ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη ψυχαγωγικού λογισμικού με Unity»

Φοιτητής

Μπούτης Θωμάς 514311

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2025

Ανάπτυξη ψυχαγωγικού λογισμικού με Unity

Κωδικός: 24185

Φοιτητής: Μπούτης Θωμάς

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 31-03-2024

Ημερομηνία περάτωσης Π.Ε. 12-09-2025

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

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

Περίληψη

Οι μηχανές παιχνιδιών αποτελούν βασικά εργαλεία για τη δημιουργία σύγχρονων παιχνιδιών και εφαρμογών. Σήμερα, δημοφιλείς μηχανές όπως η Unity, η Unreal Engine και η Godot χρησιμοποιούνται ευρέως από επαγγελματίες και ανεξάρτητους δημιουργούς, χάρη στη φιλικότητά τους προς τον χρήστη και τις πολυάριθμες δυνατότητες που προσφέρουν. Η πρόοδος αυτών των μηχανών έχει επαναστατήσει στην ανάπτυξη παιχνιδιών, επιτρέποντας τη δημιουργία πολύπλοκων 3D κόσμων, την ενσωμάτωση τεχνητής νοημοσύνης και την αποδοτική διαχείριση των συστημάτων. Αυτό έχει επιτρέψει στους χρήστες να απολαμβάνουν παιχνίδια υψηλής ποιότητας σε μια πληθώρα πλατφορμών, από κινητές συσκευές μέχρι κονσόλες και υπολογιστές. Οι εφαρμογές παιχνιδιών έχουν αποκτήσει μεγάλη σημασία στην κοινωνία σήμερα, καθώς πέρα από την ψυχαγωγία, παρέχουν εκπαιδευτικά οφέλη, ευκαιρίες κοινωνικοποίησης και ενίσχυση δεξιοτήτων μέσω διαδραστικών εμπειριών. Παιχνίδια που συνδυάζουν μάθηση και κοινωνικές αλληλεπιδράσεις έχουν κερδίσει τη δημοτικότητα των χρηστών, δημιουργώντας μια παγκόσμια κοινότητα παικτών που συνδέονται και αλληλεπιδρούν σε διάφορα επίπεδα. Στόχος της παρούσας πτυχιακής εργασίας είναι η ανάδειξη της σημασίας τέτοιων εφαρμογών σε μία σειρά από τομείς της καθημερινότητας, όπως η διασκέδαση,ο αθλητισμός, μέσω ενός απλού παιχνιδιού/εφαρμογής (quiz application). Αξιοποιώντας τα εργαλεία της, κατάφερα να δημιουργήσω ένα quiz application ψυχαγωγικού περιεχομένου, το οποίο υποστηρίζει δύο εκδόσεις. Την ατομική (SinglePlayer Option), στην οποία ο χρήστης παίζει μόνος του το κουίζ και την ομαδική (MultiPlayer Option), στην οποία δύο παίκτες παίζουν αντίπαλοι. Ταυτόχρονα, ο χρήστης έχει την επιλογή να παίξει τόσο σε τοπικό (local), όσο και σε διαδικτυακό (online) επίπεδο. Την ίδια στιγμή, μπορεί να δημιουργήσει το δικό του προφίλ αποθηκεύοντας σκορ, νίκες και ήττες που σημειώνει στο παιχνίδι. Σε αυτό το έγγραφο, αρχικά, γίνεται μία παρουσίαση μιας σειράς μηχανών παιχνιδιών που επιτρέπουν τέτοιου τύπου εφαρμογές (applications), καθώς και μία μεταξύ τους σύγκριση, η οποία μας οδήγησε στην επιλογή της Unity. Στη συνέχεια, πραγματοποιείται μία αναλυτική περιγραφή των βασικών δομών της Unity, τόσο από πλευράς λογισμικού, όσο και από πλευράς γραφικού περιβάλλοντος. Ακολουθεί η ανάλυση της δικής μας εφαρμογής σε τεχνικό επίπεδο. Τέλος, παρατίθενται μία σειρά από συμπεράσματα και δυνατότητες εξέλιξης της εφαρμογής.

«Development of entertainment software with Unity»

Abstract

Game engines play a vital role in the development of modern games and applications. Today, popular engines like Unity, Unreal Engine, and Godot are extensively used by both professional developers and independent creators due to their ease of use and the powerful features they provide. The evolution of these engines has transformed game development, allowing for the creation of intricate 3D environments, the integration of artificial intelligence, and optimized resource management. As a result, users can now enjoy high-quality games on a variety of platforms, ranging from mobile devices to consoles and PCs. Game applications have become an integral part of today's society, offering not only entertainment but also educational value, opportunities for social interaction, and skill development through engaging, interactive experiences. Games that combine educational elements and social interactions have gained immense popularity, fostering a global community of players who connect and engage with each other at different levels. The objective of this thesis is to emphasize the importance of such applications in various aspects of daily life, including entertainment and sports, by presenting a simple game/application (a quiz game) that highlights these qualities. By utilizing the available tools, I was able to create an entertainment quiz application, which supports two modes. The single-player option, where the user plays the quiz alone, and the multiplayer option, where two players compete against each other. Simultaneously, the user can choose to play both locally and online. At the same time, they can create their profile by saving their scores, wins, and losses in the game. This document initially presents a range of game engines that allow for such applications, along with a comparison between them, which led us to the choice of Unity. Then, a detailed description of Unity's core structures is provided, both from a software and graphical environment perspective. This is followed by an analysis of our application on a technical level. Finally, a series of conclusions and potential avenues for further development of the application are presented.

Ευχαριστίες

Ευχαριστώ πολύ τον καθηγητή κ.Κυριάκο Τσιακμάκη για την ανάθεση και επίβλεψη της παρούσας πτυχιακής εργασίας.

Ευχαριστώ ακόμη τους συμφοιτητές και φίλους μου που ήταν δίπλα μου και εύχομαι σε όλους να πετύχουν τα όνειρά τους ότι και αν επιλέξουν να κάνουν.

Εν κατακλείδι, θέλω να ευχαριστήσω τους ανθρώπους μου και την οικογένειά μου, που στήριξαν, στηρίζουν και θα στηρίξουν τις προσπάθειές μου!

Περιεχόμενα

Περίληψη iv
Abstract v
Ευχαριστίες vi
Περιεχόμενα vii
Κατάλογος Σχημάτων ix
Κεφάλαιο 1ο: Εισαγωγή 10
1.1 Στόχος και περιγραφή εφαρμογής 11
1.1.1 Κεφάλαιο 1 - Επισκόπηση εφαρμογής 12
1.2 Δομή της εργασίας 12
1.2.1 Κεφάλαιο 1 - Παρουσίαση εφαρμογής 12
1.2.2 Κεφάλαιο 2 - Μηχανές ανάπτυξης παιχνιδιών και βιντεοπαιχνίδια 13
1.2.3 Κεφάλαιο 3 - Η Μηχανή Σχεδίασης και Ανάπτυξης εφαρμογών Unity 14
1.2.4 Κεφάλαιο 4 - Σχεδιασμός και υλοποίηση εφαρμογής στη Unity 15
1.2.5 Κεφάλαιο 5 - Σύνοψη, συμπεράσματα και προτάσεις βελτίωσης 15
Κεφάλαιο 2ο: Μηχανές παιχνιδιών και βιντεοπαιχνίδια 16
2.1 Ορισμός βιντεοπαιχνιδιών 16
2.2 Ιστορία και εξέλιξη βιντεοπαιχνιδιών 16
2.3 Μηχανές παιχνιδιών 20
2.3.1 Unreal Engine 20
2.3.2 GameMaker 22
2.3.3 Godot 23
2.3.4 GDevelop 24
2.3.5 Unity 3D 25
2.4 Σύγκριση μηχανών παιχνιδιών 26
2.4.1 Πλεονεκτήματα και μειονεκτήματα Unity 26
Κεφάλαιο 3ο: Η μηχανή εφαρμογών Unity 28
3.1 Unity - Μια ιστορική αναδρομή 28
3.2 Δομικά στοιχεία Unity 30
3.2.1 Asset Store 30
3.2.2 Γλώσσα προγραμματισμού C# 30
3.2.3 Κλάση Monobehaviour στη Unity 30
3.3 Περιβάλλον διεπαφής Unity Editor 31
3.3.1 Παράθυρο Project (Project Window) 32
3.3.2 Προβολή σκηνής (Scene View) 33

3.3.3 Προβολή παιχνιδιού (Game View) 34
3.3.4 Παράθυρο Ιεραρχίας (Hierarchy Window) 35
3.3.5 Παράθυρο Επιθεωρητή/Παρατηρητή (Inspector Window) 35
3.3.6 Κονσόλα (Console) 36
Κεφάλαιο 4ο: Υλοποίηση εφαρμογής στη Unity 37
4.1 Δημιουργία και στήσιμο σκηνής 37
4.2 Εισαγωγή κώδικα (script) - Σύνδεση σκηνών 43
4.3 Τοποθέτηση μουσικής - βίντεο κανόνων 45
4.4 Εισαγωγή ερωτήσεων με Google Sheet URL μέσω UnityWebRequest 50
4.5 Online MatchMaking MultiPlayer με Photo Pun 2 55
Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης 61
5.1 Συμπεράσματα υλοποίησης εφαρμογής 61
5.2 Δυνατότητες εξέλιξης 62
ΒΙΒΛΙΟΓΡΑΦΙΑ 63
ΠΑΡΑΡΤΗΜΑ Α 65
ΠΑΡΑΡΤΗΜΑ Β 71

Κατάλογος Σχημάτων

Εικόνα 2.1: Η κονσόλα Magnavox Odyssey

Εικόνα 2.2: Η κονσόλα Nintendo Entertainment System(NES)

Εικόνα 2.3: Η κονσόλα GameBoy της Nintendo

Εικόνα 2.4: Η κονσόλα Nintendo Wii με αισθητήρες κίνησης

Εικόνα 2.5: Oculus Rift S VR headset (Οπτικοακουστικό σετ εικονικής πραγματικότητας)

Εικόνα 2.6: Σκηνή στο πρόγραμμα Unreal Engine

Εικόνα 2.7: Σκηνή στο πρόγραμμα GameMaker

Εικόνα 2.8: Σκηνή στο πρόγραμμα Godot

Εικόνα 2.9: Σκηνή στο πρόγραμμα GDevelop

Εικόνα 3.1: Περιβάλλον διεπαφής Unity

Εικόνα 3.2: Βασικές συναρτήσεις της κλάσης Monobehaviour

Εικόνα 3.3: Στιγμιότυπο οθόνης του Παραθύρου Project (Project Window) στη Unity

Εικόνα 3.4: Στιγμιότυπου οθόνης από αναζήτηση στο Παράθυρο Project (Project Window)

Εικόνα 3.5: Στιγμιότυπο οθόνης της Προβολής Σκηνής (Scene View) στη Unity

Εικόνα 3.6: Στιγμιότυπο οθόνης της Προβολής Παιχνιδιού (Game View) στη Unity

Εικόνα 3.7: Στιγμιότυπο οθόνης του Παραθύρου Ιεραρχίας (Hierarchy Window) στη Unity

Εικόνα 3.8: Στιγμιότυπο οθόνης του Παραθύρου Επιθεωρητή/Παρατηρητή (Inspector Window)

Εικόνα 3.9: Στιγμιότυπο οθόνης της Κονσόλας (Console) στη Unity

Εικόνα 4.1: Ορισμός Background εικόνας στο παράθυρο Επιθεωρητή (Inspector Window)

Εικόνα 4.2: Τοποθέτηση εικόνας στο Image Component στο Inspector Window

Εικόνα 4.3: Ο Canvas της πρώτης σκηνής (SampleScene) στο Inspector Window

Εικόνα 4.4: Προβολή των αντικειμένων της SampleScene σκηνής στο Scene View & Game View

Εικόνα 4.5: Προβολή της Multimenuscene

Εικόνα 4.6: Ρύθμιση OnClick() σε κουμπί ώστε να εκτελεί μια συνάρτηση

Εικόνα 4.7: Ρύθμιση Background μουσικής

Εικόνα 4.8: Εισαγωγή Google Sheet CSV

Εικόνα 4.9: Δημιουργία project στο Photon Dashboard

Εικόνα 4.10: AppID στο Photon Dashboard

Εικόνα 4.11: Προβολή της OnlinaMatchMenu σκηνής

Κεφάλαιο 1 – Εισαγωγή

1.1 Στόχος και περιγραφή εφαρμογής

Στην σημερινή εποχή της ψηφιακής εκπαίδευσης, της συνεχούς συνδεσιμότητας και της διαδραστικής ψυχαγωγίας, οι εφαρμογές που συνδυάζουν γνώση και ψυχαγωγία (παιχνίδι), αποκτούν διαρκώς αυξανόμενη ζήτηση, τόσο σε εκπαιδευτικό επίπεδο όσο και σε προσωπικό. Η ανάπτυξη εφαρμογών τύπου **quiz game** μέσα από πλατφόρμες όπως η **Unity** αντανακλά αυτή την τάση, προσφέροντας εμπειρίες που είναι ταυτόχρονα εκπαιδευτικές, ανταγωνιστικές και διασκεδαστικές. Μέσω της Unity, παρέχεται η δυνατότητα δημιουργίας ενός ευέλικτου και δυναμικού περιβάλλοντος, στο οποίο συνδυάζονται εκπαιδευτικά στοιχεία, μηχανισμοί πρόκλησης και επιβράβευσης και ένα ελκυστικό γραφικό περιβάλλον που κρατά το ενδιαφέρον του χρήστη ενεργό. Ταυτόχρονα, το γραφικό περιβάλλον και η υποστήριξη animation, ήχου και διασύνδεσης με δεδομένα, ενισχύουν την εμπλοκή του χρήστη, προσφέροντας μια ολοκληρωμένη και απολαυστική εμπειρία. Τέτοιου είδους εφαρμογές εξελίσσονται πλέον πέρα από την απλή παρουσίαση ερωτήσεων-απαντήσεων, ενσωματώνοντας multiplayer λειτουργίες, επιλογές δυσκολίας, προφίλ παίκτη, εξατομικευμένα στατιστικά, καθώς και βοηθήματα στρατηγικής που εμπλουτίζουν σημαντικά τη συνολική εμπειρία. Η τεχνολογική πρόοδος και η διάδοση εργαλείων ανάπτυξης έχουν καταστήσει πιο προσιτή από ποτέ τη δημιουργία τέτοιων εφαρμογών, ακόμα και από μικρές ομάδες ή μεμονωμένους προγραμματιστές. Παράλληλα, η δυνατότητα για διασύνδεση με βάσεις δεδομένων, online scoreboards, συνεχείς ενημερώσεις περιεχομένου και προσαρμοστικότητα στην απόδοση του παίκτη ανοίγει τον δρόμο για ακόμη πιο σύνθετα και δυναμικά παιχνίδια γνώσεων στο μέλλον.

Η δημιουργία μιας τέτοιας εφαρμογής δεν είναι απλώς μία εκπαιδευτική ή ψυχαγωγική άσκηση· αποτελεί ένα σύγχρονο εργαλείο μάθησης και αλληλεπίδρασης, αξιοποιώντας πλήρως τις δυνατότητες της ψηφιακής εποχής.

Η πτυχιακή εργασία εστιάζει στη μελέτη, τον σχεδιασμό και την ανάπτυξη ενός ψυχαγωγικού λογισμικού με τη μηχανή παιχνιδιών Unity. Στο πλαίσιο της δημιουργίας του παιχνιδιού, ανέλαβα τον πλήρη σχεδιασμό των σκηνών και την οργάνωση της σύνδεσης τους, προκειμένου να εξασφαλιστεί η ομαλή ροή και η αλληλεπίδραση του χρήστη με την εφαρμογή. Με την επιλογή προσεγμένων ήχων για τις διάφορες ενέργειες, από τις απαντήσεις στις ερωτήσεις μέχρι τις αλληλεπιδράσεις με τα power-ups, το παιχνίδι αποκτά έναν πιο ζωντανό και ρεαλιστικό χαρακτήρα.

Η δομή της συγκεκριμένης εφαρμογής/παιχνιδιού **(quiz application)**, είναι βασισμένη στη δομή ενός αγώνα μπάσκετ. Ο χρήστης θα έχει τη δυνατότητα να περιηγηθεί στο χώρο της εφαρμογής, να

παίξει το παιχνίδι είτε ατομικά **(singleplayer)** είτε ομαδικά **(multiplayer)**, τόσο σε τοπικό (local) όσο και σε διαδικτυακό (online) επίπεδο, καθώς και να εξελίξει τις γνώσεις του σε αθλητικό επίπεδο. Για το multiplayer κομμάτι σε online επίπεδο, υλοποίησα ένα σύστημα **matchmaking** που επιτρέπει στους χρήστες να συνδέονται εύκολα και γρήγορα στο διαδίκτυο, εξασφαλίζοντας μια απρόσκοπτη εμπειρία online παιχνιδιού. Το σύστημα αυτό επιτρέπει στους παίκτες να αναζητούν αντιπάλους και να συνδέονται με άλλους χρήστες μέσω ενός απλού και αποτελεσματικού μηχανισμού, με τη δυνατότητα να παίζουν σε πραγματικό χρόνο. Το παιχνίδι επίσης ενσωματώνει έναν ευέλικτο μηχανισμό για την εισαγωγή ερωτήσεων εκτός της Unity, επιτρέποντας την εύκολη προσθήκη και τροποποίηση του περιεχομένου χωρίς να απαιτούνται αλλαγές στον βασικό κώδικα του παιχνιδιού, παρέχοντας στους χρήστες μια πιο προσαρμοσμένη και επεκτάσιμη εμπειρία.

Το παιχνίδι αποτελείται από τρεις κατηγορίες ερωτήσεων αυξανόμενης βαθμίδας, οι οποίες ανεβάζουν την ένταση και δυσκολία, όπως ακριβώς και οι διαφορετικές φάσεις του μπάσκετ. Το επίπεδο ένα ή αλλιώς βολή όπως χρησιμοποιείται επίσημα και σε έναν αγώνα μπάσκετ, η οποία αποτελεί το χαμηλότερο βαθμό δυσκολίας και δίνει στον παίκτη έναν πόντο, Το επίπεδο δύο ή αλλιώς δίποντο, αποτελεί την μεσαία βαθμίδα δυσκολίας και δίνει δύο πόντους ενώ η τρίτη κατηγορία και πιο δύσκολη αποτελεί το τρίποντο το οποίο προσθέτει τρεις πόντους. Επίσης, ο κάθε παίκτης έχει στη διάθεση του και ορισμένες βοήθειες (power-ups) που αντικατοπτρίζουν πλήρως τις ορολογίες που συναντά κανείς σε πραγματικούς αγώνες μπάσκετ, προσφέροντας στον χρήστη τη δυνατότητα να επηρεάσει την πορεία του παιχνιδιού με διαφορετικούς τρόπους. Οι βασικές βοήθειες περιλαμβάνουν:

1. **Μπλοκ (Block)**: Μπλοκάρει την ερώτηση του αντιπάλου πριν αυτός απαντήσει, στερώντας του στην ουσία μία ερώτηση.
2. **Κλέψιμο (Steal)**: Επιτρέπει στον παίκτη να κλέψει την ερώτηση του αντιπάλου πριν αυτός απαντήσει και να την απαντήσει ο ίδιος.
3. **Ριμπάουντ (Rebound)**: Δίνει μια δεύτερη ευκαιρία στον παίκτη αν ο αντίπαλος απαντήσει λάθος στην ερώτηση, επιτρέποντάς του να ξαναπροσπαθήσει.
4. **Ασίστ (Assist)**: Μειώνει τον αριθμό των λανθασμένων απαντήσεων στις ερωτήσεις αφήνοντας μια σωστή και μια λανθασμένη απάντηση (50-50), βοηθώντας τον παίκτη να έχει μεγαλύτερη πιθανότητα επιτυχίας.
5. **Φάουλ (Foul)**: Αναγκάζει τον αντίπαλο να απαντήσει σε ερωτήσεις χαμηλότερης δυσκολίας, προκαλώντας τον να «πληρώσει» για μια λάθος κίνηση.

Αυτές οι στρατηγικές επιλογές προσφέρουν επιπλέον βάθος στο παιχνίδι, δίνοντας τη δυνατότητα στους παίκτες να προσαρμόζουν την πορεία του παιχνιδιού και να βελτιώνουν την απόδοσή τους, ανάλογα με τις ανάγκες της στιγμής.

1.1.1 Κανόνες παιχνιδιού (Quiz)

Στην παρακάτω παράγραφο αναφέρονται οι κανόνες του παιχνιδιού. Ο χρήστης μπορεί να παίξει είτε σε Single Player είτε σε Local Multiplayer επίπεδο, ενώ υπάρχει και δυνατότητα Online Multiplayer για να ανταγωνιστεί φίλους μέσω διαδικτύου. Το παιχνίδι, τόσο στο SinglePlayer όσο και στο Multiplayer, αποτελείται από τέσσερις γύρους, με οκτώ ερωτήσεις σε κάθε γύρο. Στο Single Player όλες οι ερωτήσεις του γύρου απαντώνται από έναν παίκτη, ενώ στο Multiplayer κάθε παίκτης απαντά σε τέσσερις ερωτήσεις ανά γύρο εναλλάξ. Στο παιχνίδι υπάρχουν βοηθήματα που μπορούν να χρησιμοποιηθούν για να διευκολύνουν τις απαντήσεις: στο Single Player είναι διαθέσιμα τρία, ενώ στο Multiplayer πέντε για κάθε παίκτη. Τα βοηθήματα μπορεί να χρησιμοποιηθούν μόνο μία φορά από κάθε παίκτη. Οι ερωτήσεις έχουν διαφορετικά επίπεδα δυσκολίας: το Level 1 (βολή) δίνει 1 πόντο, το Level 2 (δίποντο) δίνει 2 πόντους και το Level 3 (τρίποντο) δίνει 3 πόντους. Το παιχνίδι ολοκληρώνεται όταν όλοι οι γύροι έχουν παιχτεί και ο νικητής καθορίζεται ανάλογα με το σκορ: στο Single Player το παιχνίδι τερματίζει και αποθηκεύεται το καλύτερο σκορ του παίκτη, ενώ στο Multiplayer ο παίκτης με τους περισσότερους πόντους ανακηρύσσεται νικητής και το αποτέλεσμα αποθηκεύεται ως νίκες/ήττες.

1.2 Δομή της εργασίας

1.2.1 Κεφάλαιο 1 - Εισαγωγή

Στο πρώτο κεφάλαιο της πτυχιακής εργασίας, γίνεται επισκόπηση της εφαρμογής και παρουσίαση των βασικών κανόνων του παιχνιδιού. Παρουσιάζεται η βασική δομή του παιχνιδιού, οι δυνατότητες που προσφέρει στους χρήστες, καθώς και τα στοιχεία που το συνθέτουν. Εστιάζει στην αυξανόμενη ανάγκη για την ανάπτυξη διαδραστικών εφαρμογών, όπως τα quiz παιχνίδια, που συνδυάζουν μάθηση και ψυχαγωγία. Παρουσιάζεται η σημασία της ανάπτυξης αυτών των εφαρμογών, καθώς οι χρήστες αναζητούν διασκεδαστικούς και έξυπνους τρόπους αλληλεπίδρασης.

1.2.2 Κεφάλαιο 2 - Βιντεοπαιχνίδια και μηχανές παιχνιδιών

Το κεφάλαιο αυτό παρουσιάζει μια γενική εικόνα για τα βιντεοπαιχνίδια και τα εργαλεία που χρησιμοποιούνται για τη δημιουργία τους. Αρχικά, γίνεται προσπάθεια να εξηγηθεί τι είναι ένα βιντεοπαιχνίδι, ποια είναι τα βασικά του στοιχεία και πώς ο παίκτης συμμετέχει σε αυτό μέσα από τη διάδραση. Τα βιντεοπαιχνίδια εξετάζονται τόσο ως μέσο ψυχαγωγίας όσο και ως τρόπος έκφρασης που συνδυάζει την τεχνολογία με τη δημιουργικότητα. Παράλληλα, τονίζεται πώς η εξέλιξή τους επηρεάζεται από την πρόοδο της ψηφιακής εποχής.

Στη συνέχεια, παρουσιάζεται η ιστορική πορεία των βιντεοπαιχνιδιών και των παιχνιδομηχανών, από τις πρώτες, απλές ηλεκτρονικές συσκευές της δεκαετίας του 1970 έως τα σύγχρονα εξελιγμένα συστήματα που προσφέρουν ρεαλιστικά γραφικά, online δυνατότητες και προηγμένη τεχνητή νοημοσύνη. Μέσα από αυτή την αναδρομή, γίνεται φανερή η συνεχής τεχνολογική πρόοδος, καθώς και η διαρκώς αυξανόμενη απήχηση που έχουν τα παιχνίδια στο ευρύ κοινό.

Ιδιαίτερη έμφαση δίνεται επίσης στις μηχανές ανάπτυξης παιχνιδιών (game engines), δηλαδή στις πλατφόρμες που επιτρέπουν στους δημιουργούς να σχεδιάσουν, να οργανώσουν και να υλοποιήσουν τις ιδέες τους, μετατρέποντάς τες σε ολοκληρωμένα διαδραστικά προϊόντα. Παρουσιάζονται συνοπτικά μερικές από τις πιο δημοφιλείς μηχανές παιχνιδιών, όπως η Unreal Engine, το GameMaker Studio και η Godot, αναλύοντας για κάθε μία τα βασικά πλεονεκτήματα, τις δυνατότητες που προσφέρει και τις βασικές της αδυναμίες.

Ιδιαίτερη αλλά σύντομη αναφορά γίνεται και στη Unity, την οποία εξετάζουμε μόνο σε επίπεδο γενικής παρουσίασης των θετικών και αρνητικών της χαρακτηριστικών, καθώς στο επόμενο κεφάλαιο ακολουθεί αναλυτική παρουσίαση της λειτουργίας της μέσα από την πράξη. Η Unity, λόγω της ευρείας χρήσης της και της ευκολίας που προσφέρει τόσο σε αρχάριους όσο και σε έμπειρους προγραμματιστές, αποτελεί το βασικό εργαλείο ανάπτυξης του παιχνιδιού γνώσεων που παρουσιάζεται σε αυτή την εργασία.

Συνολικά, το κεφάλαιο αυτό στοχεύει να δώσει στον αναγνώστη το απαραίτητο υπόβαθρο τόσο για την κατανόηση της εξέλιξης των βιντεοπαιχνιδιών, όσο και για την κατανόηση του ρόλου που διαδραματίζουν οι σύγχρονες μηχανές ανάπτυξης στον τομέα της ψηφιακής ψυχαγωγίας. Παράλληλα, λειτουργεί ως γέφυρα που συνδέει τη θεωρητική προσέγγιση με την πρακτική εφαρμογή που αναλύεται στο επόμενο κεφάλαιο.

1.2.3 Κεφάλαιο 3 - Η μηχανή εφαρμογών Unity

Στο κεφάλαιο αυτό, παρουσιάζεται αναλυτικά η λειτουργία της Unity, μιας από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες μηχανές ανάπτυξης παιχνιδιών. Η Unity προσφέρει ένα πλήρως οπτικό και αλληλεπιδραστικό περιβάλλον εργασίας, το οποίο αποτελείται από μια σειρά παραθύρων και εργαλείων, σχεδιασμένων ώστε να διευκολύνουν τον δημιουργό στη σχεδίαση, οργάνωση και λειτουργία μιας εφαρμογής ή ενός παιχνιδιού, ακόμη και αν βρίσκεται σε πρώιμο στάδιο γνώσεων. Κάθε παράθυρο (π.χ. Hierarchy, Scene, Game, Inspector, Project) εξυπηρετεί διαφορετικό σκοπό: το παράθυρο Scene επιτρέπει τη διαμόρφωση του εικονικού κόσμου και των αντικειμένων, το παράθυρο Game εμφανίζει σε πραγματικό χρόνο πώς φαίνεται το παιχνίδι στον τελικό χρήστη, ενώ το παράθυρο επιθεωρητή (Inspector) παρέχει πρόσβαση στις ιδιότητες κάθε επιλεγμένου αντικειμένου. Το παράθυρο ιεραρχίας (Hierarchy) εμφανίζει όλα τα αντικείμενα της σκηνής, και το Project περιλαμβάνει όλα τα αρχεία του έργου. Αυτή η διάρθρωση προσφέρει μια οργανωμένη και ευέλικτη προσέγγιση στην ανάπτυξη, επιτρέποντας ακόμη και σε αρχάριους χρήστες να δημιουργήσουν λειτουργικά και ελκυστικά διαδραστικά περιβάλλοντα.

Η Unity αξιοποιείται ως το βασικό εργαλείο σχεδίασης και υλοποίησης του παιχνιδιού, προσφέροντας ένα ευρύ φάσμα δυνατοτήτων που καλύπτουν τόσο τον τεχνικό όσο και τον οπτικό σχεδιασμό. Μέσω της διαχείρισης σκηνών, ο δημιουργός έχει τη δυνατότητα να διαμορφώσει διαφορετικά στάδια του παιχνιδιού, όπως οθόνες μενού, επιλογές παικτών, ερωτήσεις και αποτελέσματα. Παράλληλα, η Unity παρέχει εργαλεία για τη δημιουργία και τοποθέτηση διαδραστικών στοιχείων, όπως κουμπιά, πλαίσια διαλόγου και πεδία εισαγωγής, τα οποία επιτρέπουν στον χρήστη να αλληλεπιδρά με το παιχνίδι με φυσικό και κατανοητό τρόπο.

Επιπλέον, η Unity υποστηρίζει την ενσωμάτωση πολυμέσων – εικόνων, ήχων και μουσικής – ενισχύοντας την εμπειρία του χρήστη και καθιστώντας το περιβάλλον πιο ελκυστικό και λειτουργικό. Η δυνατότητα προσαρμογής της εμφάνισης, της συμπεριφοράς και της ροής του παιχνιδιού μέσα από το οπτικό περιβάλλον εργασίας της Unity επιτρέπει την ακριβή ρύθμιση του gameplay, τόσο για έναν παίκτη όσο και για περιπτώσεις με εναλλαγή μεταξύ δύο παικτών.

Μέσω αυτών των δυνατοτήτων, καθίσταται δυνατή η δημιουργία ενός πλήρως λειτουργικού και ολοκληρωμένου παιχνιδιού γνώσεων, το οποίο περιλαμβάνει μηχανισμούς αξιολόγησης, εναλλαγής σειράς, καταμέτρησης σκορ και ενεργοποίησης ειδικών βοηθητικών λειτουργιών (π.χ. power-ups). Το τελικό αποτέλεσμα είναι μια δυναμική και διαδραστική εφαρμογή που συνδυάζει τη λειτουργικότητα με τη σχεδιαστική ευχρηστία.

Στην ενότητα αυτή, παρουσιάζεται με λεπτομέρεια πώς τα εργαλεία και τα παράθυρα της Unity συνεργάζονται για την οργάνωση των επιμέρους στοιχείων του παιχνιδιού, τη διαχείριση των σκηνών

και την κατασκευή της συνολικής εμπειρίας του χρήστη. Η Unity αναδεικνύεται όχι μόνο ως μια τεχνικά ισχυρή μηχανή ανάπτυξης, αλλά και ως ένα προσιτό, ευέλικτο και αποδοτικό περιβάλλον για την υλοποίηση δημιουργικών ιδεών, ανεξάρτητα από το επίπεδο εμπειρίας του δημιουργού.

1.2.4 Κεφάλαιο 4 - Σχεδιασμός και υλοποίηση εφαρμογής

Στο τέταρτο κεφάλαιο, αναλύεται η διαδικασία σχεδιασμού και ανάπτυξης του παιχνιδιού, με έμφαση στη δομή και τη δημιουργία των σκηνών, καθώς και στη σύνδεσή τους μέσω κώδικα (scripts). Εξετάζεται η ενσωμάτωση script για τη διαχείριση του ήχου, τη φόρτωση και εμφάνιση των ερωτήσεων μέσω διαφορετικών script και την εισαγωγή ερωτήσεων από ένα Google Sheet μέσω URL. Το κεφάλαιο καλύπτει επίσης την υλοποίηση του matchmaking multiplayer με χρήση του Photo Pun 2, διευκρινίζοντας πώς συνδέονται οι παίκτες για να προσφέρουν μια ομαλή και αποδοτική εμπειρία στο multiplayer περιβάλλον του παιχνιδιού.

1.2.5 Κεφάλαιο 5 - Συμπεράσματα και προτάσεις βελτίωσης

Στο πέμπτο και τελευταίο κεφάλαιο, παρουσιάζονται τα συμπεράσματα και οι δυνατότητες εξέλιξης της εφαρμογής. Αρχικά γίνεται ανασκόπηση της υλοποίησης του quiz game στην Unity, με αναφορά στη ροή singleplayer/multiplayer, στη δικτύωση μέσω Photon PUN 2, στη διαχείριση ερωτήσεων από εξωτερική πηγή και στην ενσωμάτωση συστημάτων όπως τα power-ups και το προφίλ παίκτη. Στη συνέχεια παρουσιάζονται προτάσεις για τη μελλοντική εξέλιξη της εφαρμογής, όπως βελτιώσεις στο matchmaking, στη διαχείριση και εισαγωγή ερωτήσεων και στην υποστήριξη offline λειτουργίας. Εξετάζονται επίσης επιχειρησιακές προοπτικές, όπως η αναβάθμιση για περισσότερους χρήστες και η υποστήριξη νέων συσκευών. Τέλος, αιτιολογείται η επιλογή της Unity ως βασικής πλατφόρμας ανάπτυξης, λόγω της ευελιξίας της και της συνεργασίας της με το Photon PUN.

Κεφάλαιο 2 – Μηχανές παιχνιδιών και Βιντεοπαιχνίδια

Πριν ξεκινήσουμε τη σχεδίαση και κατασκευή της δικής μας εφαρμογής αλλά και την ανάλυση των λειτουργιών της μηχανής Unity, είναι απαραίτητο να μελετήσουμε υπάρχοντα συστήματα που έχουν παρόμοιο σκοπό. Η βιβλιογραφική έρευνα μας βοήθησε να κατανοήσουμε ποιες τεχνολογίες χρησιμοποιούνται σήμερα, ποια χαρακτηριστικά θεωρούνται χρήσιμα για τον χρήστη, αλλά και τα πλεονεκτήματα-μειονεκτήματα που καθιστούν την Unity ιδανική για τέτοιες εφαρμογές.

2.1 Ορισμός βιντεοπαιχνιδιών

Ο όρος βιντεοπαιχνίδι (video game) [1] αναφέρεται σε οποιοδήποτε παιχνίδι παίζεται μέσω ηλεκτρονικής συσκευής, όπως ένας υπολογιστής, μια παιχνιδοκονσόλα, ένα smartphone και άλλα παρόμοια μέσα. Όλα τα ηλεκτρονικά παιχνίδια έχουν μια μορφή εισόδου δεδομένων από τον χρήστη: πληκτρολόγιο, ποντίκι, joystick, gamepad, οθόνη αφής, κ.α., και μια μορφή εξόδου που ικανοποιεί τις αισθήσεις του παίκτη: οθόνη υπολογιστή ή τηλεόραση, ηχεία και απτική τεχνολογία [2], μεταξύ άλλων. Ο βασικός τρόπος αλληλεπίδρασης γίνεται μέσω γραφικής διεπαφής, με αποτέλεσμα ο παίκτης να λαμβάνει οπτική ανατροφοδότηση. Αν και ο όρος «βίντεο» στα πρώτα χρόνια του είδους σχετιζόταν με οθόνες τύπου raster , σήμερα χρησιμοποιείται για να περιγράψει οποιαδήποτε απεικόνιση 2D ή 3D σε οποιοδήποτε τύπο οθόνης. Οι ηλεκτρονικές συσκευές που επιτρέπουν την αναπαραγωγή βιντεοπαιχνιδιών ονομάζονται πλατφόρμες και περιλαμβάνουν από προσωπικούς υπολογιστές έως φορητές κονσόλες. Παρότι στο παρελθόν τα παιχνίδια arcade ήταν ιδιαίτερα διαδεδομένα, πλέον η χρήση τους έχει μειωθεί αισθητά. Σήμερα, τα βιντεοπαιχνίδια έχουν εξελιχθεί σε σημαντικό πολιτιστικό μέσο, αποτελώντας ταυτόχρονα μορφή τέχνης και ισχυρή βιομηχανία ψυχαγωγίας.

2.2 Ιστορία και εξέλιξη βιντεοπαιχνιδιών

Η ιστορία των βιντεοπαιχνιδιών ξεκινά στα τέλη της δεκαετίας του 1950 στις ΗΠΑ, όταν οι υπολογιστές άρχισαν να γίνονται πιο γνωστοί. Τα πρώτα παιχνίδια εμφανίστηκαν κυρίως σε εργαστήρια ή ειδικούς χώρους και όχι στα σπίτια. Σιγά σιγά όμως, με την πρόοδο της τεχνολογίας, έφτασαν και στο σαλόνι κάθε σπιτιού.

Τα πρώτα δείγματα αυτού του νέου είδους ψυχαγωγίας εμφανίστηκαν σε διάφορες μορφές: σε σταθερές κονσόλες, σε φλίπερ, σε υπολογιστές, αλλά και σε φορητές συσκευές παιχνιδιών. Η εξέλιξή τους διακρίνεται σε οκτώ βασικές γενιές, κάθε μία από τις οποίες συνδέεται με τις εκάστοτε κονσόλες που κυκλοφορούσαν και τις τεχνολογικές καινοτομίες που έφεραν τα παιχνίδια εκείνης της περιόδου — όπως θα δούμε παρακάτω.

Το πρώτο παιχνίδι για υπολογιστή δημιουργήθηκε το 1952 από τον Alexander Douglas και ονομαζόταν OXO, μια ψηφιακή εκδοχή της τρίλιζας. Το 1958, ο William Higinbotham παρουσίασε το Tennis for Two, ένα πρωτότυπο παιχνίδι πινγκ-πονγκ όπου δύο παίκτες περνούσαν το μπαλάκι πάνω από το φιλέ, αποτελώντας το πρώτο ηλεκτρονικό παιχνίδι με εμπορικό χαρακτήρα. Λίγα χρόνια αργότερα, το 1961, τρεις φοιτητές του MIT δημιούργησαν το Spacewar, ένα διαστημικό παιχνίδι για δύο παίκτες, όπου ο καθένας προσπαθούσε να καταστρέψει το διαστημόπλοιο του άλλου, αποφεύγοντας παράλληλα διάφορα εμπόδια.

Η πρώτη γενιά βιντεοπαιχνιδιών ξεκίνησε με την εμφάνιση της πρώτης εμπορικής κονσόλας, του **Magnavox Odyssey το 1972**, το οποίο μετέτρεψε την τηλεόραση σε μέσο διασκέδασης και παιχνιδιού. Ο χειρισμός γινόταν με δύο περιστρεφόμενα κουμπιά (ροδέλες), τα οποία κινούσαν το αντικείμενο στην οθόνη πάνω–κάτω και αριστερά–δεξιά. Μερικά από τα παιχνίδια που υπήρχαν τότε ήταν το Baseball, το Hockey, το Tennis και άλλα απλά παιχνίδια λογικής.

Εικόνα 2.1: Η κονσόλα Magnavox Odyssey

[https://museumfinder.gr/magnavox-odyssey-1972-i-proti-oikiaki-konsola-vinteopaichnidion-ston-kosmo-ena-psifiako-mnimeio-sto-video-games-museum/]

Η δεύτερη γενιά βιντεοπαιχνιδιών ξεκίνησε το 1976, όταν η εταιρεία Fairchild παρουσίασε το Fairchild Video Entertainment System (VES) [4]. Η κονσόλα αυτή λειτουργούσε με τρόπο παρόμοιο με το Magnavox Odyssey, αξιοποιώντας διακόπτες για την εναλλαγή λειτουργιών. Ωστόσο, ήταν η πρώτη φορά που εμφανίστηκαν οι κασέτες (cartridges), που έδωσαν τη δυνατότητα στους παίκτες να αλλάζουν παιχνίδια στην ίδια κονσόλα χωρίς να χρειάζεται νέο μηχάνημα. Μία από τις πιο γνωστές κονσόλες εκείνης της περιόδου ήταν το Atari 2600, το οποίο έγινε ιδιαίτερα δημοφιλές κυρίως χάρη στο παιχνίδι Space Invaders. Η επιτυχία του τίτλου ήταν τόσο μεγάλη που πολλοί αγόρασαν την κονσόλα αποκλειστικά για να το παίξουν στο σπίτι τους, κάτι που ανέβασε κατακόρυφα τις πωλήσεις.

Παρά την επιτυχία αυτή, η αγορά σύντομα γέμισε με φθηνές κονσόλες και παιχνίδια χαμηλής ποιότητας. Το γεγονός αυτό οδήγησε σε κρίση ολόκληρη τη βιομηχανία των βιντεοπαιχνιδιών, με πολλές εταιρείες να κλείνουν ή να εγκαταλείπουν τον χώρο, ενώ αρκετοί παίκτες στράφηκαν πλέον στους home computers, οι οποίοι προσέφεραν μεγαλύτερη σταθερότητα, καλύτερα γραφικά και περισσότερες δυνατότητες.

Την ίδια περίοδο, η Nintendo παρουσίασε στην Ιαπωνία το Family Computer (Famicom), ανοίγοντας τον δρόμο για την τρίτη γενιά κονσολών. Το σύστημα αυτό έφερε μεγάλη βελτίωση στα γραφικά, καθώς υποστήριζε πιο λεπτομερή sprites και μεγαλύτερη ποικιλία χρωμάτων, προσφέροντας έτσι μια πιο ζωντανή και εντυπωσιακή εμπειρία σε σχέση με τις προηγούμενες κονσόλες.

Λίγα χρόνια αργότερα, η Nintendo κυκλοφόρησε στην αμερικανική αγορά την αντίστοιχη έκδοση, το Nintendo Entertainment System (NES). Η κονσόλα είχε υποδοχή στο μπροστινό μέρος για κασέτες και συνοδευόταν από τον εμβληματικό τίτλο Super Mario Bros.. Η επιτυχία του NES δεν περιορίστηκε μόνο στο ίδιο το μηχάνημα, αλλά έπαιξε καθοριστικό ρόλο στην αναγέννηση της βιομηχανίας των videogames στις Ηνωμένες Πολιτείες, επαναφέροντας την εμπιστοσύνη του κοινού και δημιουργώντας μια νέα εποχή για την οικιακή ψυχαγωγία.

Εικόνα 2.2: Η κονσόλα Nintendo Entertainment System(NES)
[https://collection.sciencemuseumgroup.org.uk/objects/co8094235/nintendo-entertainment-system-nes]

Η τέταρτη γενιά βιντεοπαιχνιδιών ξεκίνησε με το Mega Drive της Sega, που γνώρισε μεγάλη επιτυχία, και σύντομα η Nintendo απάντησε με το Super Nintendo (SNES), προσφέροντας πιο δυνατά γραφικά και ήχο. Στην ίδια περίοδο εμφανίστηκαν και άλλες κονσόλες, όπως το TurboGrafx-16 και το Neo Geo, με το τελευταίο να ξεχωρίζει για την ποιότητα αλλά και το υψηλό κόστος του. Η πραγματική επανάσταση όμως ήρθε με το Game Boy, τη φορητή κονσόλα της Nintendo, που με το μικρό της μέγεθος και τη λειτουργία με μπαταρίες έκανε το gaming φορητό (οθόνη και χειριστήρια ήταν ενσωματωμένα σε μία ενιαία συσκευή) και άλλαξε για πάντα τον τρόπο που παίζαμε.

Εικόνα 2.3: Η κονσόλα GameBoy της Nintendo

[https://collection.sciencemuseumgroup.org.uk/objects/co8184133/nintendo-game-boy]

Η πέμπτη γενιά κονσολών ξεκίνησε με το Atari Jaguar, που αν και πιο δυνατό από τις προηγούμενες κονσόλες, δεν κατάφερε να κερδίσει το κοινό. Η πραγματική άνοδος αυτής της γενιάς ήρθε με τα Sega Saturn, Sony PlayStation και Nintendo 64. Για πρώτη φορά χρησιμοποιήθηκαν CD αντί για κασέτες, κάτι που έδωσε μεγαλύτερη χωρητικότητα και καλύτερο ήχο. Αυτή η εποχή μας χάρισε μερικά από τα πιο γνωστά παιχνίδια όλων των εποχών, όπως τα Super Mario 64, Mario Kart, The Legend of Zelda, Donkey Kong και Tekken.

Στα τέλη της δεκαετίας του ’90, η Sega παρουσίασε το Dreamcast, μια κονσόλα γεμάτη καινοτομίες, αλλά σύντομα βγήκε εκτός ανταγωνισμού, καθώς το 2000 η Sony κυκλοφόρησε το PlayStation 2, που έγινε η πιο επιτυχημένη κονσόλα όλων των εποχών. Στην ίδια περίοδο εμφανίστηκαν το Nintendo GameCube και το πρώτο Xbox από τη Microsoft, που μπήκε δυναμικά στην αγορά.

Με την έκτη γενιά [6] οι κονσόλες άρχισαν να μοιάζουν περισσότερο με υπολογιστές, με καλύτερα λειτουργικά, αποθηκευτικά μέσα και δυνατότητα σύνδεσης στο ίντερνετ.

Η έβδομη γενιά ξεκίνησε το 2005 με το Xbox 360, που έφερε ασύρματα χειριστήρια και ενισχυμένο online gaming. Το 2006, η Sony απάντησε με το PlayStation 3, που εκτός από παιχνίδια έπαιζε και ταινίες Blu-ray. Την ίδια χρονιά, η Nintendo λάνσαρε το επαναστατικό Wii, το οποίο χάρη στους αισθητήρες κίνησης μετέτρεψε το παιχνίδι σε πιο διαδραστική εμπειρία. Το Wii έγινε τεράστια επιτυχία γιατί κατάφερε να φέρει στο gaming οικογένειες και παίκτες κάθε ηλικίας.

Εικόνα 2.4: Η κονσόλα Nintendo Wii με αισθητήρες κίνησης

[https://www.nintendo.com/en-gb/Support/Wii/Support-for-Wii-301698.html?srsltid=AfmBOooDYpt7b-1CJnSoV_A1xjs5qPGQz8id9fQF1fc-iHNylcOpeIum]

Το 2011, η Sony κυκλοφορεί το PlayStation Vita, καθώς και το PlayStation 4, προσφέροντας σημαντικές τεχνολογικές βελτιώσεις. Το 2013, η Microsoft παρουσιάζει το Xbox One, ενώ η Nintendo κυκλοφορεί το Switch, μια καινοτόμα υβριδική κονσόλα για φορητή και οικιακή χρήση. Σήμερα, διανύουμε την ένατη γενιά κονσολών, με το PlayStation 5 να κυκλοφορεί στα τέλη του 2020. Το PS5, όπως και οι σύγχρονοι υπολογιστές, αξιοποιεί την εικονική πραγματικότητα (VR), επιτρέποντας στους παίκτες να ζήσουν το παιχνίδι με πιο ζωντανό και ρεαλιστικό τρόπο, μέσα από ειδικά γυαλιά που τους μεταφέρουν στον εικονικό κόσμο του παιχνιδιού.

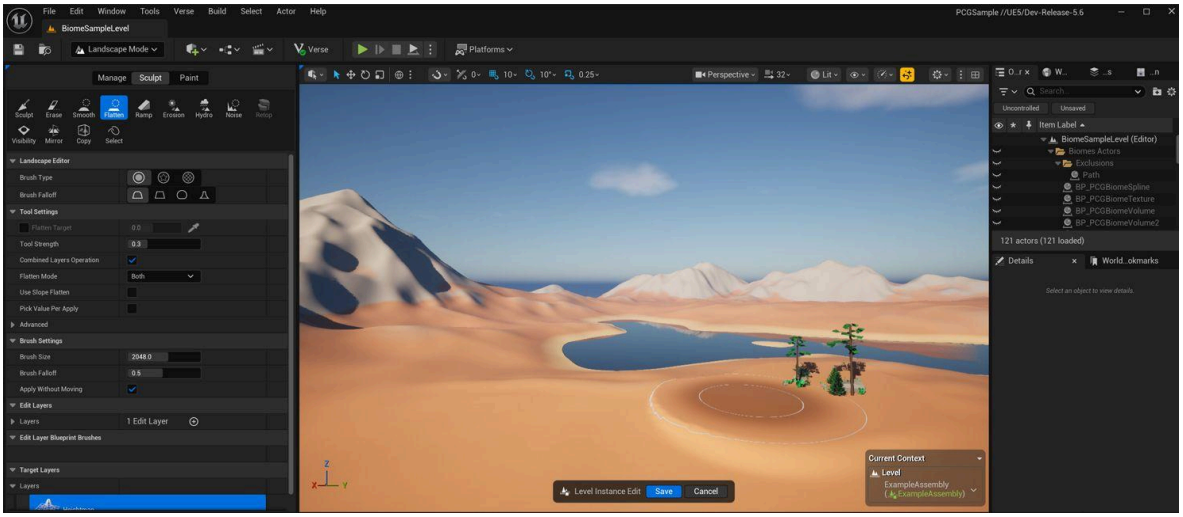
Εικόνα 2.5: Oculus Rift S VR headset (Οπτικοακουστικό σετ εικονικής πραγματικότητας)
[https://www.gamesradar.com/oculus-rift-s-review/]

2.3 Μηχανές Παιχνιδιών

2.3.1 Unreal Engine

Η **Unreal Engine** [7] αποτελεί μία μηχανή δημιουργίας παιχνιδιών (game engine), που αναπτύχθηκε από τη γνωστή εταιρεία Epic Games. Η πρώτη της εμφάνιση έγινε το 1998 με το δημοφιλές παιχνίδι Unreal, και από τότε εξελίσσεται συνεχώς, παραμένοντας μία από τις πιο διαδεδομένες επιλογές στον χώρο της ανάπτυξης παιχνιδιών. Όπως και οι περισσότερες σύγχρονες μηχανές παιχνιδιών (game engines), η Unreal έχει αντικειμενοστραφή αρχιτεκτονική και βασίζεται κυρίως στη γλώσσα προγραμματισμού C++. Στο παρελθόν χρησιμοποιούσε τη δική της γλώσσα, την UnrealScript, που ήταν βασισμένη στη Java. Η συγκεκριμένη μηχανή υποστηρίζει ένα ευρύ φάσμα πλατφορμών για υπολογιστές, κινητά, κονσόλες και εικονική πραγματικότητα. Η ανάπτυξη ενός παιχνιδιού στην Unreal γίνεται μέσω της δημιουργίας και επεξεργασίας σκηνών. Μέσα σε κάθε σκηνή, ο προγραμματιστής τοποθετεί αντικείμενα (objects) είτε μέσω κώδικα είτε χρησιμοποιώντας

έτοιμα πρότυπα (templates). Τα αντικείμενα αυτά χαρακτηρίζονται από ιδιότητες όπως η θέση, ο προσανατολισμός, η εμφάνιση και διάφορα φυσικά χαρακτηριστικά (π.χ. βάρος, ταχύτητα κίνησης). Αυτό που κάνει την Unreal —και γενικά τις game engines— να ξεχωρίζει από άλλα προγράμματα μοντελοποίησης είναι το πώς διαχειρίζεται και συνδυάζει προγραμματιστικά, οπτικά και φυσικά στοιχεία, επιτρέποντας την ολοκληρωμένη ανάπτυξη ενός διαδραστικού και λειτουργικού παιχνιδιού. Πιο συγκεκριμένα, το Unreal διαθέτει:



Εικόνα 2.6: Σκηνή στο πρόγραμμα Unreal Engine

[https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-6-release-notes]

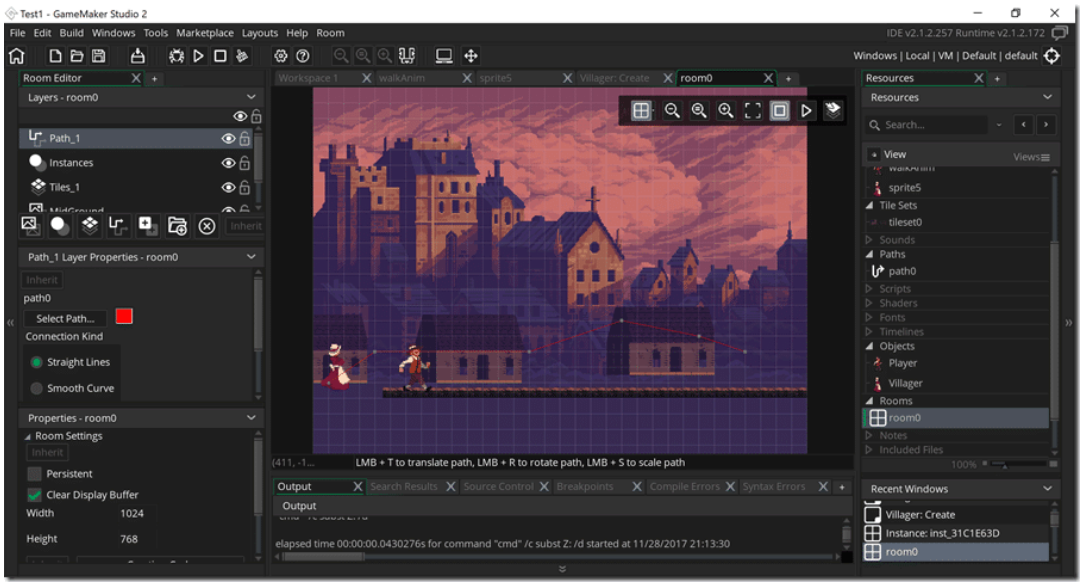
1. μία μηχανή φυσικής, η οποία είναι υπεύθυνη για τη ρεαλιστική απεικόνιση και εφαρμογή των κανόνων που διέπουν την αλληλεπίδραση των αντικειμένων μεταξύ τους και με το περιβάλλον — όπως η βαρύτητα, οι συγκρούσεις, και άλλες φυσικές δυνάμεις.
2. Μια μηχανή ήχου, η οποία αναλαμβάνει τόσο την αναπαραγωγή ήχων εντός της σκηνής — συχνά από πολλές ταυτόχρονες πηγές — όσο και την ρεαλιστική απόδοσή τους με βάση τη θέση της πηγής και του ακροατή στον χώρο, προσφέροντας έτσι τρισδιάστατο ηχητικό εφέ.
3. Μια μηχανή γραφικών, η οποία έχει ως ρόλο την οπτική απεικόνιση της σκηνής, λαμβάνοντας υπόψη τη γεωμετρία και τα χρώματα των αντικειμένων, τις πηγές φωτός, τις σκιές, καθώς και το φαινόμενο της οπτικής επικάλυψης (occlusion), όπου αντικείμενα του προσκηνίου κρύβουν μερικώς ή πλήρως εκείνα του παρασκηνίου.

Όλοι οι παραπάνω μηχανισμοί συνεργάζονται για να παραχθεί η τελική εικόνα που προβάλλεται στην οθόνη του χρήστη. Σε περίπτωση που διαδικασίες συμβαίνουν δυναμικά κατά την αλληλεπίδραση του χρήστη με τον υπολογιστή, η λειτουργία των μηχανισμών αυτών δρομολογείται με τη βοήθεια κώδικα.

Η μηχανή Unreal Engine είναι συμβατή με τα περισσότερα γνωστά λογισμικά και έχει χρησιμοποιηθεί ευρέως, όχι μόνο στη βιομηχανία των βιντεοπαιχνιδιών (π.χ. *Batman*, *Aliens*, *Fortnite*), αλλά και σε τηλεοπτικές παραγωγές όπως *The Mandalorian* και *Westworld [8]*. Χάρη στις δυνατότητές της, η Unreal Engine δεν χρησιμοποιείται μόνο στα βιντεοπαιχνίδια αλλά και σε τομείς όπως η εκπαίδευση, η αρχιτεκτονική, ο πολιτισμός και η βιομηχανία, για τη δημιουργία ρεαλιστικών προσομοιώσεων και παρουσιάσεων. Είναι δωρεάν στη χρήση, εκτός αν το τελικό προϊόν αποφέρει πάνω από 1 εκατομμύριο δολάρια, οπότε η Epic Games λαμβάνει 5% ποσοστό.

2.3.2 GameMaker

Το **GameMaker** [9] είναι μια ευέλικτη μηχανή ανάπτυξης παιχνιδιών που δημιουργήθηκε το 1997 από τον Mark Overmars και κυκλοφόρησε για πρώτη φορά στο εμπόριο το 2007 μέσω της εταιρείας YoYo Games. Υποστηρίζει πληθώρα πλατφορμών, όπως Windows, macOS, Ubuntu, HTML5, Raspberry Pi, καθώς και κονσόλες όπως PlayStation (3, 4, 5) και Xbox (One, Series X/S).



Εικόνα 2.7: Σκηνή στο πρόγραμμα GameMaker

[https://www.macrumors.com/2017/09/02/gamemaker-studio-2-debuts-on-macos/]

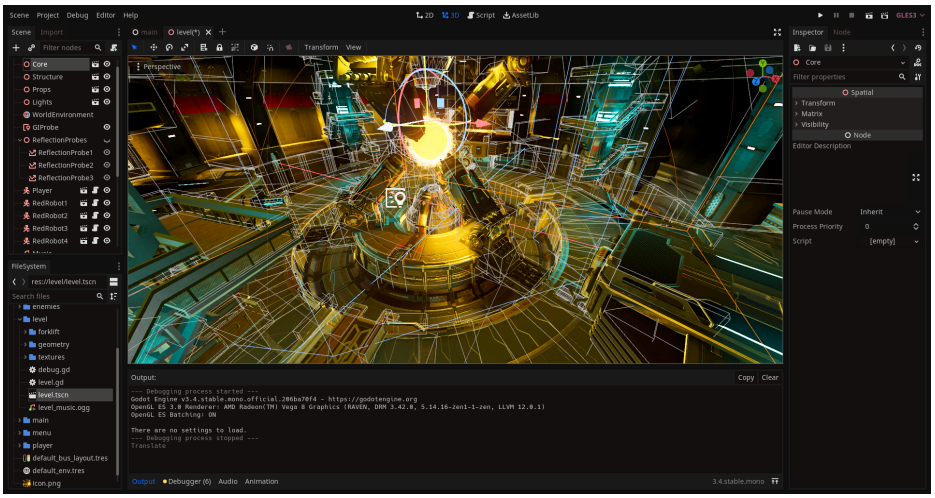
Ως μηχανή παιχνιδιών, το GameMaker διαθέτει όλες τις βασικές λειτουργίες που αναφέρθηκαν στην παραπάνω παράγραφο, δίνοντας τη δυνατότητα αξιοποίησής του σε ποικίλες εφαρμογές και κλάδους. Σε επίπεδο προγραμματισμού, προσφέρει δύο διαθέσιμες προσεγγίσεις, διευκολύνοντας έτσι τη διαδικασία ανάπτυξης τόσο για αρχάριους όσο και για πιο έμπειρους χρήστες. Πιο συγκεκριμένα, επιτρέπει τον προγραμματισμό:

1. **Με χρήση του Drag and Drop (DnD)**: ενός γραφικού εργαλείου που επιτρέπει τη δημιουργία αντικειμένων, την ενεργοποίηση λειτουργιών και τη διαχείριση ροής δεδομένων μέσω διασύνδεσης κόμβων, χωρίς την ανάγκη γραφής κώδικα. Είναι ιδανικό για αρχάριους ή για γρήγορη ανάπτυξη λογικής.
2. **Μέσω κώδικα** και της δικής του γλώσσας προγραμματισμού, της GameMaker Language, η οποία συντίθεται από ένα συνδυασμό στοιχείων της C και της JavaScript.

Η πλατφόρμα θεωρείται ιδιαίτερα φιλική προς τον χρήστη και προσφέρει ευελιξία τόσο σε αρχάριους όσο και σε πιο έμπειρους προγραμματιστές και σχεδιαστές. Ωστόσο, συχνά παρουσιάζει ζητήματα ασταθούς λειτουργίας (crushes), ενώ πολλά από τα προκαθορισμένα έτοιμα αντικείμενα και κάποιες από τις λειτουργίες έχουν χαρακτηριστεί ως ξεπερασμένα. Αξίζει επίσης να σημειωθεί ότι το GameMaker διατίθεται μόνο μέσω επί πληρωμή άδειας χρήσης.

2.3.3 Godot

Το **Godot** [10] αποτελεί μια πλήρης μηχανή παιχνιδιών, που χρησιμοποιείται για τη δημιουργία 2D και 3D παιχνιδιών, διαθέσιμη υπό την άδεια MIT Massachusetts Institute of Technology). Λόγω της ευέλικτης δομής του, το Godot δίνει τη δυνατότητα στους δημιουργούς του να εναλλάσσονται εύκολα σε 2D και 3D ώστε να φτιάχνουν διαφορετικά είδη παιχνιδιών. Υποστηρίζει πολλαπλά λειτουργικά συστήματα, όπως Windows, macOS, Linux, iOS, Android και άλλα, προσφέροντας ευρεία συμβατότητα και ευελιξία στους δημιουργούς.



Εικόνα 2.8: Σκηνή στο πρόγραμμα Godot

[https://en.wikipedia.org/wiki/Godot_%28game_engine%29]

Το Godot λειτουργεί ως πλήρης μηχανή παιχνιδιών με ένα ενσωματωμένο περιβάλλον ανάπτυξης. Το βασικό του εργαλείο είναι το παράθυρο της σκηνής, μέσω του οποίου ο χρήστης μπορεί να αξιοποιήσει όλα τα διαθέσιμα εργαλεία για τη δημιουργία του επιθυμητού περιβάλλοντος. Η βασική γλώσσα προγραμματισμού του είναι η C++. Εκτός την C++, το Godot υποστηρίζει και τον προγραμματισμό σε C#, ενώ διαθέτει και τη δική του γλώσσα προγραμματισμού, την GDScript, η οποία παρουσιάζει πολλά κοινά στοιχεία με την Python. Η GDScript δίνει προτεραιότητα στην ευκολία χρήσης και την αναγνωσιμότητα, αποτελώντας εξαιρετική επιλογή τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές. Η σύνδεσή της με τη μηχανή βοηθά να φτιάχνονται πιο εύκολα τα παιχνίδια, από τα απλά μέχρι και τα πιο σύνθετα κομμάτια τους.

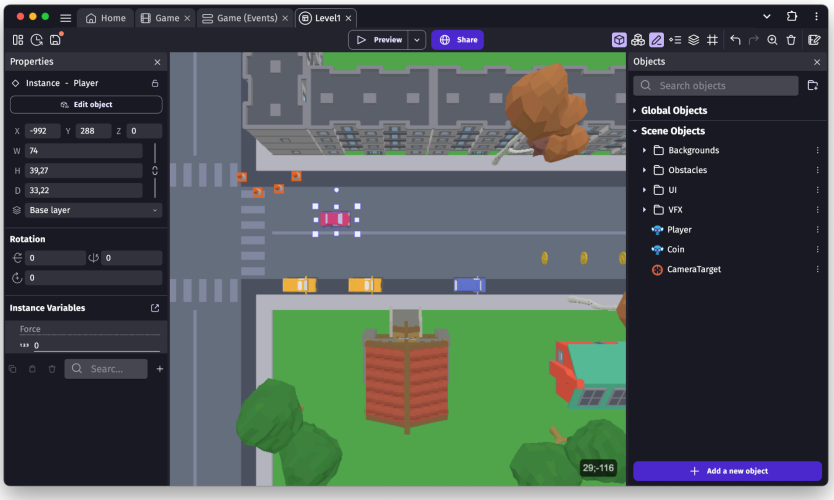
Εκτός από την ανάπτυξη παιχνιδιών, το Godot έχει αρχίσει να χρησιμοποιείται και στον τομέα της εκπαίδευσης, ιδιαίτερα σε ορισμένα λύκεια στις Ηνωμένες Πολιτείες, χάρη στη φιλική προς τον χρήστη προσέγγισή του και την άμεση εξοικείωση που προσφέρει με τον προγραμματισμό, ακόμα και σε αρχάριους. Ωστόσο, σύμφωνα με γνώμες έμπειρων χρηστών, το Godot στο τεχνικό και πρακτικό του κομμάτι παρουσιάζει κάποιες ασάφειες ως προς τις διάφορες δομές του, ενώ το πλήθος των δημοφιλών εφαρμογών τα οποία βασίζονται σε αυτό είναι αρκετά μικρό.

2.3.4 GDevelop

Το **GDevelop** [11] είναι μια ελεύθερη και ανοιχτού κώδικα μηχανή ανάπτυξης παιχνιδιών 2D και 3D, η οποία επικεντρώνεται κυρίως στη δημιουργία παιχνιδιών για PC και κινητές συσκευές, καθώς και για παιχνίδια HTML5 που παίζονται στον browser. Ένα από τα βασικά πλεονεκτήματα του GDevelop είναι ο προγραμματισμός βασισμένος σε events (event-based programming). Σε αντίθεση με τις παραδοσιακές μηχανές παιχνιδιών που απαιτούν γνώσεις προγραμματισμού, το GDevelop επιτρέπει να σχεδιάζεις τη λογική του παιχνιδιού σου μέσα από ένα εύχρηστο οπτικό σύστημα. Τα events χρησιμοποιούνται για να ορίσουν τις συμπεριφορές και τις αλληλεπιδράσεις στο παιχνίδι, όπως οι κινήσεις χαρακτήρων, οι συγκρούσεις αντικειμένων ή η τεχνητή νοημοσύνη των εχθρών. Το GDevelop διαθέτει ένα ενσωματωμένο Asset Store που φιλοξενεί εκατοντάδες χιλιάδες δωρεάν και επί πληρωμή assets. Αυτά μπορούν να προστεθούν εύκολα σε ένα project του GDevelop με μερικά κλικ μέσω της διεπαφής του επεξεργαστή. Όλο το περιεχόμενο του παιχνιδιού, όπως τα σχέδια χαρακτήρων, τα φόντα, τα κείμενα κ.λπ., μπορεί να προστεθεί απευθείας μέσω μιας διεπαφής point-and-click στον επεξεργαστή.

Ο κύριος στόχος του GDevelop είναι η ανάπτυξη παιχνιδιών μέσω του συστήματος γεγονότων, χωρίς να απαιτείται γνώση προγραμματισμού. Παρ' όλα αυτά, οι χρήστες μπορούν να χρησιμοποιήσουν μπλοκ JavaScript για να αντικαταστήσουν οποιοδήποτε γεγονός. Αυτό δίνει τη δυνατότητα στους προχωρημένους χρήστες να επεκτείνουν τις δυνατότητες του συστήματος

γεγονότων, παρέχοντας τους τη δυνατότητα να αλληλεπιδράσουν άμεσα με τη μηχανή και να αυξήσουν τις δυνατότητές της. Το GDevelop διατίθεται και ως κινητή εφαρμογή, βελτιστοποιημένη για iOS και Android. Υπάρχουν περιορισμοί για δωρεάν λογαριασμούς όσον αφορά τις σκηνές και τα γεγονότα, αλλά η αναβάθμιση σε Premium συνδρομή ξεκλειδώνει όλες τις δυνατότητες της εφαρμογής. Αυτό επιτρέπει στους χρήστες να αναπτύξουν παιχνίδια σε συσκευές Android και iOS, με υποστήριξη cross-save, δίνοντας τη δυνατότητα στους χρήστες να ξεκινήσουν την ανάπτυξη σε κινητές συσκευές και να συνεχίσουν στον υπολογιστή τους, ή αντίστροφα.



Εικόνα 2.9: Σκηνή στο πρόγραμμα GDevelop

[https://github.com/4ian/GDevelop]

Με το GDevelop ο κάθε χρήστης μπορεί εύκολα να εξάγει τα παιχνίδια του σε μια μεγάλη γκάμα πλατφορμών. Είτε θέλεις να τα μοιραστείς στο διαδίκτυο, σε κινητές συσκευές ή ως εφαρμογές για υπολογιστές, το GDevelop κάνει τη διαδικασία απλή και γρήγορη. Τα παιχνίδια μπορούν να εξαχθούν απευθείας για τις πλατφόρμες Android, Windows, Linux και Web. Οι παρακάτω πλατφόρμες υποστηρίζονται για εξαγωγή με ένα κλικ: macOS, Linux, Android, HTML5 (Web) και Windows 7/8/10/11. Επιπλέον, τα projects μπορούν να εξαχθούν τοπικά και να συντεθούν χειροκίνητα για τις εξής πλατφόρμες: iOS, Linux, Android, HTML5 (Web), Windows 7/8/10/11 και Windows Store UWP.

2.3.5 Unity

Η Unity [12], είναι μια ισχυρή μηχανή ανάπτυξης παιχνιδιών που αναπτύχθηκε από την εταιρεία Unity Technologies, με την πρώτη της έκδοση να κυκλοφορεί το 2005. Ανάμεσα στις διάφορες μηχανές που έχουν παρουσιαστεί, η Unity ξεχωρίζει για τη μεγάλη της συμβατότητα με πολλές

διαφορετικές πλατφόρμες και λειτουργικά συστήματα, καθιστώντας την ιδιαίτερα δημοφιλή τόσο στους επαγγελματίες όσο και στους ερασιτέχνες δημιουργούς. Μερικές από αυτές είναι:

1. Πλατφόρμες κινητών τηλεφώνων (iOS, Android, tvOS)
2. Πλατφόρμες ηλεκτρονικού υπολογιστή (Windows, Mac, Linux)
3. Πλατφόρμες διαδικτύου (WebGL)
4. Πλατφόρμες κονσολών (Playstation 4, Playstation 5, Xbox One, Xbox Series X/S, Nintendo Switch, Stadia)
5. Πλατφόρμες επαυξημένης και εικονικής πραγματικότητας (Oculus, Playstation VR, Steam VR, HoloLens, Google's ARCore, Apple ‘s ARKit)

ενώ είναι συμβατή και με παλιότερα, όπως οι Wii, Wii U, Playstation 3, Xbox 360, Windows Phone 8, Samsung Smart TV, Facebook Gameroom και άλλες. Μέχρι το 2015, η διάθεση της γινόταν μέσω αγοράς άδειας χρήσης, όμως από το 2016 και μετά, η Unity Technologies υιοθέτησε μοντέλο συνδρομής. Παράλληλα, διατίθεται και δωρεάν έκδοση για προσωπική χρήση ή μικρές επιχειρήσεις.

Η πλατφόρμα Unity περιγράφεται αναλυτικά στο επόμενο κεφάλαιο, όπου εξηγείται όλο το περιβάλλον διεπαφής της (user interface), καθώς αυτή χρησιμοποιήθηκε για την υλοποίηση του project. Επιπλέον, θα συζητήσουμε για τα πλεονέκτημα αλλά και τα μειονεκτήματα της, που την καθιστούν ως την νούμερο ένα επιλογή για τη δημιουργία παιχνιδιών και πλατφορμών.

2.4 Σύγκριση μηχανών παιχνιδιών

2.4.1 Πλεονεκτήματα και μειονεκτήματα Unity

Η Unity αποτελεί την κορυφαία επιλογή για τη δημιουργία online multiplayer quiz games, λόγω της μοναδικής συνδυαστικής ισχύος που προσφέρει στην ανάπτυξη παιχνιδιών με ευκολία χρήσης, ευελιξία και πολυάριθμες δυνατότητες για multiplayer λειτουργίες. Σε σύγκριση με άλλες μηχανές παιχνιδιών όπως η Unreal Engine, η Godot, το GDevelop και το GameMaker, η Unity ξεχωρίζει για τους εξής λόγους [13]:

1. **Εύκολη και αποτελεσματική ανάπτυξη multiplayer παιχνιδιών:** Η Unity παρέχει ισχυρή υποστήριξη για τη δημιουργία online multiplayer παιχνιδιών με την ενσωμάτωση εργαλείων όπως το Photon και το UNet για τη διαχείριση server-client αρχιτεκτονικής και matchmaking. Επιπλέον, η networking υποδομή της Unity είναι πολύ καλά τεκμηριωμένη, κάτι που την καθιστά ιδανική για παιχνίδια όπου η συγχρονισμένη αλληλεπίδραση παικτών είναι κρίσιμη.

2. **Υποστήριξη πολλαπλών πλατφορμών:** Ένα από τα μεγάλα πλεονεκτήματα της Unity είναι η cross-platform δυνατότητά της, επιτρέποντας τη δημιουργία παιχνιδιών που μπορούν να αναπτυχθούν για Android, iOS, Windows, Linux, Web και πολλές άλλες πλατφόρμες χωρίς να απαιτούνται διαφορετικές εκδόσεις του παιχνιδιού. Αυτό επιτρέπει στους προγραμματιστές να επεκτείνουν το παιχνίδι τους σε πολλές συσκευές και να προσεγγίσουν μεγαλύτερο κοινό.
3. **Απλή ανάπτυξη με C# και Visual Scripting:** Η Unity υποστηρίζει τη γλώσσα C#, προσφέροντας μεγάλη ευχέρεια και ευκολία για πιο προχωρημένα έργα, ενώ παράλληλα παρέχει εργαλεία visual scripting (όπως το Bolt), τα οποία είναι ιδανικά για χρήστες με λιγότερη ή καθόλου εμπειρία προγραμματισμού. Αυτό την καθιστά προσβάσιμη σε έναν μεγάλο αριθμό δημιουργών, ανεξαρτήτως του επιπέδου των τεχνικών τους γνώσεων.
4. **Μεγάλη κοινότητα και εκτεταμένοι πόροι:** Η Unity διαθέτει μια τεράστια κοινότητα χρηστών, που σημαίνει ότι μπορείς να βρεις εύκολα υποστήριξη μέσω forums, tutorials και εκπαιδευτικών πόρων. Επίσης, το Asset Store της Unity παρέχει εκατοντάδες έτοιμα assets που μπορούν να χρησιμοποιηθούν για την ταχύτερη ανάπτυξη του παιχνιδιού σου, κάτι που κάνει τη διαδικασία ανάπτυξης πολύ πιο γρήγορη και αποτελεσματική.

5. <u>Γιατί υπερτερεί σε σχέση με τις άλλες πλατφόρμες:</u>

- **Unreal Engine:** Αν και η Unreal Engine προσφέρει εξαιρετικά γραφικά και είναι ιδανική για μεγάλης κλίμακας 3D παιχνίδια, η καμπύλη εκμάθησης είναι πολύ πιο απότομη και απαιτεί υψηλότερους υπολογιστικούς πόρους, κάτι που μπορεί να περιορίσει την ευχρηστία σε μικρότερα projects.

- **Godot:** Η Godot είναι μια πολύ καλή επιλογή για 2D παιχνίδια, αλλά η υποστήριξή της για 3D multiplayer παιχνίδια είναι περιορισμένη. Επιπλέον, δεν προσφέρει τις ίδιες εξελιγμένες δυνατότητες matchmaking και online συνδέσεων που προσφέρει η Unity.

- **GDevelop και GameMaker:** Αν και είναι εύκολες στη χρήση και καλές για γρήγορη ανάπτυξη 2D παιχνιδιών, δεν προσφέρουν τις ίδιες δυνατότητες για advanced multiplayer και networking όπως η Unity, ενώ δεν είναι τόσο ισχυρές για την ανάπτυξη παιχνιδιών σε πολλές πλατφόρμες και σε πιο σύνθετα έργα.

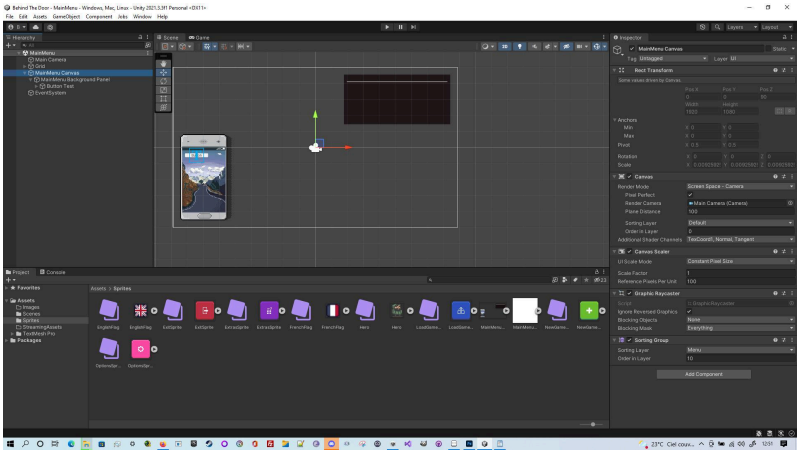
Συνολικά, η Unity είναι η πιο ευέλικτη, ισχυρή και ευχάριστη για τους χρήστες μηχανή παιχνιδιών, προσφέροντας τα εργαλεία και τις δυνατότητες για να δημιουργήσεις online multiplayer quiz games με εξαιρετική ευκολία και αποτελεσματικότητα.

Κεφάλαιο 3 – Η μηχανή εφαρμογών Unity

Στο παρόν κεφάλαιο, παρουσιάζεται αναλυτικά η μηχανή ανάπτυξης εφαρμογών πολλαπλών πλατφορμών Unity, πραγματοποιώντας ταυτόχρονα μια ιστορική αναδρομή αναφορικά με τη χρήση της σε διάφορους τομείς της τεχνολογικής εξέλιξης. Στη συνέχεια, περιγράφονται τα κύρια στοιχεία και τα εργαλεία που διαθέτει, τα οποία την καθιστούν κατάλληλη για χρήση για μια ποικιλία εφαρμογών, όπως και η δική μου.

3.1 Unity - Μια ιστορική αναδρομή

Η Unity είναι μία πλατφόρμα ανάπτυξης εφαρμογών και παιχνιδιών (game engine), η οποία δημιουργήθηκε από την εταιρεία Unity Technologies και έγινε διαθέσιμη στους προγραμματιστές για πρώτη φορά το 2005 [14]. Αυτό που την ξεχωρίζει και την καθιστά μοναδική ανάμεσα σε άλλες παρόμοιες μηχανές από την πρώτη κιόλας εμφάνιση της είναι, η ευκολία που προσφέρει κατά τη δημιουργία, ανάπτυξη και εξαγωγή εφαρμογών προσαρμοσμένων πάντα στις σύγχρονες τεχνολογικές τάσεις. Σήμερα η Unity υποστηρίζει πάνω από 25 πλατφόρμες λογισμικού όπως τα Windows, Linux, Android, iOS, Playstation, Xbox, Oculus Rift και άλλων.



Εικόνα 3.1: Περιβάλλον διεπαφής Unity

[https://1000logos.net/unity-logo/]

Με την πάροδο των χρόνων, ακολούθησε μια συνεχής αναβάθμιση στη πλατφόρμα με νέες προσθήκες και λειτουργικότητες, καθιστώντας την ως μία από τις πιο γνωστές μηχανές για ανάπτυξη παιχνιδιών και εφαρμογών γενικότερα. Δύο χρόνια μετά την παρουσίαση της στο κοινό το 2007, ενσωματώνεται μια σειρά από δυνατότητες, που αφορούν τη δημιουργία 3D γραφικών, τον φωτισμό και την αναπαραγωγή βίντεο, διευκολύνοντας ταυτόχρονα τη συνεργασία πολλών χρηστών[31]. Το 2008, με την εκκίνηση της λειτουργίας του App Store ενσωματώνονται νέες επεκτάσεις που

καθιστούν δυνατή τη δημιουργία για iPhone [15]. Δύο χρόνια μετά το 2010, προστίθενται νέες λειτουργικότητες για υπολογιστές και κονσόλες, όπως και δυνατότητες για Android [16]. Σύμφωνα με έρευνα το 2012, η Unity αναγνωρίστηκε ως η καλύτερη πλατφόρμα σχεδιασμού και υλοποίησης παιχνιδιών. Ακολουθεί η τέταρτη έκδοση που υποστηρίζει το DirectX 11 και το Adobe Flash [17]. Την ίδια χρονιά, η Unity ανακοινώνει τη συνεργασία της με την NASA (National Aeronautics and Space Administration) με στόχο την ανάπτυξη προσομοιώσεων. Το 2013, το Facebook κυκλοφορεί ένα software development kit **(SDK)** για ανάπτυξη παιχνιδιών με τη Unity [18], ενώ, το 2016 η συνεργασία των δύο εταιρειών είχε ως αποτέλεσμα την ανάπτυξη μιας νέας πλατφόρμας δημιουργίας παιχνιδιών, με σκοπό την απλοποίηση της διαδικασίας για προγραμματιστές [19]. Η νέα έκδοση της Unity η οποία θα γίνει διαθέσιμη το 2017, θα δώσει τη δυνατότητα για ταχύτατο rendering, εξαγωγή στατιστικών απόδοσης και κατανάλωσης πόρων, ενώ πλέον θα καταστεί εφικτή η εισαγωγή αρχείων προσδιορισμού κίνησης **(animation)** τρισδιάστατων μοντέλων με τη λογική **drag and drop** [20]. Επιπλέον, την ίδια χρονιά, εντάσσεται η δυνατότητα επεξεργασίας αρχείων 3ds Max της Autodesk, μιας εταιρείας λογισμικού με ευρεία παρουσία στον τεχνολογικό τομέα. Την επόμενη χρονιά, η πλατφόρμα ενισχύεται με εργαλεία rendering υψηλής ανάλυσης και ακρίβειας, τα οποία καλύπτουν τόσο κονσόλες και υπολογιστές όσο και κινητές συσκευές, καθώς και εφαρμογές επαυξημένης και εικονικής πραγματικότητας [21]. ενώ, για πρώτη φορά, λίγο διάστημα μετά, η Unity παρουσιάζει για πρώτη φορά δυνατότητες χρήσης μηχανικής μάθησης (machine learning). Το 2019, η Unity συνεργάζεται με τη Wolfram Research για την εισαγωγή νέων προηγμένων μαθηματικών εργαλείων στη μηχανή. Το 2020, παρουσιάζει το MARS, μια πλατφόρμα για εύκολο σχεδιασμό εφαρμογών επαυξημένης πραγματικότητας με βάση προκαθορισμένους κανόνες [22], ενώ αρχίζει και η υποστήριξη για τους νέους επεξεργαστές Apple Silicon, με την πλήρη ενσωμάτωση να ολοκληρώνεται το 2022. Το 2022, η εταιρεία εστιάζει στη βελτιστοποίηση του χρόνου εκκίνησης των εφαρμογών, αντιμετωπίζοντας ένα βασικό ζήτημα για τους προγραμματιστές. Το 2023, μετά την αποχώρηση του CEO, ανακοινώνεται νέα έκδοση της μηχανής που εισάγει τα εργαλεία τεχνητής νοημοσύνης Unity Muse και Sentis, επιτρέποντας την αυτοματοποιημένη δημιουργία περιεχομένου.

18 χρόνια μετά την ίδρυση της, η Unity θεωρείται ως μία από τις κορυφαίες πλατφόρμες για την ανάπτυξη 2D και 3D παιχνιδιών, ενώ έχει επεκταθεί σημαντικά και στο κομμάτι των προσομοιώσεων, της επιστημονικής έρευνας και των πειραματικών έργων. Από πλατφόρμες εικονικής και επαυξημένης πραγματικότητας (virtual και augmented reality), προσομοιώσεις για ποιοτικό έλεγχο στον κατασκευαστικό τομέα (όπως crash tests οχημάτων [23]), έως και αυτοματοποιημένες παραγωγικές διαδικασίες με απομακρυσμένο έλεγχο και ενσωματωμένα συστήματα διαχείρισης, η Unity συνεχίζει να ηγείται στο χώρο της τεχνολογίας [24]. Επιπλέον, η Unity αξιοποιείται από τη DeepMind Technologies για την εκπαίδευση νευρωνικών δικτύων, ενώ έχει χρησιμοποιηθεί και στη δημιουργία ταινιών μικρού μήκους, όπως οι Adam: The Mirror και Adam: The Prophet [25]. Η Unity διατίθεται δωρεάν για μη κερδοσκοπική χρήση και εφαρμογές με έσοδα έως 200.000 δολάρια, ενώ

προσφέρει επαγγελματικά πακέτα με επιπλέον δυνατότητες για εμπορικές εφαρμογές μεγάλης κλίμακας.

3.2 Δομικά στοιχεία Unity

3.2.1 Asset Store

Όπως έχει ήδη αναφερθεί, η Unity αποτελεί την πιο ισχυρή πλατφόρμα για σχεδιασμό εφαρμογών και παιχνιδιών, καθώς προσφέρει τα περισσότερα από τα σημαντικότερα συστατικά που απαιτούνται για την ανάπτυξη ενός ολοκληρωμένου παιχνιδιού. Με αυτόν τον τρόπο, οι προγραμματιστές αποφεύγουν να πραγματοποιούν λειτουργίες χαμηλού επιπέδου (low level features), αλλά μπορούν να εστιάσουν και σε πιο σύνθετες λειτουργίες (high level features) που αφορούν την εξέλιξη της εφαρμογής ή του παιχνιδιού. Για τον λόγο αυτό, η Unity διαθέτει το Asset Store, ένα ενσωματωμένο κατάστημα εντός της πλατφόρμας όπου οι προγραμματιστές μπορούν να βρουν και να χρησιμοποιήσουν έτοιμα μοντέλα και λειτουργίες, είτε δωρεάν είτε επί πληρωμή, χωρίς να χρειάζεται να ξεκινήσουν από το μηδέν. Έτσι, ο προγραμματιστής μπορεί να εστιάσει στο δημιουργικό μέρος της εφαρμογής, προγραμματίζοντας μόνο τις λειτουργίες που εξυπηρετούν τον σκοπό της.

3.2.2 Γλώσσα προγραμματισμού C#

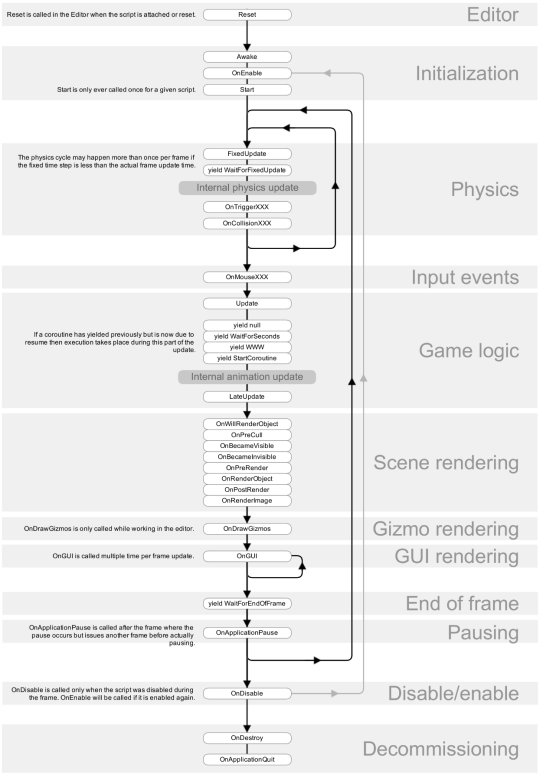
Η C# είναι μία αντικειμενοστραφής (object-oriented [26]) γλώσσα προγραμματισμού, που βασίζεται σε κλάσεις (classes [27]) και συστατικά (components [28]). Αναπτύχθηκε περίπου το 2000 από τη Microsoft, ως μέρος του .NET, και και σύντομα έγινε διεθνώς γνωστή. Προσφέρει πρόσβαση σε μία μεγάλη ποικιλία βιβλιοθηκών και namespaces του .NET, μία από τις πιο σημαντικές τεχνολογίες που επιτρέπει την αλληλεπίδραση μεταξύ εργαλείων διαφορετικών γλωσσών προγραμματισμού. Ένα από τα πιο αξιοσημείωτα namespaces είναι το System, το οποίο περιλαμβάνει την κλάση Linq, η οποία διευκολύνει τη διαχείριση λιστών αντικειμένων και παρέχει ισχυρές δυνατότητες για την επεξεργασία δεδομένων.

3.2.3 Κλάση Monobehaviour στη Unity

Η Unity παρέχει τα δικά της namespaces, όπως το UnityEngine, το οποίο είναι το βασικό namespace και περιλαμβάνει την κλάση **MonoBehaviour**. Η MonoBehaviour δίνει στον κώδικα έναν κύκλο ζωής, με κεντρική συνάρτηση την **Update()**, η οποία καλείται κάθε φορά που ολοκληρώνεται ένα frame. Η **Update()** χειρίζεται τις αλλαγές στη σκηνή μεταξύ δύο διαδοχικών frames, ενώ η **FixedUpdate()** καλείται σε τακτά χρονικά διαστήματα και σχετίζεται με τα physics της σκηνής. Για

παράδειγμα, η **FixedUpdate()** χρησιμοποιείται για τη συχνή κίνηση ενός οχήματος στο παιχνίδι, ενώ η **Update()** είναι υπεύθυνη για γενικότερες ενέργειες.

Η **Awake()** καλείται όταν το script φορτώνεται και είναι κατάλληλη για τη σύνδεση με άλλες ασύγχρονες διαδικασίες. Η **Start()** καλείται μόνο μία φορά κατά την εκκίνηση του script και χρησιμοποιείται για αρχικοποίηση μεταβλητών. Τέλος, οι συναρτήσεις **OnEnable()** και **OnDisable()** καλούνται όταν το αντικείμενο ενεργοποιείται ή απενεργοποιείται αντίστοιχα, ενώ η **OnDestroy()** εκτελείται όταν το αντικείμενο καταστρέφεται. Παρακάτω παρατίθεται διάγραμμα που περιλαμβάνει όλες αυτές τις συναρτήσεις της MonoBehaviour.



Εικόνα 3.2: Βασικές συναρτήσεις της κλάσης Monobehaviour

[https://stackoverflow.com/questions/41630986/what-is-monobehaviour-in-unity-3d]

3.3 Περιβάλλον διεπαφής Unity Editor

Ο Unity Editor ή αλλιώς περιβάλλον διεπαφής της Unity **(user interface)**, περιλαμβάνει ένα πλήθος από επιλογές, ρυθμίσεις και λειτουργικότητες, για τη δημιουργία εφαρμογών ή παιχνιδιών. . Η διεπαφή διαθέτει ένα σύνολο από χρήσιμα παράθυρα, με κυριότερα το Παράθυρο Project **(Project Window)**, η Προβολή Σκηνής **(Scene View)**, η Προβολή Παιχνιδιού **(Game View)**, το Παράθυρο

Ιεραρχίας **(Hierarchy Window)**, το Παράθυρο Παρατηρητή **(Inspector Window)** και η Κονσόλα **(Console).** Στην ενότητα αυτή, περιγράφονται οι βασικές δυνατότητες και η χρησιμότητα του καθενός.

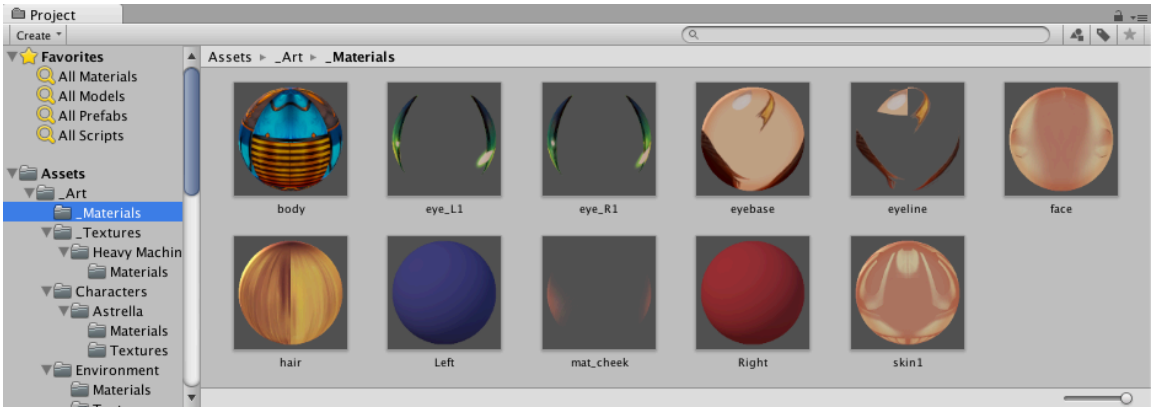
3.3.1 Παράθυρο Project (Project Window)

Το συγκεκριμένο παράθυρο βρίσκεται στο κάτω μέρος της οθόνης και εμφανίζει το σύνολο των αρχείων της εφαρμογής που έχουν εισαχθεί προς χρήση και αφορούν το εκάστοτε project (Εικόνα 3.3). Εκεί τοποθετούνται όλα τα αντικείμενα που δημιουργούνται κατά την διάρκεια της ανάπτυξής της, μέσω της διεπαφής. Τα περιεχόμενα του παραθύρου μπορεί να είναι, από αντικείμενα και scripts μέχρι υφές (textures) και υλικά (materials) τα οποία μπορούν να προστεθούν στη σκηνή με τη λογική drag and drop. Ο βασικός φάκελος που περιλαμβάνει τα στοιχεία της εφαρμογής είναι ο λεγόμενος "Assets". Αφού γίνει η λήψη (download) και η εισαγωγή (import) των πακέτων ή μοντέλων που θα επιλέξει ο προγραμματιστής από το Asset Store, αυτά εμφανίζονται μέσα στον φάκελο "Assets". Για καλύτερη οργάνωση, ιδιαίτερα σε συνεργατικές εφαρμογές, είναι σύνηθες ο φάκελος αυτός να περιέχει του εξής υποφακέλους:

- **Prefabs:** Αντικείμενα και μοντέλα που έχουν αποθηκευτεί με αυτόν τον τρόπο (prefabs) και στη συνέχεια έχουν προστεθεί στο φάκελο είτε μέσω drag and drop είτε μέσω εισαγωγής από το Asset Store.
- **Scripts:** Κώδικας που ο προγραμματιστής έχει αναπτύξει ή έχει προσθέσει από κάποια άλλη πηγή
- **Materials:** Αρχεία που καθορίζουν την εμφάνιση και τα χρώματα των επιφανειών των αντικειμένων στη σκηνή
- **Textures:** Επιπλέον αρχεία που επηρεάζουν την οπτική εμφάνιση και λεπτομέρεια των υλικών.
- **Models:** Μοντέλα που δεν χαρακτηρίζονται prefabs
- **Plugins:** Βιβλιοθήκες σε μορφή DLLs (dynamic – link library) που προσθέτουν επιπλέον λειτουργίες στην εφαρμογή.
- **Utils:** Άλλα αρχεία κώδικα (script) γενικής χρήσης που συνεισφέρουν σε γενικότερες λειτουργίες
- **StreamingAssets:** Αρχεία που σχετίζονται με τη σύνδεση σε δίκτυο και την αποστολή/λήψη πληροφορίας

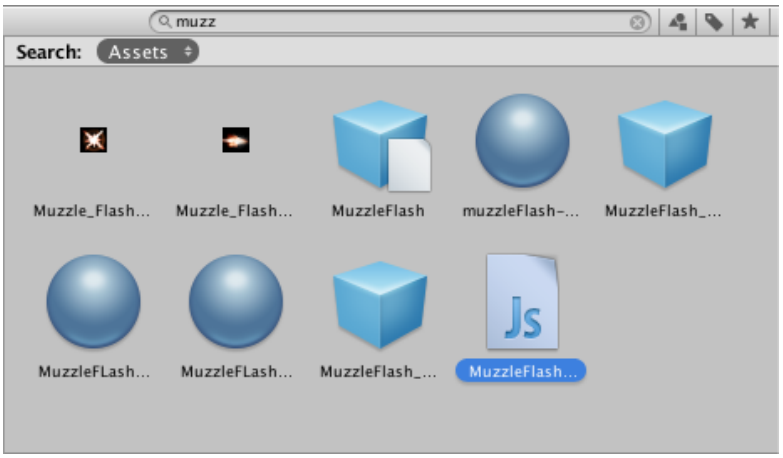
Σε εφαρμογές μεγάλης κλίμακας, όπου συμμετέχουν πολλοί προγραμματιστές, η λειτουργία αναζήτησης του παραθύρου αυτού είναι εξαιρετικά χρήσιμη (Εικόνα 3.4). Επιτρέπει την εύκολη

εύρεση αρχείων, φιλτράροντας το περιεχόμενο ανάλογα με το κείμενο που εισάγει ο χρήστης, διευκολύνοντας έτσι τον γρήγορο εντοπισμό των επιθυμητών στοιχείων.



Εικόνα 3.3: Στιγμιότυπο οθόνης του Παραθύρου Project (Project Window) στη Unity

[https://docs.unity3d.com/550/Documentation/Manual/ProjectView.html]



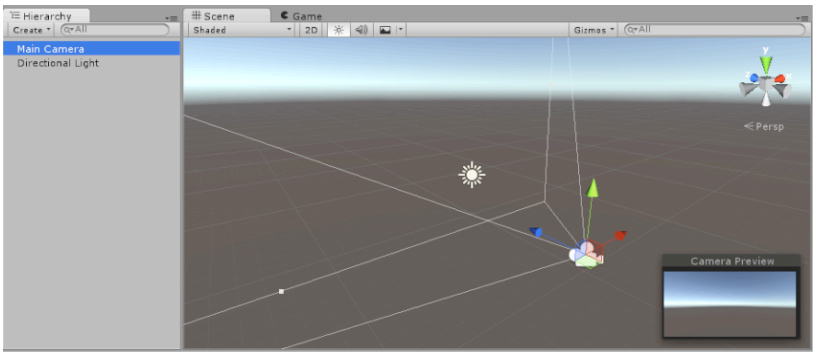
Εικόνα 3.4: Στιγμιότυπου οθόνης από αναζήτηση στο Παράθυρο Project (Project Window)

[https://docs.unity3d.com/550/Documentation/Manual/ProjectView.html]

3.3.2 Προβολή σκηνής (Scene View)

Η Προβολή Σκηνής είναι το κεντρικό και πιο σημαντικό παράθυρο του περιβάλλοντος διεπαφής της Unity (user interface), καθώς αποτελεί τον κύριο χώρο δημιουργίας της εφαρμογής. Μέσω αυτού, ο χρήστης μπορεί να έχει συνεχώς ανοιχτή μια προεπισκόπηση του παιχνιδιού ή της εφαρμογής που έχει δημιουργήσει. Ο προγραμματιστής μπορεί να δημιουργήσει όσες σκηνές θέλει, τοποθετώντας σε αυτές διάφορα στοιχεία (assets) που προέρχονται από το παράθυρο Project. Ο τρισδιάστατος χειρισμός και η παρουσία του πλέγματος που κλειδώνει (grid snapping), επιτρέπουν στον προγραμματιστή να διαμορφώνει το σκηνικό κατά βούληση, τοποθετώντας αντικείμενα, κάμερες, φώτα και άλλα στοιχεία με ακρίβεια στον χώρο (Εικόνα 3.5). Επιπλέον, ο χρήστης έχει τη δυνατότητα να μετακινεί, να περιστρέφει, να αλλάζει την κλίμακα των αντικειμένων και να περιηγείται ελεύθερα

μέσα στο περιβάλλον της σκηνής. Η καλή γνώση του τρόπου διαχείρισης και επεξεργασίας των αντικειμένων σε αυτό το παράθυρο αποτελεί βασική προϋπόθεση για την αποτελεσματική χρήση της Unity.

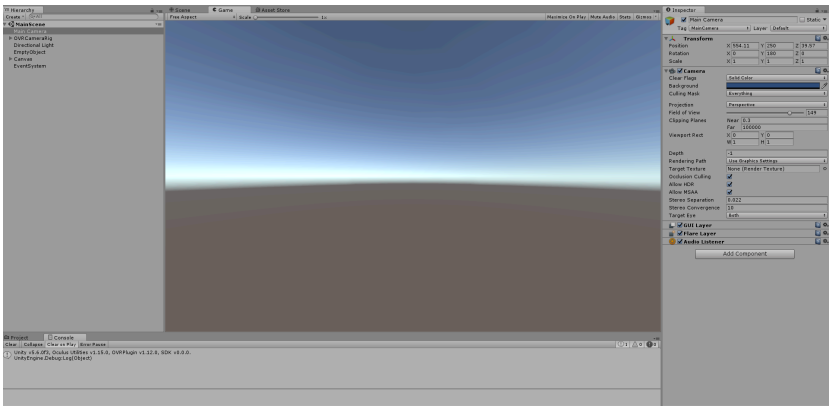


Εικόνα 3.5: Στιγμιότυπο οθόνης της Προβολής Σκηνής (Scene View) στη Unity

[https://docs.unity3d.com/Manual/CreatingScenes.html]

3.3.3 Προβολή παιχνιδιού (Game View)

Το παράθυρο Game View βρίσκεται πίσω από αυτό της σκηνής (Scene View) και εμφανίζεται επιλέγοντας την αντίστοιχη καρτέλα στο πάνω μέρος της διεπαφής. Μέσα από την Προβολή Παιχνιδιού, ο χρήστης έχει τη δυνατότητα να δει μια προεπισκόπηση της εφαρμογής ή του παιχνιδιού που έχει δημιουργήσει από την οπτική της κύριας κάμερας που έχει εισαχθεί και οριστεί ως τέτοια στην Προβολή Σκηνής (Εικόνα 3.6). Κατά την εκτέλεση της εφαρμογής μέσα από τον Unity Editor, το συγκεκριμένο παράθυρο επιτρέπει τον χρήστη να αλληλεπιδρά με το παιχνίδι μέσω πληκτρολογίου και ποντικιού. Ουσιαστικά, το συγκεκριμένο παράθυρο λειτουργεί ως η ρεαλιστική αναπαράσταση της τελικής εμπειρίας που θα έχει ο χρήστης.



Εικόνα 3.6: Στιγμιότυπο οθόνης της Προβολής Παιχνιδιού (Game View) στη Unity

[https://stackoverflow.com/questions/45212989/game-view-not-reflecting-any-changes-in-unity]

3.3.4 Παράθυρο Ιεραρχίας (Hierarchy Window)

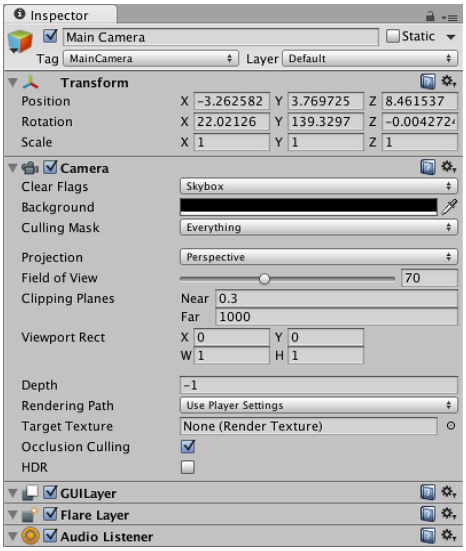
Στην αριστερή πλευρά της διεπαφής βρίσκεται το Παράθυρο Ιεραρχίας, το οποίο περιέχει όλα τα αντικείμενα του παιχνιδιού (game objects) που υπάρχουν στη σκηνή. Ο χρήστης έχει τη δυνατότητα να τροποποιεί εύκολα διάφορες παραμέτρους και χαρακτηριστικά των αντικειμένων, και να δημιουργήσει σχέσεις εξάρτησης χρησιμοποιώντας τη μέθοδο Parenting, να μεταφέρει δηλαδή βασικές ιδιότητες όπως θέση, μέγεθος και κατεύθυνση από ένα αντικείμενο σε ένα άλλο. Η προσθήκη ενός νέου αντικειμένου γίνεται από τον χρήστη με τη λογική του drag and drop. Η λίστα των αντικειμένων που εισάγονται ή αφαιρούνται από τη σκηνή ενημερώνεται αυτόματα. Παρότι τα αντικείμενα στο Παράθυρο Ιεραρχίας εμφανίζονται κατά σειρά δημιουργίας (με τα πιο πρόσφατα στο κάτω μέρος), ο χρήστης έχει τη δυνατότητα να αναδιατάξει τη σειρά τους, σύροντάς τα και τοποθετώντας τα στη θέση που επιθυμεί.

Εικόνα 3.7: Στιγμιότυπο οθόνης του Παραθύρου Ιεραρχίας (Hierarchy Window) στη Unity

[https://docs.unity3d.com/550/Documentation/Manual/Hierarchy.html]

3.3.5 Παράθυρο Επιθεωρητή/Παρατηρητή (Inspector Window)

Το συγκεκριμένο παράθυρο βρίσκεται στη δεξιά πλευρά της διεπαφής και ενεργοποιείται όταν επιλεγεί κάποιο αντικείμενο στη σκηνή. Το παράθυρο αυτό προσφέρει στον χρήστη λεπτομερείς πληροφορίες των βασικών χαρακτηριστικών ενός αντικειμένου, όπως η θέση, ο προσανατολισμός και η κλίμακα συμπεριλαμβανομένων όλων των συστατικών (components) και των επισυναπτόμενων σε αυτό scripts, επιτρέποντας την επεξεργασία κάθε διαθέσιμης παραμέτρου μέσα στη σκηνή.

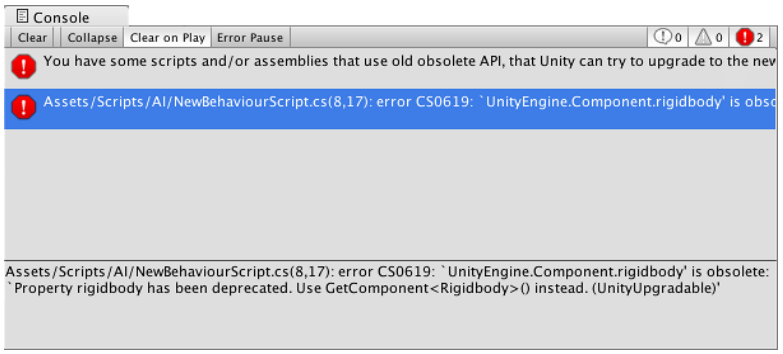


Εικόνα 3.8: Στιγμιότυπο οθόνης του Παραθύρου Επιθεωρητή/Παρατηρητή (Inspector Window)

[https://docs.unity3d.com/550/Documentation/Manual/UsingTheInspector.html]

3.3.6 Κονσόλα (Console)

Η Κονσόλα είναι το παράθυρο της Unity το οποίο είναι υπεύθυνο να εμφανίζει κατά βάση όλα τα λάθη, τις προειδοποιήσεις ή ακόμα και τις ενημερώσεις που συμβαίνουν κατά την εκτέλεση ή ανάπτυξη μιας εφαρμογής ή ενός παιχνιδιού. Αποτελεί ένα από τα σημαντικότερα παράθυρα στην Unity, καθώς διευκολύνει σημαντικά τη διαδικασία εντοπισμού και επίλυσης σφαλμάτων μέσω αυτοματοποιημένων μηνυμάτων σε πραγματικό χρόνο, για τον αριθμό αλλά και το σημείο των σφαλμάτων που εντοπίστηκαν. Με ένα απλό διπλό κλικ πάνω στο αντίστοιχο μήνυμα, ο χρήστης μεταφέρεται αυτόματα στο ακριβές σημείο όπου προέκυψε το σφάλμα, είτε αυτό εντοπίζεται σε κάποιο σημείο του κώδικα (script) που έχει δημιουργήσει ή εισάγει, είτε για πρόβλημα στο περιβάλλον διεπαφής.



Εικόνα 3.9: Στιγμιότυπο οθόνης της Κονσόλας (Console) στη Unity

[https://docs.unity3d.com/2018.4/Documentation/Manual/Console.html]

Κεφάλαιο 4 – Υλοποίηση εφαρμογής

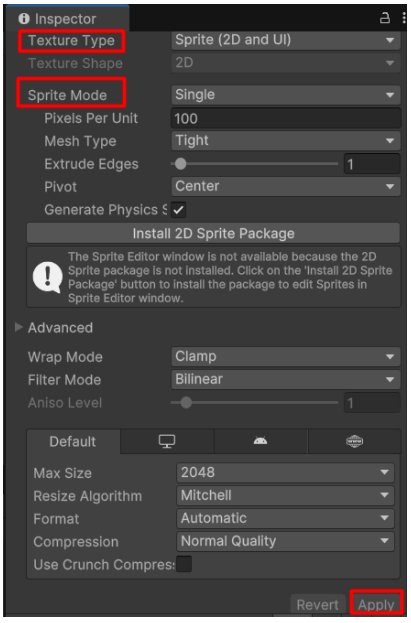
Στο κεφάλαιο αυτό, περιγράφεται η σχεδίαση και η λειτουργία της εφαρμογής που δημιουργήθηκε, καθώς και διάφορες δυνατότητες που αυτή έχει. Αρχικά, θα αναφερθεί το αντικείμενο που πραγματεύεται και σε κάποιες λεπτομέρειες που τη χαρακτηρίζουν. Στη συνέχεια, αναφέρονται τα αντικείμενα της σκηνής, καθώς και πως αυτά αποκτήθηκαν. Ακολούθως, υπάρχει αναλυτική περιγραφή των scripts και των λειτουργιών που αυτά προσφέρουν αλλά και το πως επετεύχθη η απομακρυσμένη σύνδεση μέσω διαδικτύου.

4.1 Δημιουργία και στήσιμο σκηνής

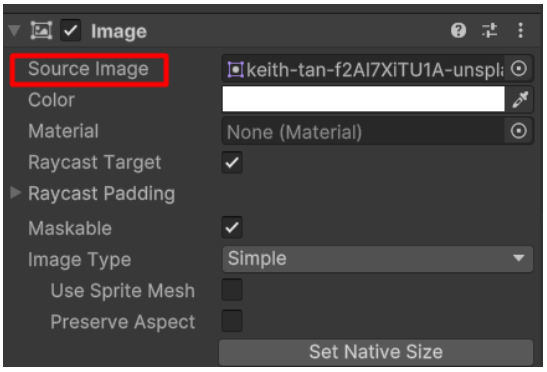
Ως πρώτο βήμα, με το που ολοκληρωθεί η εγκατάσταση της μηχανής Unity στον υπολογιστή, ο χρήστης είναι έτοιμος να επιλέξει τον τύπο (2D, 3D, Hdrp ή Urp) της εφαρμογής που πρόκειται να αναπτύξει, στην προκειμένη περίπτωση 3D και να δημιουργήσει το δικό του project (Create Project) από το menu του Unity Hub. Στην συνέχεια, ο χρήστης βλέπει όπως έχω περιγράψει προηγουμένως το περιβάλλον διεπαφής του Unity Editor (user interface), αλλά και τα παράθυρα από τα οποία αποτελείται. Αυτό είναι ουσιαστικά και μια σκηνή στη Unity. Επόμενο βήμα, μιας και η εφαρμογή πρόκειται να γίνει εξαγωγή κυρίως για κινητές συσκευές (Android), ο χρήστης μπορεί να επιλέξει από το μενού της διεπαφής (File -> Build Settings) και να δει όλες τις υποστηριζόμενες συσκευές που διαθέτει η Unity. Όσο πιο γρήγορα προβεί σε αλλαγής πλατφόρμας, στην προκειμένη περίπτωση Android, τόσο πιο γρήγορα θα γίνει και η τελική εξαγωγή του παιχνιδιού λόγω του μικρότερου όγκου των δεδομένων. Από τη στιγμή που γίνει η αλλαγή πλατφόρμας (switch platform), ο χρήστης μπορεί να αρχίσει και να χτίζει την εφαρμογή του.

Πρώτο βήμα, από το μενού στη κορυφή της διεπαφής (Gameobject) ή πραγματοποιώντας δεξί κλικ στο παράθυρο ιεραρχίας (Hierarchy Window), ο χρήστης είναι έτοιμος να προσθέσει όλα τα αντικείμενα (objects) που θα τοποθετήσει στη σκηνή. Όλα τα αντικείμενα στη συγκεκριμένη εφαρμογή βρίσκονται στην κατηγορία UI (texts, input fields, buttons, slider, image, κ.ά.). Πρώτο αντικείμενο, ένα UI panel (προαιρετικό). Έπειτα να προσαρμόσει την αναλογία διαστάσεων (aspect ratio) σε 16:9 από το παράθυρο προβολής (Game View), εφόσον δεν έχει αλλάξει όταν έγινε η αλλαγή πλατφόρμας . Επόμενο βήμα, να προσθέσει με τον ίδιο τρόπο μια εικόνα (image), η οποία θα είναι και το background του παιχνιδιού. Υπάρχει διαφορά όσον αφορά τον τύπο εικόνας (image και raw image). Το image είναι το πιο συνηθισμένο UI στοιχείο για την εμφάνιση εικόνας και χρησιμοποιείται κυρίως για 2D γραφικά (όπως κουμπιά, background, εικονίδια κ.λπ.). Χρησιμοποιεί το Sprite ως πηγή εικόνας, το οποίο είναι συνήθως μια εικόνα τύπου PNG, JPEG ή TGA. Το Sprite

είναι ρυθμισμένο για τη βελτιστοποίηση απόδοσης (σε σύγκριση με απλές εικόνες) και μπορεί να χρησιμοποιηθεί σε UI, σε χαρακτήρες, ή σε αντικείμενα. Το Raw Image χρησιμοποιείται όταν κάποιος θέλει να εμφανίσει εικόνες ακριβώς όπως είναι, χωρίς καμία επεξεργασία ή βελτιστοποίηση. Με το που γίνει η εισαγωγή της εικόνας, θα εμφανιστεί στο panel ένα μικρό άσπρο τετράγωνο. Αυτό το τετράγωνο, ο χρήστης πρέπει να το βλέπει κάτω αριστερά στο παράθυρο της σκηνής (Scene View), ειδάλλως θα σημαίνει ότι βλέπει την εφαρμογή σε ανάποδη όψη. Επιλέγοντας την main camera και με το δεξί κλικ μπορεί να κινηθεί στη σκηνή και να προσαρμόσει την όψη της εφαρμογής με τον σωστό τρόπο. Ο χρήστης προσθέτει την εικόνα που επιθυμεί να έχει στο background απλά σύροντας την στο παράθυρο Project, εκεί όπου θα τοποθετηθούν όλα τα αρχεία του παιχνιδιού. Αφού την επιλέξει, πρέπει να ορίσει το texture type στο παράθυρο επιθεωρητή (Inspector Window) σε Sprite (2D and UI), το sprite mode σε single και να πατήσει apply. Στη συνέχεια, επιλέγει το αντικείμενο της εικόνας που δημιούργησε προηγουμένως στο παράθυρο ιεραρχίας (Hierarchy Window) και μετακινεί την εικόνα από το παράθυρο Project στο πεδίο του source image στο συστατικό (component) της εικόνας στο παράθυρο επιθεωρητή(Inspector Window). Από το ίδιο παράθυρο μπορεί να προσαρμόσει τις λεπτομέρειες της εικόνας που θα οριστεί για background όπως αυτός επιθυμεί (μέγεθος, θέση, χρώμα ,κτλ.).

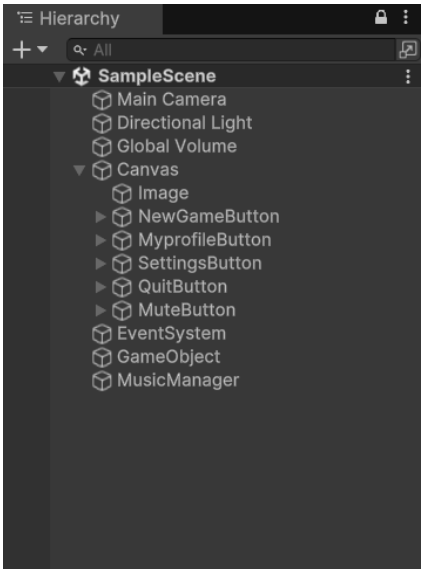


Εικόνα 4.1: Ορισμός Background εικόνας στο Inspector Window



Εικόνα 4.2: Τοποθέτηση εικόνας στο Image Component στο Inspector Window

Η διαδικασία τώρα για το υπόλοιπο στήσιμο του παιχνιδιού, δεν αλλάζει μεθοδολογία. Ο χρήστης τοποθετεί όλα τα αντικείμενα που επιθυμεί στη σκηνή (κουμπιά, input fields, sliders, toggle κ.ά.), προσαρμόζει το μέγεθος τους, την τοποθεσία τους αλλά και τις ιδιότητες τους τόσο από το παράθυρο επιθεωρητή (Inspector Window), όσο και από την μπάρα εργαλείων στην προβολή σκηνής (Scene View).



Εικόνα 4.3: Ο Canvas της πρώτης σκηνής (SampleScene) στο Hierarchy Window που αποτελείται από: 5 κουμπιά (buttons) και το background image

Μόλις είναι έτοιμος, μπορεί να αποθηκεύσει τη σκηνή με όποιο όνομα επιθυμεί. Για τη δημιουργία νέα σκηνής επιλέγει από το μενού της διεπαφής file και στη συνέχεια New Scene. Για κάθε σκηνή ο χρήστης είναι υποχρεωμένος να την τοποθετήσει στη Scene List η οποία βρίσκεται στα Build Settings (File -> Build Settings -> Scene List -> Add Open Scenes), με τη λογική drag and drop. Στη δική μου

περίπτωση για την υλοποίηση της εφαρμογής (quiz application) δημιούργησα 12 σκηνές, κάθε μια με τα δικά της αντικείμενα (objects) και χαρακτηριστικά. Πάμε να δούμε κατά σειρά ποιες είναι αυτές:

1. **SampleScene**: Πρόκειται για την πρώτη σκηνή του παιχνιδιού και αυτή που αντικρίζει ο παίκτης με το που ανοίξει την εφαρμογή. Αποτελείται από 5 κουμπιά. Το Play, το MyProfile, το Settings, το Quit και το Mute. Ο παίκτης πατώντας το Play μεταφέρεται στην επόμενη σκηνή η οποία είναι η GameModeMenuScene, με το MyProfile στη ProfileScene, με το settings στην OptionsMenuScene, το Mute όπου μπορεί να ενεργοποιήσει/απενεργοποιήσει άμεσα τον ήχο και το Quit με το οποίο οδηγείται στην έξοδο του παιχνιδιού.
2. **OptionsMenuScene:** Αποτελείται από ένα slider και 2 κουμπιά. Το slider είναι υπεύθυνο για την background μουσική του παιχνιδιού και με αυτό ο παίκτης μπορεί να προσαρμόσει την ένταση στο επιθυμητό. Επιπλέον αποτελείται από το κουμπί Rules, με το οποίο ο χρήστης μεταφέρεται σε μια άλλη σκηνή όπου περιγράφονται οι κανόνες του παιχνιδιού σε μορφή βίντεο και το κουμπί back με το οποίο ο παίκτης επιστρέφει στην προηγούμενη σκηνή (SampleScene).

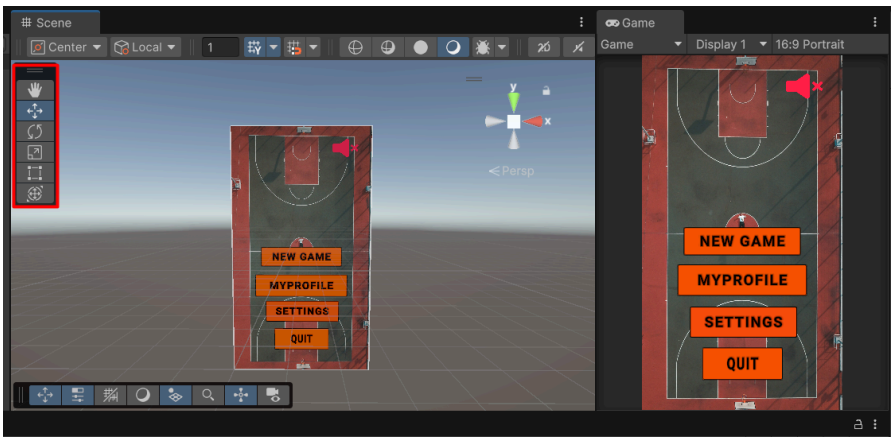
3. **ProfileScene**: Στη ProfileScene ο παίκτης μπορεί ουσιαστικά να χτίσει το προφίλ του. Αποτελείται από ένα Input Field, 2 κουμπιά και 4 text. Ο παίκτης πληκτρολογεί το όνομα που θέλει να έχει στο παιχνίδι στο Input Field και με το κουμπί Save το αποθηκεύει για όλη τη διάρκεια του παιχνιδιού. Το ένα text αφορά το όνομα του παίκτη. Με το που κρατήσει το όνομα του εμφανίζεται μήνυμα αποθήκευσης. Τα 2 από τα υπόλοιπα 3 text αφορούν τις νίκες και τις ήττες του παίκτη όταν αυτός επιλέγει να παίξει την MultiPlayer έκδοση. Μετά το τέλος του παιχνιδιού ανάλογα με το αποτέλεσμα αποθηκεύει στο προφίλ του παίκτη το κατάλληλο text (νίκες/ήττες). Το τρίτο και τελευταίο text αποθηκεύει το καλύτερο σκορ του παίκτη(Highscore) στη SinglePlayer έκδοση. Αντίστοιχα με το κουμπί Back, ο παίκτης μεταφέρεται στη προηγούμενη σκηνή, στη προκειμένη περίπτωση την SampleScene.

4. **GameModeMenuScene**: Αποτελείται από 3 κουμπιά. Το Local το οποίο εφόσον πατηθεί μεταφέρεται στη LocalGameModeMenu σκηνή, το Online όπου μεταφέρεται στη OnlineMatchMenu σκηνή που είναι υπεύθυνη για την εξ αποστάσεως σύνδεση μεταξύ δύο παικτών και το back με το οποίο μεταφέρεται στην πρώτη και προηγούμενη σκηνή του παιχνιδιού(SampleScene).
5. **LocalGameModeMenu**: Έχει το ίδιο μοτίβο με την GameModeMenuScene καθώς περιέχει εξίσου 3 κουμπιά. Το Single, το οποίο, όταν πατηθεί, μεταφέρει τον χρήστη στη σκηνή SinglePlayerScene , το Multi το οποίο τον οδηγεί στη MultiReady σκηνή και το Back, το οποίο επιστρέφει στην αμέσως προηγούμενη σκηνή, δηλαδή την GameModeMenuScene.

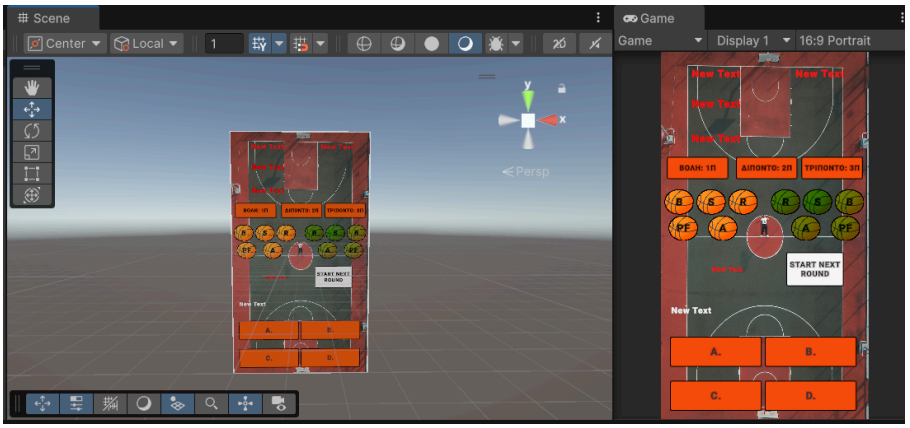
6. **SinglePlayerScene**: Η συγκεκριμένη σκηνή αποτελείται από 2 κουμπιά και ένα κείμενο (text). Το κουμπί Ready με το οποίο ο παίκτης δίνει την εντολή για να ξεκινήσει το παιχνίδι στη SinglePlayer έκδοση, το κουμπί Back με το οποίο μεταφέρεται στην προηγούμενη σκηνή (SinglePlayerScene) και το text το οποίο εμφανίζει το όνομα του παίκτη που έχει αποθηκεύσει ο ίδιος στη ProfileScene.
7. **SingleMenuScene**: Πρόκειται στην ουσία για την σκηνή που παίζεται το παιχνίδι όταν ο παίκτης επιλέγει την SinglePlayer έκδοση. Αποτελείται από τα 4 κουμπιά των απαντήσεων(A, B, C, D), ένα text στο οποίο εμφανίζεται η ερώτηση για τον παίκτη, 2 text που αφορούν το σκορ, ένα για το σκορ του τρέχοντος παιχνιδιού και ένα για το Highscore που ανέφερα προηγουμένως, 3 κουμπιά που είναι υπεύθυνα για το επίπεδο δυσκολίας των ερωτήσεων (βολή, δίποντο, τρίποντο), 2 ακόμη text εκ των οποίων το ένα δείχνει τον αριθμό της ερώτησης και το άλλο το όνομα του παίκτη, ένα πάνελ με 3 κουμπιά για τις βοήθειες του παίκτη (ασίστ, φάουλ, ριμπάουντ) και το πάνελ (panel) το οποίο περιέχει το κουμπί που είναι υπεύθυνο για την έναρξη του επόμενο γύρου αλλά και του τελικού μηνύματος με το τελικό σκορ του παίκτη στο τέλος του παιχνιδιού.
8. **MultiReady**: Περιέχει 2 Input Field και 2 κουμπιά. Οι δύο παίκτες τοποθετούν τα ονόματα τους στα αντίστοιχα πεδία (κατά προτίμηση ο ένας το ίδιο όνομα που έχει αποθηκεύσει νωρίτερα) και με το κουμπί Ready δίνουν την εντολή να ξεκινήσει το παιχνίδι στη Multiplayer έκδοση, ώστε να μεταφερθούν στη MultiMenuScene. Με το κουμπί Back ως γνωστόν πλέον μεταφέρεται στη προηγούμενη σκηνή, LocalGameModeMenu.
9. **MultiMenuScene**: Πρόκειται στην ουσία για την σκηνή που παίζεται το παιχνίδι όταν ο παίκτης επιλέγει την MultiPlayer έκδοση σε τοπικό επίπεδο, δηλαδή όταν δύο παίκτες παίζουν αντίπαλοι από την ίδια συσκευή. Αποτελείται από τα 4 κουμπιά των απαντήσεων(A, B, C, D), ένα text στο οποίο εμφανίζεται η ερώτηση για τους παίκτες, 2 text που αφορούν το σκορ, ένα για τον κάθε παίκτη, τρία κουμπιά όπως και στο SinglePlayer τα οποία είναι υπεύθυνα για το επίπεδο δυσκολίας των ερωτήσεων (βολή, δίποντο, τρίποντο), 2 ακόμη text εκ των οποίων το ένα δείχνει τον αριθμό της ερώτησης και το άλλο το όνομα του παίκτη εναλλάξ, το πάνελ (panel) το οποίο περιέχει το κουμπί που είναι υπεύθυνο για την έναρξη του επόμενο γύρου αλλά και του τελικού μηνύματος για το ποιος κέρδισε αλλά και το τελικό σκορ του παιχνιδιού και το πάνελ (panel) που περιέχει τα κουμπιά με τις βοήθειες των παικτών, 5 για κάθε παίκτη στη συγκεκριμένη περίπτωση.
10. **RulesScene**: Περιγραφή των κανόνων του παιχνιδιού.
11. **Onlinematchmenu**: Πρόκειται για τη σκηνή η οποία είναι υπεύθυνη για την online σύνδεση μεταξύ δύο παικτών. Αποτελείται από 5 κουμπιά, ένα text και ένα Input Field. Το κουμπί Find, με το οποίο ο χρήστης συνδέεται τυχαία με έναν χρήστη απομακρυσμένα, το κουμπί create room με το οποίο ο χρήστης δημιουργεί ένα ιδιωτικό δωμάτιο παίρνοντας έναν

συγκεκριμένο κωδικό, το κουμπί join room το οποίο είναι υπεύθυνο για να εμφανιστεί το Input Field, το Input Field ώστε να τοποθετηθεί ο συγκεκριμένος κωδικός από έναν δεύτερο χρήστη ώστε να συνδεθεί αποκλειστικά με τον χρήστη που επιθυμεί και μετέπειτα το κουμπί ready, με το οποίο ξεκινά το online παιχνίδι μεταξύ τους. Αντίστοιχα, το κουμπί back οδηγεί στην προηγούμενη σκηνή του παιχνιδιού.

12. **MultiMenuScenee**: Πρόκειται στην ουσία για τη σκηνή του παιχνιδιού που αφορά το online παιχνίδι. Δεν διαφέρει σε τίποτα από την Multimenuscene.



Εικόνα 4.4: Προβολή των αντικειμένων της SampleScene σκηνής στο Scene View & Game View



Εικόνα 4.5: Προβολή της Multimenuscene

Το παιχνίδι περιλαμβάνει πολλές σκηνές, όπως το SampleScene (αρχική σκηνή), το ProfileScene (για την αποθήκευση του προφίλ του παίκτη) και το GameModeMenuScene (επιλογή τοπικού ή online παιχνιδιού). Ο χρήστης μπορεί να επιλέξει τη SinglePlayer ή MultiPlayer έκδοση, με δυνατότητες βοήθειας και παρακολούθησης σκορ σε κάθε γύρο. Κάθε σκηνή περιλαμβάνει κουμπιά,

βοήθειες, πεδία εισαγωγής και κείμενα για την αλληλεπίδραση του χρήστη. Όλες αυτές οι σκηνές και τα στοιχεία αφορούν τη σχεδίαση του παιχνιδιού και τώρα θα εστιάσω στη λειτουργικότητα του παιχνιδιού μέσω κώδικα (scripts).

4.2 Εισαγωγή Script - Σύνδεση σκηνών

Μέχρι στιγμής, έχει δημιουργηθεί η δομή του παιχνιδιού χωρίς όμως να είναι τίποτα λειτουργικό. Εδώ έρχεται η εισαγωγή των script ή αλλιώς ο κώδικας που θα τοποθετηθεί, ώστε να να αρχίσει να παίρνει ρεαλιστική μορφή το παιχνίδι. Πρώτα και κύρια, για να δημιουργήσει ο χρήστης ένα script, ακολουθεί την ίδια διαδικασία όπως προηγουμένως. Από το μενού της διεπαφής της Unity στο πάνω μέρος της οθόνης στην επιλογή assets ή πραγματοποιώντας δεξί κλικ στο παράθυρο project, ο χρήστης δημιουργεί το script (create → MonoBehaviour Script) και δίνει το κατάλληλο όνομα. Για λόγους ευκολίας και διαχείρισης, ο χρήστης μπορεί να δώσει στο script το όνομα που έχει δώσει στην εκάστοτε σκηνή. Για κάθε σκηνή που έχει δημιουργηθεί, ο χρήστης οφείλει να δημιουργήσει το αντίστοιχο script που θα συνδέει την μια σκηνή με την επόμενη αλλά και το αντίστροφο. Επομένως για κάθε σκηνή, είναι απαραίτητο να υπάρχει και το αντίστοιχο script το οποίο θα είναι υπεύθυνο για τη σύνδεση μεταξύ των σκηνών του παιχνιδιού.

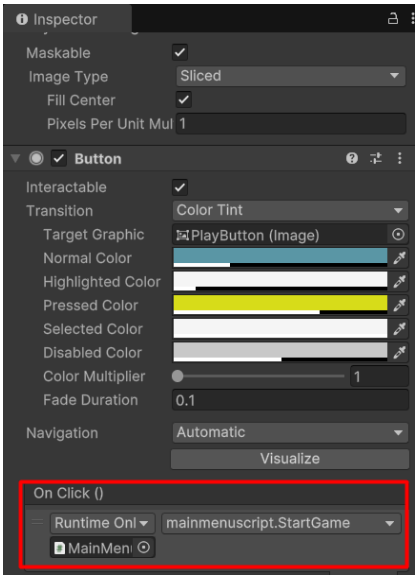
Με το που εισάγει ο χρήστης το πρώτο script και το ανοίξει σε κάποιο περιβάλλον ανάπτυξης (IDE) για προγραμματισμό (στη δική μου περίπτωση Visual Studio), θα αντικρίσει δύο μεθόδους, την void Start() και την void Update() αλλά και κάποιες γραμμές κώδικα στην κορυφή που ονομάζονται namespace. Η void Start() καλείται μια φορά όταν ξεκινήσει το παιχνίδι και εκεί τοποθετούνται συχνά αντικείμενα, τιμές, ρυθμίσεις, κ.ά., ενώ η void Update() καλείται κάθε δευτερόλεπτο και σε αυτή τοποθετούνται πράγματα που έχουν να κάνουν με κινήσεις χαρακτήρα ή με κάποια είσοδο από το πληκτρολόγιο/ποντίκι. Στο namespace τοποθετούνται κάποιες γραμμές κώδικα που δίνουν πρόσβαση σε κλάσεις, συναρτήσεις, μεταβλητές που είναι ορισμένες αλλού (π.χ. από τη Unity ή από άλλες βιβλιοθήκες). Παρακάτω δίνεται ένα παράδειγμα κώδικα (script), στο οποίο η κεντρική σκηνή SampleScene, η οποία διαθέτει τα 5 κουμπιά (Play, Options, MyProfile, Quit, Mute) συνδέεται με την επόμενη όπως έχω αναφέρει προηγουμένως, εφόσον ο χρήστης πατήσει κλικ πάνω στο αντίστοιχο κουμπί.

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SampleSceneScript : MonoBehaviour // Όνομα script
```

```
{
    public void NewGame()
    {
        SceneManager.LoadScene("gamemodemenuscene"); // Μετάβαση στη σκηνή επιλογής τρόπου παιχνιδιού
    }
    public void MyProfile()
    {
        SceneManager.LoadScene("profilescene"); // Μετάβαση στη σκηνή profilescene
    }

    public void Settings()
    {
        SceneManager.LoadScene("optionsmenuscene"); // Μετάβαση στη σκηνή ρυθμίσεων
    }

    public void QuitGame()
    {
        Debug.Log("Εξοδος από το παιχνίδι"); // Εμφάνιση μηνύματος στην οθόνη
        Application.Quit(); // Κλείσιμο του παιχνιδιού
    }
}
```

Στον παραπάνω κώδικα, φαίνεται το script που είναι υπεύθυνο για την μετάβαση από την κεντρική σκηνή στην επόμενη, όταν ο χρήστης ξεκινήσει την εφαρμογή. Πρώτο βήμα, να αφαιρέσει τις μεθόδους void Start() και void Update() οι οποίες δεν χρειάζονται προς το παρόν και να τοποθετήσει στο namespace του script που είναι υπεύθυνο για τις αλλαγές μεταξύ των σκηνών το παρακάτω: using UnityEngine.SceneManagement; . Τώρα για το κάθε κουμπί δημιουργεί και μία μέθοδο. Όταν ο χρήστης πατήσει το κουμπί play, τότε καλείται η μέθοδος StartGame(), εκτελείται η εντολή SceneManager.LoadScene (στην παρένθεση το όνομα της σκηνής που θέλει ο χρηστής να μεταβεί όταν πατήσει το κουμπί) και ο παίκτης μεταβαίνει στη επόμενη σκηνή. Η ίδια λογική ισχύει και για τα υπόλοιπα κουμπιά. Με το που ολοκληρώσει τον απαραίτητο κώδικα και αποθηκεύσει το script, ο χρήστης πρέπει να δημιουργήσει ένα κενό αντικείμενο (GameObject) στο παράθυρο Ιεραρχίας (Hierarchy Window), πάνω στο οποίο θα τοποθετήσει το script σέρνοντας το από το παράθυρο Project με τη λογική drag and drop (Εικόνα 4.3 - Gameobject). Μετέπειτα, για το κάθε κουμπί (button) στο παράθυρο επιθεωρητή (Inspector Window), πρέπει να ρυθμίσει και την αντίστοιχη μέθοδο στο αντίστοιχο συστατικό (component). Οφείλει να τοποθετήσει στο εκάστοτε κουμπί το Gameobject που περιέχει το script και να ρυθμίσει την αντίστοιχη μέθοδο που έχει προγραμματίσει.

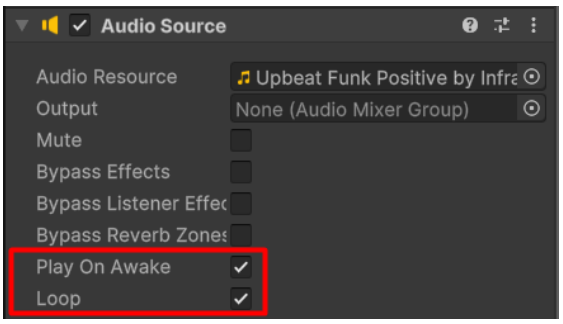
Πιο συγκεκριμένα, η παρακάτω εικόνα δείχνει πώς το κουμπί(button) είναι ρυθμισμένο ώστε να εκτελεί μια συνάρτηση όταν πατηθεί.



Εικόνα 4.6: Ρύθμιση OnClick() σε κουμπί ώστε να εκτελεί μια συνάρτηση

4.3 Τοποθέτηση background μουσικής, video κανόνων

Για τη δημιουργία μουσικής στο background μέρος του παιχνιδιού, ο χρήστης οφείλει να δημιουργήσει στην πρώτη σκηνή της εφαρμογής ένα κενό αντικείμενο (Gameobject) στο παράθυρο Ιεραρχίας (Hierarchy Window) και να το ονομάσει αναλόγως. Επόμενο βήμα, να προσθέσει το αρχείο της μουσικής που επιθυμεί στο παράθυρο project με τη γνωστή λογική drag and drop. Στη συνέχεια, αφού επιλέξει το κενό αντικείμενο (Backgroundmusic Gameobject), τοποθετεί στο παράθυρο επιθεωρητή (Inspector Window) ένα συστατικό (component) με το όνομα AudioSource. Στο πεδίο Audio Resource σέρνει και τοποθετεί το αρχείο ήχου και διαλέγει την επιλογή Play on Awake() και Loop. Αντίστοιχα για να λειτουργεί η μουσική μεταξύ των σκηνών καθόλη τη διάρκεια του παιχνιδιού χωρίς διακοπές κατά τη διάρκεια αλλαγής σκηνών, απαιτείται εξίσου κάποιος κώδικας ο οποίος θα τοποθετηθεί στο Gameobject που δημιούργησε προηγουμένως, στην πρώτη σκηνή δηλαδή την SampleScene.



Εικόνα 4.7: Ρύθμιση background Μουσικής

```
using UnityEngine;

public class MusicManager : MonoBehaviour
{
    public static MusicManager Instance;  // Δημιουργεί μια στατική μεταβλητή για να υπάρχει μόνο ένα MusicManager

    private AudioSource audioSource;  // Ορισμός της μεταβλητής AudioSource που θα αναπαράγει τη μουσική

    void Awake()
    {
        // Αν υπάρχει ήδη ένα MusicManager, καταστρέφει το καινούριο για να διασφαλίσει ότι υπάρχει μόνο ένα
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject);  // Καταστρέφει το νέο αντικείμενο αν υπάρχει ήδη το υπάρχον
            return;  // Τερματίζει την εκτέλεση αυτής της μεθόδου
        }

        Instance = this;  // Θέτει την τρέχουσα κλάση ως μοναδική
        DontDestroyOnLoad(gameObject);  // Κάνει το αντικείμενο του MusicManager να μην καταστρέφεται όταν αλλάζει σκηνή

        audioSource = GetComponent<AudioSource>();  // Αποκτά το AudioSource που είναι συνδεδεμένο με το αντικείμενο

        // Φορτώνει την ένταση από τα PlayerPrefs
        float savedVolume = PlayerPrefs.GetFloat("MusicVolume", 1f);
        audioSource.volume = savedVolume;  // Εφαρμόζει την αποθηκευμένη ένταση στο AudioSource
    }

    // Μέθοδος για να αλλάξεις την ένταση της μουσικής
    public void SetVolume(float volume)
```

```
    {
        audioSource.volume = volume;  // Ενημερώνει την ένταση του ήχου στο AudioSource
        PlayerPrefs.SetFloat("MusicVolume", volume);  // Αποθηκεύει την ένταση στη PlayerPrefs
        PlayerPrefs.Save();  // Αποθηκεύει τις αλλαγές στα PlayerPrefs για να παραμείνουν για μελλοντικές συνεδρίες
    }
}
```

Το script MusicManager στη Unity διαχειρίζεται την αναπαραγωγή και τον έλεγχο της μουσικής στο παιχνίδι. Χρησιμοποιεί το AudioSource για να αναπαράγει μουσική και εξασφαλίζει ότι η μουσική παραμένει ακέραιη ακόμα και όταν ο χρήστης μεταβαίνει σε άλλες σκηνές, μέσω της χρήσης της μεθόδου DontDestroyOnLoad. Το MusicManager είναι μοναδικό στο παιχνίδι, διασφαλίζοντας ότι υπάρχει μόνο ένα αντίγραφο του (μέσω του singleton pattern). Όταν το παιχνίδι ξεκινά, το script φορτώνει την ένταση της μουσικής από τα PlayerPrefs (ή χρησιμοποιεί την προεπιλεγμένη τιμή 1f αν δεν υπάρχει αποθηκευμένη τιμή) και την εφαρμόζει στο AudioSource. Επίσης, παρέχεται η δυνατότητα αλλαγής της έντασης μέσω της μεθόδου SetVolume, η οποία ενημερώνει την ένταση του AudioSource και αποθηκεύει την τιμή της έντασης στο PlayerPrefs, ώστε να διατηρείται η ρύθμιση και σε μελλοντικές συνεδρίες του παιχνιδιού.

Με λίγα λόγια, το συγκεκριμένο script εξασφαλίζει μια συνεχή και ελεγχόμενη εμπειρία ακρόασης μουσικής στο φόντο, προσφέροντας επαγγελματική αίσθηση συνέχειας στον ήχο του παιχνιδιού σου.

Όσον αφορά την ένταση της μουσικής, ο χρήστης μπορεί να την διαχειριστεί από το κουμπί Mute button το οποίο βρίσκεται στη πρώτη σκηνή της εφαρμογής (SampleScene) και από το slider που βρίσκεται στη OptionsMenuScene. Επομένως, πρέπει να δημιουργηθεί αντίστοιχα ένα νέο script στη SampleScene που θα είναι υπεύθυνο για το Mute Button και ένα στη ProfileScene όπου ο παίκτης θα μπορεί να ρυθμίζει την ένταση μέσω slider.

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class MuteButton : MonoBehaviour
{
    public Button muteButton;  // Δηλώνει το κουμπί για τη σίγαση του ήχου
    public TMP_Text buttonText;  // Δηλώνει το κείμενο του κουμπιού για να αλλάζει ανάλογα με την κατάσταση

    private bool isMuted = false;  // Αρχική κατάσταση του ήχου (σιωπή ή όχι)
    private AudioSource musicSource;  // Το AudioSource που αναπαράγει τη μουσική
```

```
    void Start()
    {
        // Βρίσκει το MusicManager και το AudioSource για να ελέγξει και να τροποποιήσει τον ήχο
        musicSource = GameObject.Find("MusicManager").GetComponent<AudioSource>();

        // Προσθέτει τον listener για το κλικ στο κουμπί
        muteButton.onClick.AddListener(ToggleMute);

        // Ελέγχει αν η μουσική ήταν σιγή πριν από το προηγούμενο παιχνίδι (Αποθηκεύεται στις PlayerPrefs)
        isMuted = PlayerPrefs.GetInt("MusicMuted", 0) == 1;  // Αν η τιμή είναι 1, το παιχνίδι ήταν σε σιγή
        musicSource.mute = isMuted;  // Ρυθμίζει τη σιγή του ήχου σύμφωνα με την αποθηκευμένη τιμή
        UpdateButtonLabel();  // Ενημερώνει το κείμενο του κουμπιού
    }

    // Μέθοδος για την αλλαγή κατάστασης του ήχου (mute/unmuted)
    void ToggleMute()
    {
        isMuted = !isMuted;  // Αντιστρέφει την κατάσταση της σιγής
        musicSource.mute = isMuted;  // Εφαρμόζει τη νέα κατάσταση στην μουσική
        PlayerPrefs.SetInt("MusicMuted", isMuted ? 1 : 0);  // Αποθηκεύει την κατάσταση σιγής για την επόμενη φορά
        UpdateButtonLabel();  // Ενημερώνει το κείμενο του κουμπιού
    }

    // Μέθοδος για την ενημέρωση του κειμένου του κουμπιού
    void UpdateButtonLabel()
    {
        if (buttonText != null)  // Ελέγχει αν υπάρχει το αντικείμενο κειμένου
        {
            buttonText.text = isMuted ? "🔇 " : "🔊 ";  // Αν είναι σιωπή, εμφανίζει 🔇, αλλιώς 🔊
        }
    }
}
```

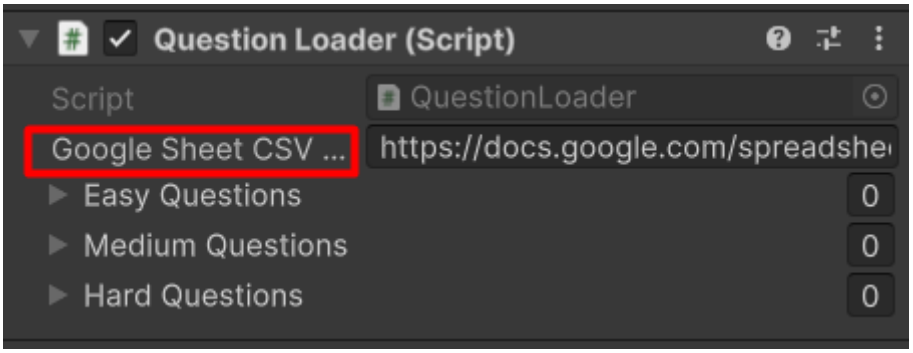
Το script MuteButton επιτρέπει στον χρήστη να ενεργοποιεί ή να απενεργοποιεί τον ήχο του παιχνιδιού μέσω ενός κουμπιού. Η κατάσταση σιγής (mute) αποθηκεύεται στα PlayerPrefs, εξασφαλίζοντας ότι παραμένει η ίδια στις επόμενες συνδέσεις. Ενημερώνει το κείμενο του κουμπιού, εμφανίζοντας αν ο ήχος είναι ενεργοποιημένος (🔊) ή σιγημένος (🔇). Αφού προσθέσει το script στο GameObject, ο χρήστης πρέπει να τοποθετήσει το Mute Button και το Text του κουμπιού στο Inspector για να λειτουργήσει σωστά.

```
using UnityEngine;
using UnityEngine.UI;

public class MusicVolumeSlider : MonoBehaviour
{
    public Slider volumeSlider;  // Δηλώνει το Slider που χρησιμοποιείται για την ένταση του ήχου

    void Start()
    {
        // Φορτώνει την αποθηκευμένη ένταση του ήχου από τα PlayerPrefs (ή 1f αν δεν υπάρχει αποθηκευμένη τιμή)
        float savedVolume = PlayerPrefs.GetFloat("MusicVolume", 1f);
        volumeSlider.value = savedVolume;  // Ρυθμίζει την αρχική τιμή του Slider στην αποθηκευμένη ένταση

        // Προσθέτει listener ώστε όταν αλλάξει η τιμή του Slider, να καλείται η μέθοδος SetVolume
        volumeSlider.onValueChanged.AddListener(SetVolume);
    }

    // Μέθοδος που καλείται όταν αλλάζει η τιμή του Slider
    void SetVolume(float value)
    {
        // Ελέγχει αν υπάρχει το MusicManager instance
        if (MusicManager.Instance != null)
        {
            // Καλεί τη μέθοδο SetVolume του MusicManager για να αλλάξει την ένταση της μουσικής
            MusicManager.Instance.SetVolume(value);
        }

        // Αποθηκεύει τη νέα τιμή της έντασης στο PlayerPrefs για να παραμείνει αποθηκευμένη για μελλοντικές συνεδρίες
        PlayerPrefs.SetFloat("MusicVolume", value);
        PlayerPrefs.Save();  // Αποθηκεύει τις αλλαγές στο PlayerPrefs
    }
}
```

Το script MusicVolumeSlider επιτρέπει στον χρήστη να ρυθμίζει την ένταση της μουσικής μέσω ενός Slider και αποθηκεύει τη ρύθμιση στα PlayerPrefs. Όταν αλλάζει η τιμή του Slider, η ένταση της μουσικής ενημερώνεται στο MusicManager. Επίσης, όπως και πριν χρειάζεται να τοποθετήσει το slider στο Inspector του GameObject που περιέχει το script. Πλέον οι δύο σκηνές (**SampleScene** και **OptionsMenuScene**) έχουν **δύο script**: ένα για την **εναλλαγή σκηνών** και ένα για τη **ρύθμιση της μουσικής.**

4.4 Εισαγωγή ερωτήσεων με Google Sheet URL μέσω UnityWebRequest

Για την εισαγωγή και την εμφάνιση των ερωτήσεων θα χρησιμοποιήσω ξεχωριστά script. Η **εισαγωγή και φόρτωση των ερωτήσεων** στη Unity από το **Google Sheets** γίνεται με τον εξής τρόπο: Ο χρήστης δημιουργεί ένα Google sheet σε μορφή πίνακα με όσες στήλες χρειάζεται. Στην περίπτωση μου ο πίνακας αποτελείται από 7 στήλες. Η μία αφορά τις ερωτήσεις, οι 5 αφορούν τις 4 απαντήσεις της κάθε ερώτησης αλλά και την σωστή απάντηση και η τελευταία, η οποία αφορά το επίπεδο δυσκολίας. Μόλις ολοκληρωθεί η συμπλήρωση του sheet και από το μενού (Αρχείο → κοινοποίηση → δημοσίευση στον ιστό) και στην ενσωμάτωση θα πρέπει να επιλέξει τιμές διαχωρισμένες με κόμμα, δηλαδή CSV (Comma-Separated Values). Πρόκειται για ροή δεδομένων όπου οι στήλες διαχωρίζονται με κόμμα. Αντίστοιχα, η άλλη επιλογή που θα αντικρίσει ο χρήστης είναι τιμές διαχωρισμένες με tab, δηλαδή TSV (Tab-Separated Values). Είναι το ίδιο πράγμα, απλά οι στήλες χωρίζονται με τον χαρακτήρα tab αντί για κόμμα. Αμέσως θα πάρει το link του sheet και θα πρέπει να το τροποποιήσει αν δεν είναι έτοιμο ήδη ώστε να έχει κατάληξη CSV. Και τα δύο κάνουν το ίδιο πράγμα: εξάγουν τις τιμές του φύλλου σε απλό κείμενο, ώστε να μπορεί ο χρήστης να τα χρησιμοποιήσει σε άλλο πρόγραμμα ή σε script.



Εικόνα 4.8: Εισαγωγή Google Sheet CSV

Όπως ανέφερα νωρίτα το **CSV** (Comma-Separated Values) είναι μια μορφή αρχείου που χρησιμοποιείται για την αποθήκευση δεδομένων σε πίνακες, όπου οι τιμές (ή τα πεδία) διαχωρίζονται από κόμματα (ή άλλους χαρακτήρες όπως το **tab** ή **ανά διαστήματα**). Κάθε γραμμή στο αρχείο CSV αντιπροσωπεύει μια **εγγραφή** (ή **γραμμή δεδομένων**), ενώ οι στήλες αντιπροσωπεύονται από τα **πεδία** που χωρίζονται με κόμματα. Έτσι τα δεδομένα είναι «καθαρά» και μπορούν να διαβαστούν εύκολα από προγράμματα όπως η Unity, Excel ή Python. Αυτό επιτρέπει στον χρήστη να τα ενσωματώνει απευθείας σε εφαρμογές, π.χ. για quiz, χωρίς να χρειάζεται να ανοίξει το ίδιο το φύλλο. Με αυτόν τον τρόπο μπορεί να αλλάζει το περιεχόμενο στο Google Sheets και να ενημερώνεται αυτόματα η εφαρμογή, χωρίς νέο build. Η χρήση του CSV στη Unity γίνεται με UnityWebRequest. Παρακάτω δίνεται το script για τη λήψη των δεδομένων.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class QuestionLoader : MonoBehaviour
{
   // Ορίζουμε την κλάση για την ερώτηση
   [System.Serializable]
   public class Question
   {
      public string questionText;  // Κείμενο της ερώτησης
      public string[] options;  // Πίνακας με τις απαντήσεις
      public string correctAnswer;  // Σωστή απάντηση
      public int difficulty;  // Επίπεδο δυσκολίας (1 = εύκολο, 2 = μέτριο, 3 = δύσκολο)


      public override bool Equals(object obj)
      {
         if (obj is Question other)
         {
            return questionText == other.questionText;  // Σύγκριση μόνο της ερώτησης
         }
         return false;
      }


      public override int GetHashCode()
```

```
    {
        return questionText.GetHashCode();  // Χρησιμοποιούμε το κείμενο της ερώτησης για το hash code
    }
}

// URL του Google Sheet CSV για τις ερωτήσεις
public string googleSheetCSVUrl;

// Λίστες για τις ερωτήσεις ανά επίπεδο δυσκολίας
public List<Question> easyQuestions = new List<Question>();
public List<Question> mediumQuestions = new List<Question>();
public List<Question> hardQuestions = new List<Question>();

// Ορίζουμε delegate και event για ενημέρωση του συστήματος όταν φορτωθούν οι ερωτήσεις
public delegate void QuestionsLoaded();  // Delegate για το event
public static event QuestionsLoaded OnQuestionsLoaded;  // Event που καλείται όταν οι ερωτήσεις φορτωθούν

// Ξεκινάμε τη διαδικασία λήψης των ερωτήσεων κατά την εκκίνηση του παιχνιδιού
void Start()
{
    StartCoroutine(DownloadQuestions());  // Ξεκινάμε το Coroutine για τη λήψη των ερωτήσεων
}

// Μέθοδος για την λήψη των ερωτήσεων από το Google Sheet CSV μέσω UnityWebRequest
IEnumerator DownloadQuestions()
{
    UnityWebRequest www = UnityWebRequest.Get(googleSheetCSVUrl);  // Κάνουμε το αίτημα για το CSV
    yield return www.SendWebRequest();  // Περιμένουμε την ολοκλήρωση του αιτήματος

    if (www.result != UnityWebRequest.Result.Success)  // Αν το αίτημα αποτύχει
    {
        Debug.LogError("Download failed: " + www.error);  // Εκτυπώνουμε το λάθος στην κονσόλα
    }
    else
    {
        ParseCSV(www.downloadHandler.text);  // Επεξεργαζόμαστε τα δεδομένα από το CSV
        OnQuestionsLoaded?.Invoke();  // Ειδοποιούμε τα υπόλοιπα συστήματα ότι οι ερωτήσεις είναι έτοιμες
    }
}

// Μέθοδος για την ανάλυση του CSV και την αποθήκευση των ερωτήσεων στις σωστές λίστες
void ParseCSV(string data)
{
```

```
    string[] lines = data.Split('\n');  // Διαχωρίζουμε τις γραμμές του CSV

    // Διατρέχουμε όλες τις γραμμές του CSV εκτός από την πρώτη (που είναι τα headers)
    for (int i = 1; i < lines.Length; i++)
    {
        string line = lines[i].Trim();  // Κόβουμε το περιττό κενό
        if (string.IsNullOrEmpty(line)) continue;  // Αν η γραμμή είναι άδεια, την παραλείπουμε

        string[] fields = line.Split(',');  // Διαχωρίζουμε τις στήλες του CSV

        if (fields.Length < 7) continue;  // Αν δεν υπάρχουν αρκετά πεδία, παραλείπουμε τη γραμμή

        // Δημιουργούμε την ερώτηση από τα δεδομένα του CSV
        Question q = new Question
        {
            questionText = fields[0],  // Η ερώτηση
            options = new string[] { fields[1], fields[2], fields[3], fields[4] },  // Οι επιλογές
            correctAnswer = fields[5],  // Η σωστή απάντηση
            difficulty = int.Parse(fields[6])  // Το επίπεδο δυσκολίας
        };

        // Προσθέτουμε την ερώτηση στην αντίστοιχη λίστα με βάση το επίπεδο δυσκολίας
        switch (q.difficulty)
        {
            case 1: easyQuestions.Add(q); break;
            case 2: mediumQuestions.Add(q); break;
            case 3: hardQuestions.Add(q); break;
        }
    }
}

// Μέθοδος για την επιστροφή μιας τυχαίας ερώτησης για το συγκεκριμένο επίπεδο δυσκολίας
public Question GetRandomQuestion(int difficulty)
{
    List<Question> pool = difficulty switch
    {
        1 => easyQuestions,
        2 => mediumQuestions,
        3 => hardQuestions,
        _ => easyQuestions
    };

    if (pool.Count == 0)  // Αν δεν υπάρχουν διαθέσιμες ερωτήσεις για το επίπεδο
```

```
        {
            Debug.LogWarning("Δεν υπάρχουν διαθέσιμες ερωτήσεις για επίπεδο: " + difficulty);
            return null;
        }

        int index = Random.Range(0, pool.Count);  // Επιλέγουμε τυχαία μια ερώτηση
        Question selected = pool[index];  // Επιστρέφουμε την επιλεγμένη ερώτηση
        pool.RemoveAt(index);  // Αν θέλουμε, την αφαιρούμε από τη λίστα (αν δεν θέλουμε επανάληψη)
        return selected;
    }

    // Μέθοδος με χρήση της λίστας used (για multiplayer), ώστε να αποφεύγουμε τις επαναχρησιμοποιημένες ερωτήσεις
    public Question GetRandomQuestion(int difficulty, List<Question> used)
    {
        List<Question> pool = difficulty switch
        {
            1 => easyQuestions,
            2 => mediumQuestions,
            3 => hardQuestions,
            _ => easyQuestions
        };

        // Φιλτράρουμε τις ήδη χρησιμοποιημένες ερωτήσεις
        List<Question> available = pool.FindAll(q => !used.Contains(q));

        if (available.Count == 0)  // Αν δεν υπάρχουν διαθέσιμες ερωτήσεις για το επίπεδο
        {
            Debug.LogWarning("Δεν υπάρχουν διαθέσιμες ερωτήσεις για επίπεδο: " + difficulty);
            return null;
        }

        int index = Random.Range(0, available.Count);  // Επιλέγουμε τυχαία μια ερώτηση από τις διαθέσιμες
        return available[index];  // Επιστρέφουμε την επιλεγμένη ερώτηση
    }
}
```

Το script χρησιμοποιεί το UnityWebRequest για να κατεβάσει το CSV από το Google Sheet, αναλύει τα δεδομένα και δημιουργεί αντικείμενα Question με τις ερωτήσεις, τις απαντήσεις, τη σωστή απάντηση και το επίπεδο δυσκολίας. Στη συνέχεια, οι ερωτήσεις οργανώνονται σε τρεις λίστες ανάλογα με το επίπεδο δυσκολίας και το σύστημα ενημερώνεται για την ολοκλήρωση της διαδικασίας φόρτωσης.

Η εμφάνιση τώρα των ερωτήσεων γίνεται με τη δημιουργία τριών script, ένα για τη SinglePlayer (QuestionDisplay) επιλογή, ένα για τη Multiplayer (MultiQuestionDisplay) και ένα για την Online Multiplayer σκηνή (NetworkMultiQuestionDisplay). Αυτά τα script είναι υπεύθυνα για την **παρουσίαση** των ερωτήσεων στην **UI** και για την αλληλεπίδραση με τον χρήστη. Το ένα θα φορτωθεί στην SingleMenuScene, το δεύτερο στη MultiMenuScene και το τρίτο στη MultiMenuScenee. Αντίστοιχα το script QuestionLoader θα πρέπει να φορτωθεί και στις τρεις σκηνές. Επίσης, να γίνει η αντιστοίχιση του Gameobject που περιέχει το script στο Inspector Window, προκειμένου να καλούνται οι μέθοδοι όπου χρειάζονται. Κομμάτι του κώδικα για την εμφάνιση των ερωτήσεων σε τοπικό (Local) επίπεδο, τόσο σε Single όσο και σε MultiPlayer βρίσκεται στο παράρτημα Α.

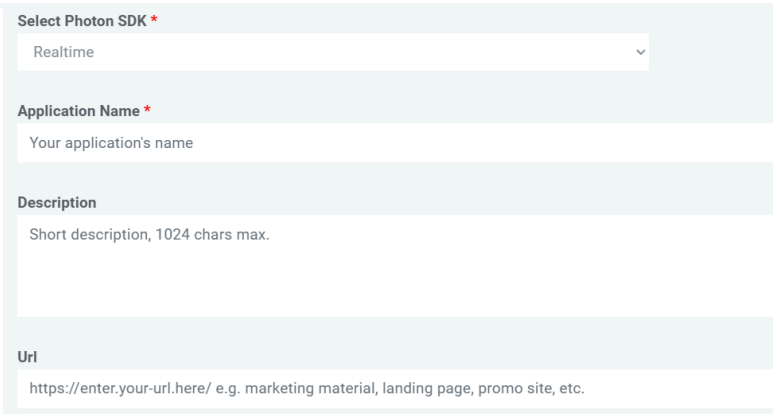
4.5 Online MatchMaking MultiPlayer με Photo Pun 2

Όσον αφορά το online matchmaking, δηλαδή τη σύνδεση δύο παικτών απομακρυσμένα μέσω διαδικτύου, αυτή πραγματοποιήθηκε με τη χρήση του **Photon PUN 2**, το οποίο ο χρήστης μπορεί να βρει δωρεάν από το Asset Store της Unity. Το Photon PUN 2 είναι ένα έτοιμο σύστημα δικτύου (networking framework) για Unity που σε συνδέει στους servers της Photon, ώστε να φτιάχνεις online παιχνίδια χωρίς δική σου υποδομή. Η λογική του βασίζεται σε **lobbies** και **rooms**: μπαίνεις πρώτα σε έναν χώρο αναμονής και μετά σε ένα δωμάτιο-παρτίδα όπου παίζεις με τους άλλους. Αντικείμενα και κατάσταση μοιράζονται μέσω **PhotonView** και **RPCs** (π.χ. ενημέρωση σκορ ή αποστολή ερώτησης σε όλους) με ελάχιστο κώδικα. Μέσα στο room υπάρχει πάντα ένας **MasterClient** που λειτουργεί σαν προσωρινός host και μπορεί να φορτώνει σκηνές για όλους· χάρη στο **Automatic Scene Sync** όταν ο Master αλλάξει σκηνή, συγχρονίζονται αυτόματα και οι υπόλοιποι. Επιπλέον, με Custom Properties (π.χ. mode=1v1, region=eu, difficulty=easy) κολλάς “ετικέτες” σε παίκτες/δωμάτια για εύκολο φιλτράρισμα στο matchmaking, ενώ τα **Reconnect & Player TTL** βοηθούν να επιστρέψεις στο ίδιο δωμάτιο αν χαθεί στιγμιαία η σύνδεση. Στη δική μου περίπτωση το online matchmaking με **Photon** PUN 2 λειτουργεί με τρία ξεκάθαρα σενάρια που συνδυάζονται αρμονικά. Όταν ο παίκτης πατήσει **Find Match**, ο client προσπαθεί πρώτα να μπει σε υπάρχον δωμάτιο μέσω **JoinRandomRoom** με φίλτρα (π.χ. mode="1v1"), ώστε να ταιριάξει άμεσα με κάποιον διαθέσιμο αντίπαλο. Αν δεν βρεθεί κατάλληλο δωμάτιο, δημιουργείται αυτόματα ένα **δημόσιο** room με τα ίδια properties για να το εντοπίσει ο επόμενος παίκτης που θα πατήσει Find Match. Παράλληλα υπάρχει το **Create Room με**

κωδικό, όπου ο host φτιάχνει ένα ιδιωτικό δωμάτιο και ως όνομα room χρησιμοποιεί τον κωδικό που βλέπει/ορίζει στην οθόνη. Έπειτα, τον κωδικό αυτόν τον μοιράζεται με τον φίλο του ώστε να παίξουν μεταξύ τους χωρίς να μπορεί να τους βρει τρίτος. Ο φίλος επιλέγει **Join Room**, πληκτρολογεί τον ίδιο ακριβώς κωδικό και συνδέεται απευθείας στο ίδιο δωμάτιο μέσω **JoinRoom(code)**, αρκεί το δωμάτιο να είναι ανοιχτό. Σε όλα τα σενάρια, μόλις ο αριθμός παικτών φτάσει το **MaxPlayers** (συνήθως 2), ο **MasterClient** ξεκινά τον αγώνα φορτώνοντας τη σκηνή του παιχνιδιού και χάρη στο **Automatic Scene Sync** όπως αναφέρθηκε προηγουμένως μεταφέρονται αυτόματα όλοι στην ίδια σκηνή. Πάμε να δούμε τώρα τα βήματα που χρειάζεται να πράξει ο χρήστης ώστε να φτάσει σε αυτό το επίπεδο.

Στο Photon PUN 2, το PhotonView είναι το component που συνδέει ένα αντικείμενο με το δίκτυο και επιτρέπει τον συγχρονισμό του ανάμεσα σε όλους τους παίκτες, ώστε να γνωρίζει ποιος το ελέγχει και ποια είναι η κατάστασή του. Ο χρήστης προσθέτει το PhotonView στο GameObject που θέλει να συγχρονιστεί, και έπειτα δηλώνει στο script ποιες μεταβλητές ή συναρτήσεις θα είναι διαθέσιμες στο δίκτυο. Τα RPCs (Remote Procedure Calls) είναι συναρτήσεις που σημειώνονται με το attribute [PunRPC] και καλούνται μέσω του PhotonView, εκτελώνταςται ταυτόχρονα σε όλους τους clients ή σε συγκεκριμένους στόχους. Έτσι, ενέργειες όπως σκορ, κινήσεις ή επιλογές γίνονται ορατές με συνέπεια σε όλους τους παίκτες.

Ως πρώτο βήμα ο χρήστης οφείλει να εγκαταστήσει το Photo Pun 2 και να το εισάγει (import) στο project του. Από το μενού της διεπαφής επιλέγει Window → Package Manager → My Assets → Photon PUN 2 → Download (αν χρειάζεται) και στη συνέχεια Import. Δεύτερο βήμα να δημιουργήσει έναν λογαριασμό στο Photon Dashboard. Από το μενού του Photon Dashboard δημιουργεί το δικό του project (create a new app), δίνει το όνομα που επιθυμεί στην εφαρμογή του, επιλέγει το κατάλληλο Photon SDK (Pun) και πατάει create. Αμέσως μετά οφείλει να αντιγράψει το AppID και να το τοποθετήσει στο Pun Wizard. Το Pun Wizard το βρίσκει από Window → Photon Unity NetWorking → Pun Wizard. Αφού το τοποθετήσει στο αντίστοιχο πεδίο είναι έτοιμος να πατήσει Setup Project. Μετά εφόσον δεν το έχει κάνει ήδη, πρέπει να δημιουργήσει την αντίστοιχη σκηνή με τα κατάλληλα κουμπιά, text και Input Fields και να προσθέσει τη σκηνή στο Build Profile.



Εικόνα 4.9: Δημιουργία project στο Photon Dashboard

Εικόνα 4.10: AppID του project στο Photon Dashboard

Στη σκηνή τώρα που θα δημιουργήσει η οποία θα είναι υπεύθυνη και για την online σύνδεση μεταξύ των δύο παικτών, πρέπει να αναπτύξει το αντίστοιχο script το οποίο θα το τοποθετήσει σε ένα κενό Gameobject στο παράθυρο ιεραρχίας (Hierarchy Window), με την γνωστή διαδικασία που έχουμε αναφέρει προηγουμένως. Το κύριο κομμάτι κώδικα που αφορά το matchmaking βρίσκεται παρακάτω:

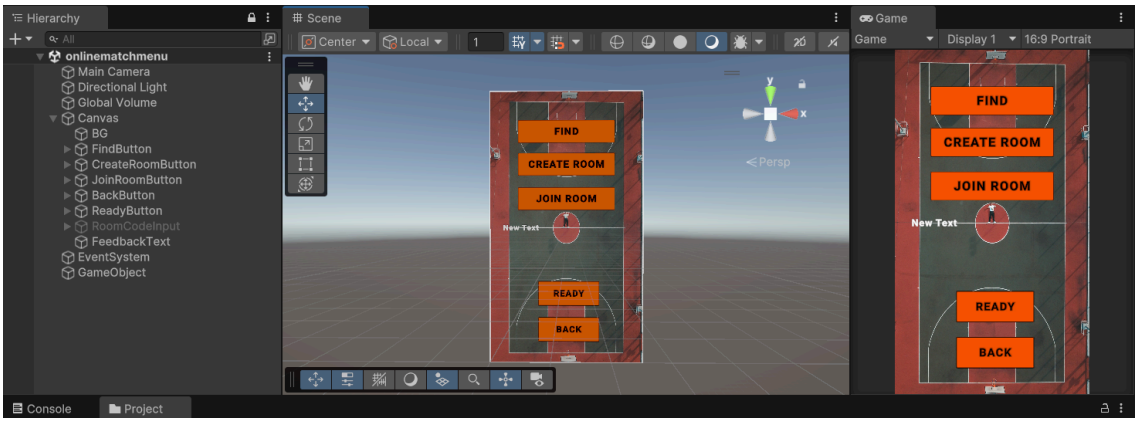
```
// Πατιέται το κουμπί "Find Match" → προσπάθησε να μπεις σε τυχαίο δωμάτιο
void OnFindButton()
{
    // Αν δεν έχει ολοκληρωθεί η σύνδεση στο Photon, σταμάτα
    if (!PhotonNetwork.IsConnectedAndReady) { FeedbackText.text = "Not connected yet..."; return; }

    FeedbackText.text = "Searching for opponent...";
    // Καλεί το Photon να σε βάλει σε ένα τυχαίο δωμάτιο (με οποιεσδήποτε default συνθήκες)
    PhotonNetwork.JoinRandomRoom();
}

// Αν το JoinRandomRoom αποτύχει (δεν υπάρχει διαθέσιμο δωμάτιο)
public override void OnJoinRandomFailed(short returnCode, string message)
{
```

```
    // Φτιάξε έναν απλό 4ψήφιο κωδικό/όνομα για το νέο δωμάτιο
    roomCode = Random.Range(1000, 9999).ToString();

    // Ρυθμίσεις δωματίου για QUICK MATCH: δημόσιο (IsVisible=true) ώστε να σε βρει ο επόμενος
    var options = new RoomOptions
    {
        MaxPlayers = MaxPlayers,        // 2 παίκτες
        PublishUserId = true,           // εκθέτει το UserId στους παίκτες
        CleanupCacheOnLeave = true,     // καθαρίζει τα cached events όταν κάποιος φύγει
        IsVisible = true,               // δημόσιο για matchmaking
        IsOpen = true                   // ανοιχτό για join
    };

    FeedbackText.text = $"No rooms. Creating {roomCode}...";
    // Δημιούργησε το δωμάτιο—ο επόμενος που θα κάνει JoinRandom θα σε βρει
    PhotonNetwork.CreateRoom(roomCode, options, TypedLobby.Default);
}

// Πατιέται το "Create Room" (ιδιωτικό παιχνίδι με φίλο)
void OnCreateRoomButton()
{
    if (!PhotonNetwork.IsConnectedAndReady) { FeedbackText.text = "Not connected yet..."; return; }

    // Γεννάμε 4ψήφιο κωδικό που θα μοιραστεί ο host στον φίλο του
    roomCode = Random.Range(1000, 9999).ToString();

    // Ιδιωτικό δωμάτιο: IsVisible=false για να μην εμφανίζεται στο lobby/JoinRandom
    var options = new RoomOptions
    {
        MaxPlayers = MaxPlayers,
        PublishUserId = true,
        CleanupCacheOnLeave = true,
        IsVisible = false,  // ΙΔΙΩΤΙΚΟ (μόνο με JoinRoom(code))
        IsOpen = true
    };

    FeedbackText.text = $"Creating room: {roomCode}";
    // Το όνομα του room είναι ο ίδιος ο κωδικός που θα δώσεις στον φίλο σου
    PhotonNetwork.CreateRoom(roomCode, options, TypedLobby.Default);
}

// Πατιέται το "Join" αφού ο χρήστης γράψει τον κωδικό που του έδωσε ο host
void OnReadyButton()
```

```
{
    if (!PhotonNetwork.IsConnectedAndReady) { FeedbackText.text = "Not connected yet..."; return; }

    var code = RoomCodeInput.text.Trim();
    if (string.IsNullOrEmpty(code)) { FeedbackText.text = "Enter a room code"; return; }

    FeedbackText.text = $"Joining room {code}...";
    // Άμεση είσοδος στο δωμάτιο με το συγκεκριμένο όνομα/κωδικό
    PhotonNetwork.JoinRoom(code);

    // Κρύψε το input και το Join button μετά την προσπάθεια
    RoomCodeInput.gameObject.SetActive(false);
    ReadyButton.gameObject.SetActive(false);
}

// Καλείται όταν ΜΠΗΚΕΣ εσύ στο δωμάτιο
public override void OnJoinedRoom()
{
                                FeedbackText.text = $"Joined: {PhotonNetwork.CurrentRoom.Name}
({PhotonNetwork.CurrentRoom.PlayerCount}/{PhotonNetwork.CurrentRoom.MaxPlayers})";
    // Έλεγξε αν φτάσαμε τους MaxPlayers για να ξεκινήσει το παιχνίδι
    TryStartWhenFull();
}

// Καλείται όταν ΜΠΗΚΕ ΚΑΠΟΙΟΣ ΑΛΛΟΣ στο δωμάτιο (στον δικό σου client)
public override void OnPlayerEnteredRoom(Player newPlayer)
{
                                        FeedbackText.text = $"Player joined
({PhotonNetwork.CurrentRoom.PlayerCount}/{PhotonNetwork.CurrentRoom.MaxPlayers})";
    // Ίδιος έλεγχος: μήπως τώρα ολοκληρώθηκε το ζευγάρι
    TryStartWhenFull();
}
```

Αυτό το script υλοποιεί το matchmaking στο Photon PUN 2, δίνοντας δύο τρόπους σύνδεσης παικτών: το γρήγορο Quick Match και το ιδιωτικό δωμάτιο με κωδικό. Όταν ο παίκτης πατήσει Find Match, το σύστημα προσπαθεί να τον βάλει σε ένα τυχαίο διαθέσιμο δωμάτιο· αν δεν βρει, δημιουργεί αυτόματα νέο δημόσιο δωμάτιο με τυχαίο τετραψήφιο κωδικό, ώστε ο επόμενος που ψάχνει να συνδεθεί εκεί. Με το Create Room ο παίκτης δημιουργεί ιδιωτικό δωμάτιο, το οποίο δεν

εμφανίζεται στο lobby και μπορεί να το μοιραστεί σε φίλο του μέσω του κωδικού. Ο άλλος παίκτης μπαίνει στο δωμάτιο με το Join, πληκτρολογώντας τον κωδικό που έλαβε. Κάθε φορά που ο παίκτης μπαίνει ή έρχεται νέος στο δωμάτιο, εμφανίζεται μήνυμα με τον αριθμό παικτών και αν το δωμάτιο φτάσει το μέγιστο επιτρεπτό (2), καλείται η συνάρτηση για να ξεκινήσει αυτόματα το παιχνίδι. Έτσι το script αναλαμβάνει όλη τη διαδικασία σύνδεσης, δημιουργίας και έναρξης του αγώνα ανάμεσα σε δύο παίκτες.



Εικόνα 4.11: Προβολή της OnlinaMatchMenu σκηνής

Κεφάλαιο 5 – Συμπεράσματα και προτάσεις βελτίωσης

5.1 Συμπεράσματα υλοποίησης εφαρμογής

Η πτυχιακή εργασία που παρουσιάστηκε είχε ως στόχο τον σχεδιασμό και την ανάπτυξη ενός διαδραστικού quiz game στην Unity, το οποίο προσομοιώνει έναν αγώνα μπάσκετ, επιτυγχάνοντας την πλήρη εφαρμογή τόσο σε singleplayer όσο και σε multiplayer έκδοση, με έμφαση στη σταθερότητα και τη δυνατότητα επέκτασης. Συγκεκριμένα, υλοποιήθηκε ροή σκηνών με αποθήκευση ονόματος παίκτη, σύστημα ερωτήσεων/απαντήσεων από εξωτερική πηγή (Google Sheets), δυναμική δυσκολία (1–3 βαθμοί/βολή-δίποντο-τρίποντο), ενσωμάτωση βοηθειών (power-ups), καθώς και η δυνατότητα δημιουργίας προφίλ παίκτη με παρουσίαση επιδόσεων (highscore, νίκες/ήττες). Στο multiplayer σχεδιάστηκαν 4 γύροι με ισοκατανομή ερωτήσεων, μη επανάληψη θεμάτων, τελικό πίνακα σκορ/νικητή και προηγμένο σύστημα βοηθειών (Block, Steal, Rebound, Assist, Foul) με ακριβή κανόνες, χρονικά παράθυρα και μία χρήση ανά παίκτη. Στο online κομμάτι ενσωματώθηκε Photon PUN 2, προσφέροντας γρήγορο matchmaking (JoinRandom→Create), ιδιωτικά δωμάτια με κωδικό, αυτόματο συγχρονισμό σκηνών (MasterClient/Automatic Scene Sync) καθώς και μηχανισμό ασφαλείας ώστε το παιχνίδι να ξεκινά σωστά μόνο μία φορά.· Παράλληλα προστέθηκαν ρυθμίσεις ήχου/μουσικής που συνεχίζουν ομαλά κατά τη διάρκεια εναλλαγής σκηνών. Συνολικά, το έργο δείχνει ότι ένα σύγχρονο multiplayer σύστημα μπορεί να συνδυάζει αποτελεσματικούς μηχανισμούς με εύχρηστο περιβάλλον, προσφέροντας σταθερή και ευχάριστη εμπειρία. Επιπλέον, παραμένουν ανοιχτές δυνατότητες εξέλιξης, όπως η προσθήκη νέων power-ups, η βελτίωση του matchmaking και η υποστήριξη σύγχρονων συσκευών (Meta Quest 3).

Σύμφωνα με τους βασικούς στόχους που είχαν τεθεί εξαρχής, ανέπτυξα ένα ολοκληρωμένο και επεκτάσιμο quiz game σε Unity με καθαρή ροή singleplayer/multiplayer, προσεγμένο UI (user interface) και αξιόπιστη δικτύωση μέσω Photon PUN 2. Η εισαγωγή ερωτήσεων οργανώθηκε από εξωτερική πηγή (π.χ. Google Sheets), ώστε οποιοσδήποτε—χωρίς ειδικές γνώσεις προγραμματισμού—να μπορεί να ενημερώνει εύκολα το περιεχόμενο εκτός Unity. Το matchmaking σχεδιάστηκε ώστε να είναι εύκολο στη χρήση, με quick match, ιδιωτικά δωμάτια με κωδικό και αυτόματο συγχρονισμό σκηνών. Με αυτόν τον τρόπο, χρειάζεται λιγότερος κώδικας και απλούστερη διαδικασία, κάτι που διευκολύνει εκπαιδευτικούς, δημιουργούς περιεχομένου και νέους developers. Το σύστημα βαθμολόγησης, οι γύροι/κανόνες, τα power-ups (Block, Steal, Rebound, Assist, Foul) και το προφίλ παίκτη (highscore, νίκες/ήττες) συνθέτουν μια ολοκληρωμένη εμπειρία που είναι ταυτόχρονα διασκεδαστική και παιδαγωγική. Έφτιαξα το παιχνίδι με modular αρχιτεκτονική, ώστε η λογική, τα δεδομένα και η παρουσίαση να είναι ξεχωριστά και εύκολα επεκτάσιμα. Έτσι, μπορεί να προστεθεί νέο περιεχόμενο ή νέες λειτουργίες χωρίς να χρειάζονται μεγάλες αλλαγές. Συνολικά,

έδειξα ότι ένα online quiz μπορεί να παραμετροποιηθεί εύκολα και να στηθεί με μικρή πολυπλοκότητα και χαμηλό κόστος.

5.2 - Δυνατότητες εξέλιξης

Το σύστημα που έφτιαξα μπορεί να γίνει ακόμη πιο απλό και "έξυπνο" για όλους. Στο matchmaking, αντί για απλό τυχαίο ταίριασμα, υπάρχει η δυνατότητα για ταίριασμα παικτών με βάση ένα εύκολο επίπεδο ικανότητας (π.χ. αρχάριος/μέτριος/προχωρημένος ή ELO), την περιοχή/καθυστέρηση δικτύου και γλώσσα.· Σκέφτομαι να προσαρμόσω ορατό timer αναμονής, auto-rematch στο τέλος και ένα σταθερό reconnect που επιστρέφει τον παίκτη στο ίδιο δωμάτιο αν κοπεί στιγμιαία η σύνδεση. Στα private rooms ο κωδικός μπορεί να λήγει μετά από λίγο ή αν γίνουν πολλές λάθος προσπάθειες, με απλούς ελέγχους για αποφυγή κατάχρησης και εύκολο κουμπί αντιγραφής. Για τις ερωτήσεις θα υπάρχει ένας απλός κανόνας (κείμενο, 4 επιλογές, 1 σωστή), δυνατότητα για backup και προεπισκόπηση πριν μπουν στο παιχνίδι. Θα υπάρχει και cache για να παίζει το παιχνίδι χωρίς ίντερνετ, καθώς και στατιστικά για σωστές/λάθος απαντήσεις ώστε να ρυθμίζεται αυτόματα η δυσκολία και να αποφεύγονται επαναλήψεις. Έτσι το παιχνίδι γίνεται πιο δίκαιο, πιο γρήγορο και πιο φιλικό για όποιον θέλει να παίξει ή να προσθέσει περιεχόμενο χωρίς να ξέρει προγραμματισμό. Αν το δωρεάν όριο παικτών δεν φτάνει, μπορώ να το αναβαθμίσω σε μεγαλύτερο πλάνο ώστε να υποστηρίζονται περισσότεροι χρήστες ταυτόχρονα. Σε επιχειρησιακό επίπεδο, αν το δωρεάν όριο δεν επαρκεί, υπάρχει η επιλογή αναβάθμισης σε πλάνο ~100 CCU (ενδεικτικά γύρω στα $95/έτος) ώστε να υποστηριχθούν περισσότεροι ταυτόχρονοι χρήστες.

Τέλος, επέλεξα τη Unity γιατί είναι ο πιο πρακτικός δρόμος: έχει πλούσιο Asset Store, γρήγορη ανάπτυξη σε C#, συνεργάζεται άριστα με το Photon PUN και παρέχει έτοιμα εργαλεία για το γραφικό περιβάλλον (UI), τη διαχείριση πόρων και την ανάπτυξη σε πολλές πλατφόρμες, όπως PC, Android και Meta Quest. Αυτό σημαίνει λιγότερο χρόνο για στήσιμο και συντήρηση σε σχέση με άλλες λύσεις, κάτι που ταιριάζει ιδανικά σε ένα quiz project με δικτύωση, όπου θέλουμε εύκολη εισαγωγή περιεχομένου και "plug-and-play" online.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://el.wikipedia.org/wiki/%CE%92%CE%B9%CE%BD%CF%84%CE%B5%CE%BF%CF%80%CE%B1%CE%B9%CF%87%CE%BD%CE%AF%CE%B4%CE%B9
2. https://www.tovima.gr/2016/02/15/science/o-neos-kosmos-twn-aptikwn-thaymatwn/
3. https://museumfinder.gr/magnavox-odyssey-1972-i-proti-oikiaki-konsola-vinteopaichnidion-ston-kosmo-ena-psifiako-mnimeio-sto-video-games-museum/
4. https://www.fastcompany.com/?_ptid=%7Bkpdx%7DAAAAvL_eNqnzcQoKVW5HNmFLY1JwdRIQbWU4eHpseWF1bG9uZXpmMRoMRVhHNzJUVFJFVlVRIiUxODA1azI4MGM0LTAwMDAzNjR2cDJoMWMwb2Y2NWtvZHV1MjBnKhdzaG93T2ZmZXJETjBHTjkwWlYwTUg0NzABOgxPVFpZMEFQMUxUSkhSEnYtcgDwJmZjd2kzMTV1YVoMOTQuMTMxLjQxLjU1YgNkd2NottLzxAZwBngMggEMT0ZGNUlSMFNPMDFC
5. https://el.wikipedia.org/wiki/Nintendo#cite_note-4
6. https://www.retrocomputers.gr/component/k2/content/history-consoles-pongs
7. https://dev.epicgames.com/documentation/en-us/unreal-engine/understanding-the-basics-of-unreal-engine
8. https://en.wikipedia.org/wiki/Unreal_Engine
9. https://www.meegle.com/en_us/topics/game-engine/game-engine-for-gamemaker-studio
10. https://medium.com/@foobar404/the-godot-engine-a-new-era-in-open-source-gaming-047d4b4c784f
11. https://en.wikipedia.org/wiki/GDevelop
12. https://unity.com/
13. https://medium.com/@mobileappandgameappdevelopment/advantages-and-disadvantages-of-unity-3d-game-development-4314c067d4b2
14. https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/
15. https://www.macworld.com/article/187693/unity-18.html
16. https://www.dice.com/career-advice/how-unity3d-become-a-game-development-beast
17. https://www.polygon.com/2012/11/14/3645122/unity-4-0-available-download/
18. https://www.adweek.com/performance-marketing/unity-sdk-out-of-beta/
19. https://web.archive.org/web/20190320210245/https:/www.gamasutra.com/view/news/307050/Unity_20172_brings_Autodesk_integration_into_the_fold.php
20. https://docs.unity3d.com/2020.1/Documentation/Manual/UIE-Events-DragAndDrop.html
21. https://www.gamesindustry.biz/unity-2018-detailed-in-gdc-keynote

22. https://www.auganix.org/unity-mars-augmented-and-mixed-reality-authoring-studio-now-available/
23. https://www.digitaltrends.com/cars/unity-automotive-virtual-reality-and-hmi/
24. https://www.businessinsider.com/unity-ceo-john-riccitiello-opportunity-beyond-gaming-2018-9?r=UK&IR=T
25. https://www.theverge.com/2017/10/4/16409734/unity-neill-blomkamp-oats-studios-mirror-cinemachine-short-film
26. https://en.wikipedia.org/wiki/Object-oriented_programming
27. https://en.wikipedia.org/wiki/Class_(computer_programming)
28. https://en.wikipedia.org/wiki/Component-based_software_engineering

*Στο κεφάλαιο 2 και συγκεκριμένα στις ενότητες 2.1, 2.2 & 2.3 έχουν χρησιμοποιηθεί πληροφορίες από το Wikipedia.

ΠΑΡΑΡΤΗΜΑ Α

1. Εμφάνιση ερωτήσεων σε Local SinglePlayerMode

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System.Collections;
using UnityEngine.SceneManagement;

public class QuestionDisplay : MonoBehaviour
{
    public QuestionLoader questionLoader;  // Αναφορά στο QuestionLoader για την ανάκτηση των ερωτήσεων
    public TMP_Text questionText;  // Εμφάνιση της ερώτησης
    public TMP_Text questionNumberText;  // Εμφάνιση του αριθμού της ερώτησης
    public TMP_Text scoreText;  // Εμφάνιση του σκορ του παίκτη
    public TMP_Text highScoreText;  // Εμφάνιση του υψηλότερου σκορ
    public TMP_Text playerNameText;  // Εμφάνιση του ονόματος του παίκτη
    public Button[] answerButtons;  // Τα κουμπιά για τις απαντήσεις

    public GameObject difficultyPanel;  // Πάνελ για την επιλογή του επιπέδου δυσκολίας
    public GameObject nextRoundPanel;  // Πάνελ για την εκκίνηση του επόμενου γύρου
    public TMP_Text nextRoundText;  // Κείμενο για την περιγραφή του επόμενου γύρου

    public Button AssistButton, PFButton, ReboundButton;  // Κουμπιά για τις βοήθειες (Assist, Foul, Rebound)

    private QuestionLoader.Question currentQuestion;  // Η τρέχουσα ερώτηση

    private int playerScore = 0;  // Σκορ του παίκτη
    private int highScore = 0;  // Υψηλότερο σκορ
    private int currentRound = 1;  // Τρέχων γύρος
    private int totalRounds = 4;  // Συνολικοί γύροι
    private int questionsPerRound = 8;  // Ερωτήσεις ανά γύρο
    private int questionsAnsweredInRound = 0;  // Ερωτήσεις που απαντήθηκαν στον τρέχοντα γύρο
    private int currentQuestionDifficulty = 1;  // Δυσκολία της τρέχουσας ερώτησης

    private string playerName;  // Όνομα του παίκτη

    private bool gameEnded = false;  // Αν το παιχνίδι έχει τελειώσει
    private bool reboundAvailable = false;  // Αν είναι διαθέσιμη η βοήθεια "Rebound"
    private bool reboundUsed = false;  // Αν η βοήθεια "Rebound" έχει χρησιμοποιηθεί
    private bool assistUsed = false;  // Αν η βοήθεια "Assist" έχει χρησιμοποιηθεί
    private bool foulUsed = false;  // Αν η βοήθεια "Foul" έχει χρησιμοποιηθεί

    private int foulQueue = 0;  // Ο αριθμός των ερωτήσεων που πρέπει να απαντηθούν αν χρησιμοποιηθεί το "Foul"
    private bool inFoulMode = false;  // Αν ο παίκτης βρίσκεται σε κατάσταση "Foul Mode"

    void Start()
    {
        nextRoundPanel.SetActive(false);  // Κρύβουμε το πάνελ για τον επόμενο γύρο στην αρχή
        ShowDifficultyPanel();  // Εμφανίζουμε το πάνελ για την επιλογή δυσκολίας

        playerName = PlayerPrefs.GetString("PlayerName", "Παίκτης");  // Φορτώνουμε το όνομα του παίκτη από τα
PlayerPrefs
        playerNameText.text = playerName;  // Εμφανίζουμε το όνομα του παίκτη στην UI

        LoadHighScore();  // Φορτώνουμε το υψηλότερο σκορ
        UpdateScoreText();  // Ενημερώνουμε το σκορ στην UI

        // Προσθήκη ακροατών στα κουμπιά για τις βοήθειες
        AssistButton.onClick.AddListener(UseAssist);
        PFButton.onClick.AddListener(UseFoul);
        ReboundButton.onClick.AddListener(UseRebound);
    }

    // Εμφανίζει το πάνελ για την επιλογή δυσκολίας
```

```
    void ShowDifficultyPanel()
    {
        difficultyPanel.SetActive(true);  // Ενεργοποιούμε το πάνελ για την επιλογή δυσκολίας
        questionText.text = $"Γύρος {currentRound}: Επέλεξε επίπεδο δυσκολίας";  // Εμφάνιση κειμένου επιλογής δυσκολίας
        questionNumberText.text = "";  // Καθαρίζουμε τον αριθμό της ερώτησης
    }

    // Μέθοδος για την επιλογή του επιπέδου δυσκολίας από τον παίκτη
    public void SelectDifficulty(int level)
    {
        currentQuestionDifficulty = level;  // Αποθηκεύουμε το επίπεδο δυσκολίας
        difficultyPanel.SetActive(false);  // Κλείνουμε το πάνελ δυσκολίας
        LoadQuestion(level);  // Φορτώνουμε την ερώτηση για το επιλεγμένο επίπεδο δυσκολίας
    }

    // Φορτώνει την ερώτηση με το δεδομένο επίπεδο δυσκολίας
    void LoadQuestion(int level)
    {
        currentQuestion = questionLoader.GetRandomQuestion(level);  // Παίρνουμε τυχαία ερώτηση από τον QuestionLoader

        if (currentQuestion == null)
        {
            Debug.Log($"Δεν υπάρχουν διαθέσιμες ερωτήσεις για level {level}.");  // Αν δεν υπάρχουν ερωτήσεις για το επίπεδο
            return;
        }

        questionText.text = currentQuestion.questionText;  // Εμφάνιση της ερώτησης
        int overallQuestionNumber = ((currentRound - 1) * questionsPerRound) + questionsAnsweredInRound + 1;
        questionNumberText.text = $"Ερώτηση {overallQuestionNumber}";  // Εμφάνιση του αριθμού της ερώτησης

        // Αντιστοίχιση των απαντήσεων στα κουμπιά
        for (int i = 0; i < answerButtons.Length; i++)
        {
            if (i < currentQuestion.options.Length)
            {
                string answer = currentQuestion.options[i];  // Η απάντηση για το κουμπί
                TMP_Text btnText = answerButtons[i].GetComponentInChildren<TMP_Text>();  // Παίρνουμε το κείμενο του
κουμπιού
                btnText.text = answer;  // Θέτουμε το κείμενο του κουμπιού με την απάντηση

                answerButtons[i].onClick.RemoveAllListeners();  // Καθαρίζουμε τους ακροατές του κουμπιού

                int capturedIndex = i;  // Αποθηκεύουμε το index για να το χρησιμοποιήσουμε στον listener
                answerButtons[i].onClick.AddListener(() =>
                    OnAnswerSelected(answerButtons[capturedIndex], currentQuestion.options[capturedIndex])  // Ορίζουμε τον
listener για την επιλογή απάντησης
                );

                answerButtons[i].GetComponent<Image>().color = Color.white;  // Επαναφορά του χρώματος του κουμπιού
                answerButtons[i].interactable = true;  // Ενεργοποίηση του κουμπιού
                answerButtons[i].gameObject.SetActive(true);  // Ενεργοποίηση του κουμπιού
            }
            else
            {
                answerButtons[i].gameObject.SetActive(false);  // Αν δεν υπάρχουν αρκετές απαντήσεις, κρύβουμε το κουμπί
            }
        }

        reboundAvailable = false;  // Αρχικοποιούμε τη βοήθεια "Rebound"
    }

    // Μέθοδος που καλείται όταν ο παίκτης επιλέξει μια απάντηση
    void OnAnswerSelected(Button clickedButton, string selectedAnswer)
    {
        DisableAnswerButtons();  // Απενεργοποιούμε τα κουμπιά των απαντήσεων

        bool isCorrect = selectedAnswer == currentQuestion.correctAnswer;  // Ελέγχουμε αν η απάντηση είναι σωστή
```

```
        if (isCorrect)
        {
            clickedButton.GetComponent<Image>().color = Color.green;  // Χρωματίζουμε το κουμπί πράσινο για σωστή
απάντηση
            playerScore += currentQuestionDifficulty;  // Αυξάνουμε το σκορ του παίκτη
            questionsAnsweredInRound++;  // Αύξηση των ερωτήσεων που απαντήθηκαν στον γύρο
            reboundAvailable = false;  // Μη διαθέσιμο το "Rebound"
            UpdateScoreText();  // Ενημέρωση του σκορ στην UI
            StartCoroutine(NextStepAfterDelay());  // Περιμένουμε λίγο πριν περάσουμε στην επόμενη ερώτηση
        }
        else
        {
            clickedButton.GetComponent<Image>().color = Color.red;  // Χρωματίζουμε το κουμπί κόκκινο για λάθος απάντηση

            if (!reboundUsed)  // Αν δεν έχει χρησιμοποιηθεί το "Rebound"
            {
                reboundAvailable = true;  // Ενεργοποιούμε το "Rebound"
            }
            else
            {
                reboundAvailable = false;  // Κλείνουμε το "Rebound"
                questionsAnsweredInRound++;  // Προχωράμε στην επόμενη ερώτηση
                StartCoroutine(NextStepAfterDelay());  // Περιμένουμε λίγο πριν περάσουμε στην επόμενη ερώτηση
            }
        }
    }
```

Το single **QuestionDisplay** διαχειρίζεται την εμφάνιση των ερωτήσεων και των απαντήσεων, ενημερώνοντας το σκορ του παίκτη και το επίπεδο δυσκολίας ανάλογα με την επιλογή του. Όταν ο παίκτης επιλέγει μια απάντηση, το σκριπτ ελέγχει αν είναι σωστή ή λάθος, ανανεώνοντας το σκορ και τη UI, και παρέχει βοήθειες όπως "Rebound", "Assist" και "Foul". Επίσης, διαχειρίζεται τη μετάβαση από ερώτηση σε ερώτηση και γύρο σε γύρο.

.

2. Εμφάνιση ερωτήσεων σε Local MultiPlayerMode

```
// === FLOW ===
void ShowDifficultySelection()
{
    if (!ready && !ValidateRefs()) return;

    // reset per-question flags
    questionShown = false;
    inputLocked = false;
    awaitingRebound = false;
    assistActive = false;
    stealActive = false;
    foulSequenceRunning = false;
    powerupLockedThisQuestion = false;

    SetAnswerButtons(false);
    ClearAnswerButtonColors();

    questionText.text = $"Παίκτης {(currentPlayer == 1 ? player1Name : player2Name)}: Διάλεξε δυσκολία για την ερώτηση
{questionCounter + 1} του γύρου {currentRound}";
    questionNumberText.text = "";
    // >>> CHANGED: δείξε "Σειρά: <όνομα>"
    playerTurnText.text = $"Σειρά: {(currentPlayer == 1 ? player1Name : player2Name)}";
```

```
    if (roundEndPanel) roundEndPanel.SetActive(false); // σιγουριά

    SetDifficultyButtons(true);
    RefreshPowerupsUI();
}

void OnDifficultySelected(int level)
{
    if (inputLocked) return;
    currentDifficulty = level;
    LoadQuestion(currentDifficulty);
}

void LoadQuestion(int difficulty)
{
    currentQuestion = questionLoader ? questionLoader.GetRandomQuestion(difficulty) : null;
    if (currentQuestion == null)
    {
        Debug.LogWarning("Δεν υπάρχουν ερωτήσεις για αυτό το επίπεδο ή λείπει το QuestionLoader!");
        ShowDifficultySelection();
        return;
    }
    currentQuestion.difficulty = difficulty;
    answeringPlayer = currentPlayer; // default ο «ιδιοκτήτης» της ερώτησης απαντά
    ShowQuestion();
}

void ShowQuestion()
{
    questionShown = true;
    awaitingRebound = false;
    assistActive = false;
    powerupLockedThisQuestion = false;

    if (reboundWindowRoutine != null) { StopCoroutine(reboundWindowRoutine); reboundWindowRoutine = null; }

    questionText.text = currentQuestion.questionText;
    questionNumberText.text = $"Ερώτηση {questionCounter + 1} - Γύρος {currentRound}";
    // >>> CHANGED
    playerTurnText.text = $"Σειρά: {(answeringPlayer == 1 ? player1Name : player2Name)}";

    for (int i = 0; i < answerButtons.Length; i++)
    {
        Button b = answerButtons[i];
        if (!b) continue;

        b.onClick.RemoveAllListeners();

        if (i < currentQuestion.options.Length)
        {
            b.gameObject.SetActive(true);
            var t = b.GetComponentInChildren<TMP_Text>();
            if (t) t.text = currentQuestion.options[i];
            int idx = i;
            b.onClick.AddListener(() => OnAnswerSelected(idx));
            b.interactable = true;

            var img = b.GetComponent<Image>();
            if (img) { img.color = Color.white; img.raycastTarget = true; } // <- mobile
        }
        else
        {
            b.gameObject.SetActive(false);
        }
    }
```

```
    SetDifficultyButtons(false);
    SetAnswerButtons(true);
    RefreshPowerupsUI();
}

void OnAnswerSelected(int index)
{
    if (inputLocked) return;
    inputLocked = true;
    SetAnswerButtons(false);

    bool isCorrect = currentQuestion.options[index] == currentQuestion.correctAnswer;
    int points = currentDifficulty;

    if (isCorrect)
    {
        var img = answerButtons[index].GetComponent<Image>();
        if (img) img.color = Color.green;
        AddPointsTo(answeringPlayer, points);
        EndQuestionAdvance(0.2f);
    }
    else
    {
        var img = answerButtons[index].GetComponent<Image>();
        if (img) img.color = Color.red;

        // Rebound μόνο αν ο «αρχικός» είναι αυτός που απαντούσε
        if (!awaitingRebound && answeringPlayer == currentPlayer)
        {
            awaitingRebound = true;

            // >>> NEW: ξεκλείδωσε input ώστε ο αντίπαλος να μπορεί να πατήσει Rebound
            inputLocked = false;

            RefreshPowerupsUI(); // ενεργοποιεί μόνο το Rebound του αντιπάλου
            reboundWindowRoutine = StartCoroutine(ReboundWindow(5f)); // 5"
        }
        else
        {
            EndQuestionAdvance(0.2f);
        }
    }
}

IEnumerator ReboundWindow(float seconds)
{
    float t = 0f;
    while (t < seconds && awaitingRebound)
    {
        t += Time.deltaTime;
        yield return null;
    }
    if (awaitingRebound)
    {
        awaitingRebound = false;
        RefreshPowerupsUI();
        EndQuestionAdvance(0.1f);
    }
}

void EndQuestionAdvance(float nextDelay)
{
    if (reboundWindowRoutine != null) { StopCoroutine(reboundWindowRoutine); reboundWindowRoutine = null; }
    awaitingRebound = false;

    UpdateScoreUI();
```

```
    // Αν την ερώτηση την απάντησε ο αντίπαλος (steal/rebound), η επόμενη επιστρέφει στον αρχικό παίκτη
    bool answeredByOpponent = (answeringPlayer != currentPlayer);

    questionCounter++;

    if (!answeredByOpponent)
        SwitchPlayer(); // φυσιολογική εναλλαγή

    if (questionCounter >= questionsPerRound)
    {
        Invoke(nameof(EndRound), nextDelay);
    }
    else
    {
        Invoke(nameof(ShowDifficultySelection), nextDelay);
    }

    inputLocked = false;
    assistActive = false;
    stealActive = false;
    powerupLockedThisQuestion = false;
    RefreshPowerupsUI();
}

void SwitchPlayer()
{
    currentPlayer = (currentPlayer == 1) ? 2 : 1;
}

void EndRound()
{
    SetAnswerButtons(false);
    SetDifficultyButtons(false);
    questionShown = false;

    bool isFinal = currentRound >= totalRounds;

    string winnerLine = "";
    if (isFinal)
    {
        if (player1Score > player2Score) winnerLine = $"\nΝικητής: {player1Name}";
        else if (player2Score > player1Score) winnerLine = $"\nΝικητής: {player2Name}";
        else winnerLine = "\nΙσοπαλία!";
    }

    if (resultText)
        resultText.text = $"Τέλος γύρου {currentRound}\n{player1Name}: {player1Score} | {player2Name}:
{player2Score}{winnerLine}";
    if (roundEndPanel) roundEndPanel.SetActive(true);
    if (startNextRoundButton) startNextRoundButton.gameObject.SetActive(true);

    if (isFinal) FinalizeMatchAndUpdateProfileStats();

    RefreshPowerupsUI();
}

void UpdateScoreUI()
{
    if (player1ScoreText) player1ScoreText.text = $"Σκορ ({player1Name}): {player1Score}";
    if (player2ScoreText) player2ScoreText.text = $"Σκορ ({player2Name}): {player2Score}";
}
```

Αυτό το κομμάτι του κώδικα χειρίζεται την επιλογή και εμφάνιση της δυσκολίας, την φόρτωση και παρουσίαση των ερωτήσεων, την επιλογή των απαντήσεων από τους παίκτες και τη διαχείριση του σκορ και των βοηθειών (Rebound, Assist, Foul,Steal,Block) κατά τη διάρκεια του παιχνιδιού.

ΠΑΡΑΡΤΗΜΑ Β

3. Περιγραφή ενός online γύρου με Photo Pun

```
// === FLOW ===
void ShowDifficultySelection()
{
    if (!ready && !ValidateRefs()) return;

    // reset per-question flags
    questionShown = false;
    inputLocked = false;
    awaitingRebound = false;
    assistActive = false;
    stealActive = false;
    foulSequenceRunning = false;
    powerupLockedThisQuestion = false;

    SetAnswerButtons(false);
    ClearAnswerButtonColors();

    questionText.text = $"Παίκτης {(currentPlayer == 1 ? player1Name : player2Name)}: Διάλεξε δυσκολία για την ερώτηση
{questionCounter + 1} του γύρου {currentRound}";
    questionNumberText.text = "";
    // >>> CHANGED: δείξε "Σειρά: <όνομα>"
    playerTurnText.text = $"Σειρά: {(currentPlayer == 1 ? player1Name : player2Name)}";

    if (roundEndPanel) roundEndPanel.SetActive(false); // σιγουριά

    SetDifficultyButtons(true);
    RefreshPowerupsUI();
}

void OnDifficultySelected(int level)
{
    if (inputLocked) return;
    currentDifficulty = level;
    LoadQuestion(currentDifficulty);
}

void LoadQuestion(int difficulty)
{
    currentQuestion = questionLoader ? questionLoader.GetRandomQuestion(difficulty) : null;
    if (currentQuestion == null)
    {
        Debug.LogWarning("Δεν υπάρχουν ερωτήσεις για αυτό το επίπεδο ή λείπει το QuestionLoader!");
        ShowDifficultySelection();
        return;
    }
    currentQuestion.difficulty = difficulty;
    answeringPlayer = currentPlayer; // default ο «ιδιοκτήτης» της ερώτησης απαντά
    ShowQuestion();
}

void ShowQuestion()
```

```
{
    questionShown = true;
    awaitingRebound = false;
    assistActive = false;
    powerupLockedThisQuestion = false;

    if (reboundWindowRoutine != null) { StopCoroutine(reboundWindowRoutine); reboundWindowRoutine = null; }

    questionText.text = currentQuestion.questionText;
    questionNumberText.text = $"Ερώτηση {questionCounter + 1} - Γύρος {currentRound}";
    // >>> CHANGED
    playerTurnText.text = $"Σειρά: {(answeringPlayer == 1 ? player1Name : player2Name)}";

    for (int i = 0; i < answerButtons.Length; i++)
    {
        Button b = answerButtons[i];
        if (!b) continue;

        b.onClick.RemoveAllListeners();

        if (i < currentQuestion.options.Length)
        {
            b.gameObject.SetActive(true);
            var t = b.GetComponentInChildren<TMP_Text>();
            if (t) t.text = currentQuestion.options[i];
            int idx = i;
            b.onClick.AddListener(() => OnAnswerSelected(idx));
            b.interactable = true;

            var img = b.GetComponent<Image>();
            if (img) { img.color = Color.white; img.raycastTarget = true; } // <- mobile
        }
        else
        {
            b.gameObject.SetActive(false);
        }
    }

    SetDifficultyButtons(false);
    SetAnswerButtons(true);
    RefreshPowerupsUI();
}

void OnAnswerSelected(int index)
{
    if (inputLocked) return;
    inputLocked = true;
    SetAnswerButtons(false);

    bool isCorrect = currentQuestion.options[index] == currentQuestion.correctAnswer;
    int points = currentDifficulty;

    if (isCorrect)
    {
        var img = answerButtons[index].GetComponent<Image>();
        if (img) img.color = Color.green;
        AddPointsTo(answeringPlayer, points);
        EndQuestionAdvance(0.2f);
    }
    else
    {
        var img = answerButtons[index].GetComponent<Image>();
        if (img) img.color = Color.red;

        // Rebound μόνο αν ο «αρχικός» είναι αυτός που απαντούσε
        if (!awaitingRebound && answeringPlayer == currentPlayer)
        {
```

```
            awaitingRebound = true;

            // >>> NEW: ξεκλείδωσε input ώστε ο αντίπαλος να μπορεί να πατήσει Rebound
            inputLocked = false;

            RefreshPowerupsUI(); // ενεργοποιεί μόνο το Rebound του αντιπάλου
            reboundWindowRoutine = StartCoroutine(ReboundWindow(5f)); // 5"
        }
        else
        {
            EndQuestionAdvance(0.2f);
        }
    }
}

IEnumerator ReboundWindow(float seconds)
{
    float t = 0f;
    while (t < seconds && awaitingRebound)
    {
        t += Time.deltaTime;
        yield return null;
    }
    if (awaitingRebound)
    {
        awaitingRebound = false;
        RefreshPowerupsUI();
        EndQuestionAdvance(0.1f);
    }
}

void EndQuestionAdvance(float nextDelay)
{
    if (reboundWindowRoutine != null) { StopCoroutine(reboundWindowRoutine); reboundWindowRoutine = null; }
    awaitingRebound = false;

    UpdateScoreUI();

    // Αν την ερώτηση την απάντησε ο αντίπαλος (steal/rebound), η επόμενη επιστρέφει στον αρχικό παίκτη
    bool answeredByOpponent = (answeringPlayer != currentPlayer);

    questionCounter++;

    if (!answeredByOpponent)
        SwitchPlayer(); // φυσιολογική εναλλαγή

    if (questionCounter >= questionsPerRound)
    {
        Invoke(nameof(EndRound), nextDelay);
    }
    else
    {
        Invoke(nameof(ShowDifficultySelection), nextDelay);
    }

    inputLocked = false;
    assistActive = false;
    stealActive = false;
    powerupLockedThisQuestion = false;
    RefreshPowerupsUI();
}

void SwitchPlayer()
{
    currentPlayer = (currentPlayer == 1) ? 2 : 1;
}
```

```
void EndRound()
{
    SetAnswerButtons(false);
    SetDifficultyButtons(false);
    questionShown = false;

    bool isFinal = currentRound >= totalRounds;

    string winnerLine = "";
    if (isFinal)
    {
        if (player1Score > player2Score) winnerLine = $"\nΝικητής: {player1Name}";
        else if (player2Score > player1Score) winnerLine = $"\nΝικητής: {player2Name}";
        else winnerLine = "\nΙσοπαλία!";
    }

    if (resultText)
        resultText.text = $"Τέλος γύρου {currentRound}\n{player1Name}: {player1Score} | {player2Name}:
{player2Score}{winnerLine}";
    if (roundEndPanel) roundEndPanel.SetActive(true);
    if (startNextRoundButton) startNextRoundButton.gameObject.SetActive(true);

    if (isFinal) FinalizeMatchAndUpdateProfileStats();

    RefreshPowerupsUI();
}

void UpdateScoreUI()
{
    if (player1ScoreText) player1ScoreText.text = $"Σκορ ({player1Name}): {player1Score}";
    if (player2ScoreText) player2ScoreText.text = $"Σκορ ({player2Name}): {player2Score}";
}
```

Ο συγκεκριμένος κώδικας ελέγχει ότι ο παίκτης είναι σε room, στήνει ασφαλές mobile UI, ορίζει p1/p2 και σειρά (turnActorNumber), δένει κουμπιά και συγχρονίζει τα αρχικά power-ups. Όταν ο παίκτης με τη σειρά διαλέξει δυσκολία, ο Master τραβά τυχαία ερώτηση από τον questionLoader και τη στέλνει σε όλους με RPC (authoritative). Σε κάθε client ενημερώνει κείμενα/αρίθμηση/ποιος παίζει, γεμίζει τις επιλογές, ενεργοποιεί μόνο τα σωστά κουμπιά (απαντήσεις/power-ups) και κλειδώνει ό,τι δεν του αναλογεί.