`) και αν το αντίστοιχο `GameObject` είναι ενεργό στην ιεραρχία της σκηνής (`activeInHierarchy`). Όταν εντοπιστεί η ενεργή κάμερα, αυτή επιστρέφεται και μπορεί να χρησιμοποιηθεί από τον μηχανισμό μεταφοράς.

Όταν ο χρήστης συλλέξει ένα μετακινήσιμο αντικείμενο, καλείται η μέθοδος `attach()`:

```
public void attach(GameObject _object)
{
    Camera activeCamera =
        getActiveCamera();

    _object.transform.SetParent(
        activeCamera.transform,
        true
    );

    _object.GetComponent<MovableObject>()
        .setCarryingPose();
}
```

Αρχικά εντοπίζεται η ενεργή κάμερα και, εφόσον αυτή υπάρχει, το μεταφερόμενο αντικείμενο ορίζεται ως παιδί (child) του Transform της μέσω της SetParent(). Ως αποτέλεσμα, το αντικείμενο ακολουθεί την κίνηση και τον προσανατολισμό της κάμερας. Στη συνέχεια καλείται η setCarryingPose() της κλάσης MovableObject, η οποία αναλαμβάνει να προσαρμόσει τη θέση, την περιστροφή και την κλίμακα του αντικειμένου στην κατάλληλη στάση μεταφοράς. Με τον τρόπο αυτό, η EgoController αναλαμβάνει τη σύνδεση του αντικειμένου με την τρέχουσα οπτική του χρήστη, ενώ η ειδικότερη διαμόρφωση της στάσης μεταφοράς υλοποιείται από τη MovableObject.

Η συμπεριφορά του μεταφερόμενου αντικειμένου διαφοροποιείται όταν ο δείκτης του ποντικιού είναι ελεύθερος. Στην περίπτωση αυτή, η θέση του αντικειμένου ενημερώνεται συνεχώς μέσα στην Update():

```
if (MouseUI.ObjectBeingCarried != null &&  
    !CursorFrozen)  
{  
    MovableObject movable =  
        MouseUI.ObjectBeingCarried  
            .GetComponent<MovableObject>();  
  
    if (movable != null)  
    {  
        Camera activeCamera = getActiveCamera();  
  
        movable.followMouse(  
            activeCamera  
        );  
    }  
}
```

Αρχικά ελέγχεται αν η μεταβλητή `MouseUI.ObjectBeingCarried` αντιστοιχεί σε κάποιο αντικείμενο καθώς και αν ο δείκτης είναι ελεύθερος, δηλαδή αν η `CursorFrozen` έχει τιμή `false`. Στη συνέχεια ανακτάται το component `MovableObject` του αντικειμένου και εντοπίζεται η ενεργή κάμερα. Η κάμερα αυτή μεταβιβάζεται στη μέθοδο `followMouse()`, η οποία αναλαμβάνει την ενημέρωση της θέσης του αντικειμένου σε σχέση με τη θέση του δείκτη του ποντικιού.

Επομένως, η ενεργή κάμερα αποτελεί το κοινό σημείο αναφοράς του μηχανισμού μεταφοράς. Κατά την αρχική συλλογή, το αντικείμενο προσαρτάται σε αυτή μέσω της `attach()`, ενώ όταν ο δείκτης είναι ελεύθερος η ίδια οπτική αναφορά χρησιμοποιείται από τη `followMouse()` για την προσαρμογή της θέσης του αντικειμένου. Έτσι, ο μηχανισμός μεταφοράς μπορεί να λειτουργεί με ενιαίο τρόπο ανεξάρτητα από την κάμερα που χρησιμοποιείται στην εκάστοτε κατάσταση λειτουργίας της εφαρμογής.

3.5 Σύστημα αλληλεπίδρασης χρήστη και διαχείρισης ποντικιού

Η κλάση `MouseUI` αποτελεί το βασικό επίπεδο διαχείρισης της αλληλεπίδρασης του χρήστη με τα αντικείμενα της τρισδιάστατης σκηνής. Προσαρτάται ως component στα αντικείμενα που μπορούν να δεχθούν ενέργειες μέσω του ποντικιού και αναλαμβάνει την αναγνώριση του αντικειμένου, τον προσδιορισμό του διαθέσιμου τύπου αλληλεπίδρασης και τη μεταβίβαση της ενέργειας στην αντίστοιχη κλάση.

Η MouseUI δεν υλοποιεί η ίδια την ειδική συμπεριφορά ενός πλήκτρου, ενός εργαλείου ή ενός περιστρεφόμενου εξαρτήματος. Αντίθετα, λειτουργεί ως ενδιάμεσο επίπεδο μεταξύ της εισόδου του χρήστη και των κλάσεων που κληρονομούν από την `InteractableObject`. Με αυτόν τον τρόπο, ο γενικός μηχανισμός εισόδου παραμένει κοινός, ενώ η πραγματική συμπεριφορά καθορίζεται από το εκάστοτε διαδραστικό αντικείμενο.

3.5.1 Κατηγοριοποίηση αντικειμένων και τύποι αλληλεπίδρασης

Τα διαδραστικά αντικείμενα διακρίνονται αρχικά μέσω της απαρίθμησης `Labels`, η οποία περιλαμβάνει τις διαφορετικές κατηγορίες αντικειμένων της εφαρμογής:

```
public enum Labels
{
    NonInteractable,
    HarpsichordKey,
    HarpsichordDummyKey,
    HarpsichordTuningPanel,
    HarpsichordRegulatorPin,
    HarpsichordTuningHammer,
    Recorder,
    RecorderHole,
    RecorderMouthpiece,
    SnareDrumTuningPanel,
    SnareDrumLug,
    SnareDrumSlice,
    SnareDrumTuningKey
}
```

Η κατηγοριοποίηση αυτή επιτρέπει στη MouseUI να αναγνωρίζει τον γενικό τύπο κάθε αντικειμένου. Οι ειδικές κατηγορίες που αφορούν τα επιμέρους μουσικά όργανα παρουσιάζονται αναλυτικότερα στις αντίστοιχες ενότητες.

Παράλληλα, η μορφή της αλληλεπίδρασης που υποστηρίζει κάθε αντικείμενο περιγράφεται μέσω πέντε βασικών λογικών μεταβλητών:

```
public bool Zoomable,  
        Pressable,  
        Rotatable,  
        Movable,  
        Receptable;
```

Η Zoomable χαρακτηρίζει αντικείμενα μέσω των οποίων μπορεί να ενεργοποιηθεί ειδική προβολή (λειτουργία εστίασης). Η Pressable χρησιμοποιείται για αντικείμενα που αντιδρούν σε πάτημα, ενώ η Rotatable χαρακτηρίζει στοιχεία που μπορούν να περιστραφούν με την κίνηση του ποντικιού. Η Movable χρησιμοποιείται για αντικείμενα που μπορούν να συλλεχθούν και να μεταφερθούν από τον χρήστη, ενώ η Receptable δηλώνει ότι ένα αντικείμενο μπορεί να δεχθεί τη χρήση κάποιου άλλου αντικειμένου που ήδη μεταφέρεται.

Με αυτόν τον τρόπο, η MouseUI δεν χρειάζεται να διαθέτει ξεχωριστή διαδικασία χειρισμού για κάθε συγκεκριμένο αντικείμενο. Αρκεί να γνωρίζει ποιες από τις παραπάνω ιδιότητες είναι ενεργές και να προωθήσει την ενέργεια στην κατάλληλη μέθοδο του αντίστοιχου InteractableObject.

3.5.2 Εντοπισμός αντικειμένου και έναρξη αλληλεπίδρασης

Η αναγνώριση ενός αντικειμένου κάτω από τον δείκτη πραγματοποιείται κυρίως μέσω της OnMouseOver(), η οποία καλείται από τη Unity όταν ο δείκτης του ποντικιού βρίσκεται πάνω από τον collider του αντικειμένου.

Στο στάδιο αυτό ελέγχεται αν το αντικείμενο είναι διαθέσιμο για αλληλεπίδραση και αν βρίσκεται σε κατάλληλη απόσταση από τον χρήστη. Για τον υπολογισμό της απόστασης λαμβάνεται υπόψη η ενεργή κάμερα:

```
Camera activeCamera =  
    getActiveCamera();  
  
VectorDistance_from_Camera =  
    transform.position -  
    activeCamera.transform.position;  
  
if (VectorDistance_from_Camera.magnitude <  
    MinimumDistance_for_Interaction)  
{  
    ...  
}
```

Η χρήση της ενεργής κάμερας είναι σημαντική, καθώς η εφαρμογή μπορεί να χρησιμοποιεί διαφορετική κάμερα ανάλογα με την τρέχουσα λειτουργία. Επομένως, η δυνατότητα αλληλεπίδρασης δεν εξαρτάται από μία σταθερή κάμερα αλλά από εκείνη που χρησιμοποιείται τη συγκεκριμένη χρονική στιγμή.

Όταν πληρούνται οι προϋποθέσεις, η MouseUI αποθηκεύει το αντικείμενο που βρίσκεται κάτω από τον δείκτη του ποντικιού και παρέχει την αντίστοιχη οπτική ανατροφοδότηση στον χρήστη, όπως την εμφάνιση του tooltip και την προσαρμογή του κέρσορα ανάλογα με τη διαθέσιμη ενέργεια. Αντίστοιχα, με την OnMouseExit(), όταν ο δείκτης βγει από την περιοχή του collider, επιστρέφει σε κατάσταση μη αλληλεπίδρασης.

Η έναρξη της ενέργειας πραγματοποιείται με την OnMouseDown(). Η μέθοδος εξετάζει τις ιδιότητες του αντικειμένου και ενεργοποιεί την κατάλληλη συμπεριφορά. Για παράδειγμα, σε ένα πιέσιμο αντικείμενο καλείται η press(), ενώ σε ένα αντικείμενο εστίασης καλείται η zoom():

```
if (Pressable)
{
    GetComponent<InteractableObject>()
        .press();
}
else if (Zoomable)
{
    GetComponent<InteractableObject>()
        .zoom();
}
```

Η MouseUI επομένως αναγνωρίζει ποια ενέργεια ζητά ο χρήστης, αλλά δεν καθορίζει τι αποτέλεσμα θα έχει αυτή η ενέργεια. Το αποτέλεσμα καθορίζεται από την εξειδικευμένη κλάση του αντικειμένου μέσω του πολυμορφισμού της InteractableObject.

3.5.3 Πάτημα, περιστροφή και ολοκλήρωση της αλληλεπίδρασης

Για τα αντικείμενα που χαρακτηρίζονται ως Rotatable, η περιστροφή πραγματοποιείται όσο ο χρήστης κρατά πατημένο το ποντίκι και το μετακινεί. Η OnMouseDown() διαβάζει τη μεταβολή της θέσης του ποντικιού και τη μεταβιβάζει στο περιστρεφόμενο αντικείμενο:

```
if (Rotatable &&
    ObjectMouseIsDownOn == gameObject)
{
    Rotating = true;

    float mouse_dx = Input.GetAxis("Mouse X");

    float mouse_dy = Input.GetAxis("Mouse Y");

    GetComponent<RotatableObject>().rotate(
        new Vector2(
            mouse_dx,
            -mouse_dy));
}
```

Η MouseUI δεν υπολογίζει εδώ τη νέα γωνία του αντικειμένου. Μεταφέρει τη μεταβολή του ποντικιού στη μέθοδο rotate(), ενώ ο τρόπος με τον οποίο η μεταβολή

αυτή μετατρέπεται σε πραγματική περιστροφή αποτελεί ευθύνη των κλάσεων περιστρεφόμενων αντικειμένων.

Κατά τη διάρκεια της διαδικασίας, η στατική μεταβλητή `Rotating` λαμβάνει την τιμή `true`. Η μεταβλητή αυτή χρησιμοποιείται και από την `EgoController`, η οποία, όπως παρουσιάστηκε προηγουμένως, δεν μεταβάλλει την οπτική γωνία του χρήστη όσο βρίσκεται σε εξέλιξη περιστροφή αντικειμένου. Έτσι αποφεύγεται η ταυτόχρονη χρήση της ίδιας κίνησης του ποντικιού για δύο διαφορετικές λειτουργίες.

Η ολοκλήρωση της αλληλεπίδρασης πραγματοποιείται μέσω της `OnMouseDown()`. Για τα πιέσιμα αντικείμενα καλείται η `release()`, ενώ για τα περιστρεφόμενα αντικείμενα καλείται η `doneRotating()`:

```
if (Pressable)
{
    GetComponent<InteractableObject>()
        .release();
}

if (Rotating)
{
    GetComponent<InteractableObject>()
        .doneRotating();

    Rotating = false;
}
```

Με αυτόν τον τρόπο η `MouseUI` διαχειρίζεται ολόκληρο τον κύκλο μιας βασικής αλληλεπίδρασης: εντοπισμό του αντικειμένου, έναρξη της ενέργειας, συνεχή ενημέρωσή της όπου απαιτείται και ολοκλήρωσή της με την απελευθέρωση του ποντικιού.

3.5.4 Συλλογή, μεταφορά και συνδυαστική χρήση αντικειμένων

Εκτός από τις απλές ενέργειες πατήματος και περιστροφής, η `MouseUI` διαχειρίζεται και αντικείμενα που μπορούν να μεταφερθούν. Όταν ένα `Movable` αντικείμενο συλλεχθεί, προετοιμάζεται για μεταφορά και στη συνέχεια καλείται η `attach()` της `EgoController`:

```
GetComponent<MovableObject>()  
    .prepareForCarrying();  
  
Ego.GetComponent<EgoController>()  
    .attach(gameObject);  
  
ObjectBeingCarried =  
    gameObject;
```

Η στατική μεταβλητή `ObjectBeingCarried` διατηρεί αναφορά στο αντικείμενο που μεταφέρεται τη συγκεκριμένη στιγμή. Η πραγματική τοποθέτησή του σε σχέση με την ενεργή κάμερα πραγματοποιείται από την `EgoController`, ενώ η ειδικότερη συμπεριφορά μεταφοράς καθορίζεται από τη `MovableObject`. Συνεπώς, η `MouseUI` είναι υπεύθυνη κυρίως για την κατάσταση της αλληλεπίδρασης, δηλαδή για το ποιο αντικείμενο έχει συλλεχθεί και τότε αυτό χρησιμοποιείται ή αποτίθεται.

Η μεταφορά συνδέεται και με τη δυνατότητα συνδυαστικής χρήσης αντικειμένων. Όταν ο χρήστης μεταφέρει ένα αντικείμενο και αλληλεπιδρά με κάποιο άλλο που χαρακτηρίζεται ως `Receptable`, μπορεί να κληθεί η `use_with()`:

```
Values_After_JointUse ResultValues =  
    await GetComponent<InteractableObject>()  
        .use_with(ObjectBeingCarried);
```

Η `use_with()` επιστρέφει πληροφορίες σχετικά με το αποτέλεσμα της κοινής χρήσης των δύο αντικειμένων. Η `MouseUI` χρησιμοποιεί το αποτέλεσμα αυτό για να ενημερώσει την κατάσταση των αντικειμένων, όπως το αν παραμένουν διαδραστικά, αν το μεταφερόμενο αντικείμενο τοποθετήθηκε στο αντικείμενο-υποδοχέα ή αν πρέπει να συνεχίσει να μεταφέρεται.

Ο μηχανισμός αυτός αποτελεί τη γενική βάση για αλληλεπιδράσεις στις οποίες ένα εργαλείο χρησιμοποιείται επάνω σε ένα άλλο αντικείμενο. Οι συγκεκριμένες εφαρμογές του στα μουσικά όργανα παρουσιάζονται στις αντίστοιχες ενότητες και επομένως δεν είναι απαραίτητο να αναλυθούν στο επίπεδο της `MouseUI`.

Τέλος, όταν ένα αντικείμενο πρέπει να αφεθεί ξανά στον τρισδιάστατο χώρο, η `MouseUI` μπορεί να προσδιορίσει το σημείο απόθεσης μέσω `Raycast` από την ενεργή κάμερα προς τη θέση του δείκτη:

```
Ray ray =  
    activeCamera.ScreenPointToRay(  
        Input.mousePosition);  
  
Physics.Raycast(  
    ray,  
    out hit  
);
```

Το δε σημείο πρόσκρουσης της ακτίνας χρησιμοποιείται για τον προσδιορισμό της θέσης στην οποία μπορεί να τοποθετηθεί το αντικείμενο.

3.6 Γενικές κλάσεις διαδραστικών αντικειμένων

Το σύστημα αλληλεπίδρασης της εφαρμογής βασίζεται σε ένα σύνολο γενικών κλάσεων, οι οποίες περιγράφουν κοινές συμπεριφορές των αντικειμένων της σκηνής. Σκοπός της δομής αυτής είναι οι βασικοί μηχανισμοί, όπως το πάτημα, η περιστροφή και η μεταφορά ενός αντικειμένου, να υλοποιούνται σε γενικό επίπεδο και στη συνέχεια να εξειδικεύονται ανάλογα με τις ανάγκες κάθε μουσικού οργάνου.

Στη βάση της ιεραρχίας βρίσκεται η *InteractableObject*. Από αυτήν προκύπτουν γενικότερες κατηγορίες αντικειμένων, όπως οι *RotatableObject*, *MovableObject* και *Key*. Για τις περιπτώσεις όπου η περιστροφή πρέπει να υπακούει σε συγκεκριμένους μηχανικούς περιορισμούς χρησιμοποιείται επιπλέον η *LimitedRotatableObject*, η οποία αποτελεί εξειδίκευση της *RotatableObject*.

Με αυτή τη δομή, η *MouseUI* που παρουσιάστηκε στην προηγούμενη ενότητα διαχειρίζεται την είσοδο του χρήστη, ενώ οι παρακάτω κλάσεις καθορίζουν τη συμπεριφορά που αντιστοιχεί στην εκάστοτε ενέργεια.

3.6.1 Βασική κλάση διαδραστικών αντικειμένων

Η *InteractableObject* αποτελεί τη βασική κλάση της ιεραρχίας των διαδραστικών αντικειμένων:

```
public class InteractableObject : MonoBehaviour
{
    public virtual string getTooltipName() { return ""; }
    public virtual void press() { }
    public virtual void zoom() { }
    public virtual void rotate(Vector2 _dv) { }
    public virtual void doneRotating() { }
    public virtual void release() { }

    public virtual async Task<Values_After_JointUse>
        use_with(GameObject _OtherObject)
    {
        Values_After_JointUse ReturnValues =
            new Values_After_JointUse(false);

        return ReturnValues;
    }
    public virtual void evacuate(GameObject _object) { }
}
```

Η εν λόγω κλάση δεν καθορίζει συγκεκριμένη συμπεριφορά για τις περισσότερες από αυτές τις ενέργειες. Αντίθετα, ορίζει ένα κοινό σύνολο εικονικών (virtual) μεθόδων, τις οποίες μπορούν να επαναπροσδιορίσουν (με override) οι εξειδικευμένες κλάσεις.

Οι `press()` και `release()` αντιστοιχούν στην έναρξη και ολοκλήρωση μιας ενέργειας πίεσης, η `rotate()` στην περιστροφή μέσω του ποντικιού και η `doneRotating()` στην ολοκλήρωση της τελευταίας. Η `zoom()` χρησιμοποιείται για αντικείμενα που ενεργοποιούν ειδική προβολή, ενώ η `getTooltipName()` επιτρέπει σε ένα αντικείμενο να παρέχει εξειδικευμένο κείμενο στο σύστημα tooltip.

Παράλληλα, η `use_with()` παρέχει τον γενικό μηχανισμό συνδυαστικής χρήσης δύο αντικειμένων. Η βασική της υλοποίηση επιστρέφει αποτέλεσμα που δηλώνει ότι δεν πραγματοποιήθηκε κοινή χρήση. Οι εξειδικευμένες κλάσεις μπορούν να την επαναπροσδιορίσουν όταν απαιτείται πραγματική αλληλεπίδραση μεταξύ ενός μεταφερόμενου αντικειμένου και ενός αντικειμένου-υποδοχέα. Αντίστοιχα, η `evacuate()` παρέχει σημείο επέκτασης για τις περιπτώσεις όπου ένα αντικείμενο αφαιρείται από κάποια θέση ή υποδοχή.

Επομένως, η `InteractableObject` λειτουργεί ως το κοινό επίπεδο επικοινωνίας μεταξύ της `MouseUI` και των διαφορετικών τύπων διαδραστικών αντικειμένων.

3.6.2 Γενική κλάση περιστρεφόμενων αντικειμένων

Για τα αντικείμενα που περιστρέφονται μέσω της κίνησης του ποντικιού δημιουργήθηκε η `RotatableObject`, η οποία κληρονομεί από την `InteractableObject`:

```
public class RotatableObject : InteractableObject
{
    public Axes MouseMovementAxis = Axes.X_Axis;
    public float RotationSensitivity = 0.2f;
    public int RotationDirection = 1;

    protected Quaternion StartingRotation;
}
```

Η κλάση περιλαμβάνει τις γενικές παραμέτρους που απαιτούνται για τη μετατροπή της κίνησης του ποντικιού σε περιστροφική κίνηση. Η `MouseMovementAxis` καθορίζει ποιος άξονας κίνησης του ποντικιού χρησιμοποιείται, η `RotationSensitivity` καθορίζει την ευαισθησία της περιστροφής και η `RotationDirection` καθορίζει αν έχει τη φορά περιστροφής των δεικτών του ρολογιού ή την αντίστροφη.

Κατά την αρχικοποίηση αποθηκεύεται ο αρχικός προσανατολισμός του αντικειμένου:

```
public override void Start()
{
    base.Start();
    StartingRotation = transform.rotation;
}
```

Η βασική επεξεργασία της εισόδου πραγματοποιείται στη `rotate()`:

```
public override void rotate(Vector2 mouseMovement)
{
    float movement;

    if (MouseMovementAxis == Axes.X_Axis)
        movement = mouseMovement.x;
    else
        movement = mouseMovement.y;

    movement *= RotationDirection;

    float requestedRotation =
        movement * RotationSensitivity;

    applyRotation(requestedRotation);
}
```

Η μέθοδος επιλέγει την κατάλληλη συνιστώσα της κίνησης του ποντικιού και την μετατρέπει σε ζητούμενη μεταβολή γωνίας. Η μεταβολή αυτή μεταβιβάζεται στην `applyRotation()`, η οποία εξειδικεύεται από τις αντίστοιχες υποκλάσεις.

Η `setRotation()` αναλαμβάνει την πραγματική εφαρμογή της συνολικής γωνίας περιστροφής σε σχέση με τον αρχικό προσανατολισμό:

```
public virtual void setRotation(float totalAngle)
{
    transform.rotation =
        Quaternion.AngleAxis(
            totalAngle,
            Vector3.up)
        * StartingRotation;
}
```

Έτσι διαχωρίζεται η ανάγνωση της κίνησης του ποντικιού από τους κανόνες που καθορίζουν πόσο επιτρέπεται τελικά να περιστραφεί το αντικείμενο.

3.6.3 Περιστροφή με μηχανικούς περιορισμούς

Η κλάση `LimitedRotatableObject` αποτελεί εξειδίκευση της `RotatableObject` για αντικείμενα των οποίων η περιστροφή πρέπει να περιορίζεται ως προς τη συνολική γωνία περιστροφής:

```
public class LimitedRotatableObject :  
    RotatableObject  
{  
    public float MinimumAngle = -60f;  
    public float MaximumAngle = 60f;  
  
    public float MaximumRotationSpeed = 20f;  
  
    protected float CurrentAngle = 0f;  
}
```

Οι `MinimumAngle` και `MaximumAngle` ορίζουν τα μηχανικά όρια της κίνησης, ενώ η `MaximumRotationSpeed` καθορίζει τη μέγιστη επιτρεπόμενη γωνιακή μεταβολή ανά μονάδα χρόνου.

Η εφαρμογή των περιορισμών πραγματοποιείται στην `applyRotation()`:

```
float maximumRotationThisFrame =  
    MaximumRotationSpeed * Time.deltaTime;  
  
float allowedRotation =  
    Mathf.Clamp(  
        requestedRotation,  
        -maximumRotationThisFrame,  
        maximumRotationThisFrame);  
  
float targetAngle =  
    CurrentAngle + allowedRotation;  
  
float clampedTargetAngle =  
    Mathf.Clamp(  
        targetAngle,  
        MinimumAngle,  
        MaximumAngle);
```

Αρχικά περιορίζεται η μεταβολή που μπορεί να πραγματοποιηθεί στο συγκεκριμένο frame. Στη συνέχεια υπολογίζεται η νέα επιθυμητή γωνία και περιορίζεται εντός του επιτρεπόμενου εύρους. Η τελική τιμή αποθηκεύεται στη `CurrentAngle` και εφαρμόζεται μέσω της `setRotation()`.

Μετά την πραγματική περιστροφή καλείται η `afterRotation()` με όρισμα την τιμή της μεταβλητής `actualRotation` η οποία εκφράζει την πραγματική μεταβολή

της γωνίας περιστροφής που πραγματοποιήθηκε στο συγκεκριμένο frame, μετά την εφαρμογή των περιορισμών ταχύτητας και των επιτρεπόμενων ορίων περιστροφής:

```
afterRotation(actualRotation);
```

Η `afterRotation()` είναι virtual κλάση και επιτρέπει στις εξειδικευμένες υποκλάσεις να συνδέσουν την πραγματική μηχανική μεταβολή με κάποιο άλλο αποτέλεσμα της προσομοίωσης. Με αυτόν τον τρόπο η γενική κλάση επιβάλλει τους περιορισμούς της κίνησης, χωρίς να χρειάζεται να γνωρίζει τι αντιπροσωπεύει η περιστροφή για κάθε συγκεκριμένο όργανο.

3.6.4 Γενική κλάση μεταφερόμενων αντικειμένων

Η `MovableObject` χρησιμοποιείται ως γενική βάση για τα αντικείμενα που ο χρήστης μπορεί να συλλέξει και να μεταφέρει εντός της σκηνής.

Για τον σκοπό αυτό αποθηκεύονται διαφορετικές παράμετροι τόσο για τη θέση ηρεμίας όσο και τη θέση μεταφοράς:

```
public Vector3 RestingRotation;  
public float Y_Offset_for_Relocation;  
  
public Vector3 CarryingLocalPosition;  
public Vector3 CarryingLocalRotation;  
public Vector3 CarryingLocalScale;
```

Κατά την αρχικοποίηση αποθηκεύεται επίσης η αρχική κατάσταση του αντικειμένου:

```
OriginalRestingPosition =  
    transform.position;  
  
OriginalRestingRotation =  
    transform.rotation;  
  
OriginalRestingScale =  
    transform.localScale;
```

Έτσι είναι δυνατή η ακριβής επαναφορά του αντικειμένου στην αρχική θέση, περιστροφή και κλίμακά του.

Όταν ξεκινά η μεταφορά, η `prepareForCarrying()` μεταφέρει προσωρινά το αντικείμενο στο layer 'Ignore Raycast':

```
public virtual void prepareForCarrying()
{
    gameObject.layer =
        LayerMask.NameToLayer("Ignore Raycast");

    AttachedToReceptor = false;
}
```

Με τον τρόπο αυτό το ίδιο το μεταφερόμενο αντικείμενο δεν παρεμβάλλεται στους ελέγχους ακτίνας (Raycast) που χρησιμοποιούνται κατά την αλληλεπίδραση.

Η `setCarryingPose()` καθορίζει την προκαθορισμένη σχετική θέση, τη γωνία περιστροφής και την κλίμακα του αντικειμένου που μεταφέρεται:

```
transform.localPosition =
    CarryingLocalPosition;

transform.localEulerAngles =
    CarryingLocalRotation;

transform.localScale =
    CarryingLocalScale;
```

Όταν ο δείκτης του ποντικιού είναι ελεύθερος, η `followMouse()` επιτρέπει στο αντικείμενο να ακολουθεί τη θέση του ποντικιού χρησιμοποιώντας ακτίνα από την ενεργή κάμερα:

```
Ray mouseRay =
    activeCamera.ScreenPointToRay(
        Input.mousePosition);

Vector3 position =
    mouseRay.GetPoint(
        MouseCarryDistance);
```

Η κλάση διαχειρίζεται επίσης την κατάσταση κατά την οποία το αντικείμενο έχει συνδεθεί σε κάποιο άλλο αντικείμενο μέσω της μεταβλητής `AttachedToReceptor`· όταν ο χρήστης μετακινεί το αντικείμενο εντός της σκηνής, η τιμή της `AttachedToReceptor` γίνεται `false`.

3.6.5 Γενική κλάση πλήκτρων

Μια ακόμη γενική κατηγορία αποτελεί η κλάση Key, η οποία κληρονομεί απευθείας από την InteractableObject και περιγράφει τη βασική συμπεριφορά ενός πιέσιμου πλήκτρου:

```
public class Key : InteractableObject
{
    public Axes RotationAxis;

    public float Angle_for_Pressed;
    public float Angle_for_Released;

    public bool Pressed;
}
```

Η κλάση διατηρεί την κατάσταση του πλήκτρου μέσω της Pressed, καθώς και τις γωνίες που αντιστοιχούν στην πατημένη και στην ελεύθερη θέση.

Οι βασικές ενέργειες υλοποιούνται με τις press() και release() και της setAngle() η οποία δέχεται ως όρισμα την επιθυμητή γωνία περιστροφής του πλήκτρου που αντιστοιχεί σε κάθε μία από τις δύο αυτές συναρτήσεις:

```
public override void press()
{
    Pressed = true;
    setAngle(Angle_for_Pressed);
}

public override void release()
{
    Pressed = false;
    setAngle(Angle_for_Released);
}
```

Η Key επομένως υλοποιεί αποκλειστικά τη γενική οπτική και λειτουργική συμπεριφορά ενός πλήκτρου. Η σύνδεση ενός συγκεκριμένου πλήκτρου με ήχο ή άλλα δεδομένα δεν αποτελεί ευθύνη της γενικής κλάσης αλλά των εξειδικεύσεών της, οι οποίες θα παρουσιαστούν στις ενότητες που αφορούν τα αντίστοιχα μουσικά όργανα.

3.6.6 Συμπληρωματικές γενικές κλάσεις του συστήματος

Πέρα από τις βασικές κατηγορίες διαδραστικών αντικειμένων, η εφαρμογή χρησιμοποιεί ορισμένες πρόσθετες γενικές κλάσεις για τη διαχείριση των μουσικών οργάνων, της εστίασης, των εργαλείων και της πρόσθετης διεπαφής με το χρήστη.

3.6.6.1 Γενική κλάση μουσικού οργάνου

Η κλάση *Instrument* αποτελεί τη γενική βάση των μουσικών οργάνων και διαχειρίζεται κυρίως τη μετάβαση μεταξύ της ελεύθερης πλοήγησης και της ειδικής προβολής ενός οργάνου.

Κάθε όργανο διαθέτει αναφορά στην κύρια και στην ειδική κάμερα:

```
public Camera MainCamera;  
public Camera FocusCamera;  
  
public Modes Focus_Mode;  
public bool View;
```

Η *toggleView()* εναλλάσσει τις δύο καταστάσεις, καλώντας τις *lockView()* και *unlockView()*. Κατά την είσοδο στην ειδική προβολή ενημερώνεται η *EgoController*, απενεργοποιείται η κύρια κάμερα και ενεργοποιείται η κάμερα του οργάνου:

```
Ego.GetComponent<EgoController>()  
    .setMode(Focus_Mode, gameObject);  
MainCamera.enabled = false;  
FocusCamera.enabled = true;
```

Αν ο χρήστης μεταφέρει ήδη κάποιο αντικείμενο, αυτό επανασυνδέεται με τη νέα ενεργή κάμερα μέσω της *attach()*. Η αντίστροφη διαδικασία πραγματοποιείται κατά την επιστροφή σε λειτουργία *Navigation*.

Η *Instrument* αποτελεί συνεπώς τον γενικό μηχανισμό εναλλαγής οπτικής λειτουργίας και ενεργής κάμερας για όλα τα μουσικά όργανα.

3.6.6.2 Πλαίσιο ενεργοποίησης ειδικής προβολής

Η κλάση *Panel* αποτελεί απλή εξειδίκευση της *InteractableObject* και συνδέει ένα διαδραστικό σημείο της σκηνής με ένα αντικείμενο της κλάσης *Instrument*:

```
public override void zoom()  
{  
    ControlInstrument  
        .GetComponent<Instrument>()  
        .toggleView();  
}
```

Όταν η MouseUI αναγνωρίσει ότι ένα αντικείμενο είναι Zoomable και καλέσει τη zoom(), η Panel ζητά από το αντίστοιχο όργανο να εναλλάξει τον τρόπο προβολής του από γενικό (δηλαδή σε λειτουργία Navigation) σε ειδικό (δηλαδή στη λειτουργία που αφορά το αντίστοιχο όργανο) ή αντίστροφα. Επομένως, λειτουργεί ως το διαδραστικό σημείο μέσω του οποίου ο χρήστης εισέρχεται στην ειδική προβολή ενός οργάνου ή εξέρχεται από αυτήν.

3.6.6.3 Εναλλαγή λειτουργίας εργαλείου κουρδίσματος

Τα εργαλεία μπορούν να χρησιμοποιηθούν με δύο διαφορετικούς τρόπους: αρχικά ως μεταφερόμενα αντικείμενα και, αφού συνδεθούν σε κατάλληλο σημείο, ως περιστρεφόμενα αντικείμενα. Τη συμπεριφορά αυτή, τη διαχειρίζεται η TuningToolMode.

Μέσω της απαρίθμησης ToolMode, η κλάση ορίζει δύο καταστάσεις:

```
public enum ToolMode  
{  
    Movable,  
    Rotatable  
}
```

Στην κατάσταση Movable «ενεργοποιείται» η MovableObject και η MouseUI χαρακτηρίζει το εργαλείο ως μεταφερόμενο. Στην κατάσταση Rotatable «ενεργοποιείται» αντίστοιχα η RotableObject. Ισχύει δηλαδή ότι:

```
mouseUI.Movable = true;  
mouseUI.Rotatable = false;
```

ή:

```
mouseUI.Movable = false;  
mouseUI.Rotatable = true;
```

Η μετάβαση σε κατάσταση περιστροφής επιτρέπεται μόνο όταν η μεταβλητή `AttachedToReceptor` έχει τιμή `true`, δηλαδή όταν το εργαλείο έχει ήδη συνδεθεί στο κατάλληλο αντικείμενο-υποδοχέα:

```
movableObject.AttachedToReceptor
```

Η `TuningToolMode` λειτουργεί επομένως ως διαχειριστής κατάστασης ενός εργαλείου, επιτρέποντας στο ίδιο `GameObject` να μεταβαίνει μεταξύ των συμπεριφορών μεταφοράς και περιστροφής.

3.6.6.4 Πρόσθετη διεπαφή χρήστη

Η `ExtraUI` αποτελεί μια απλή γενική κλάση για την εμφάνιση ή απόκρυψη πρόσθετων στοιχείων διεπαφής που σχετίζονται με τη λειτουργία ενός οργάνου.

```
public void setActivationStatus(  
    bool _NewActivationStatus)  
{  
    ControlCanvas.SetActive(  
        _NewActivationStatus);  
}
```

Η ενεργοποίησή της συνδέεται με την `Instrument`, ώστε η κατάλληλη διεπαφή (η δισδιάσταση απεικόνιση της οθόνης του ηλεκτρονικού κουρδιστηρίου στη συγκεκριμένη περίπτωση) να εμφανίζεται κατά την ειδική προβολή του οργάνου και να αποκρύπτεται όταν ο χρήστης επιστρέφει σε λειτουργία `Navigation`.

3.6.6.5 Αποτέλεσμα συνδυαστικής χρήσης

Η `Values_After_JointUse` αποτελεί βοηθητική κλάση που χρησιμοποιείται για την επικοινωνία μεταξύ της `use_with()` και της `MouseUI`. Περιέχει πληροφορίες σχετικά με το αν πραγματοποιήθηκε επιτυχώς η συνδυαστική χρήση δύο αντικειμένων καθώς και τη νέα κατάστασή τους:

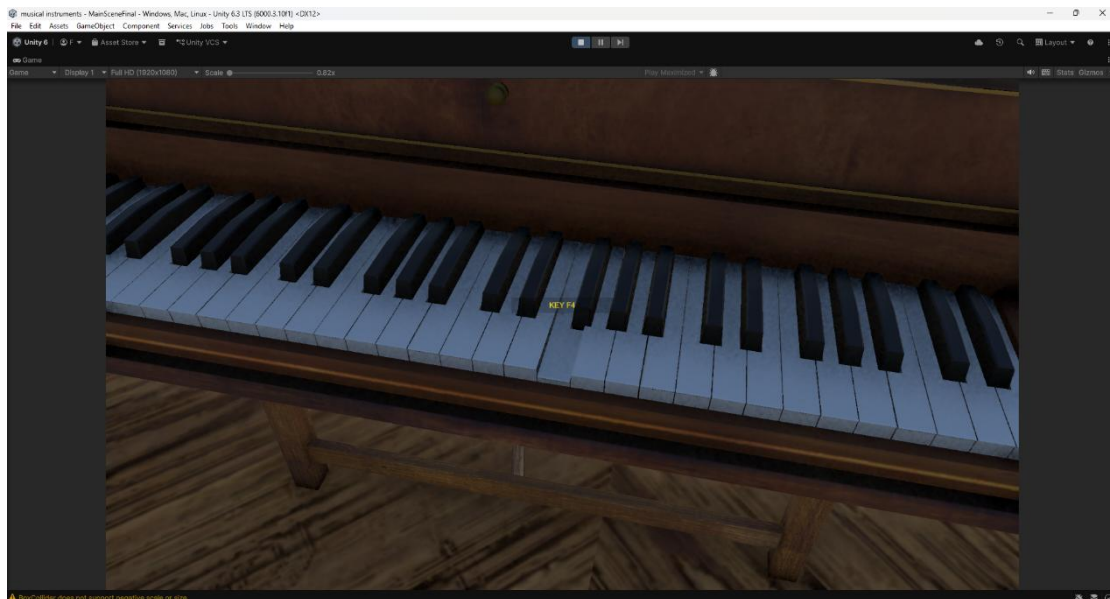
```
public bool JointUse_TookPlace;  
public GameObject ObjectBeingCarried_NewPlace;  
public GameObject Receptor_NewPlace;  
  
public bool ObjectBeingCarried_NewInteractivity;  
public bool Receptor_NewInteractivity;
```

Με αυτόν τον τρόπο, η μέθοδος `use_with()` δεν χρειάζεται να μεταβάλει απευθείας ολόκληρη την κατάσταση του συστήματος αλληλεπίδρασης· αντ' αυτού, επιστρέφει το αποτέλεσμα της ενέργειας και η `MouseUI` χρησιμοποιεί τις πληροφορίες αυτές για να πραγματοποιήσει τις απαιτούμενες αλλαγές.

3.7 Υλοποίηση αρπιχορδίου

Η υλοποίηση του αρπιχορδίου στην εφαρμογή οργανώθηκε σε ένα σύνολο συνεργαζόμενων κλάσεων, καθεμία από τις οποίες αναλαμβάνει διαφορετικό τμήμα της λειτουργίας του εικονικού οργάνου. Η δομή αυτή διαχωρίζει τα δεδομένα των μουσικών νοτών, τη συνολική διαχείριση του οργάνου, τη λειτουργία των πλήκτρων και, όπως θα παρουσιαστεί σε επόμενες ενότητες, τον μηχανισμό κουρδίσματος.

Η `HarpsichordDatabase` αποτελεί τη βάση δεδομένων του οργάνου και περιλαμβάνει για κάθε νότα τη συχνότητα αναφοράς, την ονομασία της και το αντίστοιχο ηχητικό δείγμα. Η `Harpsichord` αποτελεί την κεντρική κλάση του εικονικού οργάνου και συνδέει τα επιμέρους στοιχεία του, διατηρώντας μεταξύ άλλων την αναφορά στη βάση δεδομένων και την αντιστοίχιση των μουσικών νοτών με τα αντίστοιχα πλήκτρα. Κάθε πλήκτρο ελέγχεται από την `HarpsichordKey`, η οποία αποτελεί εξειδίκευση της γενικής κλάσης `Key` και συνδυάζει την οπτική κίνηση του πλήκτρου (Εικόνα 25) με την αναπαραγωγή του αντίστοιχου ήχου. Παράλληλα, η τιμή `pitch` του ήχου μπορεί να μεταβάλλεται από το σύστημα κουρδίσματος, ώστε το ακουστικό αποτέλεσμα να ακολουθεί την τρέχουσα κατάσταση της αντίστοιχης χορδής.



Εικόνα 25. Κρατώντας πατημένο ένα πλήκτρο του αρπιχορδίου.

Ο μηχανισμός κουρδίσματος οργανώνεται γύρω από τις κλάσεις HarpsichordTuningPin, CarryHarpsichordTuningHammer και RotateHarpsichordTuningHammer. Ο πείρος αντιπροσωπεύει την κατάσταση κουρδίσματος μιας συγκεκριμένης χορδής, ενώ το κλειδί κουρδίσματος μπορεί να προσαρτηθεί σε αυτόν και να περιστραφεί από τον χρήστη. Η περιστροφή χρησιμοποιείται στη συνέχεια για τον υπολογισμό της μεταβολής της τάσης και της συχνότητας της χορδής. Η HarpsichordTuningDisplay παρουσιάζει στον χρήστη τη νότα, την τρέχουσα συχνότητα και την κατάσταση κουρδίσματος. Η φυσική και η ακριβής συνεργασία των κλάσεων αυτών αναλύονται στις επόμενες υποενότητες.

3.7.1 Διαχείριση δεδομένων του αρπιχορδίου

Για τη συγκεντρωτική αποθήκευση των μουσικών και ηχητικών δεδομένων του αρπιχορδίου χρησιμοποιείται η κλάση HarpsichordDatabase. Η κλάση υλοποιείται ως ScriptableObject, ώστε τα δεδομένα του οργάνου να αποθηκεύονται σε ανεξάρτητο asset και να είναι διαθέσιμα στις υπόλοιπες κλάσεις χωρίς να ενσωματώνονται ξεχωριστά σε κάθε πλήκτρο.

Αρχικά, οι νότες του οργάνου ορίζονται μέσω της απαρίθμησης HarpsichordNote:

```
public enum HarpsichordNote
{
    F1, FSharp1, G1, GSharp1, A1, ASharp1, B1,

    C2, CSharp2, D2, DSharp2, E2, F2, FSharp2,
    G2, GSharp2, A2, ASharp2, B2,

    ...

    C6, CSharp6, D6, DSharp6, E6
}
```

Η απαρίθμηση περιλαμβάνει το εύρος των νοτών που αντιστοιχεί στα πλήκτρα του τρισδιάστατου μοντέλου. Με τη χρήση της, κάθε πλήκτρο μπορεί να συνδεθεί με μία συγκεκριμένη μουσική νότα μέσω μιας προκαθορισμένης τιμής, χωρίς να απαιτείται η χρήση συμβολοσειρών για την αναγνώρισή του. Η απαρίθμηση HarpsichordNote περιλαμβάνει όλες τις νότες που καλύπτει το εικονικό αρπικόρδιο, από τη νότα F1 έως τη νότα E6. Η χρήση του τύπου enum επιτρέπει την αναφορά σε κάθε νότα με έναν μοναδικό και προκαθορισμένο αναγνωριστικό τύπο, αποφεύγοντας τη χρήση συμβολοσειρών (string) για την ταυτοποίηση των νοτών και περιορίζοντας την πιθανότητα σφαλμάτων κατά τον προγραμματισμό.

Τα δεδομένα που αντιστοιχούν σε κάθε νότα οργανώνονται στην κλάση KeyData:

```
[System.Serializable]
public class KeyData
{
    public HarpsichordNote Note;
    public string DisplayName;
    public float TargetFrequency;
    public AudioClip Sound;
}
```

Κάθε εγγραφή περιλαμβάνει τη νότα (Note), την ονομασία που χρησιμοποιείται για την εμφάνισή της (DisplayName), τη συχνότητα αναφοράς (TargetFrequency) και το αντίστοιχο ηχητικό δείγμα (Sound). Η TargetFrequency αποτελεί τη συχνότητα στην οποία αντιστοιχεί η σωστά κουρδισμένη νότα και χρησιμοποιείται αργότερα ως σημείο αναφοράς από το σύστημα κουρδίσματος.

Τα αντικείμενα `KeyData` αποθηκεύονται σε μία κοινή λίστα μέσα στην `HarpsichordDatabase`:

```
[CreateAssetMenu(  
    fileName = "HarpsichordDatabase",  
    menuName = "Harpsichord/Database")]  
public class HarpsichordDatabase : ScriptableObject  
{  
    public List<KeyData> Keys = new();  
  
    public KeyData GetKeyData(  
        HarpsichordNote note)  
    {  
        return Keys.Find(  
            k => k.Note == note  
        );  
    }  
}
```

Η μέθοδος `GetKeyData()` αναζητά στη λίστα την εγγραφή που αντιστοιχεί στη ζητούμενη `HarpsichordNote`. Επομένως, οι υπόλοιπες κλάσεις δεν χρειάζεται να αποθηκεύουν τοπικά όλα τα δεδομένα μιας νότας. Αρκεί να γνωρίζουν ποια νότα διαχειρίζονται και μπορούν μέσω της βάσης δεδομένων να ανακτήσουν τη συχνότητα αναφοράς, το εμφανιζόμενο όνομα και το κατάλληλο `AudioClip`.

Με αυτόν τον τρόπο η `HarpsichordDatabase` αποτελεί το κοινό επίπεδο δεδομένων του αρπιχορδίου και διαχωρίζει τις μουσικές και ηχητικές πληροφορίες από τις κλάσεις που υλοποιούν τη συμπεριφορά του οργάνου.

3.7.2 Κεντρική διαχείριση του οργάνου

Η κλάση `Harpsichord` αποτελεί την κεντρική κλάση διαχείρισης του εικονικού αρπιχορδίου και κληρονομεί από τη γενική κλάση `Instrument`. Με τον τρόπο αυτό χρησιμοποιεί τον κοινό μηχανισμό μετάβασης από την ελεύθερη πλοήγηση στην ειδική προβολή του οργάνου και αντίστροφα, ενώ παράλληλα προσθέτει τις λειτουργίες που αφορούν αποκλειστικά το αρπιχόρδιο.

Μία από τις βασικές αναφορές της είναι η βάση δεδομένων της:

```
public HarpsichordDatabase Database;
```

Μέσω αυτής, τα επιμέρους στοιχεία του οργάνου έχουν πρόσβαση στα δεδομένα των νοτών.

Κατά την αρχικοποίηση ορίζεται επίσης η κατάσταση λειτουργίας που αντιστοιχεί στο αρπικόρδιο:

```
Focus_Mode =  
    Modes.Harpsichord_Tuning;  
  
base.Start();
```

Έτσι, η γενική λειτουργικότητα της Instrument μπορεί να χρησιμοποιηθεί για την ενεργοποίηση της ειδικής κάμερας και τη μετάβαση του EgoController στη λειτουργία Harpsichord_Tuning.

Μία ακόμη βασική ιδιότητα της Harpsichord είναι η αντιστοίχιση μεταξύ των μουσικών νοτών και των αντικειμένων HarpsichordKey. Για τον σκοπό αυτό χρησιμοποιείται ένα Dictionary:

```
private Dictionary<  
    HarpsichordNote,  
    HarpsichordKey> KeysByNote;
```

Κατά την αρχικοποίηση εντοπίζονται τα HarpsichordKey που βρίσκονται στα παιδιά του αντικειμένου και κάθε πλήκτρο καταχωρίζεται με βάση την τιμή Note που διαθέτει. Η αναζήτηση ενός πλήκτρου μπορεί στη συνέχεια να πραγματοποιηθεί μέσω της μεθόδου GetKey():

```
public HarpsichordKey GetKey(  
    HarpsichordNote note)  
{  
    KeysByNote.TryGetValue(  
        note,  
        out HarpsichordKey key  
    );  
  
    return key;  
}
```

Η αντιστοίχιση αυτή είναι σημαντική για τη συνεργασία των επιμέρους τμημάτων του οργάνου. Ένα άλλο αντικείμενο, όπως για παράδειγμα ένας πείρος κουρδίσματος, δεν χρειάζεται να διαθέτει άμεση αναφορά σε συγκεκριμένο GameObject πλήκτρου.

Γνωρίζοντας τη μουσική νότα που ελέγχει, μπορεί απλά να ζητήσει από την Harpsichord το αντίστοιχο HarpsichordKey.

Η Harpsichord αναλαμβάνει επίσης την κεντρική επικοινωνία με το ψηφιακό κουρδιστήρι. Για τον σκοπό αυτό χρησιμοποιείται η μέθοδος `updateTuningDisplay()`. Η μέθοδος δέχεται ως όρισμα το πλήκτρο που παράγει τον ήχο, ανακτά από την HarpsichordDatabase τη συχνότητα αναφοράς της αντίστοιχης νότας και υπολογίζει την τρέχουσα συχνότητα από την τιμή `TuningPitch` του πλήκτρου. Στη συνέχεια υπολογίζει την απόκλιση σε cents και, μέσω της `GetPin()`, ελέγχει αν ο αντίστοιχος πείρος έχει φτάσει σε κάποιο όριο περιστροφής. Οι τιμές αυτές μεταβιβάζονται τελικά στην HarpsichordTuningDisplay μέσω της `updateDisplay()`. Με αυτή τη δομή, η Harpsichord αποτελεί το μοναδικό σημείο άμεσης επικοινωνίας του συστήματος του οργάνου με την HarpsichordTuningDisplay.

Επομένως, η Harpsichord λειτουργεί ως κεντρικό σημείο σύνδεσης των επιμέρους αντικειμένων του οργάνου. Η HarpsichordDatabase παρέχει τα μουσικά δεδομένα, ενώ η Harpsichord οργανώνει τα πραγματικά αντικείμενα της σκηνης και επιτρέπει στις υπόλοιπες κλάσεις να εντοπίζουν το πλήκτρο που αντιστοιχεί σε κάθε νότα.

3.7.3 Πλήκτρα και παραγωγή ήχου

Η κλάση HarpsichordKey υλοποιεί τη συμπεριφορά κάθε πλήκτρου του αρπιχορδίου και αποτελεί εξειδίκευση της γενικής κλάσης Key:

```
public class HarpsichordKey : Key
{
    public HarpsichordNote Note;

    private AudioSource audioSource;

    public float ReleaseFadeDuration = 0.15f;

    private Coroutine fadeCoroutine;

    private float TuningPitch = 1f;
}
```

Κάθε πλήκτρο διαθέτει μία τιμή Note, μέσω της οποίας καθορίζεται η μουσική νότα που αντιπροσωπεύει. Η κλάση δεν αποθηκεύει το αντίστοιχο ηχητικό δείγμα και τη

συχνότητα αναφοράς τοπικά· αντίθετα, εντοπίζει το γονικό Harpsichord και μέσω αυτού ανακτά τα δεδομένα της νότας από την HarpsichordDatabase:

```
private KeyData getData()  
{  
    Harpsichord h =  
        getHarpsichord();  
  
    return h.Database.GetKeyData(Note);  
}
```

Κατά την αρχικοποίηση καθορίζεται αρχικά η μηχανική κίνηση του πλήκτρου:

```
RotationAxis = Axes.X_Axis;  
  
Angle_for_Released = 0F;  
Angle_for_Pressed = 2.5F;  
  
audioSource =  
    GetComponent<AudioSource>();  
  
base.Start();
```

Επομένως, κατά το πάτημα και την απελευθέρωση ενός πλήκτρου χρησιμοποιείται η γενική λειτουργικότητα της Key, ώστε το πλήκτρο να περιστρέφεται γύρω από τον άξονα x μεταξύ της αρχικής θέσης του και της θέσης των 2.5 μοιρών στην οποία είναι πατημένο.

Η παραγωγή του ήχου πραγματοποιείται στην επαναπροσδιορισμένη μέθοδο `press()` της κλάσης `HarpsichordKey`:

```
public override void press()
{
    base.press();

    KeyData data =
        getData();

    audioSource.volume = 1f;

    audioSource.clip =
        data.Sound;

    audioSource.pitch =
        TuningPitch;

    audioSource.Play();

    Harpsichord harpsichord =
        getHarpsichord();

    harpsichord.updateTuningDisplay(this);
}
```

Αρχικά καλείται η `base.press()`, ώστε να πραγματοποιηθεί η οπτική αναπαράσταση της κίνησης του πλήκτρου από τη μητρική κλάση `Key`. Στη συνέχεια ανακτάται το `KeyData` της αντίστοιχης νότας και το `AudioClip` της βάσης δεδομένων ανατίθεται στο `AudioSource`. Κατόπιν, με την `Play()` ξεκινά η αναπαραγωγή του ηχητικού δείγματος.

Ιδιαίτερη σημασία για τη σύνδεση της αναπαραγωγής με το σύστημα κουρδίσματος έχει η μεταβλητή `TuningPitch`. Πριν από την αναπαραγωγή ανατίθεται η αρχική της τιμή στο λόγο συχνοτήτων `pitch`:

```
audioSource.pitch =
    TuningPitch;
```

που είναι 1, οπότε το ηχητικό δείγμα αναπαράγεται στο αρχικό τονικό του ύψος. Η τιμή αυτή όμως μπορεί να μεταβληθεί μέσω της `setTuningPitch()`:

```
public void setTuningPitch(  
    float pitch)  
{  
    TuningPitch =  
        pitch;  
  
    audioSource.pitch =  
        TuningPitch;  
  
    if (audioSource.isPlaying)  
    {  
        Harpsichord harpsichord =  
            getHarpsichord();  
  
        harpsichord.updateTuningDisplay(  
            this  
        );  
    }  
}
```

Με τον τρόπο αυτό το HarpsichordKey αποτελεί το «τελικό σημείο» στο οποίο η κατάσταση κουρδίσματος μετατρέπεται σε ακουστικό αποτέλεσμα. Ο μηχανισμός κουρδίσματος υπολογίζει τον κατάλληλο λόγο μεταξύ της τρέχουσας και της επιθυμητής συχνότητας και τον μεταφέρει στο πλήκτρο ως τιμή pitch. Αν ο ήχος του πλήκτρου αναπαράγεται ήδη, η setTuningPitch() ζητά παράλληλα από την Harpsichord να ενημερώσει την ένδειξη του ψηφιακού κουρδιστηρίου. Έτσι, κατά την περιστροφή του κλειδιού κουρδίσματος, η ακουστική μεταβολή και η εμφανιζόμενη συχνότητα ενημερώνονται ταυτόχρονα, και κατά συνέπεια, δεν απαιτείται διαφορετικό ηχητικό αρχείο για κάθε δυνατή κατάσταση κουρδίσματος. Το βασικό AudioClip της νότας παραμένει το ίδιο, ενώ το τονικό του ύψος μεταβάλλεται δυναμικά μέσω της ιδιότητας AudioSource.pitch.

Κατά τη λειτουργία κουρδίσματος, η HarpsichordKey διαχειρίζεται επίσης την ελεγχόμενη αναπαραγωγή της χορδής μέσω του πατήματος από το χρήστη του πλήκτρου M στο πληκτρολόγιο. Στη μέθοδο Update() ελέγχεται αρχικά αν το αρπικχόρδιο βρίσκεται στην ειδική λειτουργία κουρδίσματος. Στη συνέχεια, μέσω της Harpsichord.GetPin() εντοπίζεται ο πείρος που αντιστοιχεί στη νότα του συγκεκριμένου πλήκτρου και ελέγχεται αν σε αυτόν έχει προσαρτηθεί το κλειδί

κουρδίσματος. Μόνο στην περίπτωση αυτή επιτρέπεται η αναπαραγωγή του ήχου με το πλήκτρο M. Με το πάτημά του καλείται η `startTuningSound()`, ενώ όσο ο ήχος αναπαράγεται η `HarpsichordKey` ζητά από την `Harpsichord` να ενημερώνει συνεχώς το ψηφιακό κουρδιστήρι μέσω της `updateTuningDisplay()`. Με την απελευθέρωση του πλήκτρου M καλείται η `stopTuningSound()` και σταματά η συνεχής ενημέρωση της οθόνης του κουρδιστηριού.

Κατά την απελευθέρωση του πλήκτρου εκτελείται αρχικά η λειτουργία της γενικής κλάσης `Key`, ώστε το πλήκτρο να επιστρέψει στην αρχική του θέση, και στη συνέχεια ξεκινά η σταδιακή εξασθένηση του ήχου:

```
public override void release()
{
    base.release();

    FadeOutAndStop(
        ReleaseFadeDuration
    );
}
```

Η εξασθένηση του ήχου πραγματοποιείται μέσω της coroutine `FadeOutAndStop()`, σταδιακά μέσω ενός βρόχου `while`, ο οποίος εκτελείται μέχρι να συμπληρωθεί το χρονικό διάστημα που ορίζεται από τη μεταβλητή `fadeDuration`. Σε κάθε καρέ, η μεταβλητή `timer` αυξάνεται κατά τον χρόνο που έχει παρέλθει από το προηγούμενο καρέ (`Time.deltaTime`). Στη συνέχεια, υπολογίζεται ο κανονικοποιημένος χρόνος t ως ο λόγος `timer/fadeDuration`, ο οποίος χρησιμοποιείται για τον υπολογισμό της νέας έντασης σύμφωνα με τη σχέση $V = V_0 \sqrt{1 - t}$ όπου V_0 είναι η αρχική ένταση του ήχου, V η τρέχουσα ένταση και t ο κανονικοποιημένος χρόνος εξασθένησης, ο οποίος μεταβάλλεται από 0 έως 1.

```
while (timer < fadeDuration)
{
    timer +=
        Time.deltaTime;

    float t =
        timer /
        fadeDuration;

    audioSource.volume =
        startVolume *
        Mathf.Sqrt(
            1f - t
        );

    audioSource.volume =
        startVolume *
        Mathf.Sqrt(1f - t);

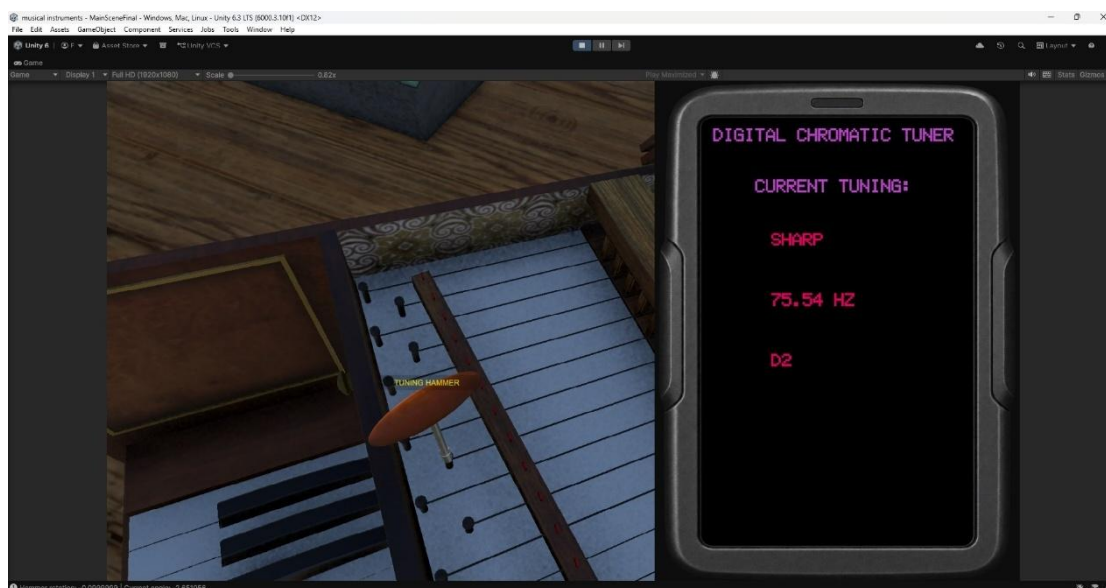
    yield return null;
}
```

Μετά την παρέλευση ενός χρονικού διαστήματος εξασθένησης, το οποίο καθορίζεται από τη μεταβλητή `ReleaseFadeDuration` (0,15 δευτερόλεπτα) επιλεγμένη με αισθητικά κριτήρια, ο ήχος σταματά και η ένταση επανέρχεται στην αρχική τιμή, ώστε το `AudioSource` να είναι έτοιμο για το επόμενο πάτημα. Η εντολή `yield return null` αναστέλλει προσωρινά την εκτέλεση της coroutine μέχρι το επόμενο καρέ, επιτρέποντας τη σταδιακή εξασθένηση του ήχου.

Η `HarpichordKey` συνδέει επομένως τρία διαφορετικά επίπεδα της προσομοίωσης: την οπτική κίνηση του τρισδιάστατου πλήκτρου μέσω της `Key`, τα δεδομένα της μουσικής νότας μέσω της `HarpichordDatabase` και την ακουστική απόδοση μέσω του `AudioSource`. Παράλληλα, η δυνατότητα μεταβολής του `TuningPitch` παρέχει το σημείο σύνδεσης με τον μηχανισμό κουρδίσματος, ο οποίος αναλύεται στη συνέχεια.

3.7.4 Υλοποίηση του μηχανισμού κουρδίσματος του αρπικχορδίου

Ο μηχανισμός κουρδίσματος του αρπικχορδίου υλοποιείται μέσω της συνεργασίας διαφορετικών κλάσεων, καθεμία από τις οποίες αναλαμβάνει ένα συγκεκριμένο μέρος της διαδικασίας. Η βασική λογική της προσομοίωσης είναι ότι ο χρήστης μεταφέρει το κλειδί κουρδίσματος σε έναν πείρο, το προσαρτά σε αυτόν και στη συνέχεια το περιστρέφει μέσω της κίνησης του ποντικιού. Η περιστροφή μεταβάλλει την γωνία του αντίστοιχου πείρου και χρησιμοποιείται για τον υπολογισμό της μεταβολής της τάσης της χορδής. Από τη νέα τάση υπολογίζεται η νέα συχνότητα, η οποία εφαρμόζεται στον ήχο του αντίστοιχου πλήκτρου και μπορεί να ελεγχθεί μέσω του ψηφιακού κουρδιστηρίου. Στην Εικόνα 26 φαίνεται ένα στιγμιότυπο της λειτουργίας αυτής.



Εικόνα 26. Στιγμιότυπο από την κατάσταση κουρδίσματος.

Η συνολική λειτουργία του κουρδίσματος του αρπικχορδίου κατανέμεται κυρίως στις κλάσεις `CarryHarpichordTuningHammer`, `RotateHarpichordTuningHammer`, `HarpichordTuningPin` και `HarpichordTuningDisplay`. Παράλληλα, γίνεται χρήση των γενικών κλάσεων `MovableObject`, `RotatableObject` και `LimitedRotatableObject` που παρουσιάστηκαν προηγουμένως, ενώ η `HarpichordKey` συνδέει το αποτέλεσμα του κουρδίσματος με τον παραγόμενο ήχο. Με αυτόν τον τρόπο διαχωρίζονται στον κώδικα η μεταφορά του εργαλείου, η περιστροφή του, οι υπολογισμοί του φυσικού μοντέλου και η παρουσίαση των αποτελεσμάτων στον χρήστη.

3.7.4.1 Μεταφορά και προσάρτηση του κλειδιού κουρδίσματος

Η κλάση `CarryHarpichordTuningHammer` κληρονομεί από τη `MovableObject` και εξειδικεύει τη γενική λειτουργία μεταφοράς αντικειμένων για το κλειδί κουρδίσματος του αρπιχορδίου. Βασική της αρμοδιότητα είναι να γνωρίζει αν το εργαλείο έχει προσαρτηθεί σε κάποιον πείρο και, εφόσον έχει προσαρτηθεί, σε ποιον συγκεκριμένα.

Για τον σκοπό αυτό χρησιμοποιείται η μεταβλητή `AttachedPin`, τύπου `HarpichordTuningPin`:

```
public HarpichordTuningPin AttachedPin;
```

Η `AttachedPin` αποθηκεύει αναφορά στο αντικείμενο `HarpichordTuningPin` στο οποίο βρίσκεται κάθε στιγμή το κλειδί. Η αναφορά αυτή είναι απαραίτητη, καθώς η περιστροφή του εργαλείου πρέπει αργότερα να εφαρμοστεί στον συγκεκριμένο πείρο και επομένως στη συγκεκριμένη χορδή.

Όταν πραγματοποιείται η προσάρτηση, καλείται η μέθοδος `attachToPin()`:

```
public void attachToPin(  
    HarpichordTuningPin pin)  
{  
    AttachedPin = pin;  
  
    prepareForAttachment();  
  
    if (hammerMode != null)  
    {  
        hammerMode.setMovableMode();  
    }  
}
```

Αρχικά, η παράμετρος `pin` αποθηκεύεται στην `AttachedPin`. Με αυτόν τον τρόπο δημιουργείται στον κώδικα η σύνδεση μεταξύ του εργαλείου και του πείρου που πρόκειται να κουρδιστεί. Στη συνέχεια καλείται η `prepareForAttachment()`, η οποία κληρονομείται από τη `MovableObject` και προετοιμάζει το εργαλείο για την κατάσταση προσάρτησης. Μεταξύ άλλων, η `MovableObject` θέτει την `AttachedToReceptor` σε `true`, ώστε το υπόλοιπο σύστημα να γνωρίζει ότι το εργαλείο δεν μεταφέρεται πλέον ελεύθερα.

Τέλος, μέσω της `TuningToolMode` το εργαλείο τίθεται αρχικά σε κατάσταση `Movable`. Η ίδια κλάση επιτρέπει στη συνέχεια την εναλλαγή του εργαλείου μεταξύ

κατάστασης μετακίνησης και περιστροφής. Επομένως, η φυσική περιστροφή δεν ενεργοποιείται απλώς επειδή ο χρήστης κρατά το κλειδί, αλλά μόνο αφού αυτό έχει τοποθετηθεί σε κατάλληλο πείρο.

Η κλάση περιλαμβάνει επίσης τις τιμές `PinAttachmentRotation` και `PinAttachmentScale`, οι οποίες καθορίζουν την κατάλληλη γωνία περιστροφής και κλίμακα του τρισδιάστατου μοντέλου όταν βρίσκεται πάνω στον πείρο. Οι συγκεκριμένες τιμές αφορούν τη γεωμετρική-οπτική τοποθέτηση του εργαλείου και δεν συμμετέχουν στους υπολογισμούς της φυσικής του κουρδίσματος.

Κατά την απομάκρυνση του εργαλείου καλείται αντίστοιχα η `detachFromPin()`:

```
public void detachFromPin()
{
    AttachedPin = null;

    AttachedToReceptor = false;

    if (hammerMode != null)
    {
        hammerMode.setMovableMode();
    }
}
```

Με την κλήση της `detachFromPin()`, η σύνδεση με τον πείρο καταργείται και το εργαλείο επιστρέφει στην κατάσταση στην οποία μπορεί να μεταφερθεί από τον χρήστη.

3.7.4.2 Περιστροφή του κλειδιού κουρδίσματος

Η περιστροφή του κλειδιού κουρδίσματος υλοποιείται από την κλάση `RotateHarpichordTuningHammer`, η οποία κληρονομεί από τη γενική κλάση `LimitedRotatableObject`. Επομένως, αξιοποιεί τον μηχανισμό περιστροφής που έχει ήδη οριστεί για τα περιστρεφόμενα αντικείμενα της εφαρμογής.

Στη `RotatableObject`, η κίνηση του ποντικιού μετατρέπεται αρχικά σε ζητούμενη γωνιακή μεταβολή:

```
float requestedRotation =  
    movement * RotationSensitivity;  
  
applyRotation(requestedRotation);
```

Η float μεταβλητή `RotationSensitivity` καθορίζει πόσο έντονα μετατρέπεται η κίνηση του ποντικιού σε περιστροφή. Η ζητούμενη μεταβολή περνά στη συνέχεια ως όρισμα στη μέθοδο `applyRotation()` της `LimitedRotatableObject` στην οποία, όπως αναφέρθηκε προηγουμένως, εφαρμόζεται πρώτα περιορισμός στη μέγιστη γωνία που μπορεί να πραγματοποιηθεί μέσα σε ένα frame. Αυτός αποτρέπει μια απότομη κίνηση του ποντικιού να προκαλέσει στιγμιαία πολύ μεγάλη περιστροφή του κλειδιού. Στη συνέχεια υπολογίζεται η νέα συνολική γωνία `targetAngle`:

```
float targetAngle =  
    CurrentAngle +  
    allowedRotation;
```

Η `CurrentAngle` δεν περιγράφει τη μεταβολή του τελευταίου frame αλλά τη συνολική γωνία περιστροφής του εργαλείου. Η νέα τιμή της `targetAngle` περιορίζεται μεταξύ `MinimumAngle` και `MaximumAngle` και αποθηκεύεται στη μεταβλητή `clampedTargetAngle`:

```
float clampedTargetAngle =  
    Mathf.Clamp(  
        targetAngle,  
        MinimumAngle,  
        MaximumAngle  
    );
```

Έπειτα υπολογίζεται η περιστροφή που πραγματοποιήθηκε πραγματικά, `actualRotation`:

```
float actualRotation =  
    clampedTargetAngle -  
    CurrentAngle;  
CurrentAngle =  
    clampedTargetAngle;  
  
setRotation(CurrentAngle);  
afterRotation(actualRotation);
```

Η `setRotation()` ενημερώνει το οπτικό αποτέλεσμα της περιστροφής του τρισδιάστατου εργαλείου, ενώ η `afterRotation()` επιτρέπει στην εξειδικευμένη

RotateHarpsichordTuningHammer να χρησιμοποιήσει την τιμή της νέας γωνίας για το κούρδισμα.

Στην κλάση του κλειδιού η μέθοδος επαναπροσδιορίζεται ως εξής:

```
protected override void afterRotation(  
    float actualRotation)  
{  
    updateTuning();  
  
    checkLimits();  
}
```

Επομένως, κάθε πραγματική περιστροφή του εργαλείου προκαλεί άμεσα επανυπολογισμό της κατάστασης κούρδισματος.

Η μέθοδος updateTuning() ελέγχει αρχικά αν υπάρχει αναφορά στο στοιχείο CarryHarpsichordTuningHammer και αν το κλειδί κούρδισματος είναι συνδεδεμένο με κάποιον πείρο. Αν κάποια από τις δύο προϋποθέσεις δεν ικανοποιείται, η μέθοδος τερματίζεται χωρίς να πραγματοποιηθεί μεταβολή στο κούρδισμα:

```
if (carryHammer == null)  
    return;  
  
if (carryHammer.AttachedPin == null)  
    return;
```

Στη συνέχεια ελέγχεται αν η τρέχουσα γωνία περιστροφής έχει φτάσει σε κάποιο από τα επιτρεπόμενα όρια:

```
bool limitReached =  
    CurrentAngle >= MaximumAngle ||  
    CurrentAngle <= MinimumAngle;
```

Τέλος, καλείται η μέθοδος updateTuningFromAngle() του πείρου στον οποίο είναι προσαρτημένο το κλειδί κούρδισματος, μεταφέροντας την τρέχουσα γωνία περιστροφής και την πληροφορία για το αν έχει επιτευχθεί κάποιο όριο:

```
carryHammer.AttachedPin  
    .updateTuningFromAngle(  
        CurrentAngle,  
        limitReached  
    );
```

Το σημαντικό στοιχείο είναι ότι λαμβάνεται υπόψη η συνολική γωνία περιστροφής CurrentAngle και όχι μόνο η actualRotation που αντιστοιχεί στη μικρή μεταβολή της

γωνίας κατά το τελευταίο frame και χρησιμοποιείται κυρίως για έλεγχο. Έτσι ο πείρος λαμβάνει γνώση της πλήρους γωνιακής κατάστασης του μηχανισμού και με βάση αυτή μπορεί να υπολογιστεί η συνολική μεταβολή του κουρδίσματος.

3.7.4.3 Σύνδεση των ορίων περιστροφής με το εύρος κουρδίσματος

Η `RotateHarpsichordTuningHammer` δεν χρησιμοποιεί αυθαίρετα όρια περιστροφής. Τα όρια του εργαλείου υπολογίζονται με βάση ένα επιτρεπόμενο εύρος κουρδίσματος, το οποίο εκφράζεται σε cents:

```
public float MinimumTuningCents = -400f;
public float MaximumTuningCents = 200f;
```

Για να μετατραπούν τα cents σε αντίστοιχη γωνία χρησιμοποιείται η `calculateRotationLimits()`.

Η σχέση μεταξύ μιας απόκλισης c , εκφρασμένης σε cents, και του λόγου δύο συχνοτήτων f και f_0 δίνεται από τη σχέση:

$$\frac{f}{f_0} = 2^{c/1200} \quad (17)$$

Στο φυσικό μοντέλο που χρησιμοποιείται για την προσομοίωση του κουρδίσματος, η μεταβολή της συχνότητας συνδέεται με την περιστροφή του κλειδιού του κουρδίσματος μέσω της μεταβολής της τάσης της χορδής. Όπως έχουμε δει από τη σχέση (3), η θεμελιώδης συχνότητα μιας χορδής δίνεται από τη σχέση: $f = 1/2 \frac{1}{2L} \sqrt{\frac{T}{\mu}}$, όπου L είναι το ενεργό μήκος της χορδής, T η τάση της και μ η γραμμική της πυκνότητα.

Κατά το κούρδισμα θεωρούνται σταθερά το μήκος και η γραμμική πυκνότητα. Συνεπώς:

$$f \propto \sqrt{T} \quad (18)$$

και συγκρίνοντας τη νέα κατάσταση με την κατάσταση αναφοράς με T_0 και f_0 τις αρχικές τιμές, έχουμε:

$$\frac{f}{f_0} = \sqrt{\frac{T}{T_0}} \quad (19)$$

Θεωρώντας ότι η τάση είναι γραμμική συνάρτηση της γωνίας:

$$\frac{T}{T_0} = 1 + s \cdot \theta \quad (20)$$

όπου $s = \frac{k \cdot r}{T_0}$ (Ενότητα 2.3.2.4) , προκύπτει:

$$\frac{f}{f_0} = \sqrt{1 + s \cdot \theta} \quad (21)$$

όπου s είναι η ευαισθησία κουρδίσματος και θ η γωνία περιστροφής.

Εξισώνοντας τις δύο παραπάνω σχέσεις προκύπτει:

$$2^{c/1200} = \sqrt{1 + s \cdot \theta} \quad (22)$$

και υψώνοντας και τα δύο μέλη στο τετράγωνο έχουμε:

$$2^{2c/1200} = 1 + s \cdot \theta \quad (23)$$

Επομένως, η απαιτούμενη γωνία περιστροφής για δεδομένη απόκλιση c σε cents υπολογίζεται από τη σχέση:

$$\theta = \frac{2^{2c/1200} - 1}{s} \quad (24)$$

Η σχέση αυτή εμφανίζεται άμεσα στον κώδικα:

```
MinimumAngle =
(
    Mathf.Pow(
        2f,
        (2f * MinimumTuningCents)
        / 1200f
    )
    - 1f
)
/ tuningSensitivity;
```

Αντίστοιχος υπολογισμός πραγματοποιείται για τη MaximumAngle.

Με τον τρόπο αυτό τα μηχανικά όρια του εικονικού εργαλείου συνδέονται μαθηματικά με το εύρος συχνοτήτων που επιτρέπεται να προκύψει. Η γωνία περιστροφής, επομένως, αποτελεί την ενδιάμεση μεταβλητή μεταξύ της ενέργειας του χρήστη και της φυσικής κατάστασης της χορδής.

3.7.4.4 Φυσικό μοντέλο του πείρου και της χορδής

Η ουσιαστική προσομοίωση του κουρδίσματος πραγματοποιείται στην `HarpsichordTuningPin`. Η κλάση αντιπροσωπεύει έναν συγκεκριμένο πείρο και συνδέεται με μία συγκεκριμένη νότα του αρπικορδίου μέσω της `HarpsichordNote`.

Όταν η `RotateHarpsichordTuningHammer` μεταφέρει τη νέα συνολική γωνία, καλείται η `updateTuningFromAngle()`:

```
updateTuningFromAngle(  
    CurrentAngle,  
    limitReached  
);
```

Η μέθοδος αποθηκεύει αρχικά τη νέα γωνία κουρδίσματος:

```
CurrentTuningAngle =  
    totalAngle;
```

Από το σημείο αυτό αρχίζει η μετατροπή της μηχανικής περιστροφής σε ακουστικό αποτέλεσμα.

Στον κώδικα ο παραπάνω λόγος των δύο συχνοτήτων υπολογίζεται σε δύο βήματα μέσω του λόγου των τάσεων (για μεγαλύτερη καθαρότητα στην ανάγνωση του κώδικα) ως εξής:

```
float tensionRatio =  
    1f +  
    TuningSensitivity *  
    CurrentTuningAngle;  
float frequencyRatio =  
    Mathf.Sqrt(  
        tensionRatio  
    );
```

και τελικά:

```
CurrentFrequency =  
    TargetFrequency *  
    frequencyRatio;
```

Η `TargetFrequency` είναι η συχνότητα αναφοράς της συγκεκριμένης νότας, ενώ η `CurrentFrequency` είναι η συχνότητα που προκύπτει από την τρέχουσα θέση του πείρου.

Άρα η γωνία χρησιμοποιείται πρώτα για τον υπολογισμό του λόγου τάσεων και ο λόγος τάσεων μετατρέπεται κατόπιν σε λόγο συχνοτήτων σύμφωνα με τη φυσική της παλλόμενης χορδής.

3.7.4.5 Υπολογισμός της απόκλισης σε cents

Μετά τον υπολογισμό της `CurrentFrequency`, η απόκλιση από τη συχνότητα αναφοράς μετατρέπεται σε cents (Εξίσωση 2): $c = 1200 \cdot \log_2 \left(\frac{f}{f_0} \right)$

και υλοποιείται ως εξής:

```
CurrentCents =  
    1200f *  
    Mathf.Log(  
        CurrentFrequency /  
        TargetFrequency,  
        2f  
    );
```

Η χρήση cents είναι ιδιαίτερα κατάλληλη για την εφαρμογή, επειδή εκφράζει τη μουσική απόσταση λογαριθμικά και ανεξάρτητα από την απόλυτη περιοχή συχνοτήτων. Απόκλιση 100 cents αντιστοιχεί σε ένα ημιτόνιο, ενώ απόκλιση 1200 cents αντιστοιχεί σε μία οκτάβα.

Όταν η `CurrentCents` είναι αρνητική, η τρέχουσα συχνότητα βρίσκεται χαμηλότερα από τη συχνότητα αναφοράς. Όταν είναι θετική, βρίσκεται υψηλότερα. Όταν προσεγγίζει το μηδέν, η συχνότητα προσεγγίζει την επιθυμητή τιμή.

3.7.4.6 Εφαρμογή του κουρδίσματος στον παραγόμενο ήχο

Ο υπολογισμός της νέας συχνότητας πρέπει να μεταφερθεί και στον πραγματικό ήχο που αναπαράγεται από το αντίστοιχο πλήκτρο.

Για τον σκοπό αυτό η `HarpsichordTuningPin` εντοπίζει το `HarpsichordKey` που αντιστοιχεί στη νότα του πείρου και καλεί από αυτό την `setTuningPitch()` με όρισμα την τιμή της `frequencyRatio`:

```
key.setTuningPitch(  
    frequencyRatio  
);
```

Η μέθοδος της `HarpsichordKey` αποθηκεύει τον νέο λόγο συχνοτήτων:

```
public void setTuningPitch(  
    float pitch)  
{  
    TuningPitch = pitch;  
  
    audioSource.pitch =  
        TuningPitch;  
}
```

Η τιμή 1 στο `AudioSource.pitch` σημαίνει αναπαραγωγή του αρχικού ηχητικού δείγματος χωρίς μεταβολή του τονικού ύψους. Επομένως, επειδή το `frequencyRatio` εκφράζει ακριβώς τον λόγο f/f_0 , μπορεί να χρησιμοποιηθεί απευθείας ως `pitch` και να ενημερώνεται απευθείας.

Κατά την πίεση του πλήκτρου, η ίδια τιμή εφαρμόζεται ξανά πριν από την αναπαραγωγή:

```
audioSource.clip =  
    data.Sound;  
  
audioSource.pitch =  
    TuningPitch;  
  
audioSource.Play();
```

Έτσι το αρχικό αρχείο WAV εξακολουθεί να αποτελεί την ηχητική πηγή, αλλά το τονικό του ύψος προσαρμόζεται σύμφωνα με την κατάσταση του αντίστοιχου πείρου. Το αποτέλεσμα του φυσικού υπολογισμού γίνεται επομένως άμεσα αντιληπτό ακουστικά από τον χρήστη.

3.7.4.7 Ψηφιακή ένδειξη κουρδίσματος

Η `HarpsichordTuningDisplay` αναλαμβάνει την παρουσίαση της κατάστασης κουρδίσματος. Δεν συμμετέχει στον υπολογισμό της νέας συχνότητας, αλλά δέχεται τα αποτελέσματα του φυσικού μοντέλου και τα μετατρέπει σε κατανοητές πληροφορίες για τον χρήστη.

Η βασική μέθοδος είναι η εξής:

```
public void updateDisplay(  
    string targetNote,  
    float targetFrequency,  
    float currentFrequency,  
    float centsDifference,  
    bool rotationLimitReached)
```

Οι σημαντικότερες τιμές που λαμβάνει είναι η `currentFrequency`, δηλαδή η τρέχουσα συχνότητα της χορδής, και η `centsDifference`, δηλαδή η απόκλιση από τη συχνότητα αναφοράς.

Η `HarpichordTuningPin` δεν επικοινωνεί απευθείας με την `HarpichordTuningDisplay`. Η περιστροφή του κλειδιού μεταβάλλει τη φυσική κατάσταση της χορδής και ενημερώνει το `TuningPitch` του αντίστοιχου `HarpichordKey`. Η ενημέρωση του ψηφιακού κουρδιστηρίου πραγματοποιείται μόνο μέσω της `Harpichord`, όταν αναπαράγεται ο αντίστοιχος ήχος. Με αυτόν τον τρόπο διαχωρίζεται η προσομοιωμένη φυσική κατάσταση της χορδής από τη διαδικασία μέτρησής της από το κουρδιστήρι.

Η συχνότητα εμφανίζεται στη δισδιάσταση απεικόνιση της οθόνης του κουρδιστηρίου με τον παρακάτω:

```
FrequencyText.text =  
    currentFrequency.ToString("F2")  
    + " Hz";
```

ενώ η απόκλιση χρησιμοποιείται για τον χαρακτηρισμό της κατάστασης. Η ανοχή καθορίζεται από τη `float` μεταβλητή `InTuneToleranceCents`:

```
public float InTuneToleranceCents = 5f;
```

και ο σχετικός έλεγχος πραγματοποιείται ως εξής:

```
else if (
    Mathf.Abs(centsDifference)
    <= InTuneToleranceCents)
{
    StatusText.text =
        "IN TUNE";

    setColor(
        Color.green
    );
}
```

Επομένως, όταν η απόλυτη απόκλιση δεν υπερβαίνει τα 5 cents, η χορδή χαρακτηρίζεται ως σωστά κουρδισμένη. Για αρνητικές μεγαλύτερες αποκλίσεις εμφανίζεται FLAT ενώ για θετικές SHARP.

Η κλάση υπολογίζει επίσης ποια νότα βρίσκεται πλησιέστερα στην τρέχουσα συχνότητα. Εφόσον στο ισοσυγκερασμένο σύστημα ένα ημιτόνιο αντιστοιχεί σε 100 cents, χρησιμοποιούνται τα ενδιάμεσα όρια των 50 cents για την αλλαγή της εμφανιζόμενης νότας:

```
int semitoneChange =
    Mathf.FloorToInt(
        (
            centsDifference
            + 50f
        )
        / 100f
    );
```

Με αυτόν τον τρόπο το ψηφιακό κουρδιστήρι δεν εμφανίζει απλώς τη νότα-στόχο, αλλά τη νότα που βρίσκεται πλησιέστερα στη συχνότητα που παράγεται εκείνη τη στιγμή.

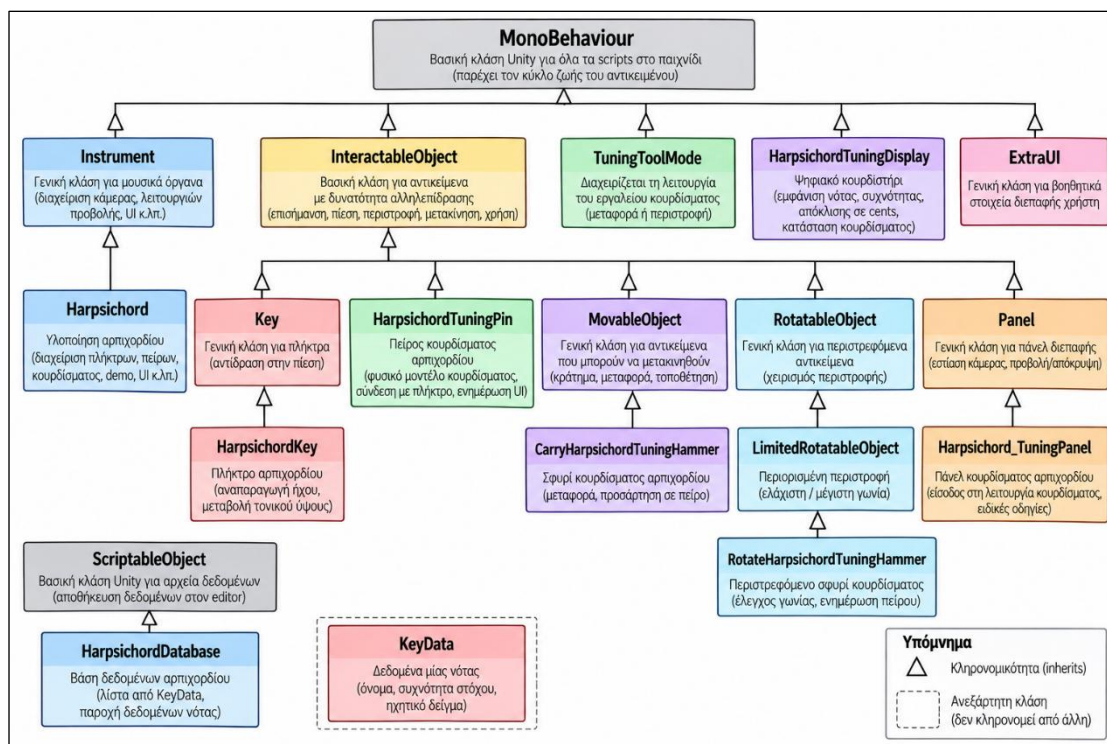
3.7.5 Συνολική λειτουργία κλάσεων αρπιχορδίου

Η υλοποίηση του αρπιχορδίου βασίζεται στη συνεργασία ενός συνόλου εξειδικευμένων κλάσεων, οι οποίες αναλαμβάνουν διακριτές λειτουργίες του εικονικού οργάνου. Στην Εικόνα 27 φαίνεται το διάγραμμα κληρονομικότητας όλων των κλάσεων που σχετίζονται με τη λειτουργία του, ενώ στην Εικόνα 28 οι αλληλεπιδράσεις των πιο σημαντικών από αυτών. Τα δεδομένα των επιμέρους νοτών

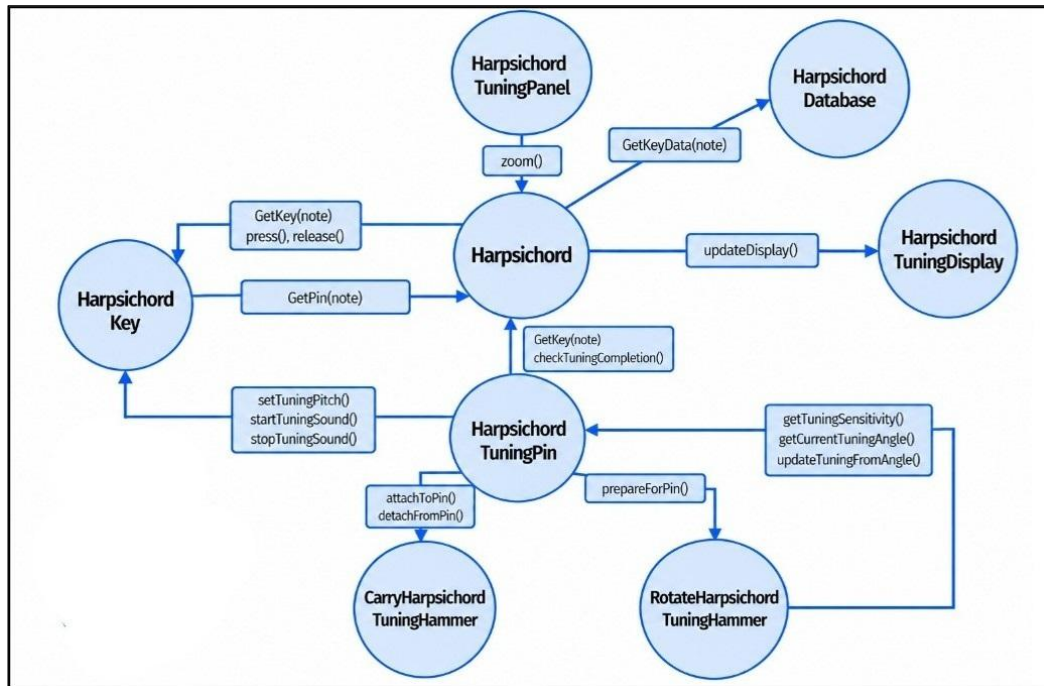
οργανώνονται στην HarpsichordDatabase, ενώ η κλάση Harpsichord αποτελεί την κεντρική κλάση του οργάνου και συνδέει τα επιμέρους στοιχεία του. Κάθε πλήκτρο υλοποιείται μέσω της HarpsichordKey, η οποία κληρονομεί τη βασική συμπεριφορά πατήματος από την Key, ανακτά τα δεδομένα της αντίστοιχης νότας από τη βάση δεδομένων και αναπαράγει το κατάλληλο ηχητικό δείγμα μέσω του AudioSource.

Η μετάβαση του χρήστη στην ειδική προβολή του αρπιχορδίου πραγματοποιείται μέσω της Harpsichord_TuningPanel και του γενικού μηχανισμού που παρέχει η Instrument. Στη λειτουργία αυτή ενεργοποιούνται τα στοιχεία που απαιτούνται για την αλληλεπίδραση με τον μηχανισμό κουρδίσματος και ο χρήστης μπορεί να χρησιμοποιήσει το εικονικό κλειδί κουρδίσματος στους αντίστοιχους πείρους.

Η λειτουργία του κλειδιού κουρδίσματος μοιράζεται μεταξύ των CarryHarpsichordTuningHammer και RotateHarpsichordTuningHammer. Η πρώτη διαχειρίζεται τη μεταφορά και την προσάρτησή του σε έναν πείρο ενώ η δεύτερη μετατρέπει την κίνηση του ποντικιού σε ελεγχόμενη περιστροφή. Η TuningToolMode επιτρέπει την εναλλαγή του ίδιου εργαλείου μεταξύ της λειτουργίας μεταφοράς και της λειτουργίας περιστροφής.



Εικόνα 27. Διάγραμμα κληρονομικότητας του αρπιχορδίου.



Εικόνα 28. Βασικές σχέσεις μεταξύ των κλάσεων του αρπιχορδίου.

Κάθε πείρος κουρδίσματος αντιπροσωπεύεται από την HarpsichordTuningPin και συνδέεται με το αντίστοιχο πλήκτρο και τη χορδή που αυτό αντιπροσωπεύει. Η γωνία περιστροφής του κλειδιού χρησιμοποιείται από τον πείρο για τον υπολογισμό της μεταβολής της τάσης και, μέσω του φυσικού μοντέλου της παλλόμενης χορδής, της νέας συχνότητας. Ο λόγος της νέας συχνότητας προς τη συχνότητα αναφοράς εφαρμόζεται στην HarpsichordKey μέσω της ιδιότητας AudioSource.pitch, ώστε η μεταβολή που προκαλεί ο χρήστης να αποδίδεται και ακουστικά. Παράλληλα, υπολογίζεται η απόκλιση από τη συχνότητα αναφοράς σε cents και η HarpsichordTuningDisplay παρουσιάζει τη νότα, την τρέχουσα συχνότητα και την κατάσταση κουρδίσματος.

Οι βασικές αρμοδιότητες των κλάσεων της υλοποίησης συνοψίζονται ως εξής:

- HarpsichordDatabase: αποθήκευση και παροχή των δεδομένων των νοτών και των αντίστοιχων ηχητικών δειγμάτων.
- Harpsichord: κεντρική διαχείριση του εικονικού οργάνου και σύνδεση των επιμέρους λειτουργιών του.
- HarpsichordKey: οπτική λειτουργία των πλήκτρων, αναπαραγωγή των ηχητικών δειγμάτων και εφαρμογή της μεταβολής του τονικού ύψους.

- Harpsichord_TuningPanel: είσοδος και έξοδος από την ειδική λειτουργία αλληλεπίδρασης με το αρπικόρδιο.
- CarryHarpsichordTuningHammer: μεταφορά του κλειδιού κουρδίσματος και προσάρτησή του σε συγκεκριμένο πείρο.
- TuningToolMode: εναλλαγή του κλειδιού μεταξύ λειτουργίας μεταφοράς και περιστροφής.
- RotateHarpsichordTuningHammer: μετατροπή της κίνησης του ποντικιού σε ελεγχόμενη περιστροφή του εργαλείου.
- HarpsichordTuningPin: σύνδεση του πείρου με τη νότα και εφαρμογή του φυσικού μοντέλου για τον υπολογισμό της συχνότητας και της απόκλισης σε cents.
- HarpsichordTuningDisplay: παρουσίαση της νότας, της τρέχουσας συχνότητας και της κατάστασης κουρδίσματος.

Συνολικά, η υλοποίηση διαχωρίζει τα δεδομένα του οργάνου, την αλληλεπίδραση με τα πλήκτρα, την παραγωγή του ήχου, τον χειρισμό του εργαλείου, τη φυσική προσομοίωση του κουρδίσματος και την οπτική ανατροφοδότηση σε διαφορετικές αλλά συνεργαζόμενες κλάσεις. Με αυτόν τον τρόπο, το αρπικόρδιο λειτουργεί ως ενιαίο διαδραστικό σύστημα, στο οποίο η ενέργεια του χρήστη μεταφέρεται από τα στοιχεία αλληλεπίδρασης στο φυσικό μοντέλο και από εκεί τόσο στον παραγόμενο ήχο όσο και στις πληροφορίες που εμφανίζονται στην οθόνη.

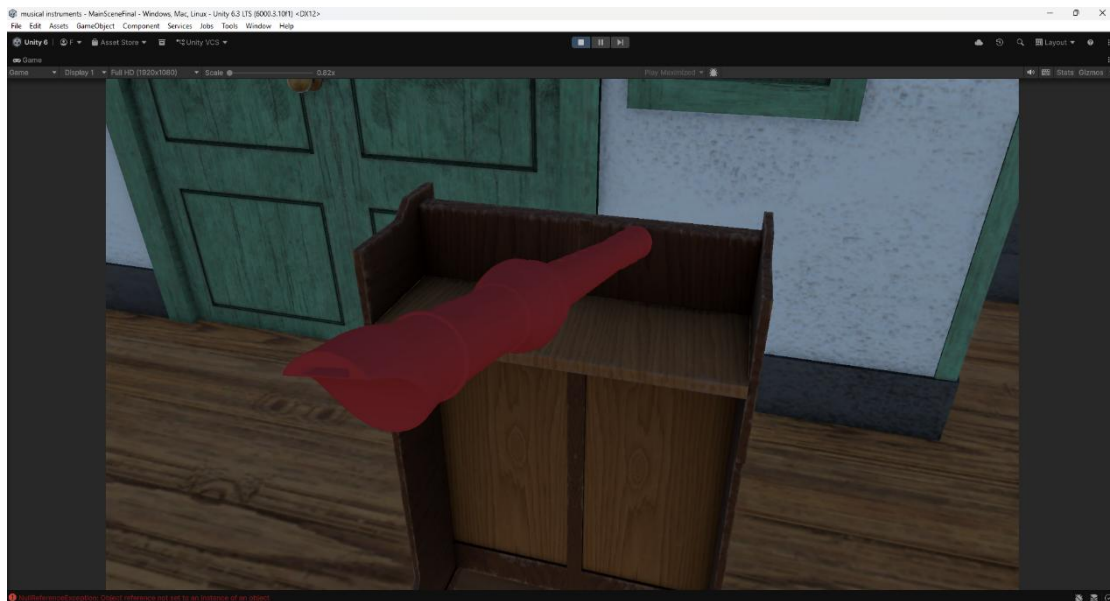
3.7.6 Ειδικές περιπτώσεις και υποστηρικτικοί έλεγχοι

Κατά την ανάπτυξη του εικονικού αρπικόρδιου χρειάστηκε να αντιμετωπιστούν ορισμένες ειδικές περιπτώσεις που προκύπτουν από τα χαρακτηριστικά του χρησιμοποιούμενου τρισδιάστατου μοντέλου. Συγκεκριμένα, το μοντέλο διαθέτει 60 πλήκτρα, ενώ το σύστημα κουρδίσματος περιλαμβάνει 54 πείρους. Για τον λόγο αυτό, τα έξι πλήκτρα F1, F#1, C#6, D6, D#6 και E6 αντιμετωπίζονται στον κώδικα ως *dummy keys*: διατηρούνται ως τμήματα του τρισδιάστατου πληκτρολογίου, αλλά δεν αντιστοιχίζονται σε πείρο κουρδίσματος και δεν παράγουν ήχο. Η διάκριση πραγματοποιείται από τη μέθοδο `isDummyKey()` της `HarpsichordKey`, ενώ για τα συγκεκριμένα πλήκτρα παρέχεται και διαφορετική ένδειξη μέσω του συστήματος `MouseUI`.

Επιπλέον, έχουν προβλεφθεί έλεγχοι για ειδικές καταστάσεις αλληλεπίδρασης. Ενδεικτικά, αποτρέπεται η είσοδος στη λειτουργία κουρδίσματος του αρπιχορδίου όταν ο χρήστης μεταφέρει τη φλογέρα ή το κλειδί κουρδίσματος του ταμπούρου, ώστε να αποφεύγεται η ταυτόχρονη ενεργοποίηση ασύμβατων λειτουργιών. Αντίστοιχα, μετά την αποσύνδεση του κλειδιού κουρδίσματος από έναν πείρο, το εργαλείο επανέρχεται σε κατάσταση μεταφοράς (Movable), ώστε να μπορεί να μετακινηθεί και να χρησιμοποιηθεί σε διαφορετικό πείρο. Οι περιπτώσεις αυτές αποτελούν υποστηρικτικές λεπτομέρειες της υλοποίησης και, για τον λόγο αυτό, δεν αναλύθηκαν εκτενώς στις κύριες υποενότητες.

3.8 Υλοποίηση φλογέρας

Η φλογέρα, σε αντίθεση με το αρπιχόρδιο, αποτελεί φορητό όργανο οπότε η αρχική αλληλεπίδραση μαζί της είναι διαφορετική μέσω κληρονομικότητας ως μετακινήσιμου αντικειμένου (Εικόνα 29). Η υλοποίηση του κουρδίσματός της βασίζεται κυρίως στις κλάσεις Recorder, RecorderDatabase, RecorderHole, RecorderMouthpiece και RecorderTuningDisplay. Η Recorder αποτελεί την κεντρική κλάση του οργάνου και κληρονομεί από την Instrument συντονίζει την κατάσταση των οπών, αναζητά την αντίστοιχη νότα στη βάση δεδομένων, αναπαράγει το κατάλληλο ηχητικό δείγμα και υλοποιεί το φυσικό μοντέλο του κουρδίσματος μέσω της μεταβολής του ενεργού μήκους της αέριας στήλης.



Εικόνα 29. Μεταφορά της φλογέρας.

Η RecorderDatabase περιέχει τις νότες της φλογέρας, τις ονομαστικές συχνότητές τους, τα αντίστοιχα δακτυλοθετήματα και τα ηχητικά δείγματα. Η RecorderHole αναπαριστά κάθε οπή και αποθηκεύει αν αυτή είναι ανοικτή, κλειστή ή, όπου επιτρέπεται, μερικώς κλειστή. Η RecorderMouthpiece λειτουργεί ως το διαδραστικό επιστόμιο μέσω του οποίου ενεργοποιείται και διακόπτεται η παραγωγή ήχου. Τέλος, η RecorderTuningDisplay παρουσιάζει στον χρήστη τη συχνότητα και την κατάσταση κουρδίσματος.

3.8.1 Βάση δεδομένων της φλογέρας

Για την αναπαράσταση των διαθέσιμων νοτών χρησιμοποιείται η απαρίθμηση RecorderNote, η οποία περιλαμβάνει τις νότες από C4 έως D6. Για κάθε νότα δημιουργείται ένα αντικείμενο RecorderNoteData, στο οποίο αποθηκεύονται η νότα, η συχνότητά της, ένα ή περισσότερα επιτρεπόμενα δακτυλοθετήματα και το αντίστοιχο AudioClip.

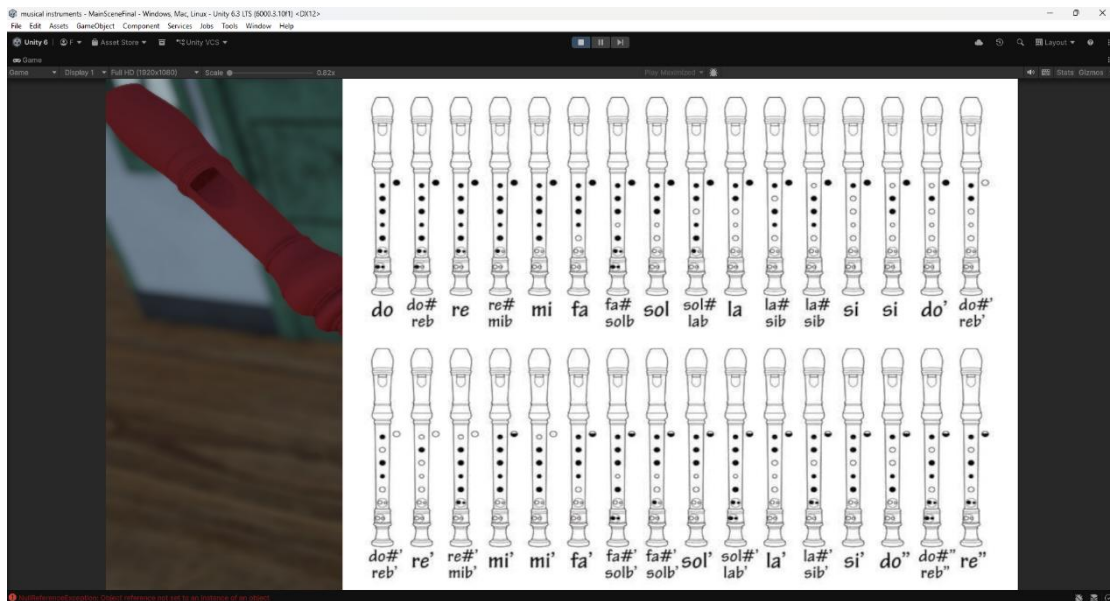
Η βασική δομή είναι η εξής:

```
public class RecorderNoteData
{
    public RecorderNote Note;
    public float Frequency;
    public Fingering[] Fingerings;
    public AudioClip Sound;
}
```

Κάθε δακτυλοθεσία περιγράφεται από την κλάση `Fingering`, η οποία αποθηκεύει τους αριθμούς των οπών που πρέπει να είναι κλειστές και αν η πίσω οπή πρέπει να είναι μερικώς καλυμμένη:

```
public class Fingering
{
    public int[] CoveredHoles;
    public bool BackHoleHalfCovered;
}
```

Επομένως, η νότα δεν επιλέγεται άμεσα από τον χρήστη. Προκύπτει από τον *συνδυασμό της κατάστασης των οπών*. Η μέθοδος `getNoteForFingering()` εξετάζει τα στιγμιότυπα της κλάσης `RecorderHole` (καθένα από τα οποία αντιστοιχεί σε μια οπή της φλογέρας), συγκρίνει την πραγματική τους κατάσταση με τους αποθηκευμένους δακτυλισμούς και επιστρέφει τη νότα που συμφωνεί με την τρέχουσα κατάσταση. Οι αποθηκευμένοι δακτυλισμοί φαίνονται και ως διάγραμμα εντός εφαρμογής στην Εικόνα 30.



Εικόνα 30. Πίνακας δακτυλισμών της φλογέρας.

3.8.2 Οπές και επιλογή της νότας

Κάθε οπή του τρισδιάστατου μοντέλου διαθέτει την κλάση RecorderHole, η οποία κληρονομεί από την InteractableObject. Η κατάσταση μιας οπής περιγράφεται από την απαρίθμηση RecorderHoleState:

```
public enum RecorderHoleState
{
    Open,
    Covered,
    HalfCovered
}
```

Με την press() ο χρήστης αλλάζει την κατάσταση της συγκεκριμένης οπής. Για τις συνηθισμένες οπές πραγματοποιείται εναλλαγή μεταξύ Open και Covered, ενώ στις οπές που επιτρέπεται μερική κάλυψη μπορεί να χρησιμοποιηθεί και η κατάσταση HalfCovered. Η updateVisual() ενημερώνει παράλληλα το τρισδιάστατο μοντέλο, ώστε η κατάσταση της οπής να είναι ορατή.

Η κλάση Recorder, μέσω της μεθόδου getCurrentNote(), συγκεντρώνει τα αντικείμενα RecorderHole που αντιστοιχούν στις οπές της φλογέρας και τα μεταβιβάζει στη μέθοδο getNoteForFingering() της RecorderDatabase, η οποία προσδιορίζει και επιστρέφει τα δεδομένα της μουσικής νότας που αντιστοιχεί στο συγκεκριμένο δακτυλοθέτημα:

```
public RecorderDatabase.RecorderNoteData getCurrentNote()
{
    RecorderHole[] recorderHoles =
        new RecorderHole[Holes.Length];

    for (int i = 0; i < Holes.Length; i++)
    {
        recorderHoles[i] =
            Holes[i].GetComponent<RecorderHole>();
    }

    return Database.getNoteForFingering(
        recorderHoles
    );
}
```

3.8.3 Παραγωγή του ήχου

Η παραγωγή του ήχου ενεργοποιείται μέσω της κλάσης RecorderMouthpiece. Με το πάτημα του επιστομίου καλείται η playCurrentNote() της κλάσης Recorder:

```
Recorder.playCurrentNote();
```

ενώ με την απελευθέρωσή του καλείται η stopCurrentNote():

```
Recorder.stopCurrentNote();
```

Η μέθοδος playCurrentNote() καλεί αρχικά την getCurrentNote(), προκειμένου να προσδιοριστεί η νότα που αντιστοιχεί στο τρέχον δακτυλοθέτημα:

```
RecorderDatabase.RecorderNoteData note =
    getCurrentNote();
```

Η getCurrentNote() όπως είδαμε πριν, συγκεντρώνει τις καταστάσεις των οπών της φλογέρας και τις μεταβιβάζει στη μέθοδο getNoteForFingering() της RecorderDatabase, η οποία επιστρέφει τα δεδομένα της αντίστοιχης νότας. Από τα δεδομένα αυτά λαμβάνονται η ονομαστική συχνότητα (note.Frequency) και το αντίστοιχο ηχητικό δείγμα (note.Sound). Εφόσον υπάρχουν έγκυρα δεδομένα νότας και ηχητικού δείγματος, η μέθοδος playCurrentNote() προχωρά στον υπολογισμό της συχνότητας αναπαραγωγής.

Η σημαντική διαφορά είναι ότι η συχνότητα που τελικά ακούγεται δεν είναι υποχρεωτικά ίση με την ονομαστική συχνότητα της νότας. Πριν από την αναπαραγωγή

υπολογίζεται η τρέχουσα συχνότητα μέσω της `getTunedFrequency()`, ώστε να λαμβάνεται υπόψη η θέση της κεφαλής της φλογέρας.

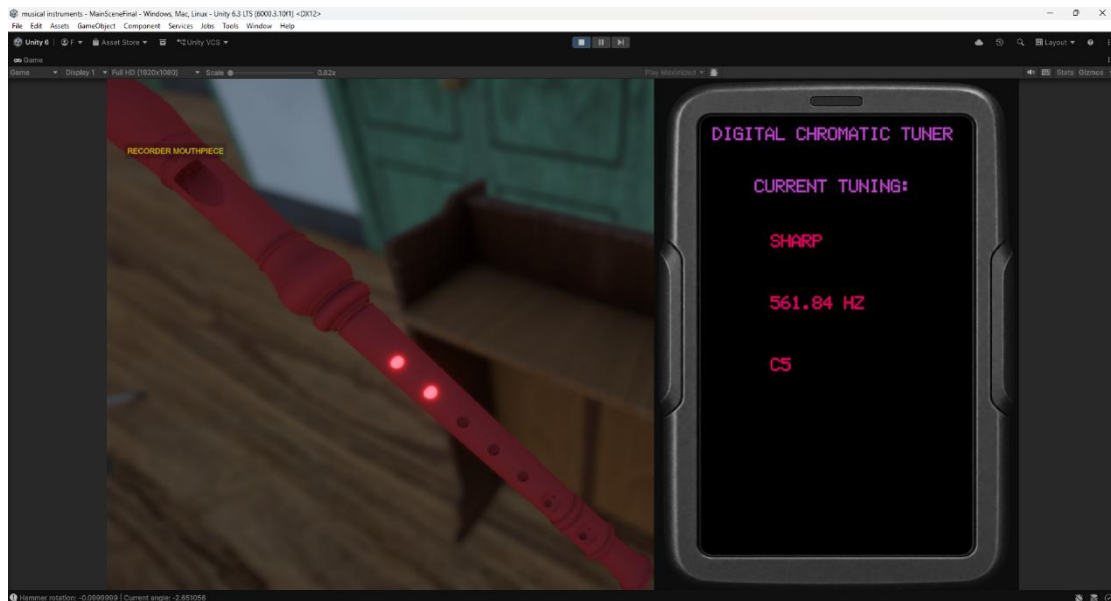
Στη συνέχεια, το ηχητικό δείγμα ανατίθεται στο `RecorderAudioSource`, η ιδιότητα `pitch` προσαρμόζεται στον λόγο της υπολογισμένης προς την ονομαστική συχνότητα και ξεκινά η αναπαραγωγή:

```
float tunedFrequency =  
    getTunedFrequency(note.Frequency);  
  
RecorderAudioSource.clip = note.Sound;  
  
RecorderAudioSource.pitch =  
    tunedFrequency / note.Frequency;  
  
RecorderAudioSource.Play();
```

Με αυτόν τον τρόπο, το ηχητικό δείγμα αντιπροσωπεύει τη συγκεκριμένη νότα, ενώ η μεταβολή της ιδιότητας `pitch` χρησιμοποιείται για να αποδώσει ακουστικά την επίδραση του κουρδίσματος. Όταν η υπολογισμένη συχνότητα είναι ίση με την ονομαστική, ο λόγος τους ισούται με τη μονάδα και το ηχητικό δείγμα αναπαράγεται χωρίς μεταβολή του τονικού ύψους. Αντίθετα, όταν οι δύο συχνότητες διαφέρουν, η τιμή του `pitch` μεταβάλλεται ανάλογα, αποδίδοντας την αντίστοιχη απόκλιση του κουρδίσματος.

3.8.4 Κούρδισμα της φλογέρας

Το κούρδισμα της φλογέρας υλοποιείται μέσω της μεταβολής της θέσης του άνω τμήματος του οργάνου (`Head Joint`). Η μετακίνησή του μεταβάλλει το ενεργό μήκος της αέριας στήλης και, κατά συνέπεια, τη συχνότητα του παραγόμενου ήχου. Με αυτό το μηχανισμό όλες οι νότες αλλάζουν ταυτόχρονα με τον ίδιο τρόπο, λόγω της κατασκευής της φλογέρας. Στην προσομοίωση η διαδικασία αυτή συνδέει άμεσα τη μετακίνηση του τρισδιάστατου μοντέλου με το φυσικό μοντέλο υπολογισμού της νέας συχνότητας και με την αντίστοιχη μεταβολή του αναπαραγόμενου ήχου. Η Εικόνα 31 δείχνει ένα στιγμιότυπο από τη διαδικασία κούρδισματος.



Εικόνα 31. Η φλογέρα σε κατάσταση κουρδίσματος.

3.8.4.1 Μετακίνηση του άνω τμήματος της φλογέρας

Στην προσομοίωση το κούρδισμα πραγματοποιείται με μετακίνηση του GameObject HeadJoint. Η θέση του μεταβάλλεται κατά μήκος ενός άξονα και περιορίζεται μεταξύ της αρχικής θέσης και της μέγιστης επιτρεπόμενης απόστασης:

```
position.x += movement;

position.x =
    Mathf.Clamp(
        position.x,
        HeadJointOriginalLocalPosition.x,
        HeadJointOriginalLocalPosition.x +
        HeadJointMaxPull
    );

HeadJoint.localPosition = position;
```

Η μετακίνηση αυτή υλοποιείται από τη `moveHeadJoint()`. Επομένως, το κούρδισμα δεν πραγματοποιείται με αυθαίρετη μεταβολή της συχνότητας αλλά ξεκινά από μια γεωμετρική μεταβολή του μήκους του οργάνου, η οποία στη συνέχεια μετατρέπεται σε μεταβολή της ακουστικής συχνότητας.

3.8.4.2 Φυσικό μοντέλο του κουρδίσματος

Η φυσική του κουρδίσματος υλοποιείται κυρίως στη μέθοδο `getTunedFrequency()`. Η βασική αρχή είναι ότι η θεμελιώδης συχνότητα μιας ιδανικής αέριας στήλης είναι αντιστρόφως ανάλογη του ενεργού μήκους της· κατά συνέπεια, όταν αυξάνεται το ενεργό μήκος της αέριας στήλης, η συχνότητα μειώνεται, ενώ όταν το ενεργό μήκος μειώνεται, η συχνότητα αυξάνεται.

Στον κώδικα υπολογίζεται αρχικά πόσο έχει μετακινηθεί το `HeadJoint` από την αρχική του θέση:

```
float currentPull =  
    HeadJoint.localPosition.x -  
    HeadJointOriginalLocalPosition.x;
```

Στη συνέχεια προσδιορίζεται η διαφορά της τρέχουσας θέσης από τη θέση στην οποία η φλογέρα θεωρείται σωστά κουρδισμένη:

```
float pullDifference =  
    currentPull -  
    TunedHeadJointPull;
```

Η διαφορά αυτή μετατρέπεται σε μεταβολή του ενεργού μήκους. Στην εφαρμογή χρησιμοποιείται επίσης ο συντελεστής `HeadJointPitchSensitivity`, ώστε η μικρή γεωμετρική μετατόπιση του τρισδιάστατου μοντέλου να προκαλεί περισσότερο αντιληπτή ακουστική μεταβολή (για λόγους εμφανούς ακουστικής επίδειξης):

```
float effectivePullDifference =  
    pullDifference *  
    HeadJointPitchSensitivity;
```

Κατόπιν προκύπτει το τρέχον ενεργό μήκος:

```
float currentEffectiveLength =  
    tunedEffectiveLength +  
    effectivePullDifference;
```

και τελικά επιστρέφεται η νέα συχνότητα:

```
return originalFrequency *  
    tunedEffectiveLength /  
    currentEffectiveLength;
```

Επομένως, όταν το `HeadJoint` απομακρύνεται περισσότερο, αυξάνεται το ενεργό μήκος της αέριας στήλης και η νότα γίνεται χαμηλότερη. Αντίθετα, η μετακίνησή του προς τα μέσα μειώνει το ενεργό μήκος και αυξάνει τη συχνότητα. Στη σωστή θέση

υπάρχει ισότητα των `tunedEffectiveLength` και `currentEffectiveLength`, οπότε η υπολογισμένη συχνότητα συμπίπτει με την ονομαστική συχνότητα της νότας. Αυτή ακριβώς είναι η φυσική λογική που υλοποιεί η `getTunedFrequency()`.

3.8.4.3 Εφαρμογή της μεταβολής στον ήχο

Η μεταβολή του ενεργού μήκους πρέπει να μεταφερθεί και στον ήχο που ακούει ο χρήστης. Για τον λόγο αυτό, όσο μια νότα αναπαράγεται, η `updatePlayingNotePitch()` υπολογίζει εκ νέου την τρέχουσα συχνότητα και ενημερώνει το `AudioSource.pitch`:

```
float tunedFrequency =  
    getTunedFrequency(  
        CurrentlyPlayingNote.Frequency  
    );  
  
RecorderAudioSource.pitch =  
    tunedFrequency /  
    CurrentlyPlayingNote.Frequency;
```

Έτσι η μετακίνηση του `HeadJoint` κατά την αναπαραγωγή μιας νότας προκαλεί άμεσα αντίστοιχη μεταβολή στο τονικό ύψος του ήχου.

3.8.4.4 Υπολογισμός της απόκλισης και κουρδιστήρι

Για την παρακολούθηση της διαδικασίας κουρδίσματος χρησιμοποιείται η μέθοδος `updateTuningDisplay()` της κλάσης `Recorder`. Η μέθοδος προσδιορίζει αρχικά τη νότα που αντιστοιχεί στο τρέχον δακτυλοθέτημα μέσω της `getCurrentNote()` και λαμβάνει την ονομαστική συχνότητά της. Στη συνέχεια υπολογίζει την τρέχουσα συχνότητα μέσω της `getTunedFrequency()`, λαμβάνοντας υπόψη τη θέση της κεφαλής της φλογέρας:

```
RecorderDatabase.RecorderNoteData note =  
    getCurrentNote();  
  
if (note == null)  
{  
    TuningDisplay.clearDisplay();  
    return;  
}  
  
float targetFrequency =  
    note.Frequency;  
  
float currentFrequency =  
    getTunedFrequency(targetFrequency);
```

Για να εκτιμηθεί η απόκλιση της τρέχουσας συχνότητας από την ονομαστική συχνότητα της νότας, η ίδια μέθοδος υπολογίζει τη διαφορά σε cents:

```
float centsDifference =  
    1200f *  
    Mathf.Log(  
        currentFrequency /  
        targetFrequency,  
        2f  
    );
```

Τα αποτελέσματα μεταβιβάζονται στη μέθοδο `updateDisplay()` της κλάσης

```
RecorderTuningDisplay:  
TuningDisplay.updateDisplay(  
    note.Note.ToString(),  
    currentFrequency,  
    centsDifference  
);
```

Η μέθοδος `updateDisplay()` αναλαμβάνει την ενημέρωση των ενδείξεων του μετρητή κουρδίσματος. Αρχικά προσδιορίζει την εμφανιζόμενη νότα μέσω της `getDisplayedNote()`, η οποία μετατρέπει την απόκλιση σε cents σε μεταβολή ημιτονίων, στρογγυλοποιημένη στον πλησιέστερο ακέραιο. Στη συνέχεια εμφανίζει τη νότα και την τρέχουσα συχνότητα με ακρίβεια δύο δεκαδικών ψηφίων:

```
string displayedNote =  
    getDisplayedNote(  
        targetNote,  
        centsDifference  
    );  
  
NoteText.text = displayedNote;  
  
FrequencyText.text =  
    currentFrequency.ToString("F2") + " Hz";
```

Τέλος, η μέθοδος ελέγχει την απόκλιση σε cents και χαρακτηρίζει την κατάσταση ως IN TUNE, FLAT ή SHARP:

```
if (Mathf.Abs(centsDifference)  
    <= InTuneToleranceCents)  
{  
    StatusText.text = "IN TUNE";  
    setColor(Color.green);  
}  
else if (centsDifference < 0f)  
{  
    StatusText.text = "FLAT";  
    setColor(Color.red);  
}  
else  
{  
    StatusText.text = "SHARP";  
    setColor(Color.red);  
}
```

Η ανοχή σωστού κουρδίσματος καθορίζεται από τη μεταβλητή `InTuneToleranceCents`, η οποία έχει προεπιλεγμένη τιμή 5 cents. Όταν η απόκλιση βρίσκεται εντός αυτού του διαστήματος, οι ενδείξεις εμφανίζονται με πράσινο χρώμα και την ένδειξη IN TUNE. Σε αντίθετη περίπτωση εμφανίζεται η ένδειξη FLAT ή SHARP με κόκκινο χρώμα, ανάλογα με το αν η τρέχουσα συχνότητα είναι χαμηλότερη ή υψηλότερη από την ονομαστική.

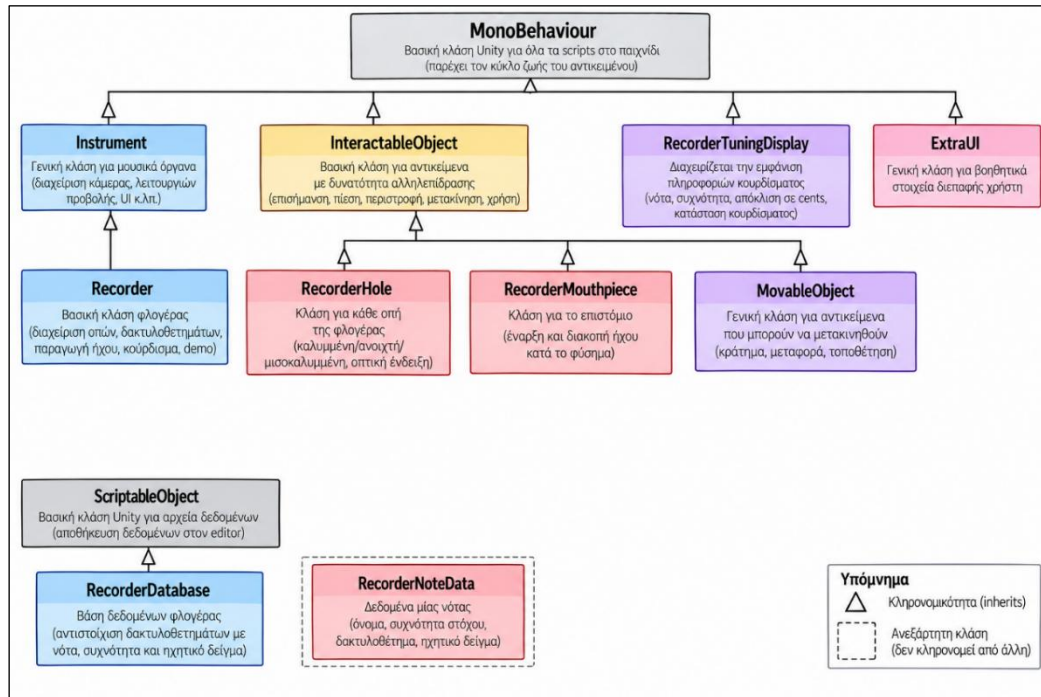
Με αυτόν τον τρόπο ολοκληρώνεται η «αλυσίδα» του κουρδίσματος: η φυσική ενέργεια που προσομοιώνεται είναι η μετακίνηση του `HeadJoint`, η μετακίνηση αυτή μεταβάλλει το ενεργό μήκος της αέριας στήλης, το νέο μήκος καθορίζει τη νέα

συχνότητα, η νέα συχνότητα μεταβάλλει το AudioSource.pitch και η νέα απόκλιση από την επιθυμητή συχνότητα αποτυπώνεται στο ψηφιακό κουρδιστήρι.

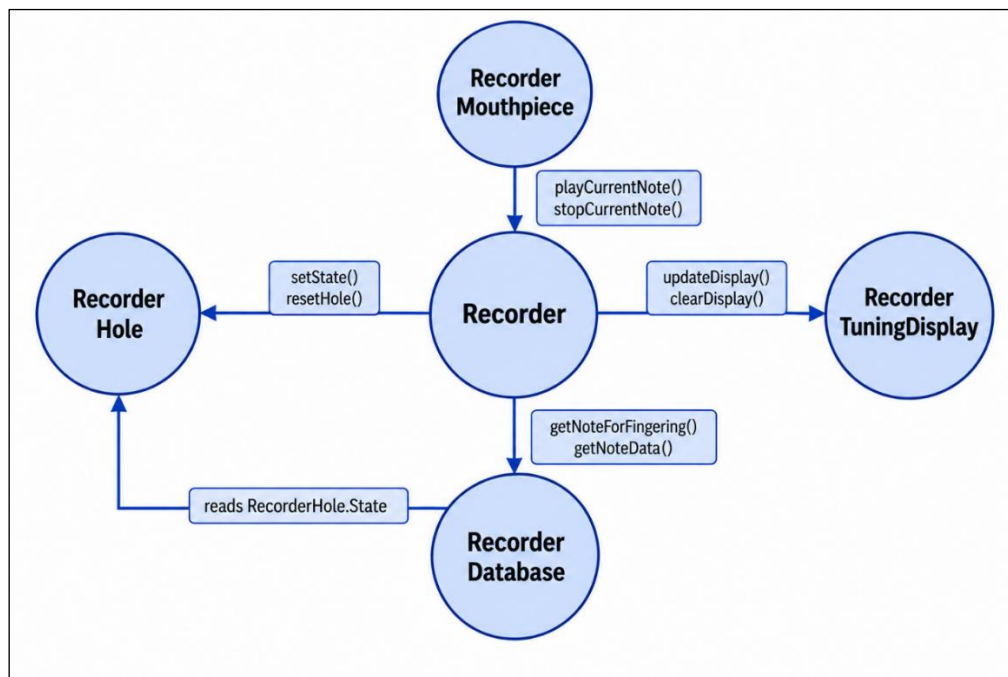
3.8.5 Συνολική λειτουργία της υλοποίησης της φλογέρας

Η υλοποίηση της φλογέρας βασίζεται στη συνεργασία επιμέρους κλάσεων που αναλαμβάνουν τη διαχείριση του οργάνου, την επιλογή της νότας, την παραγωγή του ήχου και το κούρδισμα. Στην Εικόνα 32 παρουσιάζεται το διάγραμμα κληρονομικότητας όλων των κλάσεων που σχετίζονται με τη λειτουργία της, ενώ στην Εικόνα 33 οι αλληλεπιδράσεις των πιο σημαντικών από αυτών. Η Recorder αποτελεί την κεντρική κλάση του οργάνου και συνδέει τις υπόλοιπες λειτουργίες. Οι RecorderHole καταγράφουν την κατάσταση των οπών και σχηματίζουν το τρέχον δακτυλοθέτημα, ενώ η RecorderDatabase αντιστοιχίζει το δακτυλοθέτημα σε συγκεκριμένη νότα, συχνότητα και ηχητικό δείγμα. Η RecorderMouthpiece ενεργοποιεί και διακόπτει την αναπαραγωγή της νότας μέσω της Recorder.

Το κούρδισμα πραγματοποιείται με τη μετακίνηση του αντικειμένου HeadJoint. Η Recorder υπολογίζει από τη θέση του HeadJoint τη μεταβολή του ενεργού μήκους της αέριας στήλης και, με βάση την αντίστροφη σχέση μεταξύ μήκους και συχνότητας, προσδιορίζει την τρέχουσα συχνότητα της νότας. Η μεταβολή εφαρμόζεται στο pitch του AudioSource, ώστε το αποτέλεσμα του κουρδίσματος να αποδίδεται άμεσα στον παραγόμενο ήχο. Παράλληλα υπολογίζεται η απόκλιση από την ονομαστική συχνότητα σε cents και η RecorderTuningDisplay παρουσιάζει στον χρήστη την τρέχουσα συχνότητα και την κατάσταση του κουρδίσματος.



Εικόνα 32. Διάγραμμα κληρονομικότητας φλογέρας.



Εικόνα 33. Βασικές σχέσεις μεταξύ των κλάσεων της φλογέρας.

Οι βασικές αρμοδιότητες των κλάσεων συνοψίζονται ως εξής:

- Recorder: κεντρικός συντονισμός της φλογέρας, αναγνώριση της νότας, αναπαραγωγή ήχου και υλοποίηση του φυσικού μοντέλου κουρδίσματος.

- RecorderDatabase: αποθήκευση των νοτών, των ονομαστικών συχνοτήτων, των δακτυλοθετημάτων και των αντίστοιχων ηχητικών δειγμάτων.
- RecorderHole: διαχείριση της κατάστασης κάθε οπής ως ανοικτής, κλειστής ή μερικώς κλειστής.
- RecorderMouthpiece: έναρξη και διακοπή της αναπαραγωγής της νότας.
- RecorderTuningDisplay: εμφάνιση της νότας, της τρέχουσας συχνότητας και της κατάστασης FLAT, SHARP ή IN TUNE.
- Instrument: παρέχει στη Recorder τη γενική λειτουργικότητα εναλλαγής μεταξύ της κανονικής προβολής και της ειδικής προβολής του οργάνου.

Με αυτή τη δομή, η αλληλεπίδραση του χρήστη συνδέεται με τη λειτουργία του πραγματικού οργάνου σε δύο επίπεδα: το δακτυλοθέτημα καθορίζει ποια νότα παράγεται, ενώ η θέση του Head Joint και το αντίστοιχο ενεργό μήκος της αέριας στήλης καθορίζουν την ακριβή συχνότητά της. Έτσι, η προσομοίωση συνδυάζει τη διαδραστική λειτουργία της φλογέρας με ένα απλοποιημένο φυσικό μοντέλο παραγωγής και ρύθμισης του τονικού ύψους.

3.9 Υλοποίηση ταμπούρου

Η υλοποίηση του ταμπούρου ακολουθεί την κοινή αρχιτεκτονική αλληλεπίδρασης που χρησιμοποιήθηκε και στα προηγούμενα μουσικά όργανα, ωστόσο εισάγει διαφορετικό φυσικό μοντέλο κουρδίσματος. Το όργανο αναπαρίσταται από την κλάση SnareDrum, ενώ η μεμβράνη έχει χωριστεί σε επιμέρους τμήματα (SnareDrumSlice), καθένα από τα οποία συνδέεται με έναν κοχλία κουρδίσματος (SnareDrumLug). Ο χρήστης χρησιμοποιεί ένα εικονικό κλειδί κουρδίσματος, του οποίου η μεταφορά ελέγχεται από την CarrySnareDrumTuningKey και η περιστροφή από την RotateSnareDrumTuningKey, παρόμοια με το αρπιχόρδιο. Η μεταβολή της περιστροφής μετατρέπεται σε μεταβολή της τάσης του αντίστοιχου τμήματος της μεμβράνης και κατ' επέκταση σε μεταβολή της παραγόμενης συχνότητας.

Επιπλέον, σε αντίθεση με τα προηγούμενα όργανα, δεν αρκεί κάθε κοχλίας να φτάσει ανεξάρτητα στη σωστή τάση. Η κλάση SnareTuningAlgorithm ελέγχει και τη σειρά με την οποία κουρδίζονται οι δέκα κοχλίες, ώστε η διαδικασία να ακολουθεί εναλλαγή μεταξύ απέναντι κοχλίων και μετακίνησης κατά δύο θέσεις. Έτσι, η

προσομοίωση περιλαμβάνει τόσο το φυσικό αποτέλεσμα της περιστροφής του κλειδιού όσο και τη διαδικασία που πρέπει να ακολουθήσει ο χρήστης κατά το κούρδισμα.

3.9.1 Αναπαράσταση και βασική λειτουργία του ταμπούρου

Η βασική κλάση του οργάνου είναι η SnareDrum, η οποία κληρονομεί από την Instrument:

```
public class SnareDrum : Instrument
```

Η κληρονομικότητα από την Instrument παρέχει τις κοινές λειτουργίες μετάβασης μεταξύ της κανονικής προβολής και της ειδικής προβολής κουρδίσματος. Τα βασικά στοιχεία του ταμπούρου αποθηκεύονται σε δύο πίνακες:

```
public SnareDrumLug[] Lugs;  
public SnareDrumSlice[] Slices;
```

Τα Lugs αντιπροσωπεύουν τους δέκα κοχλίες περιμετρικά του στεφανιού, ενώ τα Slices αντιπροσωπεύουν τα αντίστοιχα τμήματα της μεμβράνης. Κατά την αρχικοποίηση, τα αντικείμενα αυτά εντοπίζονται αυτόματα στα παιδιά του τρισδιάστατου μοντέλου:

```
Lugs = GetComponentsInChildren<SnareDrumLug>();  
Slices = GetComponentsInChildren<SnareDrumSlice>();
```

Η λογική αυτή επιτρέπει στην SnareDrum να λειτουργεί ως κεντρική κλάση συντονισμού, χωρίς να εκτελεί η ίδια τους επιμέρους υπολογισμούς της περιστροφής ή της τάσης.

Κατά την είσοδο στη λειτουργία κουρδίσματος ενεργοποιούνται οι κατάλληλοι colliders των κοχλιών και των τμημάτων της μεμβράνης, ώστε τα στοιχεία αυτά να μπορούν να δεχθούν αλληλεπίδραση. Η κάμερα κουρδίσματος μπορεί επίσης να περιστρέφεται γύρω από το όργανο, επιτρέποντας στον χρήστη να έχει πρόσβαση στους κοχλίες που βρίσκονται περιμετρικά του ταμπούρου. Οι υπόλοιπες λειτουργίες της SnareDrum που αφορούν μηνύματα, οδηγίες και λειτουργία επίδειξης δεν αναλύονται εδώ, καθώς δεν αποτελούν μέρος του βασικού μηχανισμού κουρδίσματος.

3.9.2 Τμήματα της μεμβράνης και παραγωγή ήχου

Κάθε περιοχή της μεμβράνης αναπαρίσταται από την κλάση:

```
public class SnareDrumSlice : InteractableObject
```

Για κάθε SnareDrumSlice αποθηκεύεται ένας ήχος από το ίδιο ένα ηχητικό δείγμα:

```
public AudioClip Sound;
```

το οποίο αναπαράγεται μέσω του AudioSource του αντίστοιχου αντικειμένου. Η βασική διαδικασία αναπαραγωγής είναι αντίστοιχη με αυτή που παρουσιάστηκε στα προηγούμενα όργανα και επομένως δεν χρειάζεται να αναλυθεί εκ νέου. Τα διάφορα τμήματα της μεμβράνης διαφέρουν μεταξύ τους ως προς το αρχικό κούρδισμά τους.

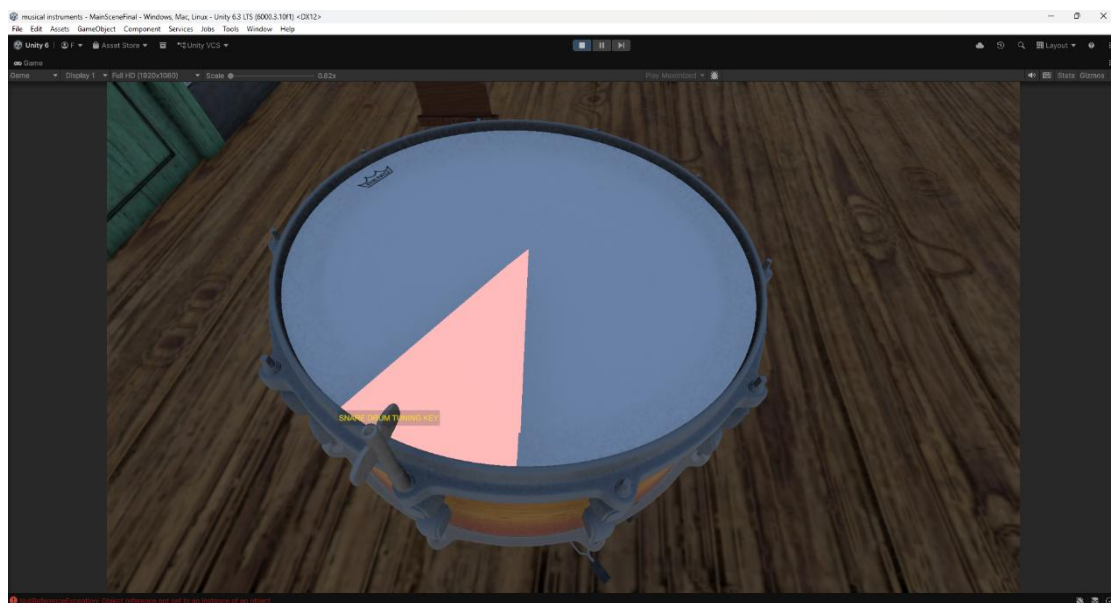
Για τον σκοπό αυτό αποθηκεύονται δύο βασικά μεγέθη:

```
public float TargetTension = 1f;  
public float CurrentTension = 1f;
```

Η TargetTension αποτελεί την κανονικοποιημένη τάση αναφοράς για την οποία το αντίστοιχο τμήμα θεωρείται σωστά κουρδισμένο, ενώ η CurrentTension εκφράζει την τρέχουσα κατάσταση της μεμβράνης. Επομένως, τιμή μικρότερη από 1 αντιστοιχεί σε χαμηλότερη τάση και τιμή μεγαλύτερη από 1 σε υψηλότερη τάση.

Η μέθοδος updatePitch() μετατρέπει αυτή την τάση σε ακουστικό αποτέλεσμα και αποτελεί ένα από τα βασικότερα σημεία της υλοποίησης.

Στην Εικόνα 34 παρουσιάζεται ένα στιγμιότυπο κρούσης ενός τμήματος της μεμβράνης.



Εικόνα 34. Χτύπημα τμήματος του τυμπάνου με το κλειδί τοποθετημένο στον αντίστοιχο κοχλία.

3.9.3 Φυσικό μοντέλο της μεμβράνης

Για μία κυκλική μεμβράνη, όταν οι υπόλοιπες παράμετροι θεωρούνται σταθερές, η χαρακτηριστική συχνότητα είναι ανάλογη με την τετραγωνική ρίζα της τάσης. Στην

προσομοίωση χρησιμοποιείται επομένως ο ίδιος λόγος όπως και στην περίπτωση του αρπιχορδίου (Εξίσωση 19): $\frac{f}{f_0} = \sqrt{\frac{T}{T_0}}$, όπου f είναι η τρέχουσα συχνότητα, f_0 η συχνότητα αναφοράς, T η τρέχουσα τάση και T_0 η τάση αναφοράς.

Η σχέση υλοποιείται στην SnareDrumSlice ως εξής:

```
float tensionRatio =  
    CurrentTension /  
    TargetTension;  
  
float pitchRatio =  
    Mathf.Sqrt(  
        tensionRatio  
    );  
  
audioSource.pitch =  
    pitchRatio;
```

Με απλά λόγια, το pitch χρησιμοποιείται ως λόγος της τρέχουσας προς τη συχνότητα αναφοράς. Όταν οι CurrentTension και TargetTension είναι ίσες, το pitchRatio γίνεται 1 και το ηχητικό δείγμα αναπαράγεται στην αρχική του συχνότητα. Αν η τάση αυξηθεί, αυξάνεται και η συχνότητα, ενώ αντίστοιχα, με μείωση της τάσης, μειώνεται η συχνότητα.

Για την αξιολόγηση του κουρδίσματος, η getCentsFromTarget() μετατρέπει τον λόγο συχνοτήτων σε cents (Εξίσωση 2): $c = 1200 \cdot \log_2\left(\frac{f}{f_0}\right)$

Στον κώδικα η σχέση αυτή αποτυπώνεται ως εξής:

```
public float getCentsFromTarget()  
{  
    float tensionRatio =  
        CurrentTension /  
        TargetTension;  
  
    float pitchRatio =  
        Mathf.Sqrt(tensionRatio);  
  
    float cents =  
        1200f *  
        Mathf.Log(pitchRatio, 2f);  
  
    return cents;  
}
```

Η ορθότητα του κουρδίσματος ελέγχεται μέσω της μεθόδου `isCorrectlyTuned()`, επίσης της κλάσης `SnareDrumSlice`. Η μέθοδος καλεί την `getCentsFromTarget()` για να λάβει την απόκλιση σε cents και εξετάζει αν η απόλυτη τιμή της βρίσκεται εντός της επιτρεπτής ανοχής:

```
public bool isCorrectlyTuned()  
{  
    float cents =  
        getCentsFromTarget();  
  
    return Mathf.Abs(cents) <=  
        TuningToleranceCents;  
}
```

Η μεταβλητή `TuningToleranceCents` καθορίζει τη μέγιστη επιτρεπτή απόκλιση και έχει οριστεί στα 5 cents. Επομένως, όταν η απόκλιση βρίσκεται μεταξύ -5 και $+5$ cents, η μέθοδος `isCorrectlyTuned()` επιστρέφει `true`, δηλώνοντας ότι το συγκεκριμένο τμήμα της μεμβράνης θεωρείται σωστά κουρδισμένο· διαφορετικά, επιστρέφει `false`.

3.9.4 Κλειδί και κοχλίας κουρδίσματος

Η φυσική αλληλεπίδραση του χρήστη με το όργανο πραγματοποιείται μέσω του κλειδιού κουρδίσματος. Η κλάση `CarrySnareDrumTuningKey` κληρονομεί από τη `MovableObject` και αναλαμβάνει τη μεταφορά του εργαλείου, καθώς και τη διαχείριση της σύνδεσής του με τους κοχλίες κουρδίσματος.

Για την αποθήκευση της αναφοράς στον κοχλία με τον οποίο είναι συνδεδεμένο το κλειδί χρησιμοποιείται η μεταβλητή `AttachedLug`:

```
public SnareDrumLug AttachedLug;
```

Η σύνδεση του κλειδιού με έναν συγκεκριμένο κοχλία πραγματοποιείται μέσω της μεθόδου `attachToLug()`, η οποία δέχεται ως παράμετρο το αντικείμενο `SnareDrumLug` που αντιστοιχεί στον επιλεγμένο κοχλία:

```
public void attachToLug(SnareDrumLug lug)
{
    AttachedLug = lug;

    prepareForAttachment();

    keyMode.setMovableMode();
}
```

Αρχικά, η αναφορά στον επιλεγμένο κοχλία αποθηκεύεται στη μεταβλητή `AttachedLug`. Στη συνέχεια καλείται η `prepareForAttachment()`, η οποία προετοιμάζει το εργαλείο για τη σύνδεση, ενώ η `setMovableMode()` της `TuningToolMode` θέτει το εργαλείο σε λειτουργία μεταφοράς.

Αντίστοιχα, η αποσύνδεση πραγματοποιείται μέσω της μεθόδου `detachFromLug()`:

```
public void detachFromLug()
{
    AttachedLug = null;

    AttachedToReceptor = false;

    keyMode.setMovableMode();
}
```

Η μέθοδος μηδενίζει την αναφορά `AttachedLug`, δηλώνοντας ότι το κλειδί δεν είναι πλέον συνδεδεμένο με κάποιον κοχλία. Παράλληλα, η μεταβλητή `AttachedToReceptor` λαμβάνει την τιμή `false` και το εργαλείο επανέρχεται σε λειτουργία μεταφοράς μέσω της `setMovableMode()`.

Η εναλλαγή μεταξύ μεταφοράς και περιστροφής του εργαλείου πραγματοποιείται από την κλάση `TuningToolMode`, η οποία έχει περιγραφεί προηγουμένως.

Η ίδια η σύνδεση του κλειδιού με τον κοχλία πραγματοποιείται από την `SnareDrumLug`. Όταν η `use_with()` δεχθεί ως όρισμα ένα αντικείμενο τύπου

CarrySnareDrumTuningKey, ελέγχει αρχικά αν επιτρέπεται να χρησιμοποιηθεί ο συγκεκριμένος κοχλίας και, εφόσον επιτρέπεται, προσαρτά το κλειδί στον κοχλία καλώντας τη μέθοδο attachTuningKey():

```
attachTuningKey(tuningKey);
```

Το κλειδί γίνεται παιδί του αντίστοιχου SnareDrumLug και τοποθετείται στην κατάλληλη σχετική θέση, γωνία περιστροφής και κλίμακα. Στη συνέχεια καλείται η μέθοδος prepareForLug() μέσω της μεταβλητής rotateKey, η οποία αποτελεί αναφορά στο στοιχείο RotateSnareDrumTuningKey που διαχειρίζεται την περιστροφή του κλειδιού κουρδίσματος:

```
rotateKey.prepareForLug();
```

Η μέθοδος prepareForLug() αποθηκεύει την τρέχουσα περιστροφή του εργαλείου στη μεταβλητή StartingRotation και μηδενίζει τη CurrentAngle:

```
public void prepareForLug()
{
    StartingRotation =
        transform.rotation;

    CurrentAngle = 0f;
}
```

Με αυτόν τον τρόπο, η θέση περιστροφής που έχει το κλειδί κατά τη σύνδεσή του με τον κοχλία ορίζεται ως η αρχική γωνιακή θέση αναφοράς για τη διαδικασία κουρδίσματος που ακολουθεί.

3.9.4.1 Μετατροπή της περιστροφής σε μεταβολή τάσης

Η RotateSnareDrumTuningKey αποτελεί το πρώτο στάδιο του φυσικού μοντέλου κουρδίσματος:

```
public class RotateSnareDrumTuningKey : LimitedRotatableObject
```

Η βασική περιστροφή και ο περιορισμός της ταχύτητας παρέχονται από τη LimitedRotatableObject. Η εξειδικευμένη κλάση του ταμπούρου αναλαμβάνει να μετατρέψει την πραγματική περιστροφή του κλειδιού σε μεταβολή της τάσης της μεμβράνης.

Ο πρώτος υπολογισμός αφορά τη λειτουργία του σπειρώματος του κοχλίου. Η μεταβλητή ThreadPitch εκφράζει την αξονική απόσταση που διανύει ο κοχλίας σε μία πλήρη περιστροφή (έχει επιλεγθεί μια αυθαίρετη τιμή η οποία ταιριάζει στη προσομοίωση):

```
public float ThreadPitch = 0.8f;
```

Για περιστροφή $\Delta\theta$, η αξονική μετατόπιση υπολογίζεται από:

$$\Delta x = p \frac{\Delta\theta}{360^\circ} \quad (25)$$

όπου p είναι το βήμα του σπειρώματος.

Στον κώδικα η αξονική μετατόπιση εκφράζεται ως εξής:

```
float displacementMillimeters =  
    ThreadPitch *  
    actualRotation /  
    360f;
```

Επομένως, η κίνηση του ποντικιού έχει πλέον μετατραπεί από γωνιακή περιστροφή σε γραμμική μετατόπιση του κοχλίου.

Το δεύτερο στάδιο είναι η μετατροπή αυτής της μετατόπισης σε μεταβολή της τάσης. Για τον σκοπό αυτό χρησιμοποιείται η μεταβλητή TensionChangePerMillimeter (και εδώ έχει επιλεγθεί τιμή που ταιριάζει στη προσομοίωση):

```
public float TensionChangePerMillimeter = 0.1f;
```

και εφαρμόζεται η απλοποιημένη γραμμική σχέση: $\Delta T = k \cdot \Delta x$, όπου k είναι ένας συντελεστής που εκφράζει την αποτελεσματική μηχανική απόκριση του συστήματος κοχλίου-στεφανιού-μεμβράνης.

Σε κώδικα, ο μαθηματικός τύπος υπολογισμού της μεταβολής της τάσης εκφράζεται ως εξής:

```
float tensionChange =  
    displacementMillimeters *  
    TensionChangePerMillimeter;
```

Κατόπιν, η μεταβολή της τάσης μεταφέρεται στον κοχλίο στον οποίο έχει προσαρτηθεί το κλειδί:

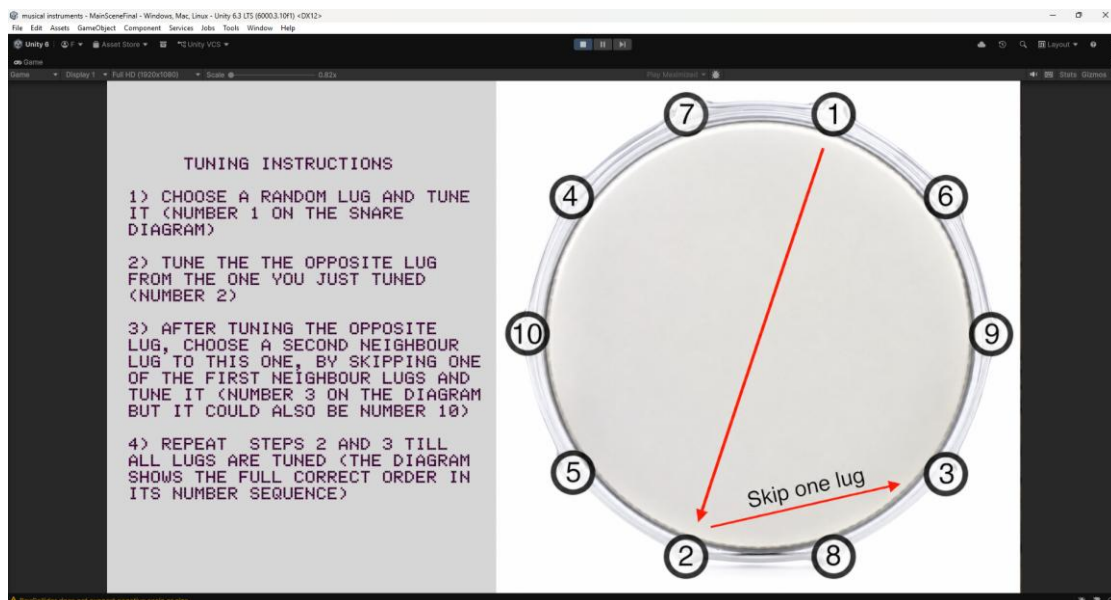
```
carryKey.AttachedLug.changeTension(  
    tensionChange  
);
```

Η SnareDrumLug μεταφέρει τελικά τη μεταβολή στο συνδεδεμένο SnareDrumSlice, αλλάζοντας την CurrentTension και καλώντας την updatePitch().

Επομένως, η διαδικασία κουρδίσματος στην προσομοίωση ακολουθεί μια διαδοχική αλληλουχία μεταβολών: η περιστροφή του κλειδιού χρησιμοποιείται για τον υπολογισμό της αξονικής μετατόπισης του κοχλίου, από την οποία προκύπτει η μεταβολή της τάσης της μεμβράνης. Η μεταβολή αυτή επηρεάζει τη συχνότητα του παραγόμενου ήχου και οδηγεί στην αντίστοιχη προσαρμογή της ιδιότητας AudioSource.pitch, η οποία καθορίζει το τονικό ύψος του αναπαραγόμενου ηχητικού δείγματος.

3.9.4.2 Έλεγχος της διαδικασίας κουρδίσματος

Ένα επιπλέον χαρακτηριστικό του ταμπούρου είναι ότι η εφαρμογή δεν ελέγχει μόνο το αν κάθε τμήμα της μεμβράνης έχει φτάσει στη σωστή τάση, αλλά και τη σειρά με την οποία ο χρήστης επεξεργάζεται τους κοχλίες. Στην Εικόνα 35 φαίνεται η ενσωματωμένη στην εφαρμογή οδηγία για το σωστό κούρδισμα.



Εικόνα 35. Οδηγίες αλγόριθμου κουρδίσματος ταμπούρου

Η λειτουργία αυτή υλοποιείται από την κλάση SnareTuningAlgorithm:

```
public class SnareTuningAlgorithm : MonoBehaviour
```

Ο αλγόριθμος αφορά συνολικά δέκα κοχλίες:

```
private const int NumberOfLugs = 10;
```

και αποθηκεύει σε μια λίστα όσους έχουν ήδη κουρδιστεί:

```
private List<int> TunedLugs = new();
```

Ο πρώτος κοχλίας μπορεί να επιλεγεί ελεύθερα και καθορίζει το σημείο από το οποίο αρχίζει η ακολουθία. Από εκεί και πέρα, ο αλγόριθμος εναλλάσσεται μεταξύ δύο κανόνων.

Ο πρώτος κανόνας είναι η επιλογή του διαμετρικά απέναντι κοχλία. Εφόσον υπάρχουν δέκα κοχλίες, ο απέναντι βρίσκεται πέντε θέσεις μετά:

```
int opposite =  
    lugNumber + 5;
```

με κατάλληλη επαναφορά της αρίθμησης όταν αυτή ξεπεράσει το 10.

Ο δεύτερος κανόνας είναι η παράλειψη ενός κοχλία, δηλαδή μετακίνηση δύο θέσεων δεξιά ή αριστερά:

```
int nextLug =  
    lugNumber + 2;
```

ή:

```
int nextLug =  
    lugNumber - 2;
```

Η μέθοδος isValidNextLug() της κλάσης SnareTuningAlgorithm ελέγχει αν ο κοχλίας που επέλεξε ο χρήστης αντιστοιχεί σε μία από τις επιτρεπόμενες θέσεις της ακολουθίας κουρδίσματος. Αρχικά, εφόσον δεν έχει ρυθμιστεί κανένας κοχλίας, επιτρέπεται η επιλογή οποιουδήποτε από τους δέκα κοχλίες. Στη συνέχεια, αποκλείονται όσοι έχουν ήδη ρυθμιστεί και ελέγχεται αν ο επόμενος κοχλίας είναι ο διαμετρικά απέναντι ή ένας από τους δύο κοχλίες που βρίσκονται δύο θέσεις δεξιά ή αριστερά από τον προηγούμενο, ανάλογα με το στάδιο της διαδικασίας.

Η λειτουργία αυτή υλοποιείται ως εξής:

```
public bool isValidNextLug(int lugNumber)
{
    if (!AlgorithmStarted)
        return true;

    if (isLugTuned(lugNumber))
        return false;

    if (NextLugShouldBeOpposite)
    {
        int oppositeLug =
            getOppositeLug (LastTunedLug);

        return lugNumber == oppositeLug;
    }

    int rightLug =
        getSkipRightLug (LastTunedLug);

    int leftLug =
        getSkipLeftLug (LastTunedLug);

    bool rightIsValid =
        !isLugTuned(rightLug);

    bool leftIsValid =
        !isLugTuned(leftLug);

    if (lugNumber == rightLug && rightIsValid)
        return true;

    if (lugNumber == leftLug && leftIsValid)
        return true;

    return false;
}
```

Επομένως, μετά την ολοκλήρωση του πρώτου κοχλία απαιτείται η ρύθμιση του διαμετρικά απέναντι. Στη συνέχεια επιτρέπεται η επιλογή ενός από τους δύο κοχλίες που βρίσκονται δύο θέσεις δεξιά ή αριστερά από τον τελευταίο ρυθμισμένο, εφόσον

δεν έχει ήδη ολοκληρωθεί η ρύθμισή τους. Ακολουθεί ξανά η επιλογή του απέναντι κοχλίας και η διαδικασία συνεχίζεται με την ίδια εναλλαγή.

Η κατάσταση του αλγορίθμου ενημερώνεται μόνο όταν ένας κοχλίας ολοκληρώσει επιτυχώς το κούρδισμά του:

```
TunedLugs.Add(lugNumber);  
  
LastTunedLug =  
    lugNumber;
```

και η μεταβλητή `NextLugShouldBeOpposite` εναλλάσσει τον επόμενο κανόνα μεταξύ της επιλογής του απέναντι κοχλίας και της παράλειψης ενός κοχλίας στα δεξιά ή στα αριστερά:

```
NextLugShouldBeOpposite =  
    !NextLugShouldBeOpposite;
```

Η σύνδεση του αλγορίθμου με το φυσικό κούρδισμα γίνεται μέσω της `SnareDrumLug`. Με κάθε αλλαγή της τάσης ελέγχεται η κατάσταση του `ConnectedSlice` πριν και μετά τη μεταβολή:

```
bool wasCorrectlyTuned =  
    ConnectedSlice.isCorrectlyTuned();  
...  
bool isCorrectlyTuned =  
    ConnectedSlice.isCorrectlyTuned();
```

Όταν το αντίστοιχο τμήμα της μεμβράνης εισέλθει για πρώτη φορά στην αποδεκτή περιοχή των ± 5 cents, ο κοχλίας καταχωρίζεται μέσω της μεθόδου `registerTunedLug()` της κλάσης `SnareDrum`:

```
snareDrum.registerTunedLug(  
    LugNumber  
);
```

Η μέθοδος `registerTunedLug()` της `SnareDrum` μεταβιβάζει τον αριθμό του κοχλίας στη μέθοδο με το ίδιο όνομα της κλάσης `SnareTuningAlgorithm`, μέσω της μεταβλητής `TuningAlgorithm`, η οποία αποτελεί αναφορά στο αντίστοιχο στοιχείο της εφαρμογής:

```
TuningAlgorithm.registerTunedLug(  
    lugNumber  
);
```

Στην κλάση SnareTuningAlgorithm, η μέθοδος registerTunedLug() προσθέτει τον αριθμό του κοχλίου στη λίστα TunedLugs, εφόσον δεν έχει ήδη καταχωριστεί, και ενημερώνει τη σειρά επιλογής των επόμενων κοχλίων. Στη συνέχεια, η SnareDrum καλεί την isComplete(), η οποία ελέγχει αν έχουν καταχωριστεί και οι δέκα κοχλίες:

```
public bool isComplete()  
{  
    return TunedLugs.Count >=  
        NumberOfLugs;  
}
```

Όταν η μέθοδος επιστρέψει true, η διαδικασία κουρδίσματος του ταμπούρου θεωρείται ολοκληρωμένη.

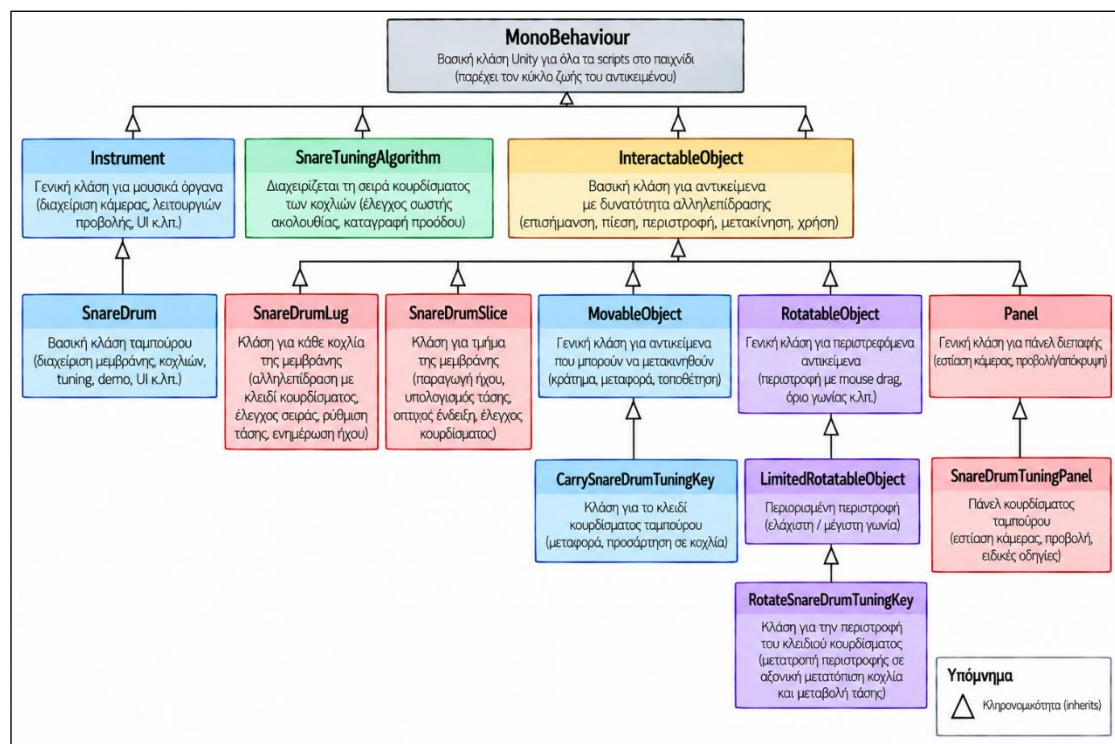
3.9.5 Συνολική λειτουργία της υλοποίησης του ταμπούρου

Η υλοποίηση του ταμπούρου συνδυάζει το κοινό σύστημα αλληλεπίδρασης της εφαρμογής με ένα εξειδικευμένο μοντέλο μηχανικού κουρδίσματος. Στην Εικόνα 36 φαίνεται το διάγραμμα κληρονομικότητας όλων των κλάσεων που σχετίζονται με τη λειτουργία του, ενώ στην Εικόνα 37 οι αλληλεπιδράσεις των πιο σημαντικών από αυτών. Η SnareDrum αποτελεί την κεντρική κλάση του οργάνου, οι SnareDrumLug αναπαριστούν τους κοχλίες και κάθε SnareDrumSlice την περιοχή της μεμβράνης που επηρεάζεται από τον αντίστοιχο κοχλία. Η CarrySnareDrumTuningKey επιτρέπει τη μεταφορά και σύνδεση του κλειδιού, ενώ η RotateSnareDrumTuningKey μετατρέπει την περιστροφή του σε μηχανική μετατόπιση του αντίστοιχου κοχλίου και τελικά σε μεταβολή της τάσης. Η SnareDrumSlice εφαρμόζει τη σχέση μεταξύ τάσης και συχνότητας και ενημερώνει αντίστοιχα τον παραγόμενο ήχο. Παράλληλα, η SnareTuningAlgorithm ελέγχει τη σειρά επεξεργασίας των κοχλίων και την ολοκλήρωση της διαδικασίας.

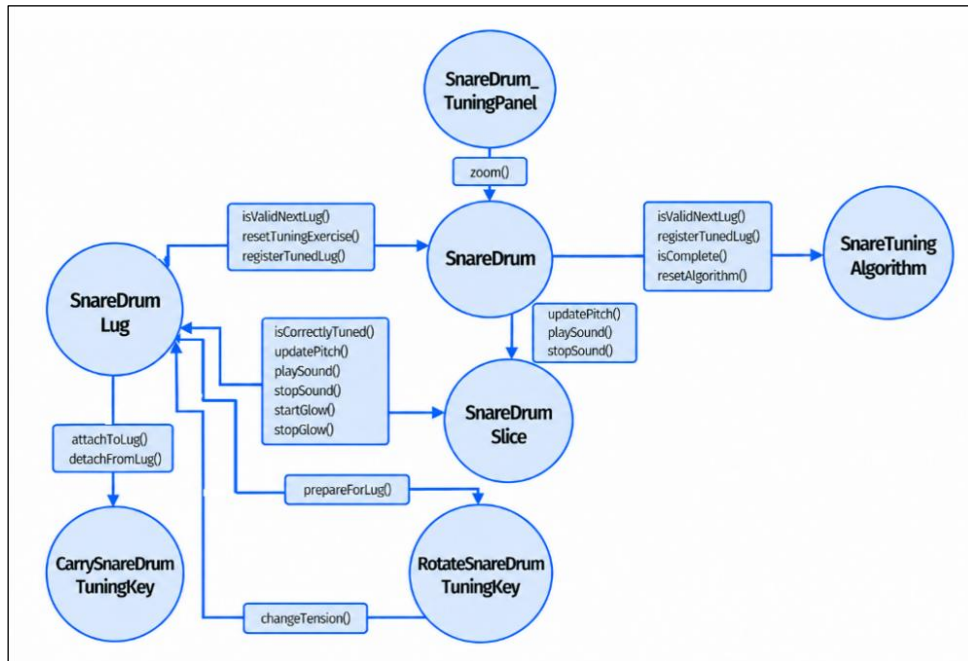
Οι βασικές αρμοδιότητες των κλάσεων είναι:

- SnareDrum: κεντρικός συντονισμός του οργάνου και της διαδικασίας κουρδίσματος.

- SnareDrumSlice: αναπαράσταση της τοπικής τάσης της μεμβράνης, υπολογισμός της συχνότητας και παραγωγή του αντίστοιχου ήχου.
- SnareDrumLug: σύνδεση ενός κοχλία με το αντίστοιχο τμήμα της μεμβράνης και εφαρμογή των μεταβολών τάσης.
- CarrySnareDrumTuningKey: μεταφορά του κλειδιού και σύνδεσή του με έναν κοχλία.
- RotateSnareDrumTuningKey: μετατροπή της περιστροφής του κλειδιού σε αξονική μετατόπιση του κοχλία και μεταβολή τάσης.
- SnareTuningAlgorithm: έλεγχος της επιτρεπόμενης σειράς κουρδίσματος των δέκα κοχλίων.
- SnareDrumTuningPanel: μετάβαση στην ειδική προβολή κουρδίσματος μέσω του κοινού μηχανισμού Panel-Instrument.



Εικόνα 36. Διάγραμμα κληρονομικότητας ταμπούρου



Εικόνα 37. Βασικές σχέσεις μεταξύ των κλάσεων του ταμπούρου

Με τη συγκεκριμένη διάρθρωση διαχωρίζονται η αλληλεπίδραση με το εργαλείο, η μηχανική λειτουργία του κοχλία, η φυσική συμπεριφορά της μεμβράνης και ο αλγόριθμος της διαδικασίας κουρδίσματος. Έτσι, η ενέργεια του χρήστη μετατρέπεται διαδοχικά σε μηχανική μεταβολή, αριθμητικό υπολογισμό της τάσης και της συχνότητας και, τελικά, σε άμεσα αντιληπτή μεταβολή του ήχου.

3.10 Συμπληρωματικές λειτουργίες της εφαρμογής

Εκτός από τους βασικούς μηχανισμούς αλληλεπίδρασης, παραγωγής ήχου και κουρδίσματος που παρουσιάστηκαν στις προηγούμενες ενότητες, στην εφαρμογή υλοποιήθηκαν ορισμένες συμπληρωματικές λειτουργίες για την οργάνωση της προσομοίωσης και την καθοδήγηση του χρήστη. Σε αυτές περιλαμβάνονται η αρχικοποίηση της κατάστασης των οργάνων, οι λειτουργίες ηχητικής επίδειξης, τα ενημερωτικά μηνύματα και το κεντρικό μενού της εφαρμογής. Οι λειτουργίες αυτές παρουσιάζονται συνοπτικά, καθώς δεν αποτελούν μέρος των φυσικών μοντέλων κουρδίσματος. Αναλυτικός κώδικας υπάρχει στο αποθετήριο πηγαίου κώδικα που παρατίθεται στην αντίστοιχη ενότητα.

3.10.1 Τυχαία αρχική κατάσταση κουρδίσματος

Κατά την έναρξη της προσομοίωσης, τα όργανα που προορίζονται για άσκηση κουρδίσματος δεν βρίσκονται απαραίτητα στην κατάσταση αναφοράς τους. Για τον σκοπό αυτό χρησιμοποιούνται μέθοδοι αρχικοποίησης που δημιουργούν μία τυχαία απόκλιση από την επιθυμητή κατάσταση. Ένα παράδειγμα αποτελεί η `randomizeInitialTuning()` της `SnareDrum`, η οποία αποδίδει στα επιμέρους τμήματα της μεμβράνης τυχαία θετική ή αρνητική απόκλιση σε cents. Η απόκλιση μετατρέπεται αρχικά σε λόγο συχνοτήτων και στη συνέχεια, μέσω της σχέσης της συχνότητας με την τάση της μεμβράνης, στην αντίστοιχη αρχική τιμή της `CurrentTension`.

Με αυτόν τον τρόπο, κάθε νέα εκτέλεση της άσκησης μπορεί να παρουσιάζει διαφορετική αρχική κατάσταση και ο χρήστης καλείται να οδηγήσει το όργανο στην επιθυμητή κατάσταση χρησιμοποιώντας τον μηχανισμό κουρδίσματος που περιγράφηκε προηγουμένως. Αντίστοιχες λειτουργίες αρχικοποίησης χρησιμοποιούνται όπου απαιτείται και στα υπόλοιπα όργανα.

3.10.2 Λειτουργίες επίδειξης των οργάνων

Στα μουσικά όργανα έχουν προστεθεί επίσης λειτουργίες επίδειξης (demo), οι οποίες επιτρέπουν την αυτόματη αναπαραγωγή προκαθορισμένων ηχητικών ακολουθιών. Οι λειτουργίες αυτές βασίζονται κυρίως σε `Coroutines`, ώστε διαφορετικοί ήχοι να ενεργοποιούνται διαδοχικά με συγκεκριμένη χρονική διάρκεια χωρίς να διακόπτεται η κανονική εκτέλεση της εφαρμογής.

Στο αρπικόρδιο, για παράδειγμα, η επίδειξη ενεργοποιεί διαδοχικά συγκεκριμένα πλήκτρα μέσω των αντίστοιχων μεθόδων αναπαραγωγής της `HarpsichordKey`. Αντίστοιχα, στο ταμπούρο χρησιμοποιούνται δύο διαφορετικές μέθοδοι της κλάσης `SnareDrum`, οι `PlaySnareDemo()` και `playSnareDemo()`, οι οποίες έχουν παρόμοια ονόματα αλλά διαφορετικούς ρόλους. Η πρώτη ελέγχει αν εκτελείται ήδη κάποια επίδειξη, ώστε να αποτρέψει την ταυτόχρονη εκκίνησή της περισσότερες από μία φορές, και στη συνέχεια εκκινεί τη δεύτερη μέσω της `StartCoroutine()`. Η `playSnareDemo()` αποτελεί `coroutine` και αναλαμβάνει την εκτέλεση της επίδειξης, καθορίζοντας τη σειρά και τη διάρκεια αναπαραγωγής των ήχων. Για τον σκοπό αυτό καλεί διαδοχικά τη μέθοδο `PlayDemoSlice()` για διαφορετικά τμήματα της μεμβράνης, περιμένοντας την ολοκλήρωση της αναπαραγωγής κάθε τμήματος πριν προχωρήσει

στο επόμενο. Οι λειτουργίες επίδειξης χρησιμοποιούν επομένως το ίδιο σύστημα παραγωγής ήχου με την κανονική αλληλεπίδραση και δεν αποτελούν ξεχωριστό φυσικό μοντέλο.

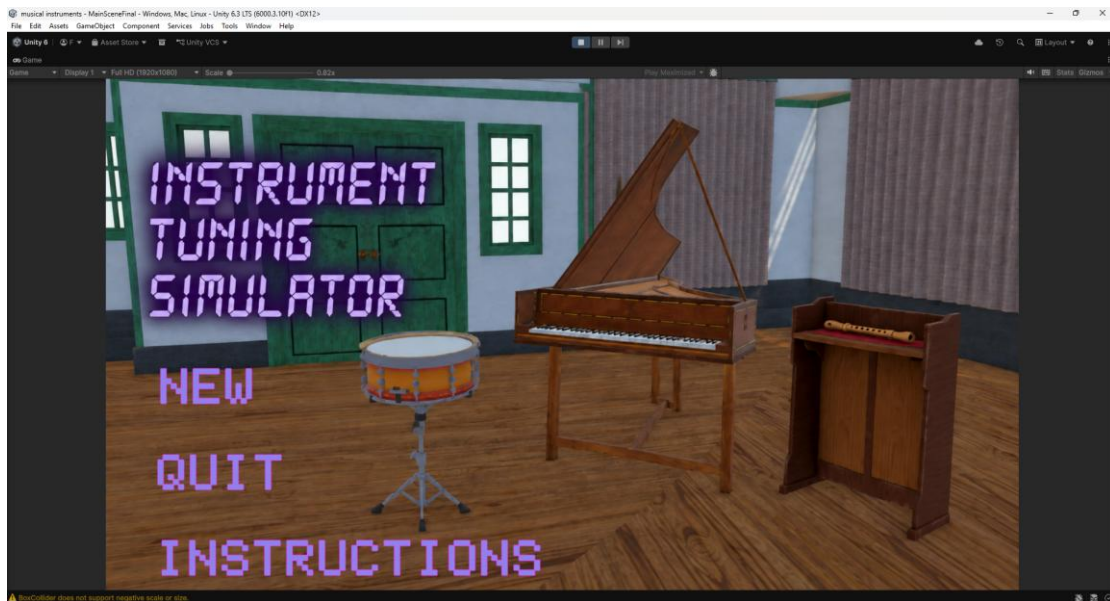
3.10.3 Μηνύματα και οδηγίες προς τον χρήστη

Η εφαρμογή χρησιμοποιεί επίσης ενημερωτικά μηνύματα για την καθοδήγηση του χρήστη κατά την αλληλεπίδραση με τα όργανα. Τα μηνύματα μπορούν να αφορούν την είσοδο σε μία λειτουργία κουρδίσματος, την επιτυχή ρύθμιση ενός στοιχείου, την ολοκλήρωση μιας διαδικασίας ή μία λανθασμένη ενέργεια.

Για παράδειγμα, στην SnareDrum χρησιμοποιούνται αντικείμενα όπως CorrectPitchMessage, SnareDrumTunedMessage και WrongTuningMessage, ενώ η εμφάνισή τους πραγματοποιείται μέσω της MessageManager. Με αυτόν τον τρόπο η λογική του οργάνου αποφασίζει πότε πρέπει να εμφανιστεί ένα μήνυμα, ενώ η κοινή κλάση MessageManager αναλαμβάνει τη διαχείριση της εμφάνισής του. Η συγκεκριμένη λειτουργία αποτελεί μέρος της διεπαφής και της εκπαιδευτικής καθοδήγησης και για τον λόγο αυτό δεν αναλύθηκε στις ενότητες του φυσικού μοντέλου.

3.10.4 Κεντρικό μενού της εφαρμογής

Η έναρξη της εφαρμογής υλοποιείται με την εμφάνιση ενός αρχικού μενού (Εικόνα 38). Η διαχείριση του αρχικού μενού πραγματοποιείται από την κλάση StartMenu. Η κλάση παρέχει τις βασικές επιλογές έναρξης νέου παιχνιδιού, εμφάνισης και απόκρυψης των οδηγιών και εξόδου από την εφαρμογή.



Εικόνα 38. Αρχικό μενού εφαρμογής.

Η μέθοδος `newGame()` διακρίνει δύο περιπτώσεις. Κατά την πρώτη εκκίνηση, η στατική μεταβλητή `GameHasStarted` είναι `false`:

```
public static bool GameHasStarted = false;
```

που σημαίνει ότι η προσομοίωση δεν έχει ακόμη αρχίσει.

Όταν ο χρήστης επιλέξει την έναρξη νέου παιχνιδιού για πρώτη φορά, η `GameHasStarted` λαμβάνει την τιμή `true` και καλείται η `startGameplay()` της κλάσης `EgoController`, η οποία διαχειρίζεται τον χαρακτήρα του χρήστη και επιτρέπει την κανονική αλληλεπίδραση με το τρισδιάστατο περιβάλλον:

```
if (!GameHasStarted)
{
    GameHasStarted = true;

    EgoController ego =
        FindFirstObjectByType<EgoController>();

    ego.startGameplay();

    return;
}
```

Αν ο χρήστης επιλέξει ξανά την έναρξη νέου παιχνιδιού αφού η προσομοίωση έχει ήδη ξεκινήσει, η ίδια επιλογή λειτουργεί ως επανεκκίνηση. Αρχικά, η κλίμακα χρόνου

επανέρχεται στην κανονική της τιμή και στη συνέχεια φορτώνεται εκ νέου η κύρια σκηνή:

```
Time.timeScale = 1f;  
  
SceneManager.LoadScene (  
    "MainSceneFinal"  
);
```

Η επαναφόρτωση της σκηνής επαναφέρει τα αντικείμενα στην αρχική τους κατάσταση. Επειδή η `GameHasStarted` είναι στατική μεταβλητή, η τιμή της διατηρείται κατά την επαναφόρτωση της σκηνής και επιτρέπει στον κώδικα να διακρίνει την πρώτη εκκίνηση από μία μεταγενέστερη.

Επιπλέον, στην κλάση `StartMenu` υπάρχουν οι μέθοδοι `showInstructions()` και `hideInstructions()`, οι οποίες αναλαμβάνουν την εμφάνιση και την απόκρυψη του `InstructionsPanel`, καθώς και η `instructionsAreOpen()`, η οποία επιστρέφει την τρέχουσα κατάσταση του συγκεκριμένου πλαισίου. Τέλος, η ίδια κλάση περιλαμβάνει τη μέθοδο `quitGame()`, η οποία χρησιμοποιείται για τον τερματισμό της εφαρμογής μέσω της `Application.Quit()`, ενώ κατά την εκτέλεση μέσα από το Unity Editor διακόπτει τη λειτουργία `Play Mode`.

3.10.5 Σύνοψη των συμπληρωματικών λειτουργιών

Οι παραπάνω λειτουργίες συμπληρώνουν τους βασικούς μηχανισμούς των μουσικών οργάνων χωρίς να μεταβάλλουν τα φυσικά μοντέλα που χρησιμοποιούνται για το κούρδισμα. Η τυχαία αρχικοποίηση δημιουργεί διαφορετικές καταστάσεις προς διόρθωση, οι λειτουργίες επίδειξης παρουσιάζουν ηχητικά τη λειτουργία των οργάνων, τα ενημερωτικά μηνύματα καθοδηγούν τον χρήστη κατά την εκτέλεση των ασκήσεων και η `StartMenu` διαχειρίζεται την έναρξη, την επανεκκίνηση, τις οδηγίες και τον τερματισμό της εφαρμογής. Με τον τρόπο αυτό οι επιμέρους προσομοιώσεις εντάσσονται σε ένα ενιαίο και ολοκληρωμένο περιβάλλον αλληλεπίδρασης.

4 Μελλοντικές επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας επικεντρώνεται στην προσομοίωση βασικών διαδικασιών κουρδίσματος και αλληλεπίδρασης με το αρπιχόρδιο, τη φλογέρα και το ταμπούρο. Η υπάρχουσα αρχιτεκτονική μπορεί μελλοντικά να αποτελέσει τη βάση για ένα πληρέστερο εκπαιδευτικό περιβάλλον στο οποίο ο χρήστης δεν θα περιορίζεται μόνο στη ρύθμιση του τονικού ύψους, αλλά θα μπορεί να αλληλεπιδρά αναλυτικότερα με τα επιμέρους μηχανικά και ακουστικά μέρη των οργάνων. Οι σημαντικότερες δυνατότητες επέκτασης αφορούν την προσομοίωση εργασιών συντήρησης, την προσθήκη πιο σύνθετων διαδικασιών κουρδίσματος, την ενσωμάτωση περισσότερων μουσικών οργάνων, την αξιοποίηση προηγμένων τεχνολογιών επεξεργασίας ήχου και, τέλος, τη μεταφορά της εφαρμογής σε περιβάλλον εικονικής πραγματικότητας.

4.1.1 Επέκταση σε προσομοίωση εργασιών συντήρησης

Μία βασική κατεύθυνση μελλοντικής ανάπτυξης είναι η αύξηση του βαθμού αλληλεπίδρασης με τα επιμέρους μέρη των μουσικών οργάνων. Στην παρούσα υλοποίηση ο χρήστης αλληλεπιδρά κυρίως με τα στοιχεία που είναι απαραίτητα για την παραγωγή ήχου και το κούρδισμα. Μελλοντικά, περισσότερα τμήματα των τρισδιάστατων μοντέλων θα μπορούσαν να μετατραπούν σε ανεξάρτητα διαδραστικά αντικείμενα, επιτρέποντας την προσομοίωση διαδικασιών συναρμολόγησης, αποσυναρμολόγησης, ελέγχου και συντήρησης.

Στην περίπτωση του αρπιχορδίου, θα μπορούσαν να προστεθούν διαδικασίες όπως ο καθαρισμός των πλήκτρων και των χορδών, ο έλεγχος της κατάστασης και της σταθερότητας των πείρων κουρδίσματος και η επιθεώρηση ή ρύθμιση των γλωσσιδίων. Η εφαρμογή θα μπορούσε, για παράδειγμα, να παρέχει στον χρήστη κατάλληλα εικονικά εργαλεία καθαρισμού και να απαιτεί την εφαρμογή τους στα σωστά τμήματα του οργάνου. Σε περισσότερο προηγμένη μορφή, η κατάσταση ενός εξαρτήματος θα μπορούσε να επηρεάζει και την ακουστική συμπεριφορά του οργάνου, συνδέοντας έτσι τη διαδικασία συντήρησης με το τελικό ηχητικό αποτέλεσμα.

Για τη φλογέρα, μία αντίστοιχη επέκταση θα μπορούσε να περιλαμβάνει τον καθαρισμό του εσωτερικού σωλήνα με εικονική ράβδο καθαρισμού, την απομάκρυνση της υγρασίας και τον έλεγχο του επιστομίου και της ακμής. Θα μπορούσε επίσης να

παρουσιαστεί στον χρήστη η επίδραση που έχουν πιθανές αλλοιώσεις ή επικαθίσεις στην εσωτερική γεωμετρία και στη ροή του αέρα και, κατ' επέκταση, στην ποιότητα και στη σταθερότητα του παραγόμενου ήχου.

Στο ταμπούρο, η συντήρηση θα μπορούσε να επεκταθεί πέρα από τη ρύθμιση των κοχλίων που ήδη αποτελεί μέρος του μηχανισμού κουρδίσματος. Θα μπορούσε να προστεθεί καθαρισμός της μεμβράνης, έλεγχος των μεταλλικών μερών και των κοχλίων, αντικατάσταση της μεμβράνης και ρύθμιση των μεταλλικών χορδών. Ιδιαίτερο εκπαιδευτικό ενδιαφέρον θα είχε η δυνατότητα του χρήστη να μεταβάλλει την τάση των μεταλλικών χορδών και να ακούει το πώς επιδρά η ρύθμιση στη χαρακτηριστική χροιά του οργάνου.

Με αυτόν τον τρόπο, η εφαρμογή θα μπορούσε σταδιακά να εξελιχθεί από προσομοίωση κουρδίσματος σε ένα γενικότερο περιβάλλον εκπαίδευσης σχετικά με τη λειτουργία, ρύθμιση και συντήρηση μουσικών οργάνων.

4.1.2 Προσθήκη πιο σύνθετων συστημάτων κουρδίσματος

Μία δεύτερη κατεύθυνση αφορά την επέκταση του εκπαιδευτικού περιεχομένου πέρα από το ίσο συγκερασμένο σύστημα που χρησιμοποιείται ως βασικό σημείο αναφοράς στην παρούσα εφαρμογή. Ιδιαίτερα στην περίπτωση του αρπιχορδίου, θα μπορούσαν να ενσωματωθούν διαφορετικά ιστορικά συστήματα κουρδίσματος και συγκερασμού, όπως το Πυθαγόρειο κούρδισμα, ο καθαρός συγκερασμός, οι μεσοτονικοί συγκερασμοί και διάφοροι καλοί συγκερασμοί.

Η εφαρμογή θα μπορούσε να επιτρέπει στον χρήστη να επιλέγει το επιθυμητό σύστημα και να υπολογίζει αυτόματα διαφορετικές συχνότητες-στόχους για κάθε νότα. Το ίδιο εικονικό όργανο θα μπορούσε επομένως να χρησιμοποιείται για τη σύγκριση διαφορετικών συστημάτων κουρδίσματος. Ο χρήστης θα είχε τη δυνατότητα όχι μόνο να πραγματοποιεί το κούρδισμα, αλλά και να ακούει τις διαφορές μεταξύ των συστημάτων και να παρατηρεί αριθμητικά τις διαφορετικές αποκλίσεις σε cents.

Μία τέτοια επέκταση θα ήταν ιδιαίτερα κατάλληλη για το αρπικόρδιο, καθώς θα συνέδεε την πρακτική διαδικασία κουρδίσματος με τη θεωρία των ιστορικών συγκερασμών και θα επέτρεπε τη χρήση της προσομοίωσης και ως εργαλείου διδασκαλίας μουσικής ακουστικής.

4.1.3 Επέκταση σε περισσότερα και πιο σύνθετα μουσικά όργανα

Η ίδια αρχιτεκτονική θα μπορούσε να χρησιμοποιηθεί για την προσθήκη περισσότερων μουσικών οργάνων ή πολυπλοκότητας στα ήδη υπάρχοντα της εφαρμογής. Ένα παράδειγμα είναι το *δίσκαλο αρπιχόρδιο*, αρπιχόρδιο δηλαδή με περισσότερες χορδές σε κάθε πλήκτρο. Ιδιαίτερο ενδιαφέρον παρουσιάζει το *πιάνο*, καθώς αποτελεί σημαντικά πιο σύνθετη περίπτωση από το αρπιχόρδιο. Η προσομοίωσή του θα απαιτούσε μεγαλύτερο αριθμό χορδών και μηχανικών εξαρτημάτων. Παράλληλα, η ηχητική προσομοίωση του πιάνου θα δημιουργούσε νέες απαιτήσεις. Σε αντίθεση με την απλή αναπαραγωγή ενός δείγματος με μεταβολή του `AudioSource.pitch`, θα μπορούσαν να χρησιμοποιηθούν πολλαπλά ηχητικά δείγματα της ίδιας νότας για διαφορετικές εντάσεις κρούσης (*velocity layers*), ώστε η ένταση με την οποία ενεργοποιείται ένα πλήκτρο να επηρεάζει όχι μόνο την ένταση αλλά και τη χροιά του παραγόμενου ήχου. Θα μπορούσαν επίσης να προσομοιωθούν φαινόμενα όπως η απόσβεση των χορδών, η λειτουργία των πεταλιών και η συνήχηση μεταξύ χορδών.

Αντίστοιχα, η εφαρμογή θα μπορούσε να επεκταθεί σε άλλα έγχορδα, πνευστά και κρουστά όργανα, με κάθε νέο όργανο να συνοδεύεται από το αντίστοιχο φυσικό μοντέλο παραγωγής ήχου και τις ιδιαίτερες διαδικασίες κουρδίσματος και συντήρησής του. Μια αρκετά απαιτητική περίπτωση είναι αυτή του *εκκλησιαστικού οργάνου* που αν προσομοιωθεί με τη πλήρη λειτουργικότητα του και κούρδισμα σε τρισδιάστατη εφαρμογή πρώτου προσώπου, θα αποτελεί ένα από τα πιο χρήσιμα εργαλεία εκμάθησης του λόγω της δυσκολίας εύρεσης χώρου για φυσική εξάσκηση σε αυτό.

4.1.4 Χρήση προηγμένων συστημάτων επεξεργασίας ήχου

Η προσθήκη πιο σύνθετων οργάνων θα μπορούσε να συνοδευτεί από επέκταση του συστήματος ήχου. Η υπάρχουσα υλοποίηση βασίζεται κυρίως στις ενσωματωμένες δυνατότητες ήχου της Unity, όπως τα `components AudioClip` και `AudioSource` και η ιδιότητα `pitch` του τελευταίου.

Για πιο απαιτητικές προσομοιώσεις θα μπορούσαν να αξιοποιηθούν ακόμα πιο εξειδικευμένες λύσεις ήχου. Ενδεικτικά, *middleware* όπως το FMOD ή το Audiokinetic Wwise θα μπορούσαν να χρησιμοποιηθούν για πιο σύνθετη διαχείριση ηχητικών γεγονότων, παραμέτρων και δυναμικών μεταβάσεων. Για εφαρμογές όπου απαιτείται

περισσότερο προγραμματιζόμενη επεξεργασία ήχου, θα μπορούσαν επίσης να διερευνηθούν τεχνολογίες όπως το Pure Data ή άλλες λύσεις DSP.

Μια τέτοια προσέγγιση θα επέτρεπε η ακουστική συμπεριφορά των οργάνων να εξαρτάται από περισσότερες παραμέτρους της προσομοίωσης και όχι αποκλειστικά από τη μεταβολή της ταχύτητας αναπαραγωγής ενός ηχητικού δείγματος. Θα μπορούσαν έτσι να εξεταστούν δυναμικές μεταβολές της χροιάς, διαφορετικά επίπεδα έντασης, συντονισμοί, αποσβέσεις και αλληλεπιδράσεις μεταξύ διαφορετικών τμημάτων ενός οργάνου.

4.1.5 Επέκταση σε περιβάλλον εικονικής πραγματικότητας

Μία περισσότερο ολοκληρωμένη μελλοντική εξέλιξη της εφαρμογής θα ήταν η μεταφορά της σε περιβάλλον εικονικής πραγματικότητας (Virtual Reality). Η αλληλεπίδραση στην παρούσα εργασία βασίζεται στο ποντίκι και στο πληκτρολόγιο, επομένως οι κινήσεις του χρήστη μετατρέπονται σε ενέργειες πάνω στα εικονικά αντικείμενα. Σε ένα σύστημα εικονικής πραγματικότητας, οι ίδιες ενέργειες θα μπορούσαν να πραγματοποιούνται μέσω των χειριστηρίων ή της παρακολούθησης των χειρών.

Για παράδειγμα, ο χρήστης θα μπορούσε να πιάσει το κλειδί κουρδίσματος, να το τοποθετήσει πάνω στον αντίστοιχο εικονικό πείρο ή κοχλία και να εκτελέσει την περιστροφική κίνηση με το χέρι του. Αντίστοιχα, θα μπορούσε να καλύπτει τις εικονικές οπές της φλογέρας με τα δάκτυλά του, να χτυπά διαφορετικά σημεία της μεμβράνης του ταμπούρου ή να χρησιμοποιεί εργαλεία καθαρισμού και συντήρησης πάνω στα κατάλληλα εξαρτήματα.

Η ενσωμάτωση τεχνολογιών απτικής ανάδρασης (haptic feedback) θα μπορούσε να ενισχύσει περαιτέρω αυτή την προσέγγιση, παρέχοντας στον χρήστη αισθητηριακή πληροφορία κατά την περιστροφή ενός κοχλία, την πίεση ενός πλήκτρου ή την επαφή με κάποιο εξάρτημα. Έτσι, η εφαρμογή θα μπορούσε να εξελιχθεί από μία προσομοίωση που εκτελείται σε συμβατική οθόνη σε ένα περισσότερο ολοκληρωμένο εικονικό εργαστήριο μουσικών οργάνων, στο οποίο η θεωρία της ακουστικής θα συνδέεται με την πρακτική εξάσκηση στο κούρδισμα, στον χειρισμό και στη συντήρηση των οργάνων.

4.1.6 Συνολική προοπτική

Οι παραπάνω επεκτάσεις δεν απαιτείται να αλλάξουν τη βασική φιλοσοφία της εφαρμογής, αλλά μπορούν να αναπτυχθούν πάνω στη δομή που έχει ήδη δημιουργηθεί. Η προσθήκη λεπτομερέστερων αλληλεπιδράσεων και εργασιών συντήρησης θα διευρύνει το αντικείμενο της προσομοίωσης, ενώ η υποστήριξη ιστορικών συστημάτων κουρδίσματος και πιο σύνθετων μουσικών οργάνων θα αυξήσει το εκπαιδευτικό της περιεχόμενο. Παράλληλα, η χρήση προηγμένων συστημάτων επεξεργασίας ήχου μπορεί να οδηγήσει σε ακριβέστερη ακουστική αναπαράσταση, ενώ η ανάπτυξη έκδοσης εικονικής πραγματικότητας μπορεί να προσφέρει περισσότερο άμεση και φυσική αλληλεπίδραση. Μακροπρόθεσμα, οι κατευθύνσεις αυτές θα μπορούσαν να οδηγήσουν στη δημιουργία ενός ολοκληρωμένου εικονικού εκπαιδευτικού περιβάλλοντος για τη μελέτη της λειτουργίας, του κουρδίσματος και της συντήρησης διαφορετικών κατηγοριών μουσικών οργάνων.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κ. Βασίλη Ζαφειρόπουλο, για την πολύτιμη καθοδήγηση, την επιστημονική του υποστήριξη και κυρίως, για την αμέριστη υπομονή που επέδειξε καθ' όλη τη διάρκεια της ανάπτυξης της εφαρμογής.

Θα ήθελα ακόμη να ευχαριστήσω τους δημιουργούς που διέθεσαν το έργο τους και συνέβαλαν με αυτόν τον τρόπο στην υλοποίηση της εφαρμογής. Συγκεκριμένα, ευχαριστώ τους 3d_molier, NRG_3D και tgarch3d για τη δημιουργία των τρισδιάστατων μοντέλων του αρπιχορδίου, του ταμπούρου και της φλογέρας αντίστοιχα, καθώς και το Gwangju Arts & Culture Humanities Story Platform για τη διάθεση του τρισδιάστατου μοντέλου των εσωτερικών χώρων που χρησιμοποιήθηκαν για τη διαμόρφωση του περιβάλλοντος της εφαρμογής. Για το ηχητικό υλικό, ευχαριστώ τον rjcohen για τα δείγματα ήχου του αρπιχορδίου που διέθεσε μέσω του Freesound, καθώς και τους δημιουργούς και τις πλατφόρμες Tomplay και Pixabay, από τις οποίες προήλθαν τα ηχητικά δείγματα της φλογέρας και του ταμπούρου αντίστοιχα. Η δουλειά και η διάθεση του υλικού τους αποτέλεσαν σημαντική βοήθεια για την ολοκλήρωση της παρούσας εφαρμογής.

Τέλος, αισθάνομαι την ανάγκη να ευχαριστήσω από καρδιάς τους στενούς μου ανθρώπους, την οικογένεια και τους φίλους μου. Η ψυχική τους υποστήριξη, η υπομονή και η συνεχής ενθάρρυνσή τους ήταν καθοριστικής σημασίας για να φέρω εις πέρας αυτό το έργο.

Αποθετήριο πηγαίου κώδικα

Ο πηγαίος κώδικας και τα αρχεία του έργου Unity που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας είναι διαθέσιμα στο ακόλουθο αποθετήριο:
<https://github.com/FedonStylianidis/Musical-Instruments-3d-Tuning-Simulator>

Βιβλιογραφία

- Barbour, J. M. (2004). *Tuning and Temperament: A Historical Survey*. Dover Publications.
- Benade, A. H. (1990). *Fundamentals of Musical Acoustics: Second, Revised Edition*. Dover Publications.
- Berg, R., & Stork, D. (2005). *The Physics of Sound, 3rd Edition*. Pearson.
- Christensen, T. (Ed.). (2002). *The Cambridge History of Western Music Theory*. Cambridge University Press. <https://doi.org/10.1017/CHOL9780521623711>
- Denis, J. (1987). *Treatise on Harpsichord Tuning* (V. J. Panetta, Ed. & Trans.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511627286>
- Denisova, A., & Cairns, P. (2015). First Person vs. Third Person Perspective in Digital Games: Do Player Preferences Affect Immersion? *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15*, 145–148. <https://doi.org/10.1145/2702123.2702256>
- Fletcher, N. H., & Rossing, T. (2008). *The Physics of Musical Instruments* (2nd ed.). Springer Science & Business Media.
- Goldstone, W. (2009). *Unity Game Development Essentials: If you have ambitions to be a game developer this guide is a must. Covering all the fundamentals of the Unity game engine, it will help you understand the different elements of 3D game creation through practical projects*. Packt Publishing Ltd.
- Hocking, J., & Hocking, J. (2022). *Unity in Action, Third Edition: Multiplatform Game Development in C#*. Simon and Schuster.
- Howard, D. M., & Angus, J. (2017). *Acoustics and Psychoacoustics* (5th ed.). Routledge. <https://doi.org/10.4324/9781315716879>

- Jorgensen, O. H. (with Internet Archive). (1991). *Tuning: Containing the Perfection of Eighteenth-Century Temperament*. East Lansing, Mich.: Michigan State University Press. <http://archive.org/details/tuningcontaining0000jorg>
- Kottick, E. L. (2016). *A History of the Harpsichord*. Indiana University Press.
- Menard, M. (2015). *Game Development with Unity* (2nd ed.). Cengage Learning.
- Mystakidis, S., Besharat, J., Papantzikos, G., Christopoulos, A., Stylios, C., Agorgianitis, S., & Tselentis, D. (2022). Design, Development, and Evaluation of a Virtual Reality Serious Game for School Fire Preparedness Training. *Education Sciences*, 12(4), 281. <https://doi.org/10.3390/educsci12040281>
- Rossing, T. D., Moore, F. R., & Wheeler, P. A. (2002). *The Science of Sound*, 3rd Edition. Pearson.
- Toulson, R., & Toulson, R. (2021). *Drum Sound and Drum Tuning: Bridging Science and Creativity*. Focal Press. <https://doi.org/10.4324/9781003104209>

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.