ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία εφαρμογής επαυξημένης πραγματικότητας για το μάθημα Μελέτη Περιβάλλοντος της Γ' Δημοτικού με προσέγγιση εκπαιδευτικού παιχνιδιού»

Του φοιτητή
Αλέξανδρου Κύτρα
Αρ. Μητρώου: 185439

Επιβλέπων
Ευκλείδης Κεραμόπουλος
Αναπληρωτής Καθηγητής

Ημερομηνία 15-01-2023

Τίτλος Δ.Ε. Δημιουργία εφαρμογής επαυξημένης πραγματικότητας για το μάθημα Μελέτη Περιβάλλοντος της Γ' Δημοτικού με προσέγγιση εκπαιδευτικού παιχνιδιού

Κωδικός Δ.Ε. 22293

Ονοματεπώνυμο φοιτητή/τών Αλέξανδρος Κύτρας

Ονοματεπώνυμο εισηγητή Ευκλείδης Κεραμόπουλος

Ημερομηνία ανάληψης Δ.Ε. 25-10-2022

Ημερομηνία περάτωσης Δ.Ε 15-01-2023

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 διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιερώνω τη διατριβή στην οικογένεια μου η οποία με στήριξε στις δυσκολίες που συνάντησα στην περίοδο υλοποίησης της»

Πρόλογος

Ο προγραμματισμός παιχνιδιού παράλληλα με το γραφικό περιβάλλον που προσφέρουν οι μηχανές παιχνιδιού όπως το Unity είναι αρκετά ενδιαφέρον καθώς ένας προγραμματιστής μπορεί να δημιουργήσει ένα παιχνίδι σε κάποιο χρονικό διάστημα ανάλογα με την εμπειρία και τα σχέδια του απασχολώντας χρήστες οι οποίοι προσπαθούν να ξεπεράσουν κάθε πρόκληση και εμπόδιο. Παρόλο που η συγκεκριμένη διπλωματική ως ανάπτυξη εκπαιδευτικού παιχνιδιού δεν αντιμετωπίζει τέτοιου είδους χρήστες, είναι εξίσου ενδιαφέρουσα καθώς ο κόσμος της επαυξημένης πραγματικότητας σε συνδυασμό με το θέμα της παρούσας διπλωματικής ελευθερώνει κάθε είδους περιορισμό ως προς την δημιουργικότητα, αλλά και παρέχει αρκετές γνώσεις και εμπειρίες, πρώτον στους χρήστες της εφαρμογής οι οποίοι έχουν την ευκαιρία να ταξιδέψουν στον επαυξημένο κόσμο μαθαίνοντας νέες γνώσεις, και δεύτερον σε εμένα ως δημιουργό της εφαρμογής, εξελίσσοντας τον τρόπο σκέψης ως προς την διαχείριση και την δημιουργικότητα κατά την διάρκεια δημιουργίας μιας ολοκληρωμένης εφαρμογής επαυξημένης πραγματικότητας.

Περίληψη

Η εφαρμογή επαυξημένης πραγματικότητας που δημιουργήθηκε για το μάθημα της μελέτης περιβάλλοντος της Γ' Δημοτικού αποτελείται από ασκήσεις που παρέχουν επιπλέον υλικό πέρα από τις πληροφορίες του βιβλίου και έχει ως στόχο να διδάξει τον χρήστη κάποιες σημαντικές γνώσεις για το περιβάλλον του συνδυάζοντας τη μαθησιακή διαδικασία με μια διασκεδαστική και ενδιαφέρουσα προσέγγιση εκπαιδευτικού παιχνιδιού.

Η εφαρμογή δημιουργήθηκε μέσω της μηχανής παιχνιδιού Unity και συγκεκριμένα με την βοήθεια του συνόλου εργαλείων του Vuforia Engine με το οποίο ο προγραμματιστής μετατρέπει μια απλή εφαρμογή σε εφαρμογή επαυξημένης πραγματικότητας, επαυξάνοντας το φυσικό περιβάλλον με εικονικά αντικείμενα με τα οποία υπάρχει αλληλεπίδραση σε πραγματικό χρόνο.

Ανοίγοντας την εφαρμογή, ο χρήστης αλληλεπιδρά με την διεπαφή χρήστη η οποία περιέχει πληροφορίες σχετικά με τον τρόπο χρήσης της εφαρμογής, καθώς και κουμπιά με τα οποία ο χρήστης μεταφέρεται στο αντίστοιχο περιβάλλον μιας από τις πέντε ασκήσεις που υλοποιήθηκαν. Όταν επιλεγεί μια άσκηση, ο χρήστης πρέπει να σαρώσει την αντίστοιχη φωτογραφία του βιβλίου προκειμένου να εμφανιστεί το εικονικό περιβάλλον της άσκησης στην κάμερα της πλατφόρμας.

Η πρώτη από τις πέντε ασκήσεις διδάσκει στον χρήστη την σημαντικότητα των κανόνων καθώς αλληλεπιδρά με το περιβάλλον και απαντά σε ερωτήσεις. Η δεύτερη διδάσκει στον χρήστη κάποιους βασικούς γεωγραφικούς όρους δημιουργώντας ένα κόσμο από που αποτελείται από αυτούς αλληλεπιδρώντας με το εικονικό περιβάλλον. Στην τρίτη άσκηση ο χρήστης βοηθάει ένα αυτοκίνητο να φτάσει στον προορισμό του απαντώντας σε ερωτήσεις, καθώς αυτό συναντά κάποια βασικά σήματα οδικής κυκλοφορίας. Η τέταρτη άσκηση αφορά την εκμάθηση βασικών πληροφοριών σχετικά με την βλάστηση του τόπου, ενώ η πέμπτη άσκηση αφορά τα ζώα του τόπου μας μαθαίνοντας κάποια βασικά χαρακτηριστικά για αυτά

Λέξεις κλειδιά: Επαυξημένη πραγματικότητα, μηχανή παιχνιδιού Unity, Vuforia Engine, επαναχρησιμοποίηση.

«Creation of an augmented reality application for the 3rd grade course Environmental-Studies with an educational game approach»

«Alexandros Kytras»

Abstract

The augmented reality application created for the 3rd grade environmental studies course consists of exercises that provide additional material beyond the information in the book and aims to teach the user some important knowledge about their environment by combining the learning process with a fun and interesting educational game approach.

The application was created through the Unity game engine and specifically with the help of the Vuforia Engine toolset with which the developer turns a simple application into an augmented reality application, augmenting the physical environment with virtual objects with which there is real-time interaction.

By opening the application, the user interacts with the user interface which contains information about how to use the application, as well as buttons that take the user to the corresponding environment of one of the five exercises implemented. When an exercise is selected, the user must scan the corresponding photo in the book in order to display the virtual environment of the exercise on the platform's camera.

The first of five exercises teaches the user the importance of rules as they interact with the environment and answer questions. The second one teaches the user some basic geographical terms by creating a world made up of them by interacting with the virtual environment. In the third exercise the user helps a car reach its destination by answering questions as it encounters some basic road signs. The fourth exercise is about learning basic information about the vegetation of the place, while the fifth exercise is about the animals of our place, learning some basic characteristics about them.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες αρχικά στον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας Ευκλείδη Κεραμόπουλο για την άριστη συνεργασία και ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον αλλά και σημαντικό θέμα το οποίο μου πρόσφερε πολλές χρήσιμες γνώσεις, και έπειτα στην οικογένεια μου η οποία με στήριξε καθόλη την διάρκεια εκπόνησης της διπλωματικής εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
UI	User Interface
3D	Three Dimensional
2D	Two Dimensional
AR	Augmented Reality
AV	Augmented Virtuality
VR	Virtual Reality
MR	Mixed Reality
XR	Extended Reality
BIRP	Built-in Render Pipeline
URP	Universal Render Pipeline
HDRP	High Definition Render Pipeline
SRP	Scriptable Render Pipeline
SDK	Software development kit

Κεφάλαιο 1ο: Εισαγωγή

Η παρούσα διπλωματική εργασία έχει ως στόχο την συμβολή στην εκμάθηση των μαθητών με έναν ενδιαφέρον τρόπο μέσω της εφαρμογής επαυξημένης πραγματικότητας η οποία συνδυάζει τη γνώση με το παιχνίδι. Η εφαρμογή είναι εμπνευσμένη από το βιβλίο "Μελέτη περιβάλλοντος" της Γ' δημοτικού και περιέχει επιπλέον υλικό σχετικά με τις ενότητες του, συσσωρευμένο σε πέντε ασκήσεις.

Με τον όρο επαυξημένη πραγματικότητα εννοείται η προβολή τρισδιάστατων εικονικών μοντέλων που δημιουργούνται από υπολογιστή στον πραγματικό κόσμο μέσω της κάμερας μιας πλατφόρμας σε πραγματικό χρόνο. Η εφαρμογή έχει δημιουργηθεί μέσω Unity και συγκεκριμένα μέσω του συνόλου εργαλείων (Software Development Kit) Vuforia Engine.

Στο δεύτερο κεφάλαιο αναφέρεται ο ορισμός της επαυξημένης πραγματικότητας η οποία αποτελεί μέρος της συνέχειας πραγματικότητας-εικονικότητας που παρουσιάζει πόσο αληθινό ή εικονικό μπορεί να είναι κάποιο περιβάλλον έχοντας από την μια μεριά το πραγματικό περιβάλλον και από την άλλη το πλήρες εικονικό περιβάλλον που δεν αποτελείται από κανένα πραγματικό αντικείμενο καθώς όλα είναι εικονικά. Η επαυξημένη πραγματικότητα επαυξάνει την πραγματικότητα γι'αυτό και ανήκει προς την μεριά της στην συνέχεια πραγματικότητας-εικονικότητας. Αντίστοιχα για το πλήρες εικονικό περιβάλλον το οποίο αναφέρεται στο κεφάλαιο ως εικονική πραγματικότητα ανήκει στην άλλη μεριά της συνέχειας πραγματικότητας-εικονικότητας, καθώς επίσης αναφέρεται και η επαυξημένη εικονικότητα η οποία επαυξάνει την εικονικότητα με πραγματικά αντικείμενα. Θεωρείται πως είναι απαραίτητη μια ιστορική αναδρομή με σκοπό να κατανοηθεί πλήρως η έννοια της επαυξημένης πραγματικότητας, έτσι αναφέρεται η πρώτη συσκευή επαυξημένης πραγματικότητας, οι τομείς στους οποίους άρχισε να χρησιμοποιείται η επαυξημένη πραγματικότητα, το πρώτο παιχνίδι, και τέλος οι νεότερες συσκευες Headset που αποτελούνται από διάφορες τεχνολογίες ενσωματωμένες σε αυτές, πολλές από τις οποίες δημιουργούν την προβολή των τρισδιάστατων εικονικών αντικειμένων στις οθόνες τους που αποτελεί το βασικό χαρακτηριστικό τους.

Το τρίτο κεφάλαιο αφορά το πρόγραμμα μηχανής παιχνιδιού με το οποίο έχει δημιουργηθεί η εφαρμογή και ονομάζεται Unity. Το Unity πέρα από την παραγωγή παιχνιδιών χρησιμοποιείται στην αρχιτεκτονική, στην βιομηχανία και στον κινηματογράφο. Η ποιότητα της σκηνής εξαρτάται κατά ένα μεγάλο μέρος από το Render Pipeline που θα επιλέξει ο προγραμματιστής στην εφαρμογή του. Ένα Render Pipeline καθορίζει πως θα εμφανίζονται τα περιεχόμενα μιας σκηνής μέσω τριών λειτουργιών που ονομάζονται Culling, Rendering και Post-Processing. Η Unity έχει δημιουργήσει τρία Render Pipelines που είναι διαθέσιμα και μπορούν να χρησιμοποιηθούν ή να υποβληθούν σε επεξεργασία μέσω Scripts (κομμάτια κώδικα). Πριν αναφερθούν οι λειτουργίες του Unity, αναλύονται κάποιες βασικές έννοιες όπως το GameObject που είναι κάποιο αντικείμενο στο χώρο, τα Scenes που είναι οι σκηνές του περιβάλλοντος και ο Canvas που αποτελεί την διεπαφή του χρήστη (UI). Στην συνέχεια αναφέρονται τα κυριότερα παράθυρα του Unity τα οποία χρησιμοποιούνται ως εργαλεία και βοηθούν τον προγραμματιστή να δημιουργήσει την εφαρμογή. Το Scene View είναι ένα από τα βασικότερα παράθυρα λόγω του ότι χρησιμοποιείται για να προβάλει το περιβάλλον μέσω του προγραμματιστή, και ο ίδιος μπορεί μέσω αυτού του παραθύρου να μετακινήσει, να περιστρέψει ή ακόμη και να κλιμακώσει κάποιο αντικείμενο του χώρου προς κάποιον από τους τρεις άξονες. Άλλα παράθυρα χρησιμοποιούνται για να προβάλλουν την λίστα με τα αντικείμενα του σκηνικού, είτε τα αρχεία του προγράμματος ή ακόμη και τις ρυθμίσεις των Components των αντικειμένων.

Components είναι τα χαρακτηριστικά ενός αντικειμένου τα οποία δίνουν ιδιότητες στα αντικείμενα όπως το Transform που αποτελεί την τοποθεσία, περιστροφή και κλίμακα ενός αντικειμένου, ενώ είναι το μόνο Component που δεν μπορεί να διαγραφεί από κάποιο αντικείμενο. Για την εμφάνιση του αντικειμένου χρησιμοποιούνται τα Components Mesh Filter και Mesh Renderer για να οριστούν τα όρια του αντικειμένου ως πλέγμα (Mesh) και το υλικό με το οποίο θα προβάλλεται το αντικείμενο.

Τέλος γίνεται μία αναφορά στο SDK Vuforia Engine που χρησιμοποιείται στο Unity με σκοπό την δημιουργία της εφαρμογής επαυξημένης πραγματικότητας. Ειδικότερα αναφέρεται η εγκατάσταση του Vuforia στο Unity, η βάση δεδομένων που χρησιμοποιείται προκειμένου να βρεθεί και να σαρωθεί η εικόνα έτσι ώστε να εμφανιστεί το εικονικό περιβάλλον.

Η επαναχρησιμοποίηση είναι πολύ σημαντική τόσο στον κόσμο όσο και στον προγραμματισμό, διότι δημιουργώντας λειτουργίες που χρησιμοποιείται παντού εξοικονομεί χρόνο και ενέργεια, σε αντίθεση με την δημιουργία πολλών παρόμοιων λειτουργιών που απαιτείται πολλαπλές προσπάθειες για σχετικά παρόμοιες λειτουργίες. Το τέταρτο κεφάλαιο ξεκινάει με μία αναφορά και ανάλυση στις βασικές λειτουργίες που επαναχρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής, μια από τις οποίες είναι η λειτουργία LevelCompleted που αποτελείται από αντικείμενα και διεπαφή χρήστη, καθώς και γραμμές κώδικα που ενεργοποιούν την διαδικασία με την οποία τελειώνει κάποια άσκηση.

Η εφαρμογή αποτελείται από το κεντρικό μενού το οποίο δίνει στον χρήστη την πρόσβαση στις πέντε ασκήσεις. Το κεντρικό μενού πέρα από τα κουμπιά με τα οποία ο χρήστης μπαίνει στις ασκήσεις, περιέχει και οδηγίες σχετικά με τον τρόπο που μπορεί να χρησιμοποιήσει ο χρήστης την άσκηση καθώς και τις εικόνες οι οποίες πρέπει να σαρωθούν για να ξεκινήσει η αντίστοιχη άσκηση.

Η κάθε άσκηση αναλύεται εις βάθος ξεκινώντας από την πρώτη που ονομάζεται “Χρειαζόμαστε κανόνες” και διδάσκει στον χρήστη την σημαντικότητα των κανόνων μέσω υποασκήσεων στις οποίες ο χρήστης απαντάει ερωτήσεις και αλληλεπιδρά με το εικονικό περιβάλλον. Στην δεύτερη άσκηση ο χρήστης μαθαίνει κάποιους γεωγραφικούς όρους, και αλληλεπιδρά με το εικονικό περιβάλλον για να δημιουργήσει έναν κόσμο τοποθετώντας τους όρους που μαθαίνει σε αυτόν. Η τρίτη άσκηση ονομάζεται “Μαθαίνουμε τα σήματα” και διδάσκει στον χρήστη μερικά απαραίτητα σήματα οδικής κυκλοφορίας βοηθώντας κάποιο αμάξι να φτάσει στον τελικό του προορισμό. Τέλος αναφέρονται οι ασκήσεις τέσσερα (“Τα φυτά του τόπου μας”) και πέντε (“Τα ζώα του τόπου μας”) με τις οποίες ο χρήστης μαθαίνει μερικές βασικές έννοιες και χαρακτηριστικά για την χλωρίδα και την πανίδα του τόπου μας απαντώντας σε ερωτήσεις τύπου κουίζ και ταυτόχρονα αλληλεπιδρώντας με το εικονικό περιβάλλον.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα κυριότερα σημεία των παραπάνω κεφαλαίων ξεκινώντας με μία εισαγωγή στην εφαρμογή επαυξημένης πραγματικότητας που δημιουργήθηκε κατά την διάρκεια υλοποίησης της διπλωματικής εργασίας επισημαίνοντας τα εργαλεία και τις λειτουργίες που χρησιμοποιήθηκαν. Ακόμη, αναφέρονται οι μελλοντικές βελτιώσεις που μπορούν να γίνουν ώστε η εφαρμογή να γίνει καλύτερη και αποτελεσματικότερη. Τέλος αναφέρονται τα συμπεράσματα που δημιουργήθηκαν στην περίοδο υλοποίησης της διπλωματικής εργασίας.

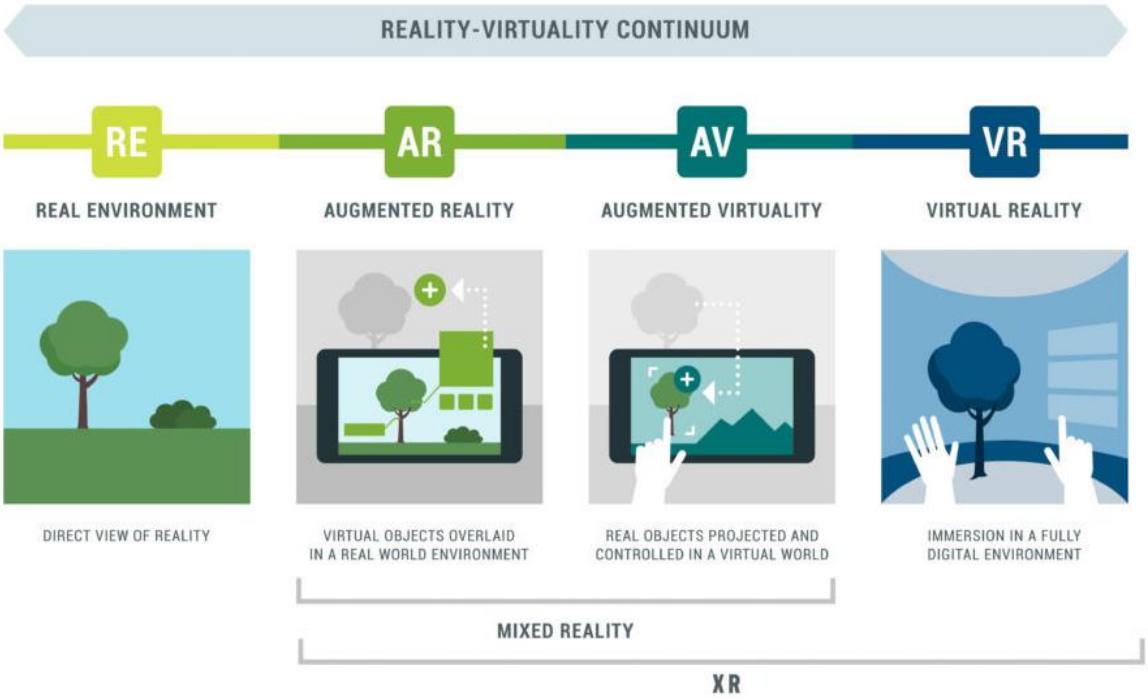
Κεφάλαιο 2ο: Πραγματικότητα και Εικονικότητα

2.1 Εισαγωγή

Ο κόσμος αναπτύσσεται χάρη στην ραγδαία εξέλιξη υπαρχουσών τεχνολογιών καθώς και την δημιουργία νέων όπως αυτές που προσομοιώνουν την πραγματικότητα. Η προσομοίωση της πραγματικότητας χρησιμοποιείται πιο συχνά σε εφαρμογές ψυχαγωγίας όπως τα βιντεοπαιχνίδια και αναλύεται ειδικότερα σε 4 κύριες κατηγορίες.

Η πρώτη είναι η ίδια η πραγματικότητα με την οποία οι άνθρωποι αλληλεπιδρούν στο πραγματικό περιβάλλον με πραγματικά αντικείμενα, ενώ η ακριβώς αντίθετη πλευρά ονομάζεται εικονική πραγματικότητα (Virtual Reality)[14]. Αν υποθέσουμε μια γραμμή (σχήμα 2.1) στην οποία όσο πιο αριστερά είναι μια τεχνολογία τόσο πιο πολύ πλησιάζει την πραγματικότητα, και όσο πιο δεξιά, τόσο πιο πολύ πλησιάζει την εικονική πραγματικότητα, μεταξύ τους υπάρχει η επαυξημένη πραγματικότητα (Augmented Reality) που τείνει προς το αριστερό μέρος καθώς αποτελείται κυρίως από τον φυσικό κόσμο[2] και η επαυξημένη εικονικότητα (Augmented Virtuality) που τείνει προς το δεξί μέρος και αποτελείται κυρίως από τον εικονικό κόσμο[7][15].

Και οι δύο τεχνολογίες που βρίσκονται στην μέση κληρονομούν ιδιότητες και από τους δύο κόσμους (πραγματικός και εικονικός), άρα αποτελούν την μικτή πραγματικότητα (Mixed Reality). Τέλος υπάρχει ο όρος εκτεταμένη πραγματικότητα (Extended reality) που αποτελείται από όλες τις τεχνολογίες που επεκτείνουν την πραγματικότητα που βιώνουμε είτε συνδυάζοντας τον εικονικό και τον πραγματικό κόσμο είτε δημιουργώντας μια πλήρως εικονική εμπειρία.



Σχήμα 2.1: Reality-Virtuality Continuum

2.2 Επαυξημένη Πραγματικότητα

Το υλικό που δημιουργείται από υπολογιστή όπως εικόνες και μοντέλα μπορεί να προβληθεί στον πραγματικό κόσμο με την χρήση τεχνολογιών. Η επαυξημένη πραγματικότητα (Augmented reality - AR) αναφέρεται σε ένα ευρύ φάσμα τέτοιων τεχνολογιών, που συνδυάζουν τα πραγματικά στοιχεία με τα εικονικά (όπως μοντέλα) σε πραγματικό χρόνο τα οποία έχουν συντεταγμένες στον πραγματικό χώρο με την βοήθεια εργαλείων και αισθητήρων και φυσικά της κάμερας της πλατφόρμας[5][6]. Το σχήμα 2.2 είναι ένα παράδειγμα επαυξημένης πραγματικότητας εμφανίζοντας στο περιβάλλον του χρήστη κάποια εικονικά μοντέλα σε πραγματικό χρόνο μέσω της κάμερας. Η εφαρμογή της παρούσας διπλωματικής εργασίας χρησιμοποιεί την επαυξημένη πραγματικότητα μέσω της εφαρμογής Unity και των εργαλείων της.

Σχήμα 2.2: Augmented Reality

2.3 Επαυξημένη Εικονικότητα

Ενώ στην επαυξημένη πραγματικότητα υπάρχουν εικονικά αντικείμενα στον αληθινό κόσμο, η επαυξημένη εικονικότητα επικεντρώνεται κυρίως στον εικονικό κόσμο. Η επαυξημένη εικονικότητα (Augmented Virtuality - AV) χρησιμοποιείται με πολλούς τρόπους, όπως η σχεδίαση μιας εικονικής κουζίνας σε ένα ψηφιακό χώρο ή σε κάποιο περιβάλλον παιχνιδιού που τα πραγματικά αντικείμενα και οι παίκτες προβάλλονται και συμμετέχουν σε κάποιον εικονικό κόσμο Με λίγα λόγια η επαυξημένη εικονικότητα ενισχύει την εικονική εμπειρία προσθέτοντας στοιχεία του πραγματικού περιβάλλοντος[7]. Στο παράδειγμα του σχήματος 2.3, πρόκειται για κάποιο εικονικό κόσμο που έχει δημιουργηθεί μέσω του περιβάλλοντος του χρήστη, και ο ίδιος μπορεί να αλληλεπιδράσει σε αυτόν μέσω αληθινών αντικειμένων που προβάλλονται στον εικονικό κόσμο ως εικονικά αντικείμενα μέσω Headset.

Σχήμα 2.3: Augmented Virtuality

2.4 Εικονική Πραγματικότητα

Εικονική πραγματικότητα (Virtual Reality - VR) είναι η αλληλεπίδραση του χρήστη σε ένα πλήρες εικονικό περιβάλλον με εικονικά αντικείμενα χωρίς την συμμετοχή πραγματικών αντικειμένων[8]. Ο χρήστης μπορεί να αλληλεπιδράσει με το εικονικό περιβάλλον και τα αντικείμενα που υπάρχουν σε αυτό με τις αισθήσεις του όπως θα αλληλεπιδρούσε και σε ένα πραγματικό περιβάλλον. Αυτό επιτυγχάνεται συνδυάζοντας διάφορες συσκευές απεικόνισης (VR Headset), ακοής (ακουστικά με 3D ήχο) και αφής (ασύρματα Controllers που αντιγράφουν τις κινήσεις των χεριών του χρήστη).

Σχήμα 2.4: Virtual Reality

2.5 Μικτή Πραγματικότητα

Ο συνδυασμός φυσικού και ψηφιακού κόσμου ονομάζεται μικτή πραγματικότητα (Mixed Reality - MR) και είναι ένας γενικότερος όρος που περιέχει τεχνολογίες πραγματικότητας όπως είναι η εκτεταμένη πραγματικότητα (AR) και η εκτεταμένη εικονικότητα (AV). Στην Μικτή πραγματικότητα

μπορούν τα πραγματικά και τα εικονικά αντικείμενα να αλληλεπιδράσουν μεταξύ τους και μεταξύ του χρήστη. Ακόμη χρησιμοποιεί στοιχεία στα οποία βασίζεται όπως την γραφική επεξεργασία, τις τεχνολογίες οθόνης αλλά ακόμη και το Cloud Computing.

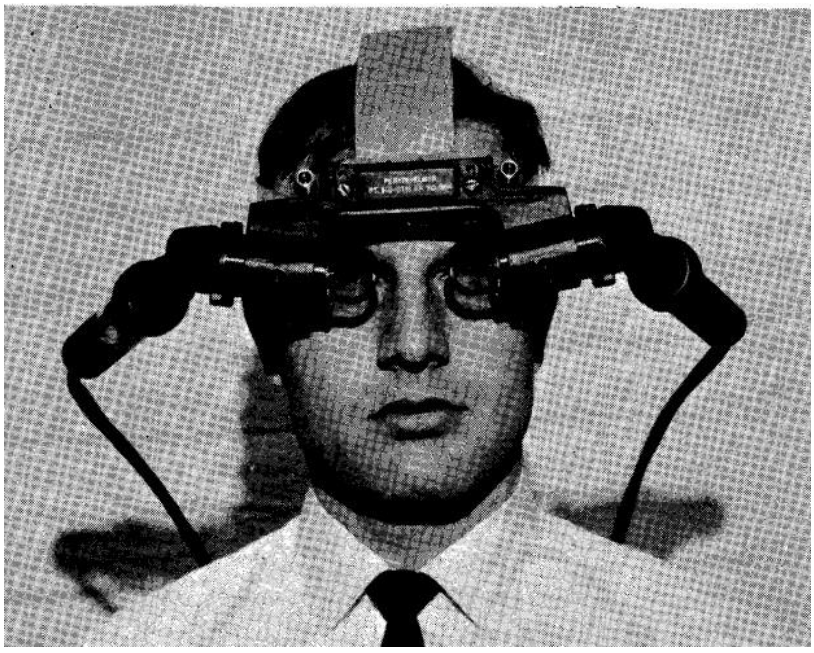
2.6 Εκτεταμένη Πραγματικότητα

Η εκτεταμένη πραγματικότητα (Extended reality - XR) είναι ένας γενικός όρος που καλύπτει διάφορες τεχνολογίες πραγματικότητας όπως την επαυξημένη πραγματικότητα και την εικονική πραγματικότητα. Πιο απλά, στη συντομογραφία XR της εκτεταμένης πραγματικότητας υποθέτουμε πως το γράμμα X είναι μια μεταβλητή και στην θέση της μπορούν να μπουν άλλα γράμματα όπως AR για επαυξημένη πραγματικότητα και VR που είναι η εικονική πραγματικότητα.

2.7 Ιστορική αναδρομή της επαυξημένης πραγματικότητας

2.7.1 The Sword of Damocles

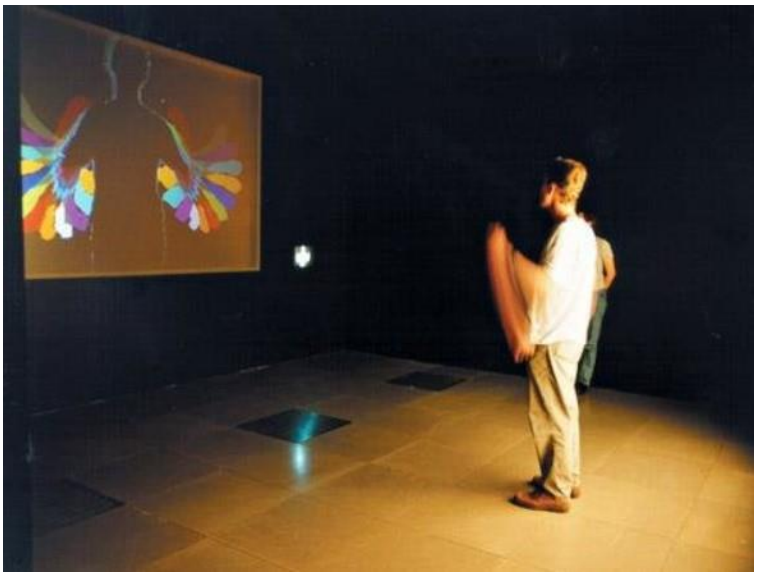
Η ιδέα της επαυξημένης πραγματικότητας ξεκίνησε πολύ παλιά και είχε συζητηθεί για χρόνια. Ο Ivan Sutherland ξεκίνησε την σχεδίαση μιας τεχνολογίας με την οποία ο χρήστης μπορούσε μέσω φακών (σχήμα 2.5) να βλέπει αντικείμενα που δεν υπήρχαν στην πραγματικότητα[9]. Το 1968 δημιουργήθηκε την πρώτη Τρισδιάστατη οθόνη που τοποθετείται στο κεφάλι (Head-Mounted Three Dimensional Display) με την ονομασία “The Sword of Damocles”. Ο στόχος της δημιουργίας ήταν μέσω της οθόνης να περιβάλλει τον χρήστη με τρισδιάστατες πληροφορίες. Ο χρήστης είχε την δυνατότητα να κινηθεί ελάχιστα προκειμένου να δει τα αντικείμενα καλύτερα, μετρώντας παράλληλα την θέση και την περιστροφή που είχε η οθόνη μέσω κάποιου υπολογιστή που χρησιμοποιούσε τις συντεταγμένες του με βάση το δωμάτιο προκειμένου να αλλάξει το σημείο θέασης (Point of View) και να φαίνονται τα αντικείμενα τρισδιάστατα και όχι σαν απλές εικόνες[10].



Σχήμα 2.5: The Sword of Damocles

2.7.2 Videoplace

Λίγο αργότερα το 1970, δημιουργήθηκε από τον Myron Krueger ένα εργαστήριο τεχνητής πραγματικότητας (Artificial Reality), που με την βοήθεια αισθητήρων καταλάβαινε τις κινήσεις των χρηστών. Το εργαστήριο αυτό ονομάστηκε Videospace και οι χρήστες του μπορούσαν να αλληλεπιδρούν για πρώτη φορά με εικονικά αντικείμενα (σχήμα 2.6).



Σχήμα 2.6: Videoplace

Η ονομασία επαυξημένη Πραγματικότητα (Augmented Reality) δημιουργήθηκε το 1990 από έναν ερευνητή αεροπλάνων Tom Caddell. Περίπου εκείνη την περίοδο αρχισε αυτή όπως και άλλες τεχνολογίες να χρησιμοποιούνται αρχικά στον στρατό, ενώ αργότερα η επαυξημένη Πραγματικότητα άρχισε να χρησιμοποιείται στο θέατρο μετά στον αθλητισμό, αργότερα στην πλοήγηση και το 2000 εντάχθηκε στον τομέα των παιχνιδιών.

2.7.3 ARQuake

Το πρώτο AR παιχνίδι δημιουργήθηκε το 2000 και πήρε την ονομασία “ARQuake”, προερχόμενο από την AR έκδοση του δημοφιλούς για την τότε εποχή παιχνιδιού “Quake”[11]. Για να παίξει κάποιος, έπρεπε να φορέσει την οθόνη (Headset), η οποία λειτουργούσε με αισθητήρες και λειτουργίες όπως GPS και γυροσκόπιο.

Σχήμα 2.7: ARQuake

2.7.4 ARTennis

Το 2006 δημιουργήθηκε το πρώτο Face to Face AR παιχνίδι[12]. Το παιχνίδι ονομάζεται "ARTennis" και δύο παίκτες κάθονται απέναντι ο ένας από τον άλλο σε ένα τραπέζι το οποίο έχει ένα χαρτί για να αναγνωρίζει τις θέσεις των παικτών.

Οι χρήστες βλέπουν ένα εικονικό περιβάλλον τένις και χρησιμοποιούν τα τηλέφωνά τους για να παίξουν τένις σε πραγματικό χρόνο.

Σχήμα 2.8: ARTennis

2.7.5 Google Glasses

Μια πολύ σημαντική δημιουργία ήταν τα Google Glasses (σχήμα 2.10) τα οποία βγήκαν για πρώτη φορά στην αγορά το 2014 και ονομάστηκαν και ως "Wearable Computer" (υπολογιστής που φοριέται). Ο χρήστης είχε πολλές δυνατότητες χάρη στην επιφάνεια αφής που υπήρχε στο δεξί μέρος των γυαλιών, στην ίδια πλευρά που υπήρχε και η οθόνη με αποτέλεσμα τον εύκολο έλεγχο της συσκευής. Ο χρήστης μπορούσε να δει διάφορες λειτουργίες στην διεπαφή που υπήρχε μπροστά του όπως τον καιρό ή άλλα γεγονότα. Ακόμη υπήρχε και κάμερα με την οποία ο χρήστης είχε την δυνατότητα να καταγράφει υψηλής ανάλυσης βίντεο και φωτογραφίες.

Σχήμα 2.9: Google Glasses

2.7.6 Pokemon Go

Το Pokemon ξεκίνησε ως παιχνίδι του Game Boy και δημιουργήθηκε το 1996 από τον Satoshi Tajiri. Η λέξη Pokemon είναι συντομογραφία για το "Pocket Monster" (δηλαδή τέρας τσέπης). Το παιχνίδι αποτελείται από εικονικά τέρατα τα οποία ονομάζονται Pokemon και εκπαιδεύονται από τους εικονικούς ανθρώπους (Pokemon Trainers) για να πολεμήσουν ο ένας τον άλλο, οι Pokemon Trainers κρατούν τα Pokemon σε Pokeballs, δηλαδή μπάλες που ανοίγουν και μέσα σε αυτά ζούνε τα Pokemon. Το Pokemon έγινε πολύ γνωστό και πλέον ανήκει στην ιαπωνική εταιρεία Nintendo η οποία επέκτεινε το Pokemon στην τηλεόραση ως κινούμενα σχέδια και στα κόμικς.

Μετά από 20 χρόνια (2016) το Pokemon έκανε μια μεγάλη επιστροφή με το παιχνίδι επαυξημένης πραγματικότητας το οποίο χρησιμοποιείται μέσω κινητού και βασίζεται στην τοποθεσία του χρήστη. Δημιουργώντας τον λογαριασμό και επιλέγοντας το όνομα του παίκτη, ο κάθε χρήστης πλέον γίνεται Pokemon Trainer, και καθώς προχωράει στην πραγματικότητα, προχωράει και στο περιβάλλον το οποίο με βάση τους δρόμους είναι ένα εικονικό αντίγραφο της πραγματικότητας (σχήμα 2.9). Όσο προχωράει ο χρήστης συναντάει περισσότερα Pokemon, πατώντας τα μπορεί να πετάξει μια Pokeball πάνω στο Pokemon για να το πιάσει. Επιπλέον, στο Pokemon Go υπάρχουν και κάποια μέρη που ονομάζονται Pokestops και οι χρήστες μπορούν να πάνε σε εκείνο το μέρος για να πάρουν δωρεάν προϊόντα του παιχνιδιού, ενώ παράλληλα υπάρχουν και κάποιοι κακοί (ομάδα πύραυλος) που υπάρχουν τυχαία στο περιβάλλον και πολεμούν τους Pokemon Trainers. Ακόμη υπάρχουν κάποιοι χώροι που ονομάζονται Gym (γυμναστήριο), οι χρήστες τοποθετούν ένα Pokemon για να κατακτήσουν το γυμναστήριο και να πάρουν επιπλέον προϊόντα, καθώς παράλληλα προσπαθούν να κατακτήσουν τα γυμναστήρια και άλλοι Pokemon Trainers, κάνοντας τους παίκτες να πολεμάνε μεταξύ τους. Μια άλλη ενέργεια που χρειάζεται πολλαπλούς παίκτες είναι τα Raids, τα οποία ουσιαστικά είναι κάποια μεγάλα τέρατα και απαιτούν ολόκληρες ομάδες για να τα πολεμήσουν.

Σε μια έρευνα που καταγράφηκε[15], οι χρήστες του Pokemon Go που δεν ήταν πολύ ενεργοί παρουσίασαν μεγάλες αυξήσεις δραστηριότητας. Το Pokemon Go οδήγησε σε αρκετά άτομα μεγάλη αύξηση στην σωματική δραστηριότητα σε σύγκριση με κάποιες άλλες συσκευές (όπως για παράδειγμα τα Wristbands που φοριούνται στον καρπό) που μετρούν τα βήματα και χρησιμοποιούνται για περπάτημα.

Σχήμα 2.10: Pokemon Go

2.7.7 Microsoft Hololens 2

Η τελευταία δημιουργία έγινε από την Microsoft η οποία ανακοίνωσε την κυκλοφορία του Hololens 2 το 2019. Το Hololens 2 είναι ένα ανεξάρτητο ολογραφικό Headset με διαφανείς φακούς που δίνει μια εμπειρία επαυξημένης πραγματικότητας μέσω των εφαρμογών. Στόχος του Hololens 2 είναι να αυξήσει την ακρίβεια και την απόδοση του χρήστη καθώς χρησιμοποιείται στις επιχειρήσεις. Με το Hololens 2 ο χρήστης μπορεί να βιώσει τρισδιάστατες ολογραφικές εικόνες σαν να είναι μέρος του περιβάλλοντος, να χρησιμοποιήσει χώρους όπως την κουζίνα ως επιφάνεια εργασίας, και να μεταδώσει βίντεο στον τοίχο ή να δημιουργήσει ένα πύργο από εικονικά τουβλάκια στο τραπέζι.

Σχήμα 2.11 Hololens 2

2.8 Επίλογος

Η συνέχεια πραγματικότητας-εικονικότητας καθορίζει πόσο πραγματική ή εικονική είναι μια τεχνολογία ανάλογα με την περιοχή που βρίσκεται πάνω στην γραμμή. Πέρα από το πραγματικό περιβάλλον υπάρχει η επαυξημένη πραγματικότητα που επαυξάνει το πραγματικό περιβάλλον, στην συνέχεια είναι η επαυξημένη εικονικότητα η οποία επαυξάνει το εικονικό περιβάλλον και τέλος η εικονική πραγματικότητα αποτελείται μόνο από εικονικά στοιχεία. Η επαυξημένη πραγματικότητα υπήρχε ως σκέψη από τα παλιά χρόνια, ενώ το 1968 δημιουργήθηκε η πρώτη συσκευή επαυξημένης πραγματικότητας. Εξίσου σημαντικές εφευρέσεις ήταν το Videospace στο οποίο οι χρήστες αλληλεπιδρούσαν με εικονικά αντικείμενα και το ARTennis που δύο χρήστες μπορούσαν να παίξουν εικονικό Tennis. Πλέον η συγκεκριμένη τεχνολογία έχει προχωρήσει αρκετά, αυτό φαίνεται στα νεότερα προϊόντα που δημιουργήθηκαν, το Google Glass και το Hololens 2, που είναι Headset με τα οποία ο χρήστης βλέπει ολογράμματα και χρησιμοποιεί εικονικές λειτουργίες.

Κεφάλαιο 3ο: Μηχανή παιχνιδιού Unity

3.1 Εισαγωγή

Η εφαρμογή έχει υλοποιηθεί μέσω της μηχανής παιχνιδιού (Game Engine) Unity. Στο κεφάλαιο αναφέρονται κάποιες βασικές έννοιες που είναι απαραίτητες για την κατανόηση των λειτουργιών που παρέχονται από το Unity. Ανοίγοντας το Unity, εμφανίζονται κάποια παράθυρα-εργαλεία με τα οποία ο προγραμματιστής θα διαχειριστεί το περιβάλλον και τις λειτουργίες του. Τα GameObjects (αντικείμενα) από μόνα τους δεν προσφέρουν πολλά, όμως μπορούμε να τοποθετήσουμε κάποια Components (μεμονωμένες λειτουργίες) με τα οποία το αντικείμενο θα αποκτήσει κάποιες ιδιότητες. Τα Render Pipelines που προσφέρει το Unity εκτελούν μια σειρά από εντολές για να προβάλουν το σκηνικό. Τέλος, στο κεφάλαιο αναφέρεται το SDK (ένα σύνολο εργαλείων) Vuforia Engine με το οποίο δημιουργήθηκε η εφαρμογή μέσω του Unity.

3.2 Βασικές Έννοιες

3.2.1 GameObjects

Οποιοδήποτε αντικείμενο υπάρχει στην σκηνή ονομάζεται GameObject. Τα GameObjects από μόνα τους δεν προσφέρουν πολλά, αλλά μπορούμε να δώσουμε χαρακτηριστικά (που ονομάζονται Components, δηλαδή μεμονωμένες λειτουργίες που δίνουν ιδιότητες στα GameObjects) σε αυτά, έτσι ώστε να έχουν παραπάνω δυνατότητες όπως πηγή φωτός ή βαρύτητα.

3.2.2 Prefabs

Το σύστημα του Prefab επιτρέπει τους προγραμματιστές να αποθηκεύσουν ένα GameObject με όλες τις ιδιότητες και τις ρυθμίσεις που έχει ως επαναχρησιμοποιήσιμο στοιχείο. Οταν ο προγραμματιστής θέλει να επαναχρησιμοποιήσει κάποιο GameObject που έχει διαμορφώσει με συγκεκριμένο τρόπο, μπορεί να το μετατρέψει σε Prefab. Είναι καλύτερη η δημιουργία prefab από την "αντιγραφή επικόλληση" επειδή το Prefab κρατάει όλα τα αντίγραφα συγχρονισμένα, δηλαδή με τις ίδιες ρυθμίσεις, αρκεί καθώς αλλάζουν οι ρυθμίσεις ενός αντιγράφου να πατηθεί το κουμπί που μεταφέρει τις αλλαγές στο Prefab. Ακόμη υπάρχει και η δυνατότητα τοποθέτησης prefabs μέσα σε άλλα prefab, για ευκολότερη επεξεργασία σε πολλαπλά επίπεδα.

3.2.3 Assets και Features

Asset ονομάζεται η αναπαράσταση οποιουδήποτε στοιχείου που μπορεί να χρησιμοποιηθεί στην εφαρμογή. Ένα Asset μπορεί να προέρχεται από ένα αρχείο που δημιουργήθηκε σε κάποια άλλη εφαρμογή εκτός του Unity, όπως ένα 3D μοντέλο ή ένα αρχείο ήχου.

Το Unity παρέχει το Unity Asset Store δηλαδή την δική του ιστοσελίδα από την οποία πωλούνται και αγοράζονται Assets. Τα Assets που αγοράζονται από εκείνη την σελίδα μεταφέρονται στην εφαρμογή και κατεβάζονται ως πακέτα, και μπορούν να χρησιμοποιηθούν όπως επιθυμεί ο προγραμματιστής. Features θεωρούνται τα Assets που δεν έχουν κάποια μορφή και δεν είναι GameObjects, όπως για παράδειγμα βιβλιοθήκες κώδικα.

3.2.4 Scenes

Scene είναι ένα περιβάλλον στο χώρο που έχει τα δικά του GameObjects, είναι το μέρος που εργάζονται οι προγραμματιστές στο Unity, και περιέχουν ένα μέρος ή ένα σύνολο μιάς εφαρμογής. Μια εφαρμογή μπορεί να έχει πολλα Scenes, το καθένα με διαφορετικά περιβάλλοντα, διακοσμήσεις και χαρακτήρες. Ο παίκτης μπορεί να ανακατευθυνθεί σε διαφορετικό Scene μέσω Script (κώδικα).

3.2.5 Canvas

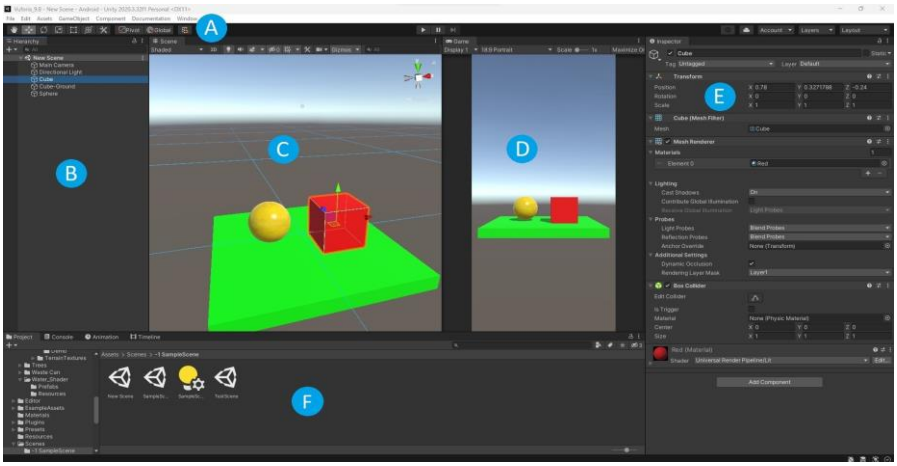
Όλα τα στοιχεία διεπαφής χρήστη όπως κουμπιά και 2D Texts πρέπει ως προς την ιεραρχία να είναι παιδιά του Canvas για να εμφανίζονται και να λειτουργήσουν σωστά. Ο Canvas είναι ένα GameObject με χαρακτηριστικό (Component) Canvas. Αν δεν υπάρχει κάποιος Canvas στην σκηνή, το Unity δημιουργεί έναν αυτόματα. Μέσα στο Scene, ο Canvas εμφανίζεται ως ορθογώνιο και συμβολίζει την οθόνη του χρήστη και χάρη σε αυτό η τοποθέτηση στοιχείων διεπαφής χρήστη γίνεται πολύ πιό εύκολη καθώς είναι ορατή.

3.2.6 Terrain

Το Terrain είναι ένα GameObject με χαρακτηριστικο (Component) Terrain και είναι η λύση στην δημιουργία τοπίων όπως βουνά και πεδιάδες. Χάρη στο ενσωματωμένο σύνολο λειτουργιών Terrain που παρέχει το Unity, δίνεται η δυνατότητα προσαρμογής ύψους και αλλαγή χρώματος με την χρήση βούρτσας (Brush). Ακόμη υπάρχει δυνατότητα ενσωμάτωσης γρασιδιού, δέντρων ή άλλων αντικειμένων που θα ανήκουν στο Terrain και δεν είναι πραγματικά GameObjects με τον ίδιο τρόπο όπως και με τις άλλες ρυθμίσεις του Terrain, δηλαδή με βούρτσα. Τέλος υπάρχουν οι γενικές ρυθμίσεις που αλλάζουν πιό περίπλοκες ρυθμίσεις όπως το μέγεθος του Terrain ή τα συνολικά σημεία που υπάρχουν και μπορούν να παραμορφωθούν.

3.3 User Interface

Η μηχανή Unity παρέχει μία διεπαφή χρήστη (User Interface) που αποτελείται από κάποια χρήσιμα παράθυρα τα οποία επιτρέπουν στους χρήστες να περιηγηθούν στα εργαλεία και τις δυνατότητες της με σκοπό την δημιουργία μιας εφαρμογής[1]. Παρακάτω επισημαίνονται λεπτομερώς μερικά από τα πιό σημαντικά παράθυρα επεξεργασίας και οι χρήσεις τους με βάση τις αριθμήσεις (A-F) του σχήματος 3.3.1 και άλλα σημαντικά παράθυρα που μπορούν να ανοίξουν από την ρύθμιση Window.



Σχήμα 3.3.1: User Interface

3.3.1 Toolbar

Η γραμμή εργαλείων (Toolbar) υπάρχει στο πάνω μέρος του προγράμματος όπως φαίνεται στο σχήμα 3.3.1 με αρίθμηση “A” και είναι το μόνο παράθυρο που δεν μπορεί να αναδιαταχθεί.

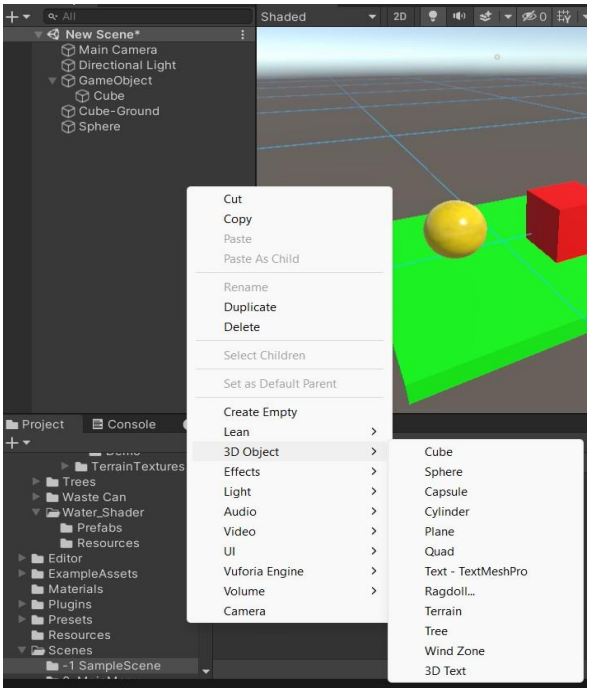
Τα εργαλεία που περιέχει στα αριστερά της οθόνης αφορούν την επεξεργασία αντικειμένων και την μετακίνηση στο χώρο. Ειδικότερα για τα σημαντικότερα από αυτά, το πρώτο κουμπί ονομάζεται “Hand Tool” και επιλέγοντάς το (ή πατώντας το πλήκτρο συντόμευσης “Q”) μπορούμε να κινηθούμε στο χώρο προς 4 κατευθύνσεις (πάνω, κάτω, δεξιά και αριστερά) με βάση το που έχουμε γυρισμένη την κάμερα της προβολής σκηνής (Scene view). Το επόμενο κουμπί ονομάζεται “Move Tool” και επιλέγοντάς το (ή πατώντας το πλήκτρο συντόμευσης “W”) εμφανίζονται 3 βελάκια (ένα για κάθε διάσταση) στην μέση ενός επιλεγμένου αντικειμένου στο Scene View τα οποία σείροντάς τα μετακινείται το αντικείμενο ανάλογα με την κατεύθυνση του ποντικιού και το βέλος που επιλέξαμε. Αντίστοιχα έχουμε το κουμπί “Rotate Tool” με πλήκτρο συντόμευσης “E” που μας επιτρέπει να περιστρέψουμε το αντικείμενο στο χώρο και το “Scale Tool” (που έχει πλήκτρο συντόμευσης “R”) με το οποίο μπορεί ένα αντικείμενο να μεγαλώσει ή να μικρύνει προς οποιονδήποτε άξονα.

Στο κέντρο υπάρχουν 3 κουμπιά (play, pause και step) τα οποία αφορούν την έναρξη, την παύση και την συνέχεια ανά πλαίσιο (frame) στο παράθυρο “προβολή παιχνιδιού” (Game View) και χρησιμοποιούνται κυρίως για εντοπισμό σφαλμάτων και δοκιμή του παιχνιδιού. Τα συγκεκριμένα κουμπιά δεν χρησιμοποιούνται σε παιχνίδια επαυξημένης πραγματικότητας καθώς η διαδικασία αυτή γίνεται μέσω του κινητού.

Τέλος στα δεξιά υπάρχουν κουμπιά τα οποία αφορούν γενικότερες διαδικασίες όπως σύνδεση σε distributed version control συστήματα (όπως το Git) για παράλληλη εργασία, επεξεργασία του λογαριασμού και άλλα σχετικά με την εμφάνιση κάποιων στρωμάτων (Layers) στον κόσμο.

3.3.2 Hierarchy

Το παράθυρο ιεραρχία (Hierarchy) υπάρχει ως προεπιλογή στα αριστερά του UI όπως φαίνεται στο σχήμα 3.3.1 με αρίθμηση “B”. Εκεί υπάρχουν όλα τα αντικείμενα όπως μοντέλα, κάμερες ή άλλα αντικείμενα τα οποία βρίσκονται στην φορτωμένη σκηνή και όταν διαγράφεται ή προστίθεται κάποιο αντικείμενο στο χώρο, τότε διαγράφεται ή προστίθεται αντίστοιχα από την ιεραρχία.



Σχήμα 3.3.2: Δημιουργία Αντικειμένου

Στο συγκεκριμένο παράθυρο μπορούν να δημιουργηθούν απλά 3D αντικείμενα (μερικά από τα οποία είναι: κύβος, κύλινδρος, κάψουλα, σφαίρα), φωτισμός, σύστημα σωματιδίων (particle System) ή άλλα αντικείμενα όπως ένα κενό αντικείμενο (empty - που δεν έχει χαρακτηριστικά(Components)) πατώντας δεξί κλικ σε κάποιο κενό χώρο του παραθύρου όπως φαίνεται στο σχήμα 3.3.2. Δημιουργώντας το αντικείμενο στην ιεραρχία, εμφανίζεται και στην προβολή σκηνής (Scene View) σε κάποιο μέρος

του κόσμου συνήθως στην συντεταγμένη (0,0,0) του κόσμου εκτός και αν δημιουργηθεί ως παιδί (child) κάποιου άλλου αντικειμένου (GameObject) κληρονομώντας την τοποθεσία του στον χώρο.

Επίσης υπάρχει η δυνατότητα ταξινόμησης και ομαδοποίησης των GameObjects. Στην περίπτωση που έχουν φορτώσει δύο σκηνές στον ίδιο χρόνο, η κάθε μία εμφανίζεται στην ιεραρχία με τα δικά της GameObjects ως παιδιά (childs) της κάθε σκηνής. Η ομαδοποίηση των GameObjects μπορεί να γίνει μέσω της έννοιας γονέα-παιδιού που χρησιμοποιείται στην ιεραρχία, δηλαδή μπορεί ένα GameObject να γίνει child ενός άλλου GameObject το οποίο θα ονομάζεται γονέας (parent) του πρώτου. Έτσι το child GameObject κληρονομεί την κίνηση και την περιστροφή του γονικού GameObject.

3.3.3 Scene View

Η προβολή σκηνής (ή αλλιώς Scene View) υπάρχει στο σχήμα 3.3.1 με αρίθμηση “C” και είναι ένα από τα πιό σημαντικά παράθυρα καθώς με αυτό προβάλλεται ο κόσμος που δημιουργείται. Στην προβολή σκηνής υπάρχει η δυνατότητα στον χρήστη να δημιουργήσει ή να επιλέξει κάποιο ήδη υπάρχον αντικείμενο του χώρου και να το τροποποιήσει με την βοήθεια του Toolbar ή του παραθύρου Inspector. Η επεξεργασία των αντικειμένων και η μετακίνηση κάμερας της προβολή σκηνής είναι από τις πρώτες ενέργειες που πρέπει να μάθει κάποιος χρήστης για να μπορέσει να χρησιμοποιήσει και να εργαστεί στο Unity. Στο πάνω μέρος του παραθύρου υπάρχουν κάποια κουμπιά που ρυθμίζουν την φωτεινότητα, τον ήχο, την ταχύτητα της κάμερας της προβολής σκηνής και γενικότερα τι ακριβώς θα φαίνεται στη σκηνή από τα αντικείμενα που αφορούν το συγκεκριμένο παράθυρο.

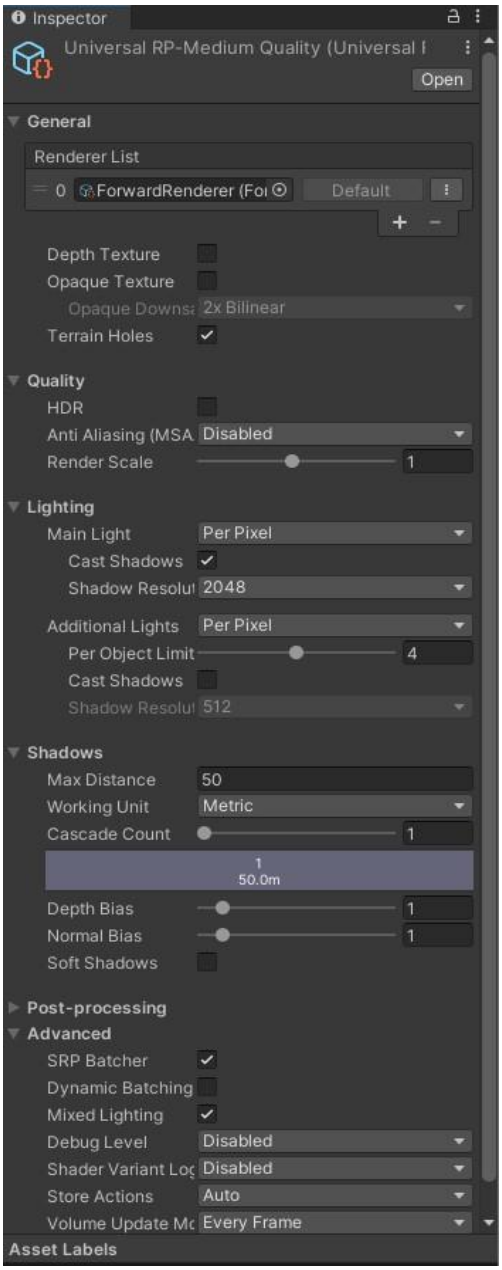
3.3.4 Game View

Η προβολή παιχνιδιού (Game View) συνήθως βρίσκεται δεξιά του Scene View όπως βλέπουμε στο σχήμα 3.3.1 με αρίθμηση “D”. Στο σκηνικό πρέπει να υπάρχει μία ή παραπάνω κάμερες έτσι ώστε ο παίκτης να μπορεί να βλέπει μέσω αυτών όταν χρησιμοποιεί την εφαρμογή. Το Game View προβάλλει το σκηνικό μέσα από την κάμερα που προσθέσαμε σε αυτό και άρα αντιπροσωπεύει το τελικό δημοσιευμένο παιχνίδι, αρκεί να πατήσουμε το κουμπί “Play” στο κέντρο του Toolbar. Το Game View δεν χρησιμοποιείται αρκετά σε παιχνίδια επαυξημένης πραγματικότητας, η διαδικασία προβολής γίνεται μέσω του κινητού λόγω του ότι η εικονική κάμερα μετακινείται μέσω του κινητού του χρήστη και δεν χρειάζεται να μετακινηθεί την εικονική κάμερα μέσα στο σκηνικό. Παρόλα αυτά είναι χρήσιμο στο ότι με αυτό μπορούμε να δούμε πως θα εμφανίζεται στο κινητό το 2D UI το οποίο περιέχει διάφορα στοιχεία διεπαφής χρήστη (UI elements) όπως κουμπιά και Texts.

Στό πάνω μέρος υπάρχουν κάποια κουμπιά τα οποία αφορούν ποιά κάμερα θα εμφανίζεται στο Game View, την αναλογία απεικόνισης (Aspect Ratio) δηλαδή την ανάλυση που θα έχει το παιχνίδι ανάλογα με τον τύπο παιχνιδιού και τις συσκευές που το χρησιμοποιούν και άλλα σχετικά με το συγκεκριμένο παράθυρο.

3.3.5 Inspector

Το παράθυρο Inspector βρίσκεται ως προεπιλογή στα δεξιά της οθόνης (όπως φαίνεται στο σχήμα 3.3.1 με αρίθμηση "E") και χρησιμοποιείται για προβολή και επεξεργασία ρυθμίσεων για σχεδόν τα πάντα όπως τις ρυθμίσεις των χαρακτηριστικών(Components) των GameObjects και των αρχείων του Unity που βοηθούν στην λειτουργία του παιχνιδιού. Για να επεξεργαστούμε ένα αρχείο ή GameObject στον Inspector αρκεί να το επιλέξουμε είτε από το παράθυρο έργου (Project Window) αν είναι αρχείο είτε από την ιεραρχία αν είναι GameObject.



Σχήμα 3.3.3: Ρυθμίσεις αρχείου του Unity

Κάθε τύπος αρχείου έχει τις δικές του ρυθμίσεις, για παράδειγμα επιλέγοντας ένα GameObject εμφανίζονται όλα τα Components που έχει όπως το Transform που είναι απαραίτητο component και μας δίνει δεδομένα σχετικά με την τοποθεσία του GameObject στον κόσμο. Από την άλλη μεριά, επιλέγοντας κάποιο βασικό αρχείο ρύθμισης του παιχνιδιού όπως το αρχείο "Universal RP-Medium Quality" του σχήματος 3.3.3 που αντιπροσωπεύει προεπιλεγμένες ρυθμίσεις σχετικά με την ποιότητα των αντικειμένων, του φωτισμού, των σκιών και άλλων χαρακτηριστικών.

Ακόμη, υπάρχει η δυνατότητα επιλογής και επεξεργασίας πολλών αρχείων ή GameObjects ταυτόχρονα. Επιλέγοντάς τα, στον Inspector εμφανίζονται τα κοινά τους Components, και μέσα σε αυτά οι κοινές τους τιμές. Όταν οι τιμές στα ίδια πεδία είναι διαφορετικές, τότε εμφανίζεται ένα σύμβολο "-" και μπορούμε να τα αλλάξουμε βάζοντας μία τιμή που θα ισχύει για όλα τα επιλεγμένα GameObjects ή πατώντας δεξί κλικ αριστερά από τα πεδία τιμών, μας δίνεται η δυνατότητα να βάλουμε την τιμή του πεδίου ενός από τα επιλεγμένα GameObjects και στα υπόλοιπα που έχουμε επιλέξει.

3.3.6 Project Window

Το Project Window ή αλλιώς παράθυρο έργου εμφανίζεται ως προεπιλογή στο κάτω μέρος της εφαρμογής όπως βλέπουμε στο σχήμα 3.3.1 με αρίθμηση "F" και αντιπροσωπεύει τον εξερευνητή αρχείων (File explorer) του Unity ξεκινώντας από κάποιο φάκελο που ονομάζεται "Assets", όπου μέσα σε αυτό τον φάκελο ή σε κάποιο υποφάκελο του εμφανίζονται όλα τα αρχεία που σχετίζονται με το παιχνίδι.

Στο αριστερό μέρος εμφανίζεται η δομή φακέλων σε ιεραρχική λίστα. Με την επιλογή ενός φακέλου εμφανίζονται τα περιεχόμενα του στο δεξί μέρος του παραθύρου όπου και εμφανίζονται τα αρχεία με εικονίδια ανάλογα με τον τύπο τους όπως για παράδειγμα C# Scripts, υποφάκελοι και prefabs. Στο πάνω μέρος του παραθύρου υπάρχουν κουμπιά και ένα text field που αφορούν την αναζήτηση στους φακέλους.

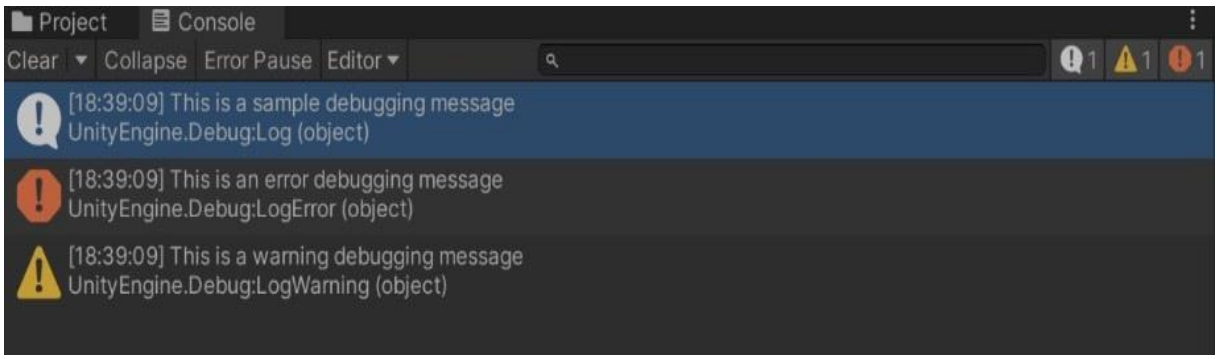
3.3.7 Console

Το παράθυρο Console (σχήμα 3.3.4) είναι το ιδανικό πλαίσιο για αποσφαλμάτωση (debugging) καθώς σε αυτό εμφανίζονται όλα τα σφάλματα και οι προειδοποιήσεις που δημιουργεί το πρόγραμμα με σκοπό να μας προειδοποιήσει ότι κάτι μπορεί να προκαλέσει πρόβλημα στην εφαρμογή μας.

Υπάρχουν τρία είδη μηνυμάτων, ένα από αυτά είναι το μήνυμα σφάλματος (error message) το οποίο συμβολίζεται με ένα κόκκινο οκτάγωνο και συνήθως όταν υπάρχει στο console, δεν μπορεί να τρέξει η εφαρμογή πριν διορθωθεί (όπως για παράδειγμα σε συντακτικά λάθη σε κάποιο script) ή εμφανίζεται όταν υπάρξει κάποιο λάθος κατά την διάρκεια που τρέχει (όπως για παράδειγμα στην προσπάθειά ενός script να διαιρέσει με το μηδέν) και δεν καταρρέει χάρη στην αυτόματη διαχείριση λαθών του Unity. Το Δεύτερο είδος είναι το μήνυμα προειδοποίησης (Warning Message), συμβολίζεται με ένα κίτρινο τρίγωνο και συνήθως είναι απλά προειδοποιητικά μηνύματα τα οποία δεν προκαλούν κάποιο πρόβλημα με την εκτέλεση της εφαρμογής. Το Τρίτο είδος μηνυμάτων στο console είναι απλά μηνύματα, συμβολίζονται με μία άσπρη επεξήγηση και τα συναντάμε μέσω scripts για λόγους debugging.

Για να γίνει ακόμα πιο εύκολο το debugging, η Unity έχει δημιουργήσει μια κλάση “Debug” με αρκετές χρήσιμες μεθόδους (methods) που μπορούμε να καλέσουμε μέσω των Scripts. Για παράδειγμα όταν ένα script που εκτελείται φτάσει στην γραμμή “Debug.Log(“Hello World”);” θα εμφανιστεί στο console η φράση “hello world”, ή μπορούμε να εμφανίσουμε κάτι πιό χρήσιμο όπως την τοποθεσία ενός αντικειμένου στο χώρο καλώντας την κατάλληλη μέθοδο.

Τέλος, στο πάνω μέρος υπάρχει ένα Search bar που μας επιτρέπει να ψάξουμε λέξεις ή φράσεις για να βρούμε κάποιο συγκεκριμένο μήνυμα, και κάποια κουμπιά που αφορούν τον καθαρισμό του console, την ομαδοποίηση ίδιων μηνυμάτων, την παύση του παιχνιδιού στην εύρεση λάθους, και την εμφάνιση ή μη του κάθε τύπου μηνυμάτων.



Σχήμα 3.3.4: Console Window

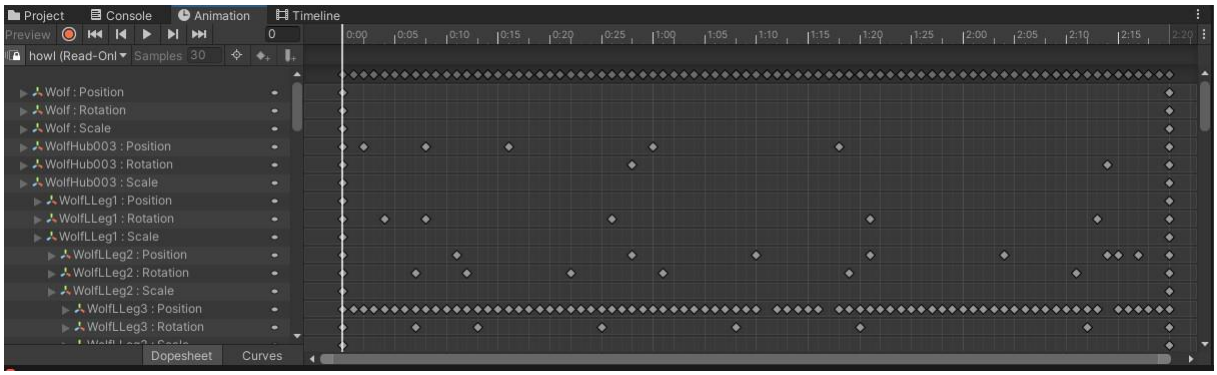
3.3.8 Animation

Το Animation Window (σχήμα 3.3.5) έχει σχεδιαστεί για την δημιουργία και τροποποίηση κλιπ κινούμενων εικόνων (Animation Clip) και θεωρείται μια απλή και εναλλακτική λύση των εξωτερικών προγραμμάτων για 3D animation. Μέσα στα Animation Clips πέρα από κίνηση αντικειμένων υπάρχει και η δυνατότητα αλλαγής τιμών στα πεδία του Inspector όπως για παράδειγμα στα Materials (υλικό αντικειμένων). Το παράθυρο χωρίζεται σε δύο μέρη (αριστερά και δεξιά).

Στα αριστερά υπάρχουν οι ονομασίες των αντικειμένων που γίνονται Animated, επίσης πάνω υπάρχουν κάποια κουμπιά τα οποία χρησιμοποιούνται για να αρχίσει ή κλείσει η καταγραφή του animation και ως προεπισκόπηση ή προβολή των εικόνων του Animation.

Στο δεξί μέρος υπάρχει ένας χώρος ο οποίος οριζόντια μετριέται σε Frames (εικόνες), και εκεί εμφανίζονται τα βασικά καρέ (Keyframes) τα οποία προσδιορίζουν μία τιμή σε κάποιο πεδίο των Animated GameObjects και συμβολίζονται ώς άσπροι ρόμβοι. Εάν δεν υπάρχουν άλλα keyframes μεταξύ δύο Keyframes, τοτε δημιουργούνται αυτόματα Keyframes με ανάλογες τιμές σε Frames που δεν έχουν, έτσι ώστε η κίνηση να φαίνεται ομαλή. Στο πάνω μέρος φαίνεται να υπάρχουν κάποια Keyframes, αλλά στην πραγματικότητα αντιπροσωπεύουν όλα τα υπόλοιπα Keyframes στο ίδιο Frame, δηλαδή από κάτω τους.

Ακόμη, μέσω του animation event μπορούμε να καλέσουμε μεθόδους από κάποιο script που ανήκει στο αντικείμενο του animation και να ορίσουμε το frame στο οποίο επιθυμούμε να τρέξει. Επίσης μπορούμε να έχουμε πρόσβαση στο animation καλώντας μεθόδους του Unity μέσω script για να σταματήσουμε ή να αρχίσουμε το animation.

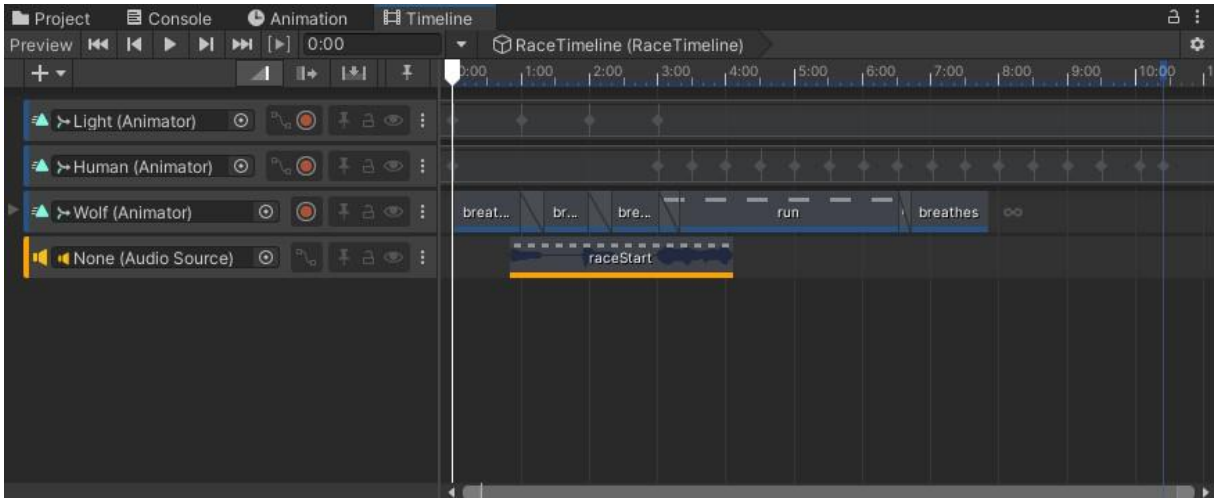


Σχήμα 3.3.5: Animation Window

3.3.9 Timeline

Στην περίπτωση που θέλουμε να δημιουργήσουμε κινηματογραφικό περιεχόμενο, αυτό που χρειαζόμαστε είναι το παράθυρο χρονοδιαγράμματος (Timeline Window) (σχήμα 3.3.6). Με αυτό το παράθυρο μπορούμε να συνδυάσουμε πολλά animations με διαφορετικά GameObjects, να βάλουμε ήχους σε συγκεκριμένα frames και να συνδυάσουμε δύο animations του ίδιου GameObject βάζοντας ένα μέρος από αυτά να εκτελούνται στον ίδιο χρόνο έτσι ώστε να αναμειχθούν και να γίνει πιο ομαλή η εναλλαγή του animation.

Ακόμη, πατώντας πάνω σε κάποιο animation μέσα στο Timeline, εμφανίζονται στον Inspector κάποιες ρυθμίσεις σχετικά με την ταχύτητα του animation και με την ομαλή εναλλαγή από ένα animation σε άλλο.



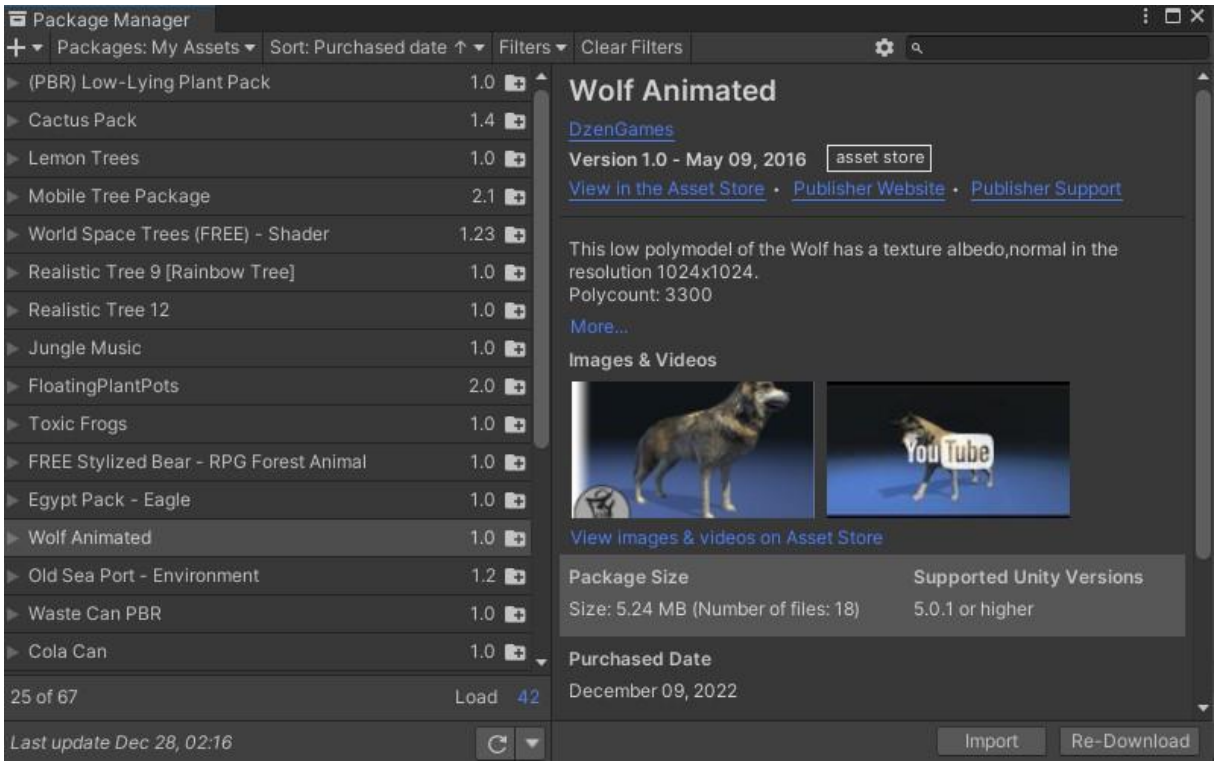
Σχήμα 3.3.6: Timeline Window

3.3.10 Package Manager

Πακέτο (Package) ονομάζεται ένας χώρος που αποθηκεύει διάφορα assets ή features (όπως βιβλιοθήκες κώδικα) του Unity ή άλλων ιδιοκτητών. Το παράθυρο Package Manager (σχήμα 3.3.7) είναι ο «συνδετικός κρίκος» του Unity με τα πακέτα, τα οποία μπορούμε να ενσωματώσουμε στην εφαρμογή μας.

Οταν στο Unity ανοίγει την εφαρμογή μας, φορτώνει ένα αρχείο που ονομάζεται “Project Manifest File” το οποίο αποθηκεύει πληροφορίες σχετικά με το ποιά πακέτα είναι εγκατεστημένα, και καθορίζει ποιά από αυτά πρέπει να ανακτηθούν και να φορτωθούν στην εφαρμογή. Στην συνέχεια το αρχείο στέλνει ένα αίτημα στον Package Registry Server (ο οποίος αποθηκεύει μεταδεδομένα και περιεχόμενα των πακέτων) για κάθε πακέτο που εμφανίζεται ως εξάρτηση (Dependency) στο Project Manifest File. Έπειτα ο Package Registry στέλνει τα ζητούμενα δεδομένα πίσω στον Package Manager ο οποίος εγκαθιστά αυτά τα πακέτα στην εφαρμογή.

Το Unity παρέχει την δική του ιστοσελίδα για αγορά και πώληση των Assets ή Features και ονομάζεται Unity Asset Store. Εάν επιθυμούμε να αγοράσουμε ένα Asset ή Feature, μετά την αγορά εμφανίζεται ένα κουμπί “Open in Unity”. Πατώντας το, η ιστοσελίδα μας ανακατευθύνει στο πακέτο που αγοράσαμε μέσα στο Package Manager, και από εκεί μπορούμε να το κατεβάσουμε και να το ενσωματώσουμε στην εφαρμογή μας έχοντας την επιλογή να διαλέξουμε ποιά ακριβώς αρχεία θα ενσωματωθούν στην περίπτωση που δεν επιθυμούμε να βάλουμε όλο το πακέτο στην εφαρμογή.



Σχήμα 3.3.7: Package Manager Window

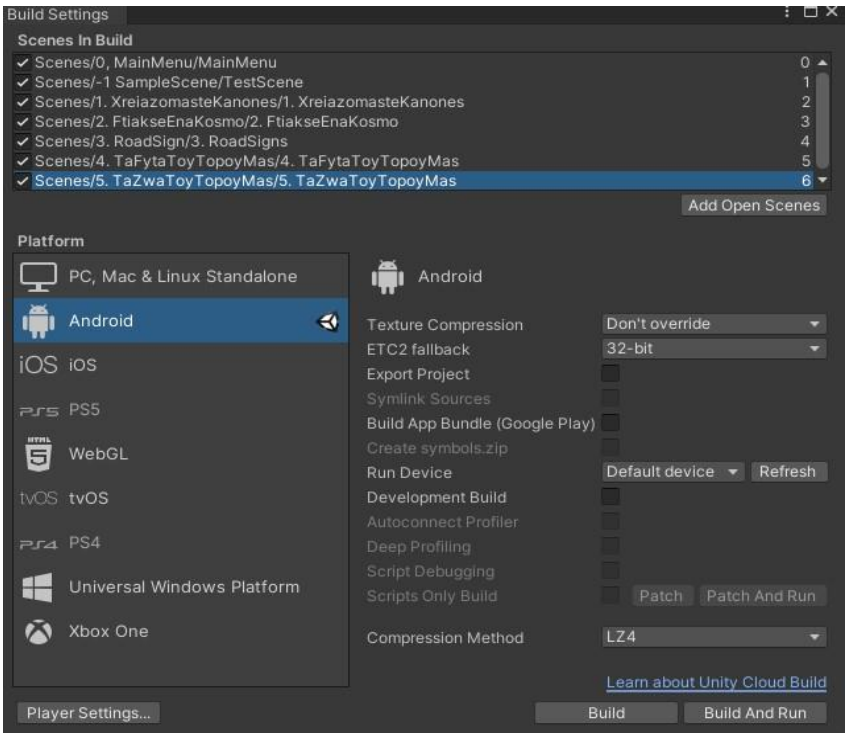
3.3.11 Build Settings

Το παράθυρο Build Settings (σχήμα 3.3.8) είναι υπεύθυνο για την δημιουργία της εφαρμογής από τον Unity Editor στην συσκευή, αλλά και ποιά Scenes υπάρχουν στην εφαρμογή και μπορούν να χρησιμοποιηθούν. Με την επιλογή πλατφόρμας στην περιοχή "Platform", καθορίζουμε τον τύπο της πλατφόρμας για την οποία θέλουμε να τρέξει η εφαρμογή μας. Στην συγκεκριμένη περίπτωση έχουμε επιλέξει "Android" επειδή η εφαρμογή επαυξημένης πραγματικότητας αφορά τα κινητά Android.

Για να φορτώνουν τα Scenes που θέλουμε στην εφαρμογή μας πρέπει πρώτα να τα τοποθετήσουμε στο Scenes In Build του παραθύρου Build Settings, προφανώς δεν είναι απαραίτητο να τοποθετήσουμε εκεί όλα τα Scenes που δημιουργούμε όπως αυτά που δεν χρησιμοποιούνται και έχουν δημιουργηθεί για Testing ή για κάποιον άλλο σκοπό. Δεξιά από τα Scenes που έχουμε τοποθετήσει παρατηρούμε κάποιους αριθμούς. Κάθε αριθμός είναι ένας Index (δείκτης) του κάθε Scene και χρησιμοποιείται σε Script καθώς μπορούμε να φορτώσουμε κάποιο Scene με τον Index του, για παράδειγμα η γραμμη "SceneManager.LoadScene(2);" θα φορτώσει το Scene με Index 2. Επίσης υπάρχει και η δυνατότητα να αλλάξουμε την σειρά των Scenes σε περίπτωση που επιθυμούμε να τα τοποθετήσουμε σε άλλη σειρά.

Για να κατεβάσουμε την εφαρμογή στο κινητό μας, πρέπει αρχικά να κατεβάσουμε την εφαρμογή "Unity Remote" απ΄το Google Store και μετά να συνδέσουμε το κινητό με τον υπολογιστή έχοντας επιτρέψει την μεταφορά αρχείων καθώς και την ρύθμιση του κινητού "USB debugging" που συνήθως βρίσκεται στις ρυθμίσεις προγραμματιστή (Developer Options). Έπειτα ελέγχουμε αν έχουμε βάλει όλα τα Scenes που χρειαζόμαστε μέσα στο Scenes In Build, και αν έχουμε επιλέξει την επιλογή Android στην περιοχή Platform. Αφού γίνουν τα παραπάνω, μπορούμε να πατήσουμε το κουμπί "Build And Run" που βρίσκεται κάτω δεξιά. Η εφαρμογή θα ανοίξει στο κινητό μόλις την

κατεβάσουμε και φορτώσει, δηλαδή μια διαδικασία ενός ή δύο λεπτών ανάλογα με το πόσο βαριά (σε χωρητικότητα) είναι η εφαρμογή μας.



Σχήμα 3.3.8: Build Settings Window

3.4 Components

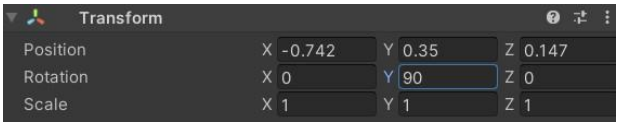
Για να δώσουμε λειτουργικότητα σε κάποιο GameObject, προσθέτουμε κάποια Components σε αυτό. Ως Components ορίζονται τα χαρακτηριστικά ενός αντικειμένου, δηλαδή μεμονωμένες λειτουργίες που δίνουν κάποιες ιδιότητες στα αντικείμενα όπως για παράδειγμα η μάζα ή η τοποθεσία στον χώρο.

Μπορούμε να προσθέσουμε Components επιλέγοντας αρχικά το GameObject στο οποίο θέλουμε να γίνει αυτή η ενέργεια και μέσω του παραθύρου Inspector στο οποίο βρίσκεται ένα κουμπί "Add Component". Πατώντας το εμφανίζεται ένα μικρό παράθυρο (σχήμα 3.4.1) το οποίο έχει μία ταξινομημένη λίστα με όλα τα Components που είναι διαθέσιμα για χρήση και προσθήκη στο GameObject. Στο πάνω μέρος βρίσκεται ένα Search Bar με το οποίο μπορούμε να αναζητήσουμε και να βρούμε το Component που χρειαζόμαστε. Για να διαγράψουμε κάποιο Component πατάμε δεξί κλικ πάνω στο component και πατάμε την επιλογή "Remove component". Κάθε Component μπορεί να υποστεί επεξεργασία αλλάζοντας τις τιμές του στα πεδία. Μερικά από τα βασικότερα Components που χρησιμοποιούνται είναι τα: Transform, Mesh Renderer, Box Collider, Rigidbody, Audio Source, Scripts και Materials.

Σχήμα 3.4.1: Add Component

3.4.1 Transform

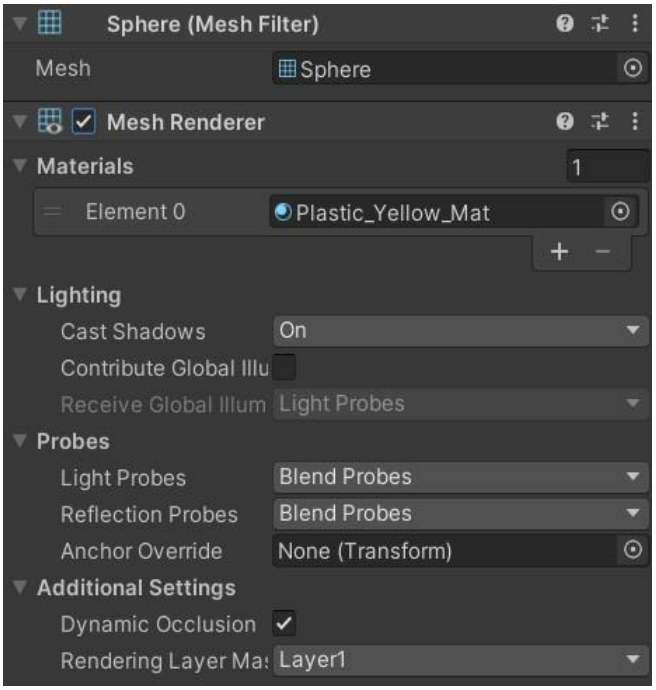
Το Transform (σχήμα 3.4.2) είναι το μόνο Component που υπάρχει από προεπιλογή σε όλα τα GameObjects και δεν μπορεί να διαγραφεί, επειδή ορίζει την τοποθεσία (Position), την περιστροφή (Rotation) και την κλιμάκωση (Scale), και χωρίς αυτά δεν μπορεί να υπάρξει στον χώρο. Οι τιμές των παραπάνω χαρακτηριστικών μπορούν να αλλάξουν αλλάζοντας τις τιμές των Fields για κάθε άξονα (X, Y, Z) ξεχωριστά.



Σχήμα 3.4.2: Transform Component

3.4.2 Mesh Filter και Mesh Renderer

Για να εμφανίζονται τα αντικείμενα στο χώρο, πρέπει να έχουν κάποια μορφή. Την δουλειά αυτή την αναλαμβάνει ο συνδυασμός των Components Mesh Filter και Mesh Renderer (βλέπε σχήμα 3.4.3).



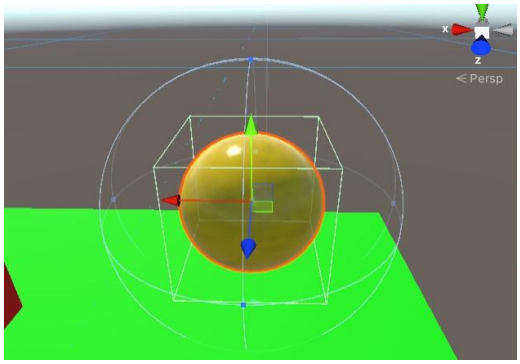
Σχήμα 3.4.3: Mesh Filter & Renderer

Το δεδομένο που παίρνει το Mesh Filter αναφέρεται σε ένα Mesh (πλέγμα που αποτελείται από τρίγωνα και σχηματίζει κάποιο σχήμα) έτσι ώστε το GameObject να πάρει κάποια μορφή στον χώρο. Δεν αρκεί όμως μόνο αυτό επειδή το αντικείμενο χρειάζεται κάποιο Material πάνω στο Mesh του. Γι'αυτό τον λόγο χρειάζεται το Mesh Renderer, δίνοντας του κάποιο Material ως δεδομένο μπορούμε να δούμε την τελική μορφή του αντικειμένου. Επιπλέον, το Mesh Renderer έχει κάποιες ρυθμίσεις οι οποίες ρυθμίζουν το πως θα φαίνεται το αντικείμενο σε κάποιες ειδικές περιπτώσεις, για παράδειγμα όταν χτυπάει φως στο αντικείμενο, το ίδιο να βγάζει σκιά ή όχι.

Στην περίπτωση που θέλουμε να χρησιμοποιήσουμε κάποιο παραμορφώσιμο πλέγμα, πρέπει αντί για Mesh Filter και Mesh Renderer να χρησιμοποιηθεί το Component "Skinned Mesh Renderer" το οποίο όπως και τα προηγούμενα Components απαιτεί κάποιο Mesh και Material. Για να πάρει κάποιο αντικείμενο το Component "Skinned Mesh Renderer", είναι υποχρεωτικό να υπάρχει κάποια κατασκευή από κόκκαλα στο αντικείμενο (γι'αυτό χρησιμοποιείται σε ανθρωπόμορφα ή ζώα μοντέλα) ή κάποιο Blendshape (που βοηθάει στην δημιουργία χαρακτηριστικών προσώπου).

3.4.3 Box Collider

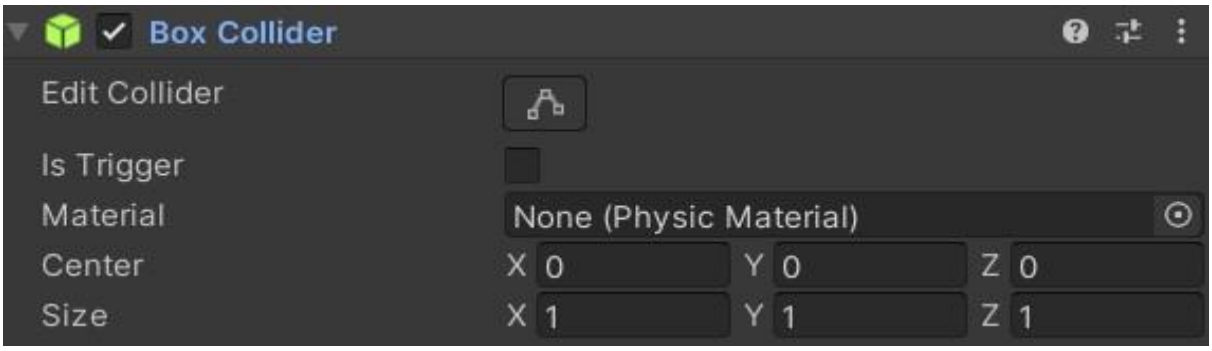
Η διαχείριση συγκρούσεων (Collisions) επιτυγχάνεται μέσω των Components Colliders, τα οποία καθορίζουν τα όρια των φυσικών συγκρούσεων και είναι αόρατα στο περιβάλλον. Το σχήμα του Collider δεν είναι απαραίτητο να έχει το ίδιο σχήμα με το Mesh, παρόλα αυτά υπάρχουν πολλά είδη Components για collider ανάλογα με το σχήμα που επιθυμούμε να ενσωματώσουμε στο GameObject. Το πιό συχνό Component είναι το Box Collider το οποίο εφαρμόζει ένα Collider με σχήμα κύβου στο GameObject (Σχήμα 3.4.4), αλλά υπάρχουν και άλλα Colliders όπως το Capsule Collider που παίρνει την μορφή κάψουλας, Sphere Collider που παίρνει την μορφή σφαίρας, και Mesh Collider που παίρνει την μορφή του Mesh.



Σχήμα 3.4.4: Box Collider

Το Component του κάθε Collider (σχήμα 3.4.5) περιέχει κάποιες ρυθμίσεις που μπορούμε να αλλάξουμε όπως την μετακίνηση και το μέγεθος του Collider σε σύγκριση με το Mesh, μια υποδοχή φυσικού υλικού (Physic material) που δίνει στο αντικείμενο επιπλέον δεδομένα σχετικά με τις συγκρούσεις που θα δέχεται, και ένα Boolean IsTrigger.

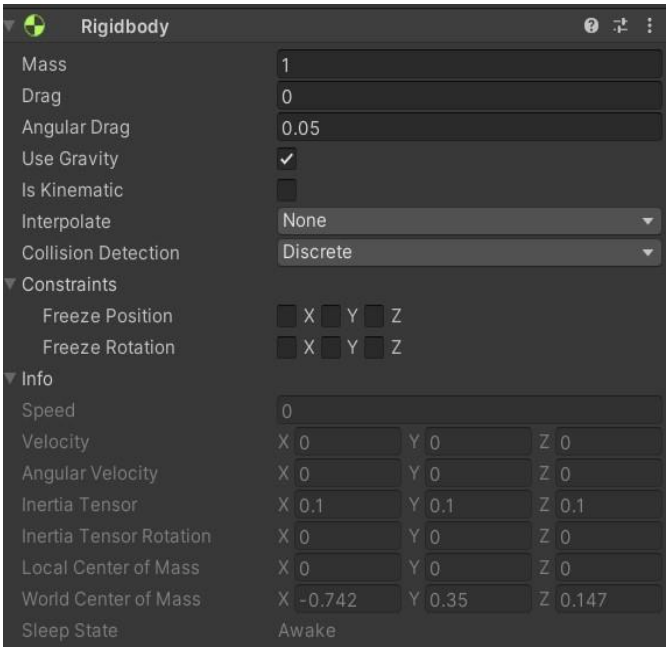
Όταν η τιμή του IsTrigger είναι true, διαγράφει τις ιδιότητες του Collider (δηλαδή δεν υπάρχει σύγκρουση με άλλα GameObjects γιατί θα περνάνε μέσα από αυτό) και χρησιμοποιείται σαν Trigger με την βοήθεια μιας void μεθόδου “OnTriggerEnter” με παράμετρο “other” τύπου Collider, η οποία (μέθοδος) ενεργοποιείται όταν κάποιο άλλο Collider ενός GameObject ακουμπήσει ή μπεί μέσα στο Trigger. Εάν εκείνη την ώρα που ενεργοποιηθεί το Trigger θέλουμε να μάθουμε πως ονομάζεται το GameObject που μπήκε μέσα στο Trigger αρκεί να αναφερθούμε στην παράμετρο “other” γράφοντας “other.transform.name”. Συνήθως το Trigger χρησιμοποιείται σε αντικείμενα που δεν έχουν Mesh (ώστε να μην φαίνονται) και εκτελούν κώδικα όπως ενεργοποίηση εμποδίων που δεν έχουν ενεργοποιηθεί στην αρχή του παιχνιδιού ή αυτόματη αποθήκευση (Autosave).



Σχήμα 3.4.5: Box Collider Component

3.4.4 Rigidbody

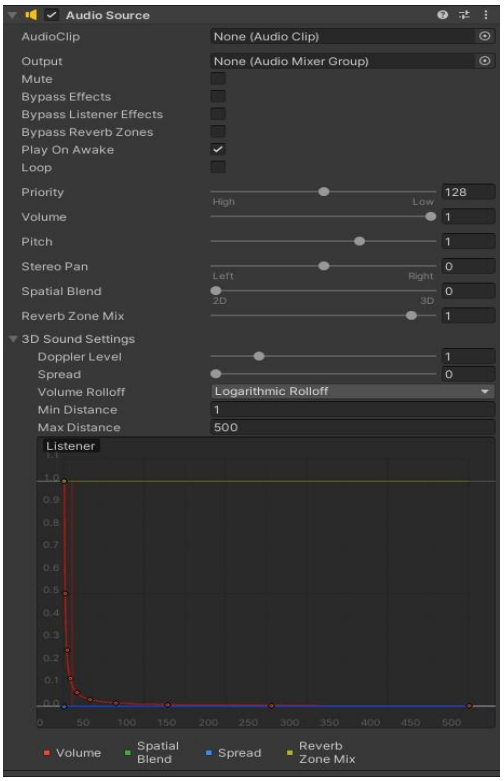
Ένα αντικείμενο χωρίς Rigidbody δεν δέχεται καμία κίνηση από το περιβάλλον του, ακόμη και αν συγκρουστεί με κάποιο άλλο αντικείμενο (έχοντας κάποιο Collider), θα λειτουργήσει ως τοίχος χωρίς κίνηση. Το Component Rigidbody (Σχήμα 3.4.6) επιτρέπει στο GameObject να δέχεται φυσικές κινήσεις όπως συγκρούσεις ή βαρύτητα. Στις ρυθμίσεις του μπορούμε να βάλουμε κάποια μάζα που θέλουμε να έχει το αντικείμενο, κάποια τιμή επιβράδυνσης σε φυσικές κινήσεις, να δέχεται βαρύτητα, να μην περιστρέφεται ή κινείται προς κάποιους άξονες. Τα Scripts (του Feature LeanTouch) που μας επιτρέπουν να σύρουμε ή να πατήσουμε με το δάχτυλό μας σε κάποιο 3D GameObject (όχι όμως σε 2D αντικείμενα που αποτελούν το User Interface) σε παιχνίδια επαυξημένης πραγματικότητας δίνουν μια φυσική κίνηση στο αντικείμενο, άρα για να μπορέσει να δεχτεί φυσικές κινήσεις είναι απαραίτητο να έχει RigidBody.



Σχήμα 3.4.6: Rigidbody Component

3.4.5 Audio-Source

Το Component Audio Source (σχήμα 3.4.7) είναι το «κλειδί» για αναπαραγωγή ήχων, και απαιτείται να υπάρχει ένα Component "Audio Listener" για την ακοή που συνήθως τοποθετείται στην κάμερα. Περιέχει ένα πεδίο υποδοχής αρχείου τύπου ήχου (για παράδειγμα .mp3 ή .wav) και αρκετές ρυθμίσεις σχετικά με τον ήχο και τον τρόπο που αναπαράγεται. Υπάρχει και η ρύθμιση "Spatial Blend" η οποία καθορίζει πόσο 2D ή 3D να είναι ο ήχος. Όταν ο ήχος είναι 2D τότε ακούγεται από όλα τα ηχεία το ίδιο και δεν υπάρχει καμία διαφορά στο που είναι ο παίκτης, Ενώ όταν είναι 3D, ακούγεται πιο δυνατά ή πιο σιγά ανάλογα με την τοποθεσία δηλαδή όσο πιο μακριά είμαστε από το Audio Source, τόσο λιγότερο θα ακούγεται προσομοιώνοντας την πραγματικότητα. Στο κάτω μέρος του Component υπάρχει μια συνάρτηση την οποία μπορούμε να επεξεργαστούμε και καθορίζει το ποσοστό ακοής του ήχου ανάλογα με την απόσταση μας απο το Audio Source. Ο 3D ήχος δεν παρέχει μόνο αλλαγές με βάση την τοποθεσία του παίκτη, αλλα και με βάση την περιστροφή, δηλαδή αν ο παίκτης είναι γυρισμένος έτσι ώστε το Audio Source να βρίσκεται στα δεξιά του, τότε το δεξί ηχείο θα ακούγεται παραπάνω από



Σχήμα 3.4.7: Audio-Source Component

το αριστερό, όπως γίνεται και στην πραγματικότητα με τα αυτιά μας.

Το Audio Source είναι προσβάσιμο μέσω Script για προσαρμογή ήχου, αναπαραγωγή (Play), Παύση (Pause) και διακοπή (Stop) και άλλες ρυθμίσεις. Επίσης μπορούν να αναπαραχθούν πολλαπλοί ήχοι σε ένα Audio Source χρησιμοποιώντας την μέθοδο "PlayOneShot". Σε περίπτωση που δεν έχουμε κάποιο Audio Source και θέλουμε να παράξουμε ήχο, μπορούμε να καλέσουμε την μέθοδο "PlayClipAtPoint" και του δίνουμε 2 παραμέτρους, το Audio Clip που θέλουμε να παίξει, και ενα Vector3 (δηλαδή μια τοποθεσία στον χώρο X,Y,Z) έτσι ώστε να παίξει το Audio Clip στη συγκεκριμένη συντεταγμένη του χώρου.

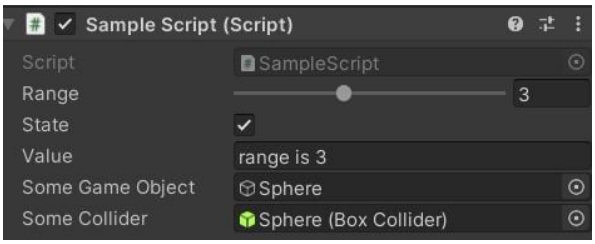
3.4.6 C# Scripts

Τα Scripts ανήκουν και αυτά στην κατηγορία των Components καθώς δίνουν χαρακτηριστικά στα GameObjects. Είναι απαραίτητα για να τρέξει μια εφαρμογή, επειδή χάρη σε αυτά συμβαίνουν γεγονότα (events) μέσα στο παιχνίδι όταν πρέπει. Τα Scripts μπορούν επίσης να χρησιμοποιηθούν για την δημιουργία γραφικών εφέ, τον έλεγχο της φυσικής συμπεριφοράς των αντικειμένων ή ακόμη και την εφαρμογή συστήματος τεχνητής νοημοσύνης σε χαρακτήρες Bots του παιχνιδιού. Δημιουργώντας ένα Script, επιλέγουμε το όνομα του και έπειτα το ανοίγουμε με το Visual Studio ή κάποιο άλλο πρόγραμμα.

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

public class SampleScript : MonoBehaviour
{
    [SerializeField][Range(1, 6)] int range;
    [SerializeField] bool state;
    [SerializeField] string value;
    [SerializeField] GameObject someGameObject;
    [SerializeField] BoxCollider someCollider;

    Collider sampleCollider;

    void Start()
    {
        state = false;
        sampleCollider = someGameObject.GetComponent<BoxCollider>();
    }


    void Update()
    {
        if (someCollider == sampleCollider) { state = true; }
        value = "range is: "+range;

    }
}
```

Σχήμα 3.4.8: Script

Ο κώδικας του Script (σχήμα 3.4.8) περιέχει την μέθοδο "Start" που εκτελείται με το που τρέξουμε την εφαρμογή, και την μέθοδο "Update" η οποία εκτελείται σε κάθε Frame, παρόλα αυτά καμία από τις δύο μεθόδους δεν είναι υποχρεωτική. Μπορούμε να δηλώσουμε μια μεταβλητή ως Public ή να χρησιμοποιήσουμε τον όρο "[SerializeField]" πριν την δήλωση μιας μεταβλητής έτσι ώστε να εμφανιστεί στο Component του Script μας (σχήμα 3.4.9) και να μπορούμε να την ρυθμίσουμε από εκεί ή να ενσωματώσουμε κάποιο αντικείμενο όπως για παράδειγμα στην περίπτωση του GameObject που έχει ενσωματωθεί ένα GameObject που ονομάζεται Sphere. Ακόμη μπορούμε να αρχικοποιήσουμε άλλα Components όπως το Box Collider, και όταν κάνουμε drag-drop κάποιο GameObject, παίρνει το Box Collider του αν υπάρχει.

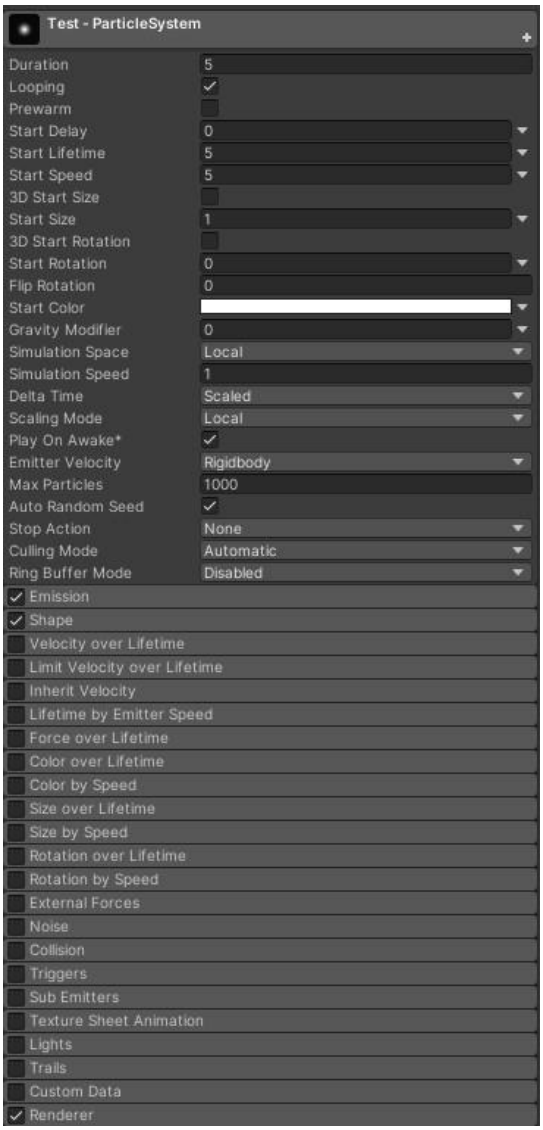


Σχήμα 3.4.9: Script Component

Τέλος, για να πάρουμε κάποιο Component από κάποιο GameObject όπως για παράδειγμα το Box Collider από το GameObject "someGameObject", χρησιμοποιούμε την μέθοδο "GetComponent" ως εξής: "someGameObject.GetComponent<boxCollider>()" και προαιρετικά μπορούμε να αποθηκεύσουμε το συγκεκριμένο box Collider σε μια μεταβλητή τύπου boxCollider.

3.4.7 Particle System

Σωματίδιο (Particle) είναι οποιαδήποτε μεμονωμένο υλικό ή οντότητα που δημιουργείται από το σύστημα σωματιδίων (Particle System). Το Particle System δημιουργεί ένα μεγάλο αριθμό σωματιδίων με μικρή διάρκεια ζωής όπως για παράδειγμα καπνό, σύννεφα, φλόγες και άλλα εφέ. Το Particle System έχει ένα τεράστιο αριθμό από ρυθμίσεις (σχήμα 3.4.10) που έχει κάθε είδους ρύθμιση όπως για παράδειγμα πόση διάρκεια να κρατήσει ένα Particle πριν διαγραφεί, ποιά θα είναι η ταχύτητα του και το χρώμα του και πόσα Particles θα υπάρχουν ταυτόχρονα στον χώρο. Παρακάτω βλέπουμε κάποιες ρυθμίσεις όπως το Emission, πατώντας το εμφανίζονται ρυθμίσεις σχετικά με το Emission, δηλαδή πόσα Particles να επέμπονται το δευτερόλεπτο. Στην ρύθμιση Shape υπάρχουν ρυθμίσεις σχετικά με το σχήμα κατά την εκπομπή των Particles. Αν επιλέξουμε το σχήμα να είναι Cone (κώνος) μας δίνει νέες ρυθμίσεις σχετικά με τον κώνο όπως οι μοίρες που θέλουμε να έχει. Άλλες σημαντικές ρυθμίσεις είναι το Renderer που καθορίζει το πώς θα φαίνονται τα Particles και τι Material θα έχουν, και το Trails που αφήνει ίχνη πίσω από τα Particles και στην εφαρμογή χρησιμοποιείται στα πυροτεχνήματα στο τέλος κάθε πίστας. Ακόμη μια χρήσιμη ρύθμιση είναι το Noise (θόρυβος) που μας επιτρέπει να εφαρμόσουμε αναταράξεις στην κίνηση των σωματιδίων, ορίζοντας την συμπεριφορά του θορύβου ξεχωριστά για κάθε άξονα, και χρησιμοποιώντας τις επιπλέον ρυθμίσεις μπορούμε να ρυθμίσουμε την ομαλότητα του θορύβου.



Σχήμα 3.4.10: Particle System Component

3.5 Render Pipeline

Ένα Render Pipeline χρησιμοποιείται για να εμφανίζει τα περιεχόμενα της σκηνής στην οθόνη μέσω εκτέλεσης μιας σειράς από (τρεις) λειτουργίες που ονομάζονται Culling, Rendering και Post-Processing[4].

Η πρώτη λειτουργία ονομάζεται Culling και αφαιρεί πρόσθετα αντικείμενα τα οποία δεν είναι ορατά στην κάμερα όταν υπάρχουν άλλα αντικείμενα μπροστά από αυτά. Η δεύτερη λειτουργία ονομάζεται Rendering, η οποία σχεδιάζει μια σκηνή στην οθόνη του υπολογιστή, αυτό περιλαμβάνει ένα συνδυασμό υπολογισμών γεωμετρίας, υφής επεξεργασίας επιφάνειας, φωτισμού και προβολή της εικονικής κάμερας. Το Post-Processing είναι η τρίτη λειτουργία, και χρησιμοποιείται για την εφαρμογή φίλτρων και εφέ στο buffer εικόνας πριν εμφανιστεί στην οθόνη.

3.5.1 Render Pipelines του Unity

Στο Unity υπάρχουν τρια Render Pipelines, καθένα από τα οποία χρησιμοποιούνται στις κατάλληλες εφαρμογές ανάλογα με τις δυνατότητες και τα χαρακτηριστικά που επιθυμεί ο προγραμματιστής. Με την δημιουργία της εφαρμογής επιλέγεται και ένα από τα τρια διαθέσιμα Render Pipelines που θα χρησιμοποιηθεί για να εφαρμόσει τις παραπάνω λειτουργίες.

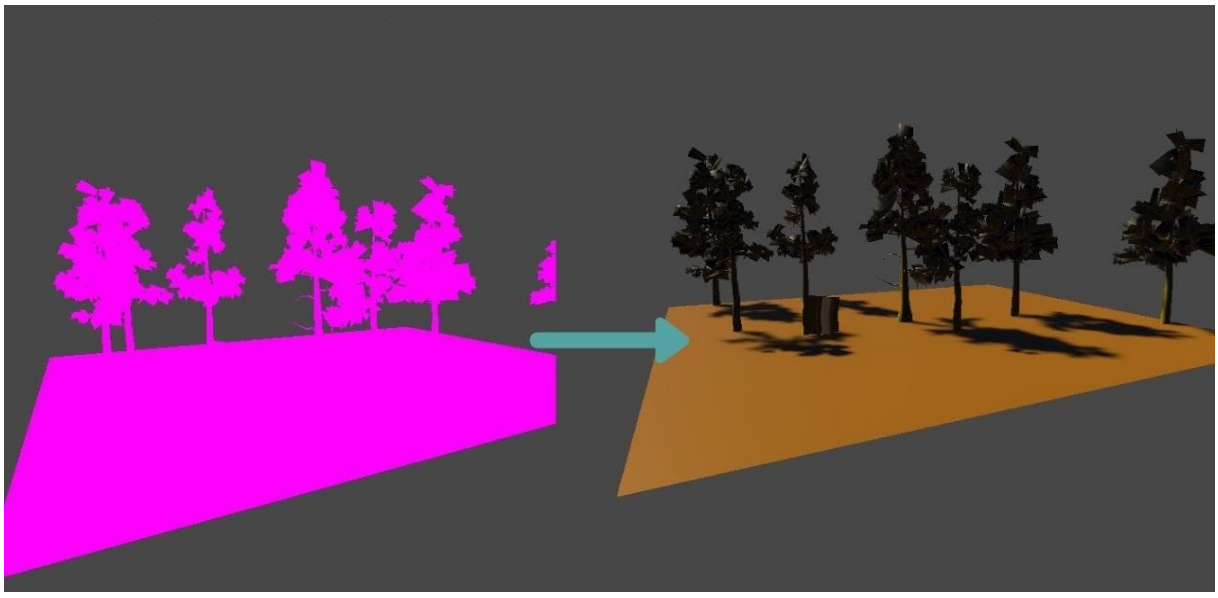
Το πρώτο είναι το Built-in Render Pipeline (BiRP) και είναι το πρώτο Render Pipeline που δημιουργήθηκε στο Unity, χρησιμοποιείται ως προκαθορισμένο (Default) Pipeline. Το Δεύτερο ονομάζεται Universal Render Pipeline (URP) και επιτρέπει φιλικές ροές εργασίας προς τον καλλιτέχνη δημιουργώντας βελτιστοποιημένα γραφικά σε όλες τις πλατφόρμες. Τέλος είναι το High Definition Render Pipeline (HDRP) το οποίο σχεδιάστηκε για εφαρμογές υψηλής ποιότητας και στοχεύει στις σύγχρονες πλατφόρμες καθώς χρησιμοποιεί διάφορες τεχνικές όπως οι τεχνικές φωτισμού που βασίζονται στο φυσικό περιβάλλον. Scriptable Render Pipeline (SRP) είναι μια δυνατότητα που παρέχει το Unity, που επιτρέπει τον έλεγχο του Rendering μέσω Scripts και υποστηρίζεται στο Universal Render Pipeline (URP) και High Definition Render Pipeline (HDRP).

Η παρούσα εφαρμογή χρησιμοποιεί URP καθώς με αυτό παρέχεται η δυνατότητα να δημιουργηθεί εύκολα μια εφαρμογή επαυξημένης πραγματικότητας με κάποια βασικά εργαλεία που δεν παρέχει το BiRP.

3.5.2 Upgrade Materials to UniversalRP

Στο Unity υπάρχουν Shaders, δηλαδή μικρά Scripts που υπολογίζουν το χρώμα κάθε Pixel που αποδίδεται με βάση την είσοδο φωτισμού και τις ρυθμίσεις των Materials. Διαφορετικά Render Pipelines χρησιμοποιούν διαφορετικά Shaders, όμως παρέχονται τα βασικά shaders μαζί με το Render Pipeline που επιλέγεται για την εφαρμογή.

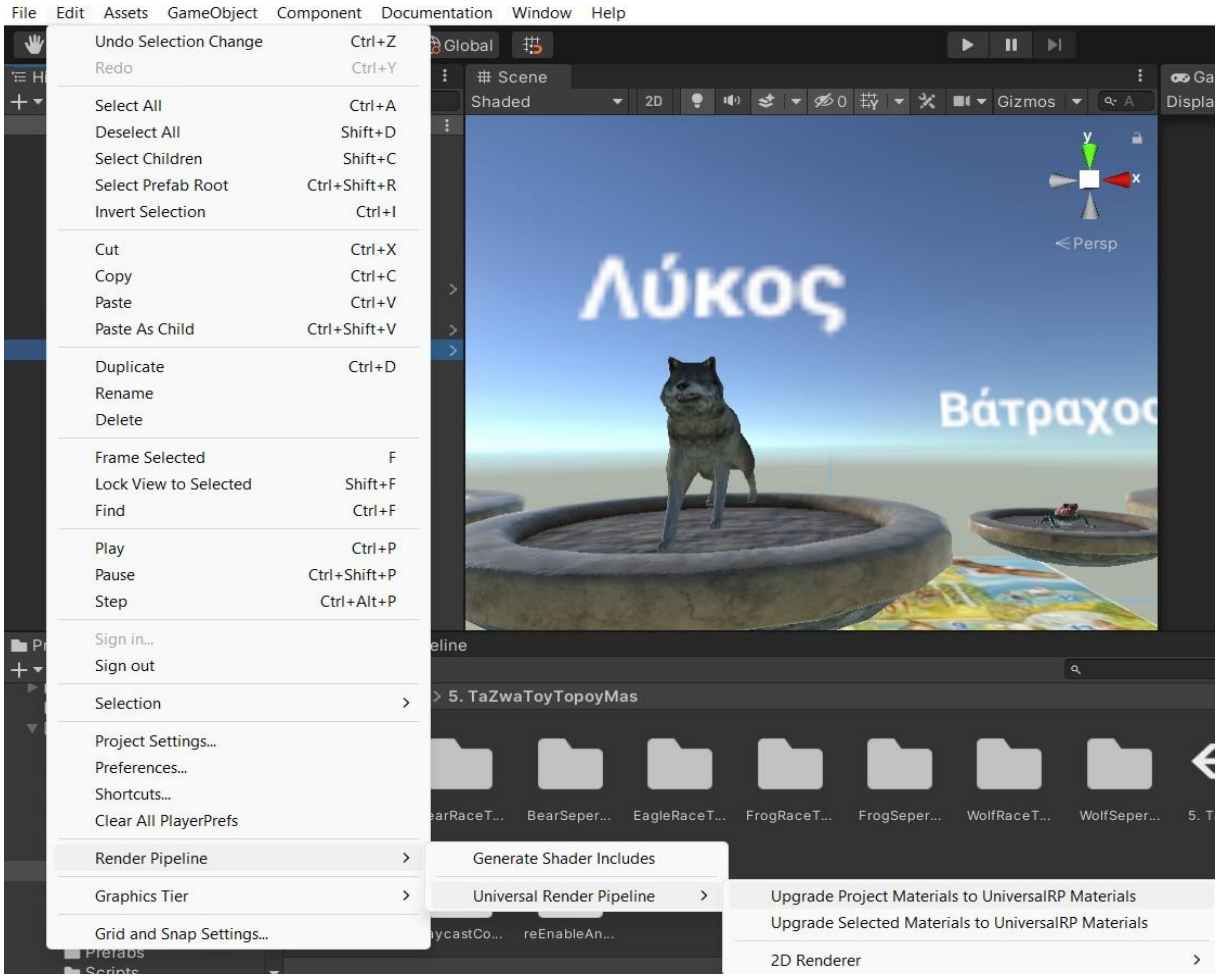
Πολλά Assets που έχουν δημιουργηθεί για πολλαπλά Render Pipelines, μπορεί να μην χρησιμοποιούν το κατάλληλο Shader. Ως αποτέλεσμα, τα Assets έχουν ένα ροζ χρώμα (σχήμα 3.4.11) το οποίο αντιπροσωπεύει την αποτυχία υπολογισμού χρώματος στα Pixels, είτε λόγω του λανθασμένου Shader πάνω στο Material ή λόγω της έλλειψης Material στο αντικείμενο.



Σχήμα 3.4.11: URP Shaders

Για να μπει κάποιο Asset στην URP εφαρμογή, πρέπει το συγκεκριμένο Asset να υποστηρίζεται από το URP. Αν το Asset είναι αγορασμένο από το Asset Store, ο πωλητής αναφέρει με ποιά Render Pipelines είναι συμβατό. Αν το Asset είναι ροζ ενώ υποστηρίζεται από το URP, μπορούμε να το αντιμετωπίσουμε με δύο τρόπους. Ο πρώτος τρόπος είναι να αλλάξουμε το Shader του Material από τις ρυθμίσεις του Material (σχήμα 3.4.12) στο Shader του URP (URP/Lit), ή σε κάποιο άλλο ειδικό Shader που δημιουργήθηκε για το συγκεκριμένο Material αν υπάρχει. Ο δεύτερος και πιό απλός τρόπος είναι να παμε στην επιλογή “Edit” του Unity, “Render Pipeline”, “Universal Render Pipeline”, και από εκεί μπορούμε να πατήσουμε την πρώτη επιλογή “Upgrade Project Materials to UniversalRP Materials” που ελέγχει όλα τα Materials της εφαρμογής και τα αναβαθμίζει σε URP Materials που σημαίνει ότι τοποθετεί τα κατάλληλα Shaders στα ροζ Materials, ή την δεύτερη επιλογή “Upgrade Selected Materials to UniversalRP Materials” στην περίπτωση που θέλουμε να αναβαθμίσουμε μόνο τα επιλεγμένα Materials.

Σχήμα 3.4.12: Change Shader



Σχήμα 3.4.13: Upgrade Materials

3.6 Vuforia Engine

Software development kit (SDK) ονομάζεται το σύνολο εργαλείων που μπορεί να συνδεθεί σε άλλο πρόγραμμα και επιτρέπουν τους προγραμματιστές να δημιουργήσουν εφαρμογές για την συγκεκριμένη πλατφόρμα. Το Vuforia Engine είναι ένα από τα πιό δημοφιλές SDK για ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας[3]. Το Vuforia παρέχει μια βάση δεδομένων (Vuforia Database) που χρησιμοποιείται για την αποθήκευση εικόνων και την χρήση τους στην διαδικασία σάρωσης εικόνων (Image Tracking). Οι εικόνες τοποθετούνται σε κάποια ρύθμιση (Image Target) ενός Component [2].

3.6.1 Εγκατάσταση του Vuforia στο Unity

Προκειμένου να ενσωματωθεί το Vuforia στο Unity, πρέπει αρχικά να είναι ανοιχτή η εφαρμογή στην οποία θέλουμε να λειτουργεί. Στην σελίδα του Vuforia υπάρχει μια καρτέλα που ονομάζεται “Downloads”, εκεί υπάρχει ένα Dropdown μενού το οποίο περιέχει τις εκδόσεις του Vuforia. Επιλέγοντας μία έκδοση εμφανίζονται κάποια Links. Πατώντας το Link “Add Vuforia Engine to a Unity Project or upgrade to the latest version” έπειτα από Login στην ιστοσελίδα κατεβαίνει το αρχείο “add-vuforia-package-10-12-3.unitypackage” που είναι ένα Package το οποίο με το άνοιγμά του ενσωματώνεται στο Unity. Πατώντας το κουμπί “Import” στο παράθυρο που εμφανίζεται στην εφαρμογή ενσωματώνεται το Vuforia μαζί με τις ρυθμίσεις και τα αρχεία που περιέχει.

3.6.2 Vuforia Database

Η εφαρμογή λειτουργεί σαρώνοντας εικόνες έτσι ώστε να εμφανιστεί το περιβάλλον. Οι εικόνες αρχικά πρέπει να τοποθετηθούν στην βάση δεδομένων που παρέχει το Vuforia πατώντας το κουμπί “Add Target” στην σελίδα της βάσης δεδομένων. Έπειτα πατώντας το κουμπί “Download Database” κατεβαίνει η βάση δεδομένων σε μορφή unitypackage. Ανοίγοντας το πακέτο που κατεβάσαμε εμφανίζεται ένα παράθυρο στην εφαρμογή, πατώντας “Import” ενσωματώνουμε τις εικόνες στο πρόγραμμα.

3.6.3 Image Target

Η διαδικασία της σάρωσης γίνεται μέσω του Image Target, το οποίο είναι ένα GameObject με Components του Vuforia. Η επιλογή εικόνας για σάρωση γίνεται μέσω του Component “Image Target Behaviour” το οποίο είναι ένα Script. Στις ρυθμίσεις του Component επιλέγουμε στο Type “From Database” για να ψάξει τις εικόνες από την βάση δεδομένων, από κάτω στην ρύθμιση Database επιλέγουμε το όνομα της βάσης δεδομένων, και τέλος στην ρύθμιση Image Target επιλέγουμε το όνομα της εικόνας που θέλουμε να σαρωθεί για να εμφανίσει το συγκεκριμένο Scene.

Έχοντας επιλέξει τις παραπάνω ρυθμίσεις, στο Scene εμφανίζεται η εικόνα από την βάση δεδομένων που βοηθάει στην σύγκριση των GameObjects του Scene, για να γνωρίζει ο προγραμματιστής το μέγεθος των GameObjects με βάση την εικόνα καθώς σαρώνεται.

3.7 Επίλογος

Η εφαρμογή έχει υλοποιηθεί μέσω της μηχανής παιχνιδιού Unity. Τα GameObjects αποκτούν διάφορες ιδιότητες χάρη στα Components τα οποία ο χρήστης μπορεί να ρυθμίσει μέσω του παραθύρου Inspector. Πέρα από το παράθυρο inspector υπάρχουν και άλλα που δίνουν την δυνατότητα στον προγραμματιστή να περιηγηθεί στο περιβάλλον και να ρυθμίσει τον τρόπο με τον οποίο λειτουργεί. Τα Render Pipelines χρησιμοποιούνται για να εμφανίσουν το περιβάλλον στον χρήστη. Τέλος, το Vuforia Engine είναι ένα σύνολο εργαλείων με το οποίο δημιουργήθηκε η εφαρμογή επαυξημένης πραγματικότητας μέσα στο Unity.

Κεφάλαιο 4ο: Υλοποίηση της εφαρμογής

4.1 Εισαγωγή

Έχοντας τοποθετήσει το Vuforia Engine στο Unity, δεν μένει τίποτα άλλο από την δημιουργία των ασκήσεων. Η εφαρμογή αποτελείται από ένα κεντρικό μενού από το οποίο ο χρήστης μπορεί να επιλέξει την άσκηση την οποία επιθυμεί μέσω κουμπιών που αναγράφουν την ονομασία της άσκησης. Σε περίπτωση που ο χρήστης επιλέξει λάθος άσκηση υπάρχει ένα κουμπί (πάνω αριστερά) με το οποίο μπορεί να επιστρέψει στο κεντρικό μενού, επίσης υπάρχει και ένα ακόμα κουμπί (πάνω δεξιά) που ξαναφορτώνει την άσκηση από την αρχή. Τα συγκεκριμένα κουμπιά υπάρχουν σε όλες τις ασκήσεις έτσι ώστε να υπάρχει ανεξαρτησία και να μην χρειάζεται να ολοκληρωθεί μία άσκηση προκειμένου να επιλεχθεί η επόμενη.

Πέρα από τα παραπάνω κουμπιά, οι ασκήσεις έχουν διαφορετικές λειτουργίες και περιεχόμενο, παρόλα αυτά υπάρχουν κάποιες κοινές λειτουργίες που χρησιμοποιούνται μεταξύ των ασκήσεων, Παρακάτω στο κεφάλαιο θα αναλύσουμε τον τρόπο με τον οποίο λειτουργεί η κάθε άσκηση λαμβάνοντας υπόψη κάθε λειτουργία με σκοπό να καλυφθεί κάθε είδους απορία σχετικά με την εφαρμογή.

4.2 Βασικές λειτουργίες

Είναι πραγματικά χρήσιμη η λογική της επαναχρησιμοποίησης, καθώς δημιουργώντας λειτουργίες που μπορούμε να χρησιμοποιήσουμε ξανά και ξανά αποτρέπουμε την σπατάλη χρόνου που απαιτείται για να δημιουργηθούν παρόμοιες λειτουργίες από την αρχή, αλλά και αποφυγή πολυπλοκότητας από άποψη ότι μία εφαρμογή πρέπει να είναι απλή και με όσο το δυνατόν λιγότερες αλλά και χρήσιμες λειτουργίες πρώτον επειδή κάνει πιό εύκολη την συνεργασία των προγραμματιστών μεταξύ τους, και δεύτερον επειδή μας βοηθάει στην αποσφαλμάτωση (debugging) έχοντας λιγότερες λειτουργίες για έλεγχο και πιό απλές ώστε να μπορούν να διαβαστούν από οποιονδήποτε. Πριν την ανάπτυξη των λειτουργιών των ασκήσεων, θα αναλύσουμε κάποιες βασικές λειτουργίες που επαναχρησιμοποιούνται σε όλη την εφαρμογή.

4.2.1 ChangeScene

Μία από τις βασικότερες λειτουργίες του παιχνιδιού είναι η δυνατότητα αλλαγής των Scenes από τον χρήστη. Το ChangeScene είναι ένα Script που σε όλα τα Scenes τοποθετείται πάνω σε ένα Empty GameObject (χωρίς Components πέρα από το Transform) που ονομάζεται SceneChanger.

Πριν χρησιμοποιήσουμε το Script, πρέπει να έχουμε υπόψη ότι όλες οι πίστες που θέλουμε να χρησιμοποιηθούν υπάρχουν στο “Scenes In Build” του παραθύρου Build Settings (σχήμα 3.8.8) επειδή για να αλλάξουμε Scene χρησιμοποιώντας την κλάση του Unity, SceneManager, προσπαθούμε να αποκτήσουμε πρόσβαση στο αντίστοιχο Scene αναγνωρίζοντάς το από τη λίστα με τα Scenes του Build Settings.

Παρατηρώντας τον κώδικα του ChangeScene (σχήμα 4.2.1), βλέπουμε αρχικά τις 3 βασικές βιβλιοθήκες που χρησιμοποιούνται σε όλα τα Scripts (Using…) και μία επιπλέον βιβλιοθήκη (UnityEngine.SceneManagement) που απαιτείται για να έχουμε πρόσβαση στην κλάση SceneManager

από την οποία μπορούμε να αλλάζουμε πίστες, και την κύρια κλάση του που έχει την ονομασία ChangeScene. Η αλλαγή της πίστας γίνεται μέσω κουμπιών, οπότε η πρώτη σκέψη γιά την αλλαγή των Scenes είναι η δημιουργία κάποιων (ξεχωριστών για κάθε Scene) Public μεθόδων οι οποίες θα καλούνται με το πάτημα των κουμπιών.

Προκειμένου να αλλάξουμε Scene, χρησιμοποιούμε την Static μέθοδο LoadScene από την βιβλιοθήκη "UnityEngine.SceneManagement" η οποία παίρνει ως παράμετρο μία μεταβλητή τύπου String, στη θέση αυτής πρέπει να βάλουμε το όνομα του Scene ακριβώς όπως είναι (προτείνεται copy+paste για την αποφυγή λάθους). Καλούμε λοιπόν την LoadScene για κάθε μέθοδο με τα ονόματα των Scenes που επιθυμούμε και έτσι καλούμε μέσω κουμπιών τις δικές μας μεθόδους για να φορτώσουμε καινούριο Scene. Οι μέθοδοι είναι public επειδή μόνο με αυτό τον τρόπο έχουμε πρόσβαση σε αυτές έξω από το Script.

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

public class ChangeScene : MonoBehaviour
{
    public void LoadMainMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void LoadXreiazomasteKanones()
    {
        SceneManager.LoadScene("1. XreiazomasteKanones");
    }

    public void LoadXtiseEnaKosmo()
    {
        SceneManager.LoadScene("2. FtiakseEnaKosmo");
    }

    public void LoadRoadSigns()
    {
        SceneManager.LoadScene("3. RoadSigns");
    }

    public void LoadTaFytaToyTopoyMas()
    {
        SceneManager.LoadScene("4. TaFytaToyTopoyMas");
    }

    public void LoadTaZwaToyTopoyMas()
    {
        SceneManager.LoadScene("5. TaZwaToyTopoyMas");
    }

    public void LoadTestScene()
    {
        SceneManager.LoadScene("TestScene");
    }
}
```

Σχήμα 4.2.1: ChangeScene

4.2.2 QuestionController

Το Script QuestionController (σχήματα 4.2.2 και 4.2.3) είναι το πρώτο Script που δημιουργήθηκε κατά την διάρκεια υλοποίησης της εφαρμογής, και όπως λέει το όνομα του, είναι αυτό που ελέγχει τις ερωτήσεις που εμφανίζονται προς τους χρήστες.

Πέρα από τις βασικές βιβλιοθήκες, προστέθηκε και η “UnityEngine.Playables” από την οποία διαχειριζόμαστε PlayableDirectors, δηλαδή Components που αφορούν το Timeline. Όλες οι μεταβλητές είναι public επειδή όλες αφορούν αντικείμενα μέσα στο Scene, όπως και η μεταβλητή director που είναι τύπου PlayableDirector και χρησιμοποιείται ως κύριο Timeline μιας άσκησης όπως την τρίτη άσκηση με αυτοκίνητα. Άλλες μεταβλητές είναι οι ερωτήσεις (1 μέχρι 12), ένα GameObject που ονομάζεται wrongAnswerImageAnimation και περιέχει ένα Animation σχετικά με τη λάθος απάντηση, και δύο ακόμα GameObjects, που αφορούν την ολοκλήρωση του παιχνιδιού. Ακόμη έχουμε 3 AudioClips για τους ήχους, σωστή απάντηση, λάθος απάντηση και τέλος παιχνιδιού.

Όλες οι μέθοδοι είναι Void, χωρίς να επιστρέφουν κάτι μετά την εκτέλεση, και οι περισσότερες είναι public για να υπάρχει πρόσβαση σε αυτές έξω από το Script. Στην μέθοδο Start αρχικοποιούνται κάποιες τιμές και καλούνται δύο ακόμα μέθοδοι, η PauseTimeline που παγώνει (δηλαδή σταματάει τον χρόνο) το Timeline θέτοντας την ταχύτητα του στο μηδέν, και η μέθοδος hideAllQuestions που απενεργοποιεί όλες τις ερωτήσεις εάν έχουν αρχικοποιηθεί και δεν εχουν τιμή null.

Η μέθοδος Arxi καλεί την μέθοδο ResumeTimeline που συνεχίζει το Timeline θέτοντας την ταχύτητα του σε ένα, και απενεργοποιεί την μεταβλητή arxi. Σε μερικές ασκήσεις δεν υπήρξε κύριο Timeline οπότε υπάρχει και η hideArxi η οποία απλά απενεργοποιεί την μεταβλητή arxi. Η ShowQuestion1Time καλεί την ShowQuestion1 (ενεργοποιεί το question1) μέσω του Invoke, που επιτρέπει να κληθεί μια μέθοδος μετά από κάποιο χρονικό διάστημα, το οποίο στην συγκεκριμένη μέθοδο είναι η Float παράμετρος. Για κάθε μεταβλητή question υπάρχει και ο αντίστοιχος συνδυασμός για την ενεργοποίηση τους.

```
public class QuestionController : MonoBehaviour
{

    public PlayableDirector director;

    public AudioClip wrongSound;
    public AudioClip correctSound;
    public AudioClip LevelCompleteSound;

    public Camera ARCamera;

    public GameObject arxi;

    public GameObject question1;
    public GameObject question2;
    public GameObject question3;
    public GameObject question4;
    public GameObject question5;
    public GameObject question6;
    public GameObject question7;
    public GameObject question8;
    public GameObject question9;
    public GameObject question10;
    public GameObject question11;
    public GameObject question12;

    public GameObject wrongAnswerImageAnimation;

    public GameObject levelCompletedParent;
    public GameObject levelCompletedUI;

    private void Start()
    {
        arxi.active = true;
        PauseTimeline();
        HideAllQuestions();
        levelCompletedParent.active = false;
        levelCompletedUI.active = false;
    }

    public void Arxi()
    {
        ResumeTimeline();
        arxi.active = false;
    }

    public void hideArxi() { arxi.active = false; }

    void ShowQuestion1() { question1.active = true; }
    public void ShowQuestion1Time(float time) { Invoke("ShowQuestion1", time); }
```

Σχήμα 4.2.2: QuestionController-1

```
public void CorrectAnswerNoTimeline()
{
    CorrectSound();
    HideAllQuestions();
}

public void CorrectAnswer(float time)
{
    CorrectSound();
    HideAllQuestions();
    Invoke("ResumeTimeline", time);
}

void CorrectSound() {
    AudioSource.PlayClipAtPoint(correctSound, ARCamera.transform.position, 0.8f);
}

public void WrongAnswer()
{
    AudioSource.PlayClipAtPoint(wrongSound,ARCamera.transform.position);
    wrongAnswerImageAnimation.active = true;
}

void ResumeTimeline()
{
    if (director != null)
        director.playableGraph.GetRootPlayable(0).SetSpeed(1);
}

public void PauseTimeline()
{
    if(director!=null)
        director.playableGraph.GetRootPlayable(0).SetSpeed(0);
}

public void LevelCompleted()
{
    levelCompletedParent.active = true;
    AudioSource.PlayClipAtPoint(LevelCompleteSound, ARCamera.transform.position, 0.65f);
    Invoke("EnableLevelCompleteUI", 2.5f);
}

void EnableLevelCompleteUI(){ levelCompletedUI.active = true; }

public void HideAllQuestions()
{
    if (question1 != null) question1.active = false;
```

Σχήμα 4.2.3: QuestionController-2

Η CorrectAnswer καλεί την CorrectSound η οποία παίζει κάποιο ήχο για σωστές απαντήσεις, καλεί την hideallQuestions και μέσω της Invoke καλεί την resumeTimeline. Η WrongAnwer παίζει κάποιο ηχητικό για τις λάθος απαντήσεις και ενεργοποιεί το wrongAnswerImageAnimation. Τέλος η LevelCompleted ενεργοποιεί το levelcompletedParent, παίζει ενα ηχητικό για την ολοκλήρωση της άσκησης, και καλεί μέσω της Invoke την EnableLevelCompletedUI η οποία ενεργοποιεί το levelCompletedUI.

4.2.3 ExerciseController

Η ExerciseController δημιουργήθηκε για τον έλεγχο των υποασκήσεων μέσα στις ασκήσεις. Όπως τις μεταβλητές ερωτήσεων στην QuestionController, έτσι και σε αυτήν αρχικοποιούνται GameObjects (Ask) που αφορούν τις ασκήσεις.

Στην Start καλείται η HideAllAsk η οποία απενεργοποιεί κάθε μεταβλητή Ask που έχει αρχικοποιηθεί και δεν είναι null. Κάθε μεταβλητή Ask έχει τη δική της μέθοδο ShowAsk με float παράμετρο η οποία με την βοήθεια της Invoke καλεί μια ShowAsk χωρίς παράμετρο που απενεργοποιεί το αντίστοιχο GameObject Ask.

```
public class ExerciseController : MonoBehaviour
{

    public GameObject Ask1;
    public GameObject Ask2;
    public GameObject Ask3;
    public GameObject Ask4;
    public GameObject Ask5;
    public GameObject Ask6;
    public GameObject Ask7;
    public GameObject Ask8;
    public GameObject Ask9;

    private void Start()
    {
        HideAllAsk();
    }

    public void HideAllAsk()
    {
        if (Ask1 != null) Ask1.active = false;
        if (Ask2 != null) Ask2.active = false;
        if (Ask3 != null) Ask3.active = false;
        if (Ask4 != null) Ask4.active = false;
        if (Ask5 != null) Ask5.active = false;
        if (Ask6 != null) Ask6.active = false;
        if (Ask7 != null) Ask7.active = false;
        if (Ask8 != null) Ask8.active = false;
        if (Ask9 != null) Ask9.active = false;
    }

    void ShowAsk1() { if(Ask1 != null) Ask1.active = true; }
    public void ShowAsk1(float time)
    {
        Invoke("ShowAsk1", time);
    }
```

Σχήμα 4.2.4: ExerciseController

4.2.4 PlayRandomMusic

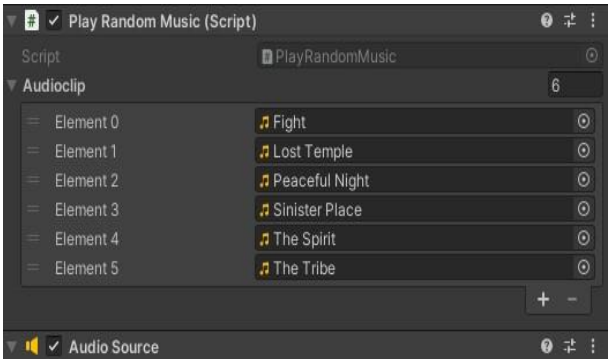
Σε κάποιες ασκήσεις χρησιμοποιείται το Script PlayRandomMusic το οποίο τοποθετείται σε ένα GameObject “MusicController” που έχει το Component AudioSource. Στο Script (σχήμα 4.2.5) υπάρχει ένα audioSource τύπου AudioSource το οποίο δεν έχει αρχικοποιηθεί, ένας serialized πίνακας audioClip τύπου AudioClip για να τοποθετήσουμε τις μουσικές και τέλος ένα randomNumber τύπου int.

Στην Start αρχικοποιούμε το audioSource δίνοντας του την τιμή AudioSource του ίδιου του αντικειμένου που είναι το Script (άρα στο MusicController). Στην Update έχουμε μια συνθήκη που ελέγχει αν το audioSource παίζει κάποιον ήχο. Αν δεν παίζει, τότε δίνεται μία τυχαία τιμή στην μεταβλητή randomNumber και το audioSource παίζει ένα τραγούδι με Index randomNumber και δύναμη ήχου 0,3.

Το σχήμα 4.2.6 δείχνει πώς εμφανίζεται το Script στα Components μετά την εισαγωγή 6 AudioClips στον πίνακα audioclip.

```
public class PlayRandomMusic : MonoBehaviour
{
    AudioSource audioSource;
    [SerializeField] AudioClip[] audioclip;

    int randomNumber;

    void Start()
    {
        audioSource = gameObject.GetComponent<AudioSource>();
    }
    void Update()
    {
        if (!audioSource.isPlaying)
        {
            randomNumber = Random.Range(0, audioclip.Length-1);
            audioSource.PlayOneShot(audioclip[randomNumber],0.3f);
        }
    }
}
```

Σχήμα 4.2.5: PlayRandomMusic

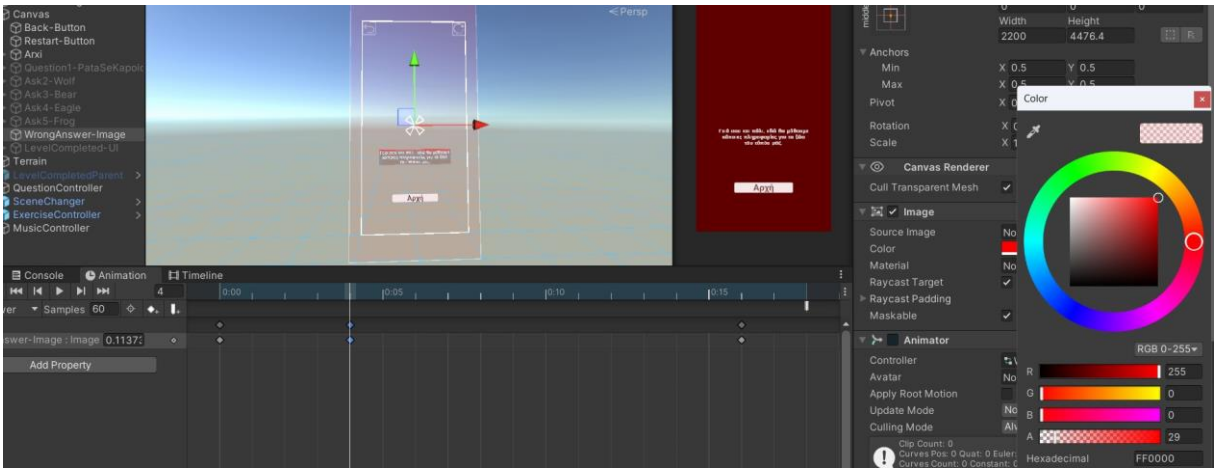


Σχήμα 4.2.6: PlayRandomMusic Component

4.2.5 WrongAnswer

Προκειμένου να γίνει κατανοητό ότι ο χρήστης επέλεξε λάθος απάντηση, είναι απαραίτητο κάποιο animation που κάνει την οθόνη να αναβοσβήσει με κόκκινο χρώμα και ταυτόχρονα ένα ηχητικό. Αρχικά πρέπει να δημιουργηθεί το Animation μέσα στο GameObject του UI με την κόκκινη εικόνα που θα αναβοσβήνει.

Στο Animation υπάρχουν 3 Keyframes που ρυθμίζουν το Alpha της εικόνας, επίσης υπάρχει ένα Animation Event μετά το τρίτο Keyframe. Το πρώτο και το τελευταίο Keyframe έχουν το Alpha της εικόνας στο 0, ενώ το μεσαίο έχει το Alpha στο 29 (βλέπε σχήμα 4.2.7). Με αυτό τον τρόπο το Alpha θα πάει από 0 στιγμιαία στο 29 και μετά πίσω στο 0.



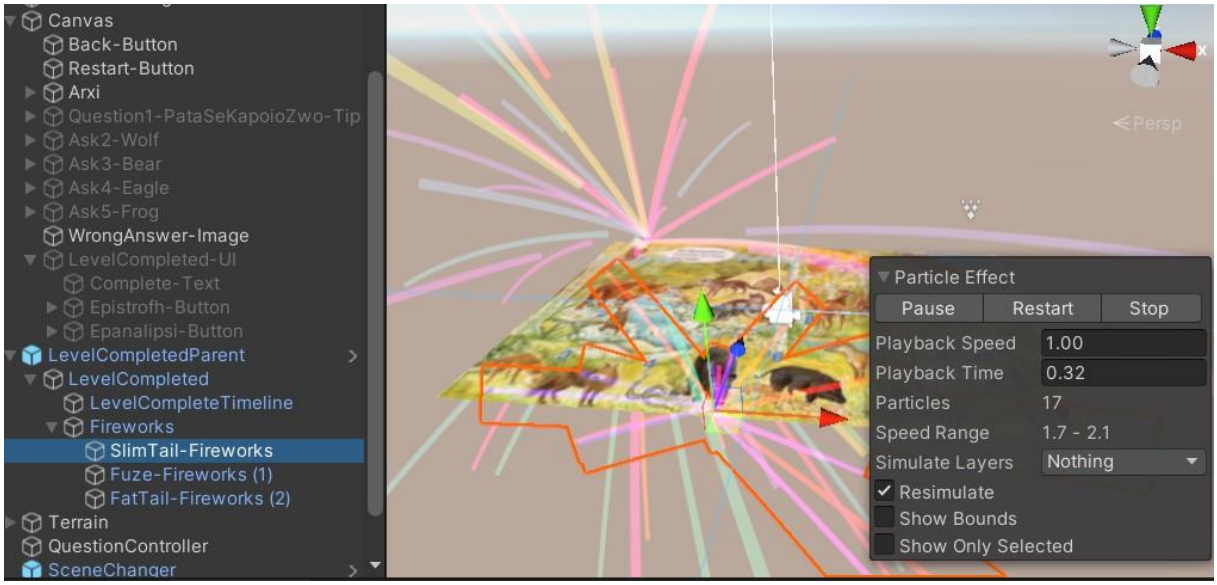
Σχήμα 4.2.7: WrongAnswerImage

Στο AnimationEvent εκτελείται η μέθοδος “DisableWrongAnswerImageAnimation” που προέρχεται από το Script DisableWrongAnswerImage το οποίο βρίσκεται πάνω στο GameObject της εικόνας. Η μέθοδος απενεργοποιεί την εικόνα, η οποία (εικόνα) σε διαφορετική περίπτωση θα εμπόδιζε τον χρήστη να αλληλεπιδράσει με το περιβάλλον καθώς όταν είναι ενεργή, καλύπτει όλα τα υπόλοιπα αντικείμενα του UI.

Το Animation έχει την επιλογή “Play on Awake” που του επιτρέπει να παίζει με το που ενεργοποιηθεί το GameObject. Ενεργοποιώντας λοιπόν το GameObject μέσω ενός κουμπιού που θεωρείται λάθος απάντηση και καλεί την μέθοδο “WrongAnswer” του Script QuestionController, ενεργοποιείται το Animation και στο τέλος ξανακλείνει το GameObject. παράλληλα ακούγεται το κατάλληλο ηχητικό.

4.2.6 LevelCompleted

Μόλις τελειώσει μία άσκηση, καλείται η μέθοδος LevelCompleted από το Script QuestionController. Στο LevelCompleted ενεργοποιείται το GameObject LevelCompletedParent που περιέχει 3 Particle Systems με πυροτεχνήματα (σχήμα 4.2.8) και ένα Timeline που τα ενεργοποιεί μόλις το LevelCompletedParent ενεργοποιηθεί. Παράλληλα ακούγεται ο κατάλληλος ήχος και καλείται μέσω της Invoke (μετά από 2.5 δευτερόλεπτα) η EnableLevelCompleteUI που ενεργοποιεί το GameObject levelCompletedUI. Το levelCompletedUI είναι ένα Empty GameObject που ανήκει στον Canvas (ο οποίος χρησιμοποιείται για όλα τα 2D αντικείμενα όπως κουμπια και Texts) και περιέχει ως Childs ένα Text που γράφει “Μπράβο, τα κατάφερες!” και δύο κουμπιά, “επιστροφή” που επιστρέφει τον χρήστη στο αρχικό μενού, και “επανάληψη” που ξαναφορτώνει την ίδια πίστα.



Σχήμα 4.2.8: LevelCompleted

4.2.7 Raycast

Raycast ονομάζεται μια νοητή ακτίνα που περνάει μέσα στο Scene στο σημείο που πατάει κάποιος χρήστης με το δάχτυλό του. Το Raycast χρησιμοποιείται στα Scripts και υπάρχει για να μπορεί ο χρήστης να αλληλεπιδρά με το περιβάλλον.

Υπάρχει ένα Feature από το Asset Store που ονομάζεται LeanTouch και χρησιμοποιείται στην εφαρμογή που αποτελείται από έτοιμες βιβλιοθήκες και Scripts τα οποία χρησιμοποιούν το Raycast. Το LeanTouch δίνει την δυνατότητα στους Developers να χρησιμοποιήσουν τα Scripts συγχωνεύοντάς τα πάνω σε GameObjects για να μπορεί ο χρήστης να αλληλεπιδράσει με αυτά με το

δάχτυλο χωρίς επιπλέον Scripts. Το LeanTouch χρησιμοποιήθηκε στην εφαρμογή για οποιοδήποτε GameObject μπορεί να συρθεί.

Παρόλα αυτά δημιουργήθηκε ένα ιδιωτικό Script (σχήμα 4.2.9) για να αποθηκεύεται το GameObject που πατιέται με το δάχτυλο σε μια μεταβλητή για να μπορεί να χρησιμοποιηθεί με οποιονδήποτε τρόπο ή να γίνει κάποιος έλεγχος σχετικά με αυτό. Αρχικά ο κώδικας τοποθετείται πάνω στην μέθοδο Update έτσι ώστε να τρέχει καθόλη την διάρκεια που χρησιμοποιείται. Το Input.touches είναι μια μέθοδος του Unity και από εκεί έχουμε πρόσβαση στα πατήματα του δαχτύλου μας στην εφαρμογή μέσω της οθόνης, και χάρη σε αυτό εκτελείται μία foreach η οποία απομονώνει τα αγγίγματα σε μια μεταβλητή touch τύπου Touch. Έπειτα αρχικοποιείται η μεταβλητή ray τύπου Ray που δημιουργεί το Raycast (δηλαδή την νοητή ακτίνα) για κάθε touch και γίνεται έλεγχος για το αν άρχισε το touch (επειδή δεν θέλουμε να δέχεται τα συνεχόμενα Touches αλλά μόνο αυτά που αρχίζουν (δηλαδή πατιούνται μια φορά). Στην συνέχεια αρχικοποιείται η μεταβλητή hit τύπου RaycastHit, και αργότερα γίνεται ένας ακόμη έλεγος που είναι το Raycast καλώντας μια μέθοδο του Unity με τρεις παραμέτρους, το ray, το hit και ένα αριθμό τύπου float που αντιπροσωπεύει την απόσταση σε Units (μονάδα απόστασης του περιβάλλοντος). Ο έλεγχος είναι ο εξης: “αν είναι αληθής η συνθήκη με την οποία μια ακτίνα (ray) ακουμπήσει ένα αντικείμενο (hit) σε απόσταση 1000 Units τοτε…”. Όταν δηλαδή ακουμπήσουμε με το δάχτυλό μας ένα GameObject, η συνθήκη γίνεται true και μπαίνει μέσα στο if. Η λέξη “out” δίπλα στην παράμετρο hit σημαίνει πως αν ο χρήστης ακουμπήσει κάποιο αντικείμενο, αυτό θα αποθηκευτεί στην μεταβλητή hit, για παράδειγμα, για να εμφανίσουμε το όνομα του αντικειμένου που ακουμπήσαμε, γράφουμε “hit.transform.name”.

```
void Update()
{
    foreach (Touch touch in Input.touches)
    {
        Ray ray = Camera.main.ScreenPointToRay(touch.position);

        if (touch.phase == TouchPhase.Began)
        {
            RaycastHit hit = new RaycastHit();

            if (Physics.Raycast(ray, out hit, 1000))
            {
                //kapoios kwdikas
            }
        }
    }

}
```

Σχήμα 4.2.9: Raycast

4.3 Κεντρικό Μενού

Με το άνοιγμα της εφαρμογής εμφανίζεται το λογότυπο του Unity, και έπειτα φορτώνει το πρώτο Scene (με Index 0) από το παράθυρο Build Settings. Ως πρώτο Scene επιλέχθηκε το κεντρικό μενού, επειδή μέσω αυτού μπορεί ο χρήστης να περιηγηθεί σε όλες τις ασκήσεις μέσω κουμπιών.

Το Scene του κεντρικού μενού αποτελείται από την AR κάμερα που απαιτείται για να μπορεί ο χρήστης να βλέπει την εφαρμογή, και τον Canvas που περιέχει οτιδήποτε βλέπουμε στην οθόνη μας στο συγκεκριμένο Scene.

Το Background αποτελείται από δύο Images που καλύπτουν όλη την οθόνη. Η πρώτη είναι μία μαύρη εικόνα έτσι ώστε να μην φαίνεται το περιβάλλον του χρήστη μέσω της κάμερας του κινητού, και η δεύτερη είναι η εικόνα του βιβλίου με πολύ μικρό Alpha (που είναι ίσο με 7) έτσι ώστε να μην υπάρχει πολύ χρώμα και να φαίνονται καλύτερα οι υπόλοιπες εικόνες και τα κουμπιά.

Στο πάνω μέρος υπάρχει μια φωτογραφία ενός πίνακα ο οποίος χρησιμοποιείται και στις ασκήσεις για να δίνει πληροφορίες και βοήθεια. Στο συγκεκριμένο Scene χρησιμοποιείται μαζί με ένα Text οδηγιών, έτσι ώστε ο χρήστης να γνωρίζει τι πρέπει να κάνει για να χρησιμοποιήσει την εφαρμογή.

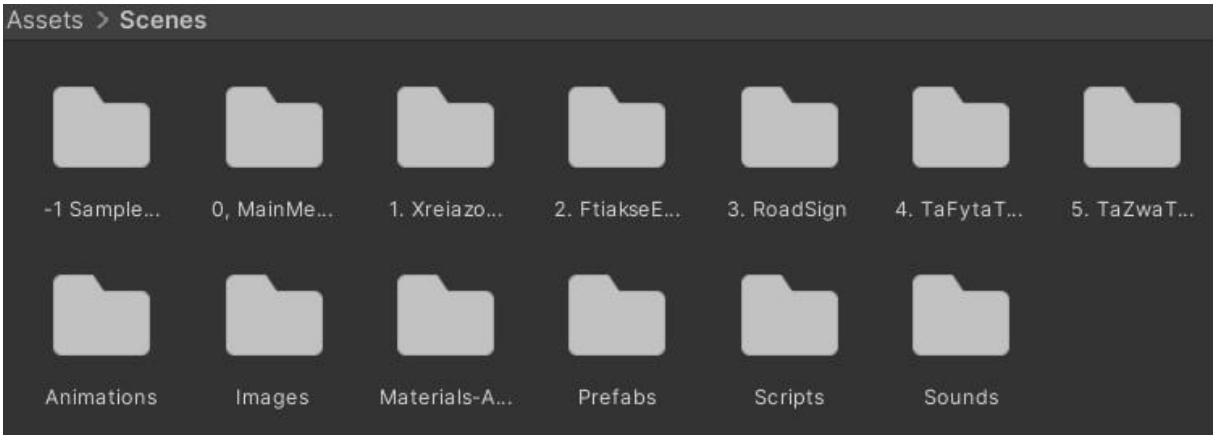
Σχήμα 4.3.1: Main Menu

Παρακάτω υπάρχουν κάποια Empty GameObjects που αντιπροσωπεύουν την κάθε άσκηση. Μέσα σε κάθε ένα από αυτούς τους Parents υπάρχει μία εικόνα που αντιπροσωπεύει την εικόνα που πρέπει να σαρώσει ο χρήστης για κάθε άσκηση και από δίπλα ένα Text που δίνει τον αριθμό της εικόνας με βάση την ενότητα, το κεφάλαιο και την σειρά της φωτογραφίας. Πάνω από την εικόνα και το Text βρίσκονται κάποια κουμπιά με τα ονόματα των ασκήσεων και την αρίθμησή τους. Κάθε κουμπί αντιστοιχεί στην κατάλληλη μέθοδο του Script ChangeScene για κάθε άσκηση και καλεί την μέθοδο LoadScene της κλάσης SceneManager δίνοντας ως παράμετρο το όνομα της άσκησης έτσι ώστε να την φορτώσει.

4.4 Ασκήσεις

Για την καλύτερη οργάνωση των αρχείων στο Project Window, πέρα από τους φακέλους που δημιούργησε το Unity και το Vuforia για τις ρυθμίσεις, υπάρχει και ο φάκελος Scenes που δημιουργήθηκε για την καταχώρηση των αρχείων που έχουν σχέση με τις ασκήσεις. Στο σχήμα 4.4.1 υπάρχουν επιπλέον οργανωμένοι φάκελοι μέσα στον φάκελο Scenes.

Η κάθε άσκηση έχει τον δικό της φάκελο έτσι ώστε να γίνεται ξεκάθαρο τι Asset ή Feature χρησιμοποιείται σε ποιά άσκηση. Οτιδήποτε είναι κοινό μεταξύ των ασκήσεων όπως Scripts, Sound Effects (SFX), εικόνες, Animation, Materials και Prefabs υπάρχουν στον φάκελο Scenes κατηγοριοποιημένα με την βοήθεια επιπλέον υποφακέλων. Τα Assets ή Features που χρησιμοποιούνται σε όχι παραπάνω από μία άσκηση υπάρχουν στους αντίστοιχους φακέλους (Scripts, Animations και άλλα) μέσα στους φακέλους των αντίστοιχων ασκήσεων. Ο πρώτος φάκελος ονομάζεται “-1 Sample Scene” και χρησιμοποιείται αποκλειστικά για debugging και δεν υπάρχει μέσα στην εφαρμογή ούτε εμπεριέχεται κάποιο Scene του στο Build Settings. Παρακάτω αναλύονται σε βάθος οι ασκήσεις και ο τρόπος με τον οποίο δημιουργήθηκαν.



Σχήμα 4.4.1: Scenes Folder

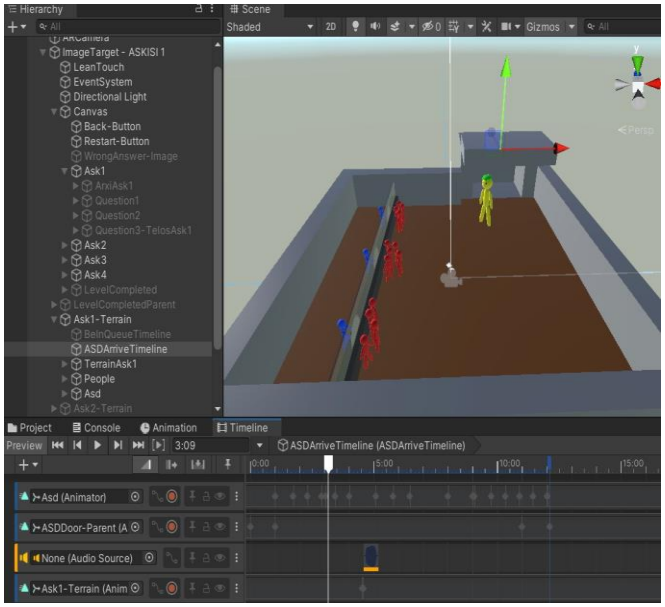
4.4.1 Χρειαζόμαστε κανόνες

Η πρώτη άσκηση είναι εμπνευσμένη από την πρώτη ενότητα του βιβλίου και συγκεκριμένα από το δεύτερο κεφάλαιο “Για να ζούμε μαζί, χρειαζόμαστε κανόνες”. Στόχος της άσκησης είναι να διδάξει στον χρήστη την σημασία των κανόνων με έναν απλό και ενδιαφέρον τρόπο δημιουργώντας κανόνες σε περιοχές (4 υποασκήσεις) που δεν τηρούνται με την βοήθεια ενός κίτρινου χαρακτήρα.

Μόλις μπει ο χρήστης στο Scene της άσκησης μέσω του κεντρικού μενού, πρέπει να σαρώσει την φωτογραφία του σχήματος 4.4.2 που βρίσκεται στο δεύτερο κεφάλαιο της πρώτης ενότητας.

Σχήμα 4.4.2: Εικόνα 1.2.3

Μόλις σαρωθεί η φωτογραφία εμφανίζεται ένα UI με την φωτογραφία του χαρακτήρα και ένα Text που αναφέρει πως ο χαρακτήρας θέλει την βοήθεια του χρήστη για να βάλει κανόνες εκεί που δεν τους τηρούν. Πατώντας το κουμπί "Αρχή" που εμφανίζεται παρακάτω, απενεργοποιείται ο Parent του αρχικού UI, εμφανίζεται η πρώτη άσκηση (σχήμα 4.4.3) και ο χρήστης απαντάει δύο ερωτήσεις, η πρώτη ερώτηση σχετίζεται με το αν οι πελάτες (κόκκινοι) τηρούν τους κανόνες προτεραιότητας στο ταμείο, μόλις ο χρήστης πατήσει την σωστή επιλογή "όχι" εμφανίζεται η δεύτερη ερώτηση "ποιό κανόνα θα έβαζες για να λειτουργεί καλύτερα ο χώρος;". Μόλις ο χρήστης πατήσει το κουμπί "Να μπουν σε σειρά", ενεργοποιείται το Timeline και έρχεται ο κίτρινος χαρακτήρας που τους κάνει (με μια σφυρίχτρα ως ηχητικό) να μπουν σε σειρά με Animation μέσα στο Timeline. Στο τέλος της πρώτης υποάσκησης εμφανίζεται το UI του σχήματος 4.4.4.



Σχήμα 4.4.3: Ασκηση 1.1

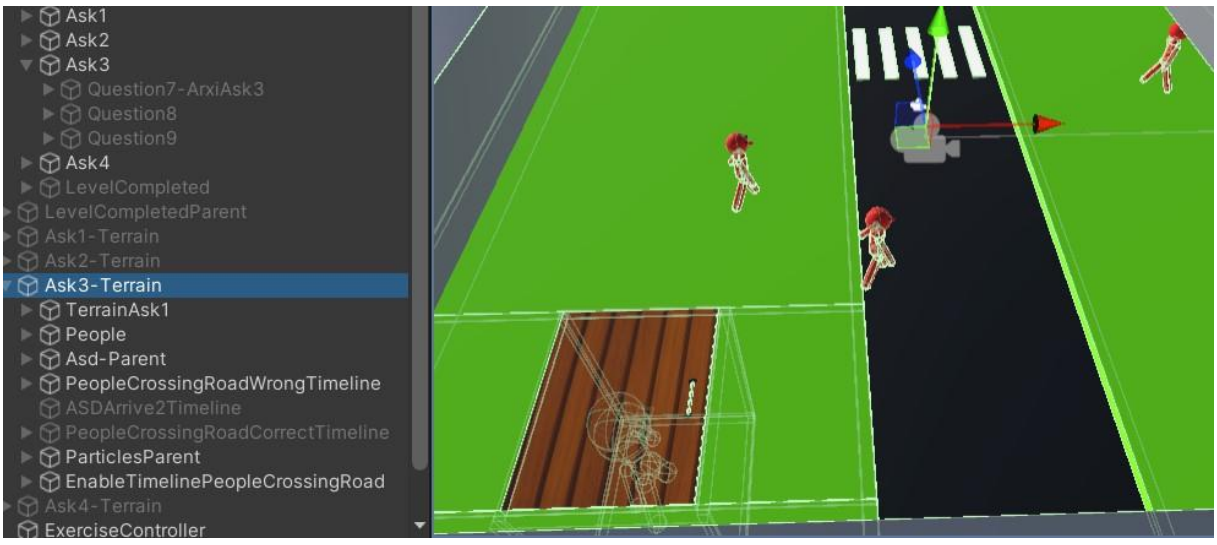
Σχήμα 4.4.4: Τέλος ασκησης 1.1

Πατώντας επόμενο εμφανίζεται η δεύτερη υποάσκηση (σχήμα 4.4.5) η οποία δείχνει ένα αυτοκίνητο το οποίο έχει σταθμεύσει σε χώρο με ράμπα όπου και απαγορεύεται. Δίνονται στον χρήστη κάποιες πληροφορίες σχετικά με την άσκηση και μετά εμφανίζεται ένα κουμπί συμβουλής που γράφει ότι πρέπει να σύρει το αυτοκίνητο στην κόκκινη περιοχή που αναβοσβήνει. Το κόκκινο GameObject (TipSquare) που αναβοσβήνει στον χώρο στάθμευσης δημιουργήθηκε από διάφανο Material και ένα Animation που ανεβοκατεβάζει το Alpha του Material μία φορά και μετά απενεργοποιεί το GameObject. Πάνω στο GameObject υπάρχει ένα Script με μία μέθοδο του Unity "OnDisable" που εκτελείται όταν το αντικείμενο απενεργοποιείται. Μέσα στην μέθοδο ενεργοποιείται ξανά το GameObject με αποτέλεσμα να αναβοσβήνει μέχρις ότου απενεργοποιήσουμε το Animation πριν αυτό απενεργοποιήσει το GameObject ή πιό απλά να απενεργοποιήσουμε το Script που το κάνει να ενεργοποιείται. Όταν λοιπόν ο χρήστης σύρει το αυτοκίνητο (με την βοήθεια του Script EnableDrag και LeanDrag του Feature LeanTouch που επιτρέπει τέτοιου είδους κινήσεις) στο Trigger πάνω από το TipSquare απενεργοποιείται το αυτοκίνητο και ενεργοποιείται ένα δεύτερο ίδιο που ανήκει στην σωστή θέση πάνω στην θέση στάθμευσης. Αυτό δημιουργεί την ψευδαίσθηση ότι είναι το ίδιο αυτοκίνητο που κουνούσε ο παίκτης ενώ στην πραγματικότητα είναι δύο διαφορετικά GameObjects.

Επίσης εμφανίζεται ένα αντίστοιχο μήνυμα όπως αυτό της πρώτης υποάσκησης με ένα κείμενο και ένα κουμπί “Συνέχεια”.

Σχήμα 4.4.5: Άσκηση 1.2

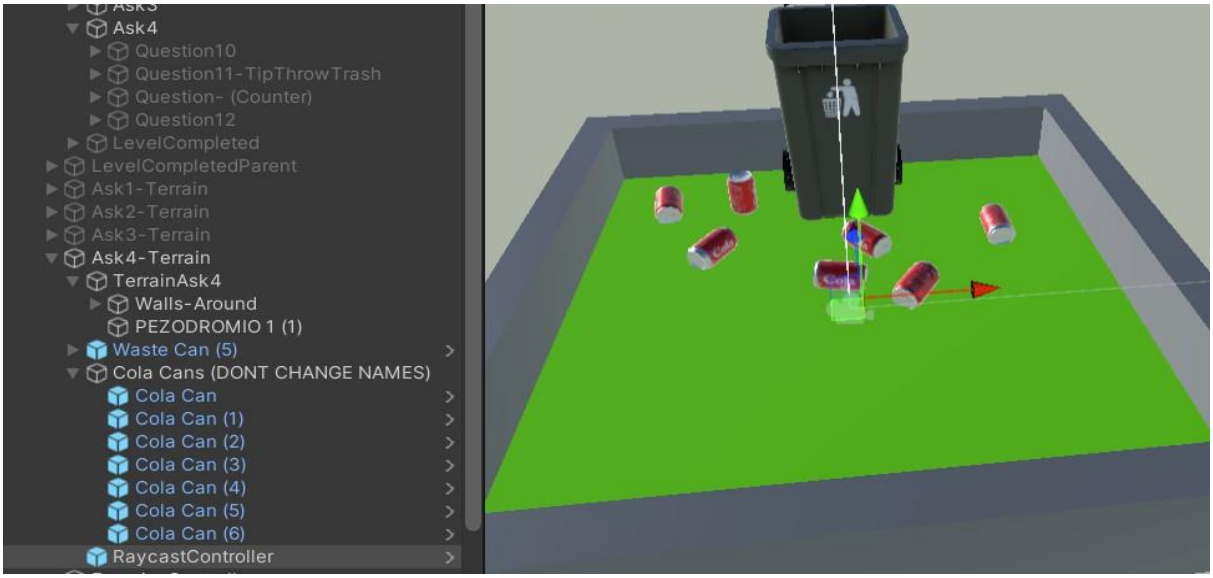
Μόλις ο χρήστης πατήσει το κουμπί “Συνέχεια” εμφανίζεται η τρίτη υποάσκηση (σχήμα 4.4.6) στην οποία υπάρχουν δύο κύρια Timelines, το πρώτο είναι το “PeopleCrossingRoadWrongTimeline” που κάνει κάποια ανθρωπάκια να περνούν τον δρόμο από οποιοδήποτε σημείο εκτός της διάβασης πεζών και χωρίς να κοιτούν αν έρχεται αυτοκίνητο, και το δεύτερο είναι το “PeopleCrossingRoadCorrectTimeline” που κάνει κάποια ανθρωπάκια να περνούν από την διάβαση και πριν περάσουν κοιτούν αριστερά και δεξιά. Και στις δύο περιπτώσεις όταν τελειώσει το Timeline καλείται μια μέθοδος που απενεργοποιεί τα GameObjects των Timeline, και τα ίδια έχουν ένα Script με την μέθοδο onDisable στην οποία ενεργοποιούνται ξανά με αποτέλεσμα τα Timelines να τρέχουν για πάντα.



Σχήμα 4.4.6: Άσκηση 1.3

Η υποάσκηση ξεκινάει με το πρώτο Timeline που είναι ο λάθος τρόπος να περάσει κάποιος τον δρόμο και δύο ερωτήσεις προς τον χρήστη. Η πρώτη ερώτηση ρωτάει αν είναι σωστός ο τρόπος που περνούν τον δρόμο τα ανθρωπάκια και η δεύτερη ρωτάει τι πρέπει να κάνουν πριν περάσουν τον δρόμο. Η σωστή απάντηση για την δεύτερη ερώτηση είναι "πρώτα να κοιτάξουν αν έρχεται αυτοκίνητο και μετά να περάσουν από την διάβαση". Μόλις πατηθεί η σωστή απάντηση, ξεκινάει ένα Animation με τον κίτρινο χαρακτήρα να εμφανίζεται μέσα από την καφέ πόρτα στο πάτωμα λέγοντας στα ανθρωπάκια πως να περνάνε τον δρόμο σφυρίζοντας (όπως και στην πρώτη υποάσκηση). Έπειτα απενεργοποιείται το "PeopleCrossingRoadWrongTimeline" και ενεργοποιείται το "PeopleCrossingRoadCorrectTimeline" που δείχνει τα ανθρωπάκια να περνούν σωστά τον δρόμο. Μετά εμφανίζεται το τέλος της υποάσκησης με ένα Text και ένα κουμπί "Επόμενο" όπως και στις προηγούμενες υποασκήσεις.

Μόλις ο χρήστης πατήσει το κουμπί "Επόμενο" εμφανίζεται η τέταρτη υποάσκηση (σχήμα 4.4.7) στην οποία κάποιος δεν τήρησε τον κανόνα και πέταξε τα σκουπίδια έξω από τον κάδο. Ο χρήστης πρέπει να πατήσει πάνω στα σκουπίδια για να τα βάλει στον κάδο. Παράλληλα υπάρχει ένα κουμπί βοήθειας που του δίνει πληροφορίες σε περίπτωση που δεν ξέρει τι πρέπει να κάνει και στο κάτω μέρος υπάρχει ένα Text που λειτουργεί ως μετρητής για να γνωρίζει ο χρήστης πόσα σκουπίδια πρέπει να μαζέψει ακόμα (για παράδειγμα 3/7).



Σχήμα 4.4.7: Άσκηση 1.4

Η διαδικασία με την οποία λειτουργεί το πάτημα για την εξαφάνιση των σκουπιδιών και τον μετρητή γίνεται μέσω του Script RaycastController του GameObject RaycastController που έχει κάποιες Serialized μεταβλητές και ελέγχει ποιό είναι το όνομα του GameObject που πάτησε ο χρήστης. Υπάρχει ένας έλεγχος αν αυτό που πάτησε ο χρήστης έχει το ίδιο όνομα με κάποιο από τα GameObjects σκουπίδια (τα οποία είναι καταχωρημένα σε ένα GameObject πίνακα "Objects", τότε απενεργοποιεί το συγκεκριμένο σκουπίδι και μετράει +1 σε κάποιον δείκτη (int Count). Παρακάτω στον κώδικα γίνεται έλεγχος αν ο δείκτης count είναι ίσος με το Length του πίνακα Objects (if count == objects.Length), αν ναι, τότε ενεργοποιεί το Text για το τέλος της άσκησης, τα πυροτεχνήματα τα οποία είναι Particle System, και το ηχητικό των πυροτεχνημάτων τα οποία προέρχονται από Serialized

μεταβλητές. Τέλος εμφανίζονται δύο κουμπιά “επανάληψη” σε περίπτωση που κάποιος χρήστης επιθυμεί να ξαναπαίξει την ίδια άσκηση, και “επιστροφή” για να γυρίσει πίσω στο κεντρικό μενού.

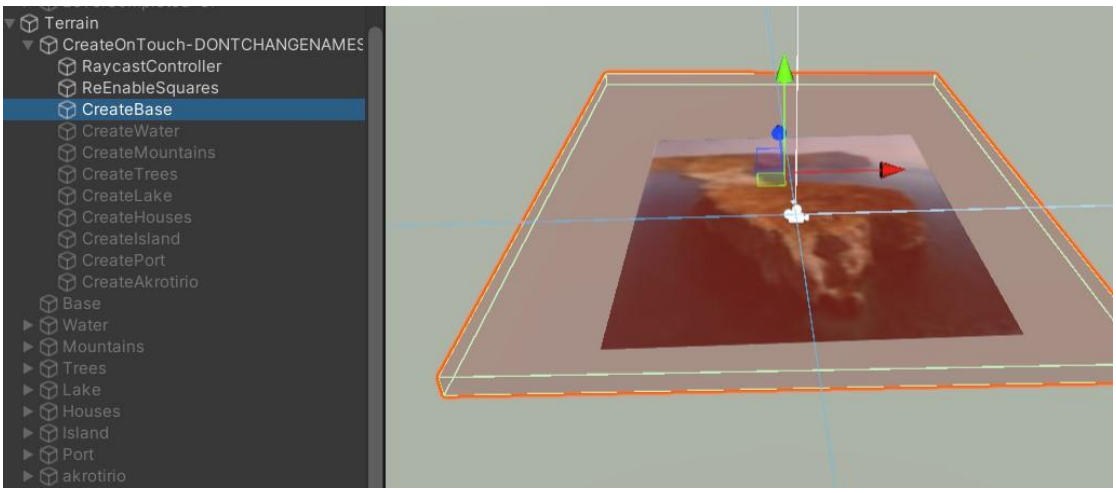
Σχήμα 4.4.8: Τέλος άσκησης 1

4.4.2 Φτιάξε ένα κόσμο

Η Δεύτερη άσκηση είναι εμπνευσμένη από την Δεύτερη ενότητα του βιβλίου και κυρίως από τα κεφάλαια έξι “Ένας χάρτης μας πληροφορεί (α)” και εφτά “Ένας χάρτης μας πληροφορεί (β)”. Στόχος της άσκησης είναι ο χρήστης να μάθει κάποιες ονομασίες γεωγραφικών όρων μέσω ερωτήσεων καθώς δημιουργεί ένα κόσμο που αποτελείται από τους όρους που μαθαίνει.

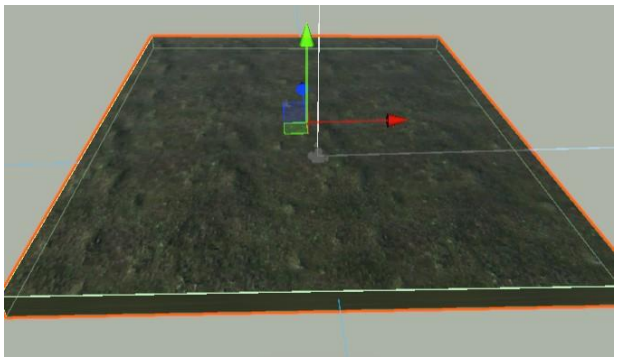
Σχήμα 4.4.9: Εικόνα 2.7.3

Μόλις μπει ο χρήστης στο Scene της άσκησης μέσω του κεντρικού μενού, πρέπει να σαρώσει την φωτογραφία του σχήματος 4.4.9 που βρίσκεται στο έβδομο κεφάλαιο της δεύτερης ενότητας. Μόλις σαρωθεί η φωτογραφία εμφανίζεται ένα UI με ένα Text που καλωσορίζει τον χρήστη και αναφέρει τον στόχο της άσκησης και ένα κουμπί “Αρχή”. Μόλις πατηθεί το κουμπί εμφανίζεται η πρώτη ερώτηση ενώ παράλληλα δεν υπάρχει τίποτα άλλο στην κάμερα του χρήστη. Η πρώτη ερώτηση είναι “Τί ονομάζουμε πεδιάδα;”, και μόλις ο χρήστης πατήσει το σωστό κουμπί “Η επίπεδη έκταση Γης” εμφανίζεται ένα κόκκινο πλαίσιο (σχήμα 4.4.10) το οποίο αναβοσβήνει μέσω Animation ρυθμίζοντας το Alpha του Transparent Material μια φορά απο 0 στο 105 και πίσω στο 0, και στο τέλος του Animation απενεργοποιείται το GameObject.



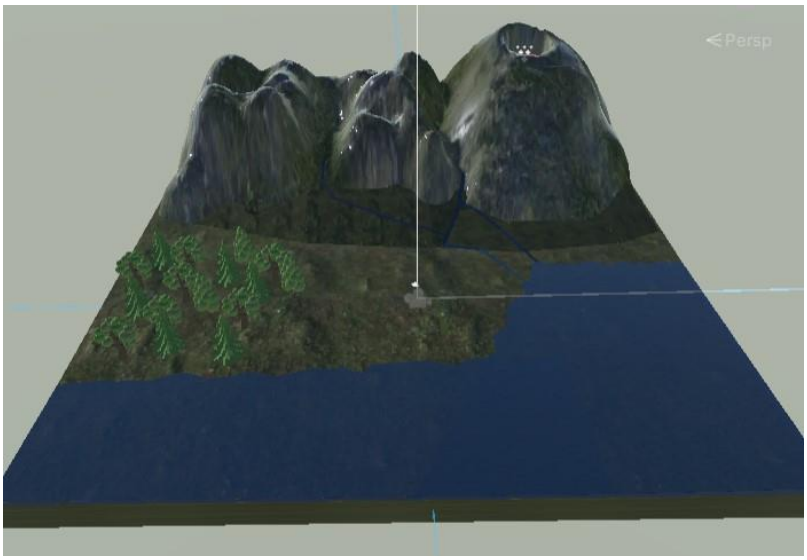
Σχήμα 4.4.10: CreateBase

Υπάρχει ένα Script EnableCheck του GameObject ReEnableSquares που ενεργοποιεί όλα τα πλαίσια της άσκησης που απενεργοποιούνται αφού αναβοσβήσουν. Μαζί με το κόκκινο πλαίσιο εμφανίζεται και ένα μήνυμα βοήθειας που γράφει ότι πρέπει ο χρήστης να πατήσει το κόκκινο πλαίσιο για να δημιουργηθεί μια πεδιάδα (σχήμα 4.4.11) η οποία είναι ένας παραμορφωμένος κύβος με Material από το Asset Store. Για το πάτημα των κόκκινων πλαισίων υπάρχει ένα Script RaycastControllerBuild μέσα στο GameObject RaycastController που παίρνει τα κόκκινα πλαίσια ως Serialized μεταβλητές, και μέσω του Raycast κώδικα για κάθε συγκεκριμένο πλαίσιο καλούνται οι κατάλληλες εντολές για την εμφάνιση των επόμενων ερωτήσεων.



Σχήμα 4.4.11: Πεδιάδα

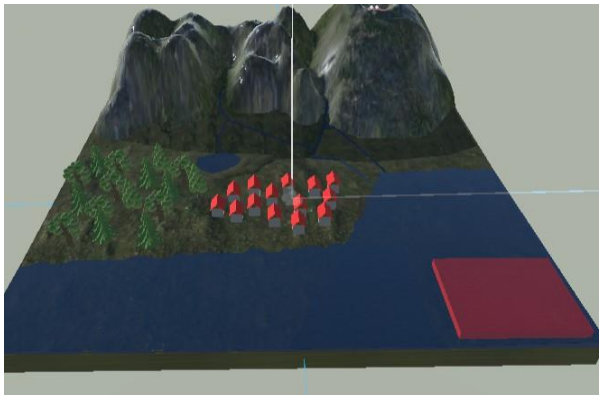
Έπειτα ζητείται να δημιουργηθεί μια θάλασσα στο νότιο μέρος της πεδιάδας με τον ίδιο τρόπο όπως και με την πεδιάδα, δηλαδή με ένα πλαίσιο που αναβοσβήνει. Η θάλασσα είναι δύο Shader (αριστερά και δεξιά) από το Asset Store και τα σημεία πάνω της κινούνται σαν κύματα για να δώσει την αίσθηση ότι είναι θάλασσα και όχι ένα απλό αντικείμενο, επιπλέον έχει δοθεί μια κλήση (Rotation) στα GameObjects έτσι ώστε να μην φαίνονται τετράγωνα όπως είναι στην πραγματικότητα. Μετά εμφανίζεται μια ερώτηση σχετικά με τι είναι οροσειρά. Όταν ο χρήστης πατήσει το σωστό κουμπί “Πολλά βουνά σε σειρά” εμφανίζεται με τον ίδιο τρόπο ένα πλαίσιο για να πατήσει ο χρήστης και να δημιουργήσει την οροσειρά. Η οροσειρά είναι Terrain και είναι φτιαγμένη μέσω των βασικών εργαλείων του Terrain, πάνω από το ηφαίστειο (δεξιά) υπάρχει ένα Particle System που αντιπροσωπεύει τον καπνό που προέρχεται μέσα από το ηφαίστειο. Όπως και με την θάλασσα έτσι εμφανίζεται ένα κείμενο που ζητάει από τον χρήστη να δημιουργήσει ένα δάσος που αποτελείται από δέντρα από το Asset Store. Σε αυτό το σημείο ο κόσμος που έχει δημιουργήσει ο χρήστης φαίνεται στο σχήμα 4.4.12.



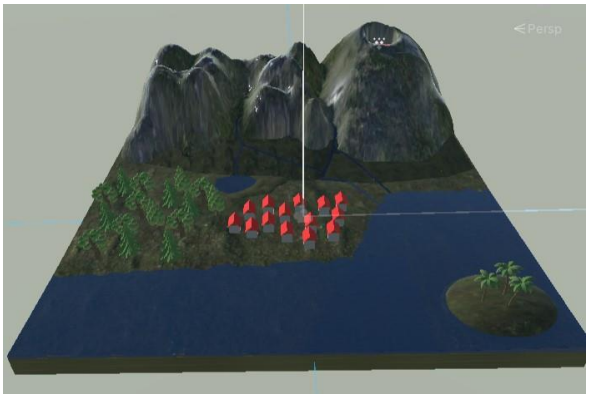
Σχήμα 4.4.12: Δάσος

Η επόμενη ερώτηση έχει ένα Text που ρωτάει ποιά από τις παρακάτω φωτογραφίες είναι λίμνη και τρεις φωτογραφίες (κουμπιά) με μια λίμνη, μια θάλασσα και ένα ποτάμι. Μόλις πατηθεί η σωστή φωτογραφία εμφανίζεται ένα κόκκινο πλαίσιο στο σημείο που θα δημιουργηθεί η λίμνη. Μόλις ο χρήστης πατήσει στο πλαίσιο και δημιουργήσει την λίμνη η οποία είναι το ίδιο shader με αυτό της θάλασσας και γύρω από αυτό υπάρχουν κάποιες παραμορφωμένες κάψουλες για να μην φαίνονται τα όρια της θάλασσας.

Μετά ζητείται από τον χρήστη να δημιουργήσει σπίτια για τους ανθρώπους τα οποία δημιουργήθηκαν από πέντε κύβους το καθένα, και ένα νησί έπειτα από ερώτηση σχετικά με το τι είναι ένα νησί. Το σωστό κουμπί είναι “Κομμάτι Γης που περιβάλλεται από νερό” και όταν πατηθεί εμφανίζεται το πλαίσιο που πρέπει να πατήσει ο παίκτης (σχήμα 4.4.13) για να δημιουργήσει το νησί το οποίο είναι μια παραμορφωμένη σφαίρα με 3 δέντρα πάνω της (σχήμα 4.4.14).



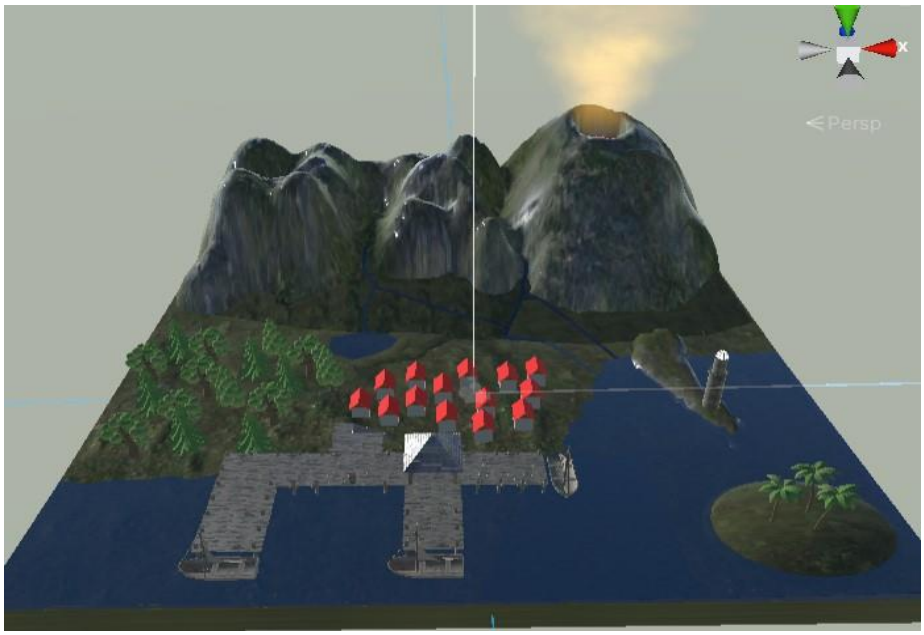
Σχήμα 4.4.13: CreateIsland



Σχήμα 4.4.14: Island

Αφού δημιουργηθεί το νησί, εμφανίζεται η επόμενη ερώτηση με αφορμή την προηγούμενη δημιουργία (το νησί) που ρωτάει τι χρειάζονται οι άνθρωποι που σε αυτό οι άνθρωποι φορτώνουν φορτία και επιβάτες για τα νησιά. Μόλις ο χρήστης πατήσει το σωστό κουμπί “Λιμάνι” τότε εμφανίζεται το κόκκινο πλαίσιο για να δημιουργηθεί το λιμάνι.

Τέλος εμφανίζεται η τελευταία ερώτηση σχετικά με κάποιον γεωγραφικό όρο “πώς ονομάζεται το τμήμα της ξηράς που εισχωρεί στη θάλασσα;”, η απάντηση είναι “Ακρωτήριο” και όταν επιλεχθεί εμφανίζεται το τελευταίο πλαίσιο που με το πάτημά δημιουργεί το ακρωτήριο το οποίο είναι Terrain, και από πάνω του υπάρχει ένας πύργος φτιαγμένος από Assets του Asset Store. Το τελικό αποτέλεσμα είναι αυτό του σχήματος 4.4.15. Στο τέλος εμφανίζονται τα πυροτεχνήματα (Particle System) και τελειώνει η πίστα με δύο κουμπιά “επανάληψη” και ¨επιστροφή”.



Σχήμα 4.4.15: Ο Κόσμος

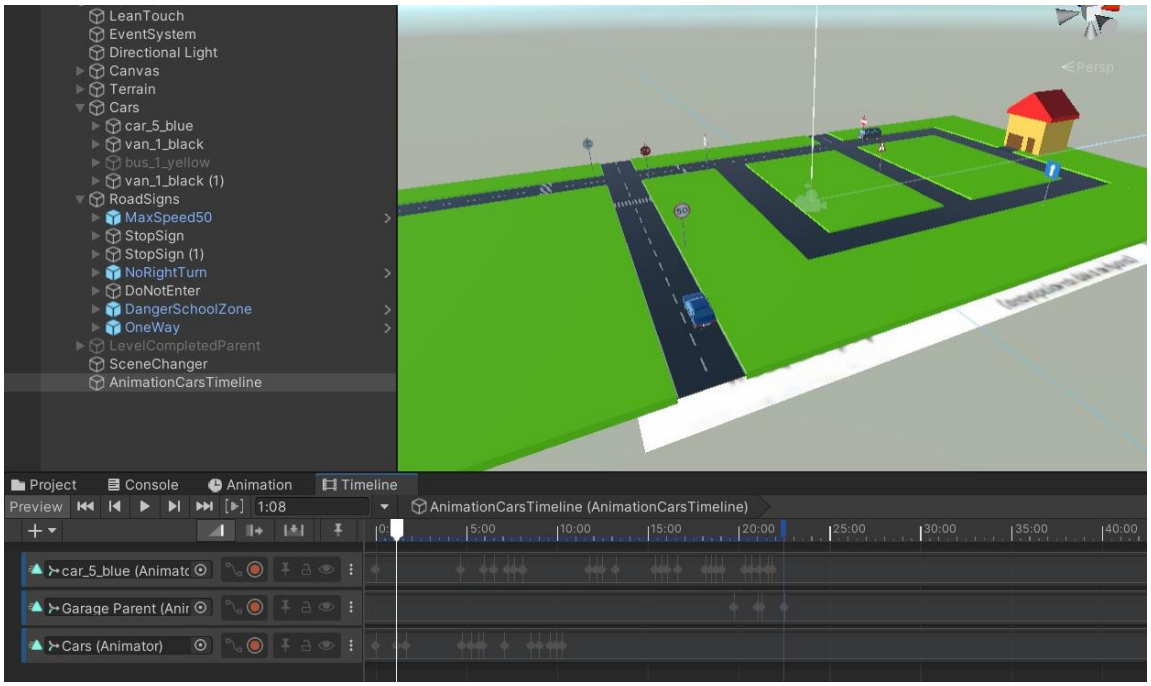
4.4.3 Μαθαίνουμε τα σήματα

Η Τρίτη άσκηση είναι εμπνευσμένη από το τρίτο κεφάλαιο "Κυκλοφορούμε με ασφάλεια" της τρίτης ενότητας του βιβλίου που αναφέρει τα σήματα του κώδικα οδικής κυκλοφορίας. Στόχος της άσκησης είναι ο χρήστης να βοηθήσει ένα αυτοκίνητο να φτάσει στο σπίτι του καθώς συναντάει κάποια σήματα στο δρόμο και απαντάει σε ερωτήσεις σχετικά με το πως πρέπει να αντιδράσει ο οδηγός σε κάθε ένα από τα σήματα που συναντάει.

Μόλις μπει ο χρήστης στο Scene της άσκησης μέσω του κεντρικού μενού, πρέπει να σαρώσει την φωτογραφία του σχήματος 4.4.16 που βρίσκεται στο τρίτο κεφάλαιο της τρίτης ενότητας.

Σχήμα 4.4.16: Εικόνα 3.3.1

Μόλις σαρωθεί η φωτογραφία εμφανίζεται το περιβάλλον του σχήματος 4.4.17 το οποίο δημιουργήθηκε κυρίως από κύβους, ενώ χρησιμοποιήθηκαν μερικά έτοιμα Assets όπως τα αυτοκίνητα και μερικές πινακίδες από το Asset Store.



Σχήμα 4.4.17: Ασκηση 3

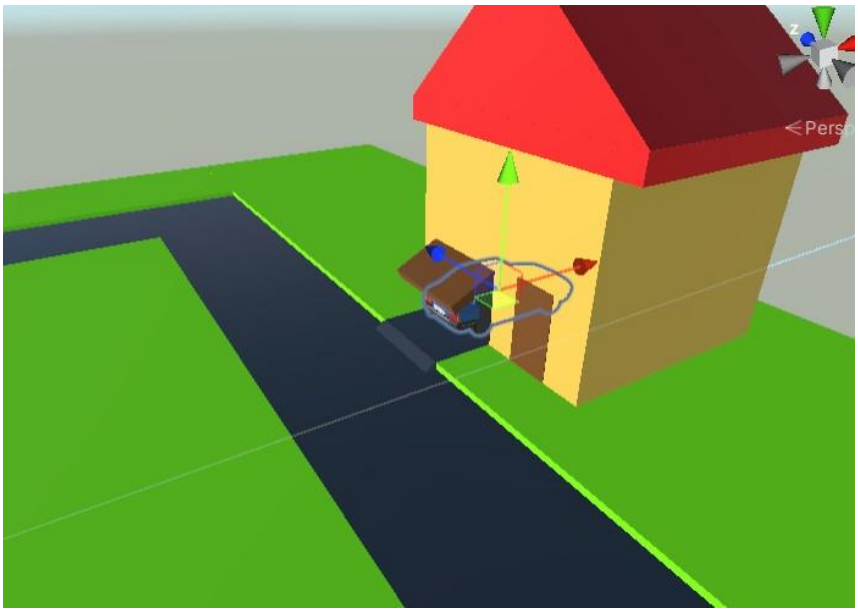
Υπάρχει ένα Timeline που ανήκει στο GameObject AnimationCarsTimeline και χρησιμοποιείται καθόλη την διάρκεια της άσκησης. Μέσω αυτού του Timeline κινείται το μπλε αμάξι, τα υπόλοιπα αμάξια και η πόρτα του γκαραζ για να μπεί μέσα το αυτοκίνητο στο τέλος της πίστας. Μόλις το αμάξι φτάσει κοντά σε κάποιο σήμα, καλείται μέσω του Timeline μια μέθοδος που σταματάει (pause) τον χρόνο του Timeline και παράλληλα εμφανίζεται η κατάλληλη ερώτηση, σε κάθε σωστή απάντηση εκτελείται μια άλλη μέθοδος που συνεχίζει το Timeline θέτοντας τον χρόνο στο 1.

Η πρώτη ερώτηση (σχήμα 4.4.18) αναφέρει πως ο οδηγός βλέπει την πινακίδα που θέτει το όριο ταχύτητας στα 50, και ρωτάει τι πρέπει να κάνει ο οδηγός αν πηγαίνει με 30. Η σωστή απάντηση είναι “Να μην υπερβεί το όριο ταχύτητας (50)” και μόλις πατηθεί συνεχίζει το Timeline.

Σχήμα 4.4.18: Ασκηση 3 πρώτη ερώτηση

Μόλις φτάσει στην πινακίδα stop εμφανίζεται η δεύτερη ερώτηση που αναφέρει πως υπάρχει μια πινακίδα Stop και πλησιάζει ένα κίτρινο λεωφορείο, και ρωτάει πώς πρέπει να αντιδράσει ο οδηγός. Η σωστή απάντηση είναι “Να διακόψει υποχρεωτικά την πορεία του μέχρι να περάσει το λεωφορείο”. Στο Timeline ο χρήστης βλέπει το αυτοκίνητο να σταματά και να περιμένει να περάσει το λεωφορείο. Μόλις περάσει, το μπλε αυτοκίνητο συνεχίζει την πορεία του στρίβοντας δεξιά και συναντά μια πινακίδα “απαγορεύεται η δεξιά στροφή” πριν το επόμενο στενό και η τρίτη ερώτηση εμφανίζεται ρωτώντας τι σημαίνει το συγκεκριμένο σήμα. Μόλις απαντήσει σωστά συνεχίζει το Timeline μέχρις ότου ο οδηγός συναντήσει το σήμα “απαγορεύεται η είσοδος”.

Η τέταρτη ερώτηση εμφανίζεται ρωτώντας τον χρήστη αν μπορεί ο οδηγός του αυτοκινήτου να προχωρήσει ευθεία (προς το σήμα). Η απάντηση είναι “Όχι επειδή απαγορεύεται η είσοδος”, οπότε το Timeline Συνεχίζει και το αυτοκίνητο στρίβει δεξιά. Ό οδηγός σταματάει σε ένα τρίγωνο σήμα με 2 ανθρωπάκια και εμφανίζεται η πέμπτη ερώτηση “Τί σημαίνει το διπλανό σήμα που συναντά ο οδηγός;”. Όταν ο χρήστης πατήσει το σωστό κουμπί “Κίνδυνος λόγω συχνής διέλευσης παιδιών” συνεχίζει το Timeline με το αυτοκίνητο να κατευθύνεται στο τελευταίο σήμα που είναι ο μονόδρομος. Εμφανίζεται η τελευταία ερώτηση η οποία αφορά το σήμα του μονόδρομου, και όταν απαντηθεί σωστά το αυτοκίνητο μετακινείται προς το γκαράζ, του οποίου η πόρτα ανοίγει για να μπει το αυτοκίνητο (σχήμα 4.4.19). Η πόρτα του γκαράζ είναι φτιαγμένη από ένα κύβο, για να γίνει όμως η κίνηση όπως αυτή του γκαράζ, υπάρχει ένας Parent της πόρτας ο οποίος είναι στο πάνω μέρος της πόρτας, έτσι περιστρέφοντας (rotate) τον Parent, η πόρτα προς τον άξονα Z, η πόρτα περιστρέφεται με βάση το Pivot, δηλαδή το σημείο που βρίσκεται ο Parent. Τέλος εμφανίζονται τα πυροτεχνήματα (Particle System) και τελειώνει η πίστα εμφανίζοντας δύο κουμπιά “επανάληψη” και ¨επιστροφή”.

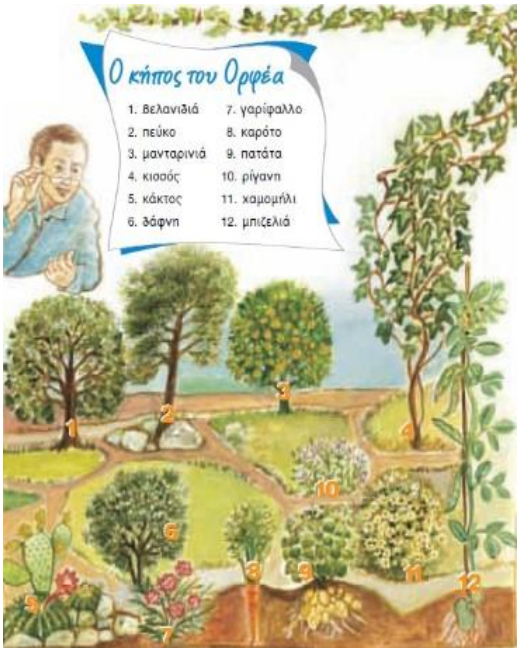


Σχήμα 4.4.19: Τέλος ασκησης 3

4.4.4 Τα φυτά του τόπου μας

Η τέταρτη άσκηση είναι εμπνευσμένη από τα κεφάλαια ένα μέχρι πέντε (1-5) της τέταρτης ενότητας του βιβλίου τα οποία αναφέρονται στα φυτά. Στόχος της άσκησης είναι να διδάξει στον χρήστη κάποιες βασικές πληροφορίες και έννοιες σχετικά με την βλάστηση του περιβάλλοντος μέσω 5 υποασκήσεων.

Μπαίνοντας στο Scene της άσκησης μέσω του κεντρικού μενού, ο χρήστης πρέπει να σαρώσει την φωτογραφία του σχήματος 4.4.20 που βρίσκεται στο τέταρτο κεφάλαιο της τέταρτης ενότητας. Μόλις σαρωθεί η φωτογραφία εμφανίζεται ένα μενού (σχήμα 4.4.21) από το οποίο ο χρήστης μπορεί να επιλέξει μια από τις 5 υποασκήσεις (κουμπιά) της άσκησης.

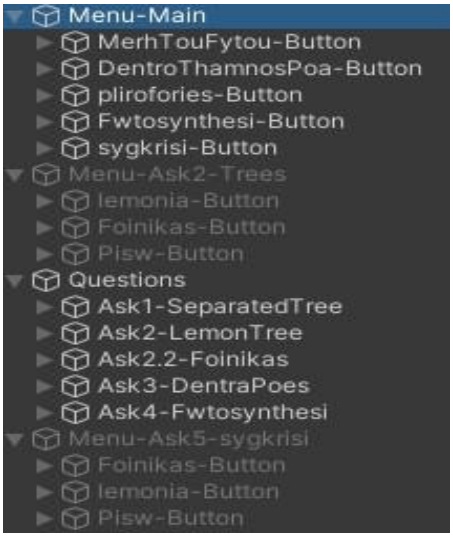


Σχήμα 4.4.20: Εικόνα 4.4.1

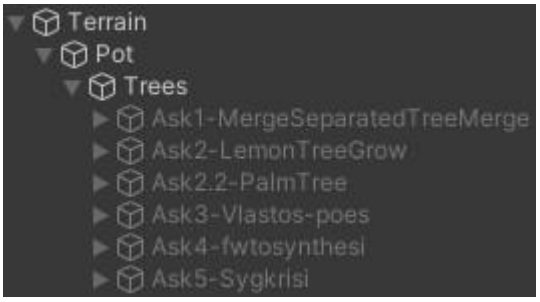
Σχήμα 4.4.21: Ασκηση 4 - MainMenu

Ως προς τον τρόπο που είναι ταξινομημένα τα GameObjects στην ιεραρχία, ξεκινώντας με τον Canvas, υπάρχει ο Parent "Arxi" που όπως σε όλες τις ασκήσεις περιέχει ένα Text που καλωσορίζει τον παίκτη και ένα κουμπί "Αρχή". Στην συνέχεια είναι το "Menu-Main" που περιέχει τα 5 κουμπιά των υποασκήσεων και κάθε κουμπί απενεργοποιεί το Menu-Main και ενεργοποιεί ένα από τα 5 menu που αφορά μια υποάσκηση το καθένα (σχήμα 4.4.22).

Από την άλλη μεριά, τα GameObjects ανήκουν σε ένα Parent "Terrain" (σχήμα 4.4.23). Μέσα στο Terrain υπάρχει ένα παιδί "Pot" που το ίδιο έχει ένα παιδί "Trees". Μέσα στο Trees υπάρχουν οι 5 Parents των υποασκήσεων και στην αρχή της άσκησης είναι όλα απενεργοποιημένα, ενώ ενεργοποιείται το κατάλληλο ανάλογα με το κουμπί που θα πατήσουμε στο Menu-Main.

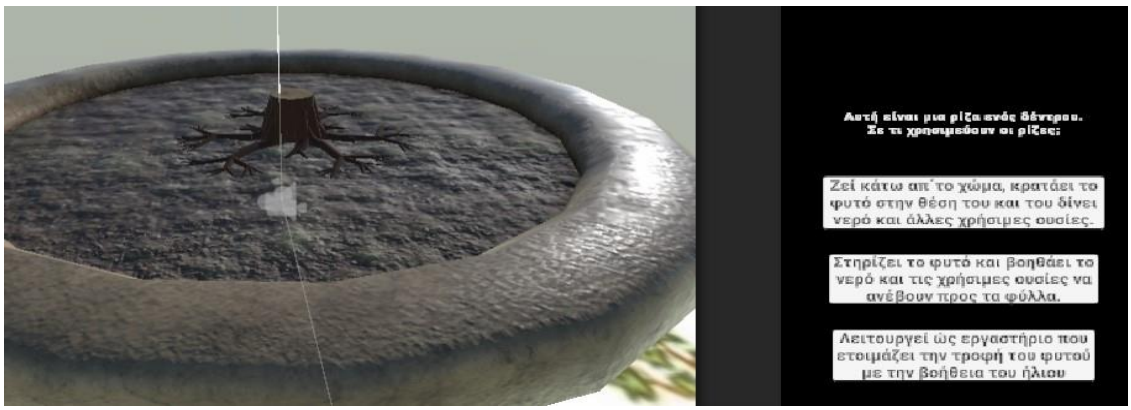


Σχήμα 4.4.22: Ασκηση 4 - UI Hierarchy



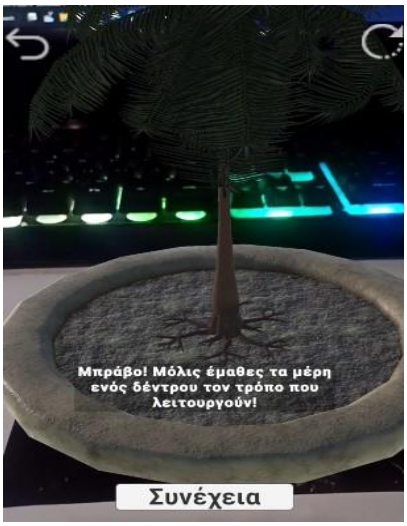
Σχήμα 4.4.23: Ασκηση 4 - Terrain Hierarchy

Η πρώτη υποάσκηση ονομάζεται "Τα μέρη του φυτού" και ο στόχος της είναι να διδάξει στον χρήστη τα τρία μέρη του φυτού και ποιός είναι ο ρόλος τους. Μόλις πατήσει ο χρήστης το κουμπί, απενεργοποιείται το κεντρικό Menu και ενεργοποιείται το "Ask1-SeparatedTree" από τον Canvas και παράλληλα ενεργοποιείται και ο Parent της πρώτης υποάσκησης από το Terrain. Η υποάσκηση ξεκινάει με ένα GameObject από το Asset Store που μοιάζει με μεγάλη γλάστρα, αλλά χρησιμοποιείται ως έδαφος για να στέκονται κάπου τα GameObjects και να μην είναι στον αέρα. Αρχικά εμφανίζεται μια ρίζα δέντρου που δημιουργήθηκε από πολλούς κυλίνδρους και μια ερώτηση σχετικά με αυτή (σχήμα 4.4.24).



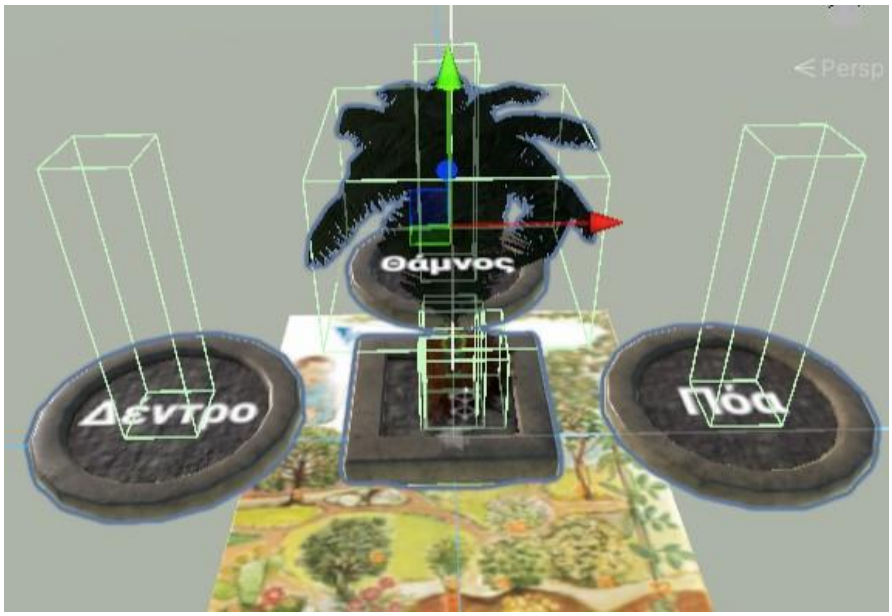
Σχήμα 4.4.24: Ασκηση 4 - Ρίζα

Έπειτα από την σωστή απάντηση εμφανίζεται ένας κορμός δέντρου δεξιά από την ρίζα και η δεύτερη ερώτηση σχετικά με τον κορμό ενώ οι απαντήσεις είναι οι ίδιες με αυτές της πρώτης ερώτησης. Μόλις απαντηθεί σωστά η ερώτηση, ο χρήστης πρέπει να τοποθετήσει τον κορμό πάνω στην ρίζα με σκοπό την δημιουργία ενός δέντρου. Τέλος εμφανίζονται τα φύλλα στα αριστερά της ρίζας και του κορμού, μαζί με την τρίτη ερώτηση αντίστοιχα με τις δύο προηγούμενες. Όταν απαντήσει σωστά ο χρήστης, πρέπει να μετακινήσει τα φύλλα πάνω από τον κορμό έτσι ώστε να ολοκληρωθεί το δέντρο. Το αποτέλεσμα της υποάσκησης εμφανίζεται στο σχήμα 4.4.25. Τέλος εμφανίζεται το αρχικό μενού και απενεργοποιούνται οι Parents της πρώτης υποάσκησης.



Σχήμα 4.4.25: Ασκηση 4 τέλος υποάσκησης 1

Η δεύτερη υποάσκηση ονομάζεται “¨Δεντρο, θάμνος και πόα” και διδάσκει στον χρήστη τι είναι δέντρο, τι θάμνος και τι πόα αντιστοιχίζοντας τα φυτά στις κατάλληλες κατηγορίες (σχήμα 4.4.26). Η υποάσκηση αποτελείται από 4 φυτά τα οποία έχουν Scripts για να κινούνται από τον χρήστη, και 3 περιοχές πέρα από το κέντρο που αντιπροσωπεύουν τις 3 κατηγορίες φυτών. Μέσα σε αυτές υπάρχουν Triggers με τα οποία γίνεται έλεγχος αν ο χρήστης έβαλε το φυτό στην σωστή κατηγορία. Επίσης υπάρχει και ένα κουμπί που δίνει πληροφορίες σε περίπτωση που ο χρήστης δεν γνωρίζει τι πρέπει να κάνει. Μόλις τελειώσει ο χρήστης την αντιστοίχιση τελειώνει η υποάσκηση και επανέρχεται το αρχικό μενού.

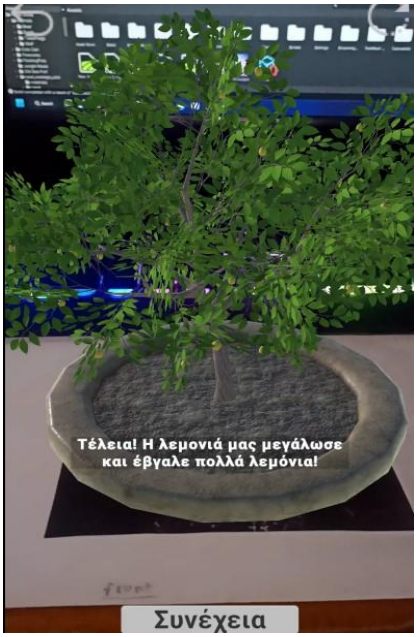


Σχήμα 4.4.26: Ασκηση 4 υποάσκηση 2

Η τρίτη υποάσκηση ονομάζεται “Quiz δέντρων”, πατώντας το κουμπί, εμφανίζονται άλλα τρία κουμπιά, ¨Ο Φοίνικας”, “Η Λεμονιά” και “Πίσω”. Το κουμπί Πίσω πηγαίνει πίσω στο αρχικό μενού με τις υποασκήσεις κλείνοντας το συγκεκριμένο μενού και ανοίγοντας το menu-main. Τα άλλα δύο κουμπιά ξεκινάνε μια σειρά από ερωτήσεις ως προς το δέντρο που θα επιλέξει ο χρήστης. Επιπλέον, το Asset της λεμονιάς είναι ένα μικρό δέντρο, και μετά τις ερωτήσεις ο χρήστης πρέπει να πατήσει πάνω στο δέντρο για να μεγαλώσει και να βγάλει λεμόνια(σχήμα 4.4.28). Στην πραγματικότητα

υπάρχουν 4 διαφορετικά Assets και ο τρόπος που λειτουργεί είναι πατώντας το Box Collider του κάθε δέντρου με το δάχτυλο (μέσω Script), απενεργοποιείται το παλιό δέντρο και εμφανίζεται το επόμενο. Μόλις εμφανιστεί και το τελευταίο δέντρο η υποάσκηση τελειώνει και εμφανίζεται το αρχικό μενού της άσκησης.

Σχήμα 4.4.27: Ασκηση 4 - Φοίνικας

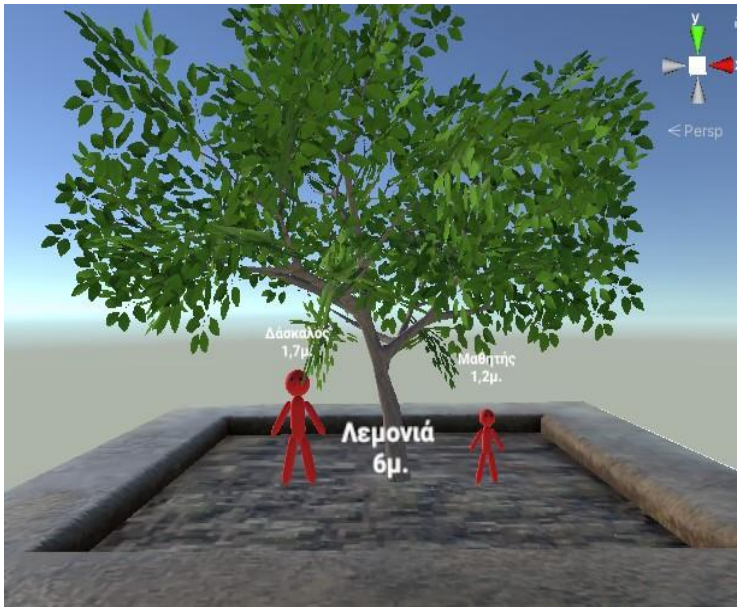


Σχήμα 4.4.28: Ασκηση 4 - Λεμονιά

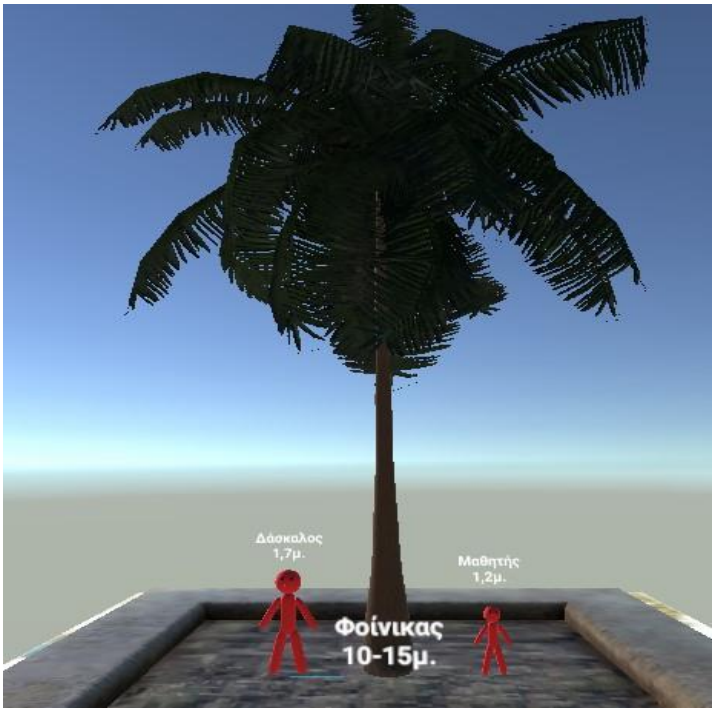
Η τέταρτη υποάσκηση ονομάζεται “φωτοσύνθεση” και σκοπός της είναι να διδάξει τον χρήστη τι είναι φωτοσύνθεση και πως λειτουργεί. Αρχικά εμφανίζεται ένα δέντρο πάνω σε ένα έδαφος με πέντε 3D Texts γύρω του (σχήμα 4.4.29) τα οποία έχουν Box Collider για να μπορούν να ενεργοποιήσουν το Trigger του δέντρου. Εμφανίζεται ένα Text το οποίο αναφέρει πως τα δέντρα μόνα τους, αρκεί να έχουν τα υλικά που χρειάζονται. Ο χρήστης πρέπει να σύρει τα τρία υλικά που χρειάζεται το φυτό για να παράξει μόνο του την ενέργεια. Μόλις σύρει και τα σωστά υλικά στο δέντρο, εμφανίζεται μία ερώτηση σχετικά με το τι παράγουν τα φυτά με τα υλικά που του δώσαμε. Οταν ο χρήστης πατήσει το σωστό κουμπί “οξυγόνο και γλυκόζη” τελειώνει η υποάσκηση και εμφανίζεται το αρχικό μενού της άσκησης.

Σχήμα 4.4.29: Ασκηση 4 υποάσκηση 4

Η τελευταία υποάσκηση είναι μια πληροφορία σχετικά με το ύψος του φοίνικα και της λεμονιάς. Πατώντας το κουμπί εμφανίζονται άλλα τρία κουμπιά, ένα “πίσω” και άλλα δύο για τα δύο δέντρα. Πατώντας ένα από τα δέντρα βλέπουμε την σύγκριση ενός από τα δέντρα με δύο ανθρωπάκια, ένα είναι ο μαθητής που έχει περίπου 1.2m ύψος και το άλλο είναι ο δάσκαλος που είναι λίγο μεγαλύτερος και έχει 1.7m ύψος. Στο σχήμα 4.4.30 εμφανίζεται η σύγκριση ύψους μεταξύ της λεμονιά, τον δάσκαλο και τον μαθητή και αντίστοιχα στο σχήμα 4.4.31 για τον φοίνικα.



Σχήμα 4.4.30: Ασκηση 4 σύγκριση ύψους



Σχήμα 4.4.31: Ασκηση 4 σύγκριση ύψους 2

4.4.5 Τα Ζώα του τόπου μας

Η πέμπτη άσκηση είναι εμπνευσμένη από τα κεφάλαια έξι μέχρι δέκα (6-10) της τέταρτης ενότητας του βιβλίου τα οποία αναφέρονται στα ζώα. Στόχος της άσκησης είναι να διδάξει στον χρήστη κάποιες πληροφορίες σχετικά με πέντε ζώα που υπάρχουν στο Scene.

Μπαίνοντας στο Scene της άσκησης μέσω του κεντρικού μενού, ο χρήστης πρέπει να σαρώσει την φωτογραφία του σχήματος 4.4.32 που βρίσκεται στο έβδομο κεφάλαιο της τέταρτης ενότητας.

Σχήμα 4.4.32: Εικόνα 4.7.1

Μόλις σαρωθεί η φωτογραφία εμφανίζονται τέσσερα ζώα, αρκούδα, λύκος, βάτραχος και αετός τα οποία προέρχονται από το Asset store μαζί με κάποια Animations που αντιπροσωπεύουν τις κινήσεις των ζώων (σχήμα 4.4.33).

Σχήμα 4.4.33: Ασκηση 5 Επιλογή Ζώου

Τα ζώα περιέχουν κάποιο Box Collider ώστε ο χρήστης να μπορεί να πατήσει κάποιο από τα ζώα (μέσω Script) ώστε να το επιλέξει για να μάθει πληροφορίες για αυτό. Μόλις πατήσει ο χρήστης κάποιο από τα ζώα, απενεργοποιείται ο Parent που περιέχει τα τέσσερα ζώα, και ενεργοποιείται ο Parent του ζώου και το UI με το μενού του. Το κάθε ζώο έχει ένα μενού με τέσσερα κουμπιά (σχήμα 4.4.34).

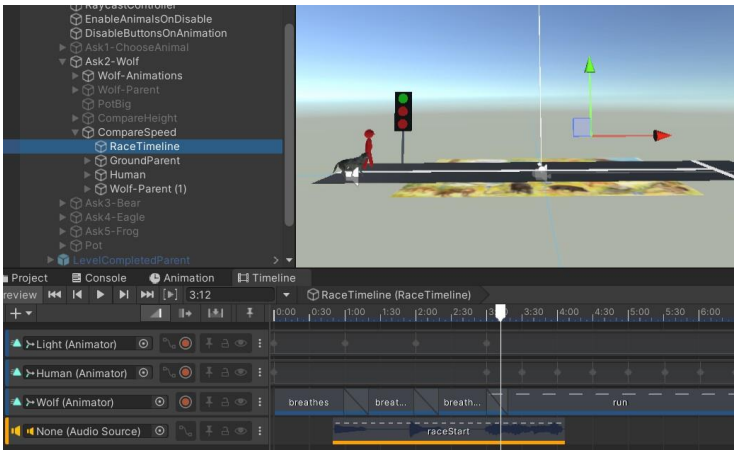
Σχήμα 4.4.34: Ασκηση 5 Wolf Menu

Το πρώτο ονομάζεται “κίνηση και ήχος”, πατώντας το, εμφανίζεται ένα άλλο μενού με πέντε κουμπιά, καθένα από αυτά αντιπροσωπεύει τις κινήσεις των ζώων οι οποίες δημιουργήθηκαν με Timeline που περιέχει και ήχους όπου χρειάζεται. Τα κουμπιά (Animations) του λύκου είναι “Τρέξιμο” που δείχνει τον λύκο να τρέχει, “Σκάψιμο” που δείχνει τον λύκο να σκάβει, “Περπάτημα” που δείχνει τον λύκο να περπατάει, “ουρλιαχτό” που δείχνει τον λύκο να ουρλιάζει με την συνοδεία ενός ηχητικού, και ¨Κάθισμα¨ που δείχνει τον λύκο να κάθεται. Υπάρχουν παρόμοια Animations και στα υπόλοιπα ζώα μαζί με ήχους.

Σχήμα 4.4.35: Ασκηση 5 Κίνηση και ήχος

Το δεύτερο κουμπί αφορά τρεις ερωτήσεις που σχετίζονται με το επιλεγμένο ζώο όπως το βάρος, την ομάδα που ανήκει (π.χ. θηλαστικό) και κάθε τρίτη ερώτηση ειδικεύεται συγκεκριμένα πάνω στο ζώο, για παράδειγμα στον βάτραχο η ερώτηση είναι “πόσα αυγά γεννάει την φορά;”.

Το τρίτο κουμπί αφορά την σύγκριση μεταξύ ανθρώπου και ζώου. Πατώντας το, εμφανίζεται ένα άλλο UI με δύο κουμπιά, “Σύγκριση ταχύτητας” και “Σύγκριση ύψους”. Στην σύγκριση ταχύτητας γίνεται ένας αγώνας ταχύτητας μεταξύ ανθρώπου και ζώου (σχήμα 4.4.36) μέσω Timeline στο οποίο τοποθετήθηκαν τα έτοιμα Animations του λύκου, ένα ηχητικό και Animation για το φανάρι (μετρητής), και δημιουργήθηκε επιπλέον Animation για την κίνηση του ανθρώπου. Η ταχύτητα των ζώων συγκριτικά με τον άνθρωπο δεν είναι τυχαίες και έχουν σχέση με την πραγματικότητα. Στην σύγκριση ύψους εμφανίζεται το περιβάλλον του σχήματος 4.4.37 για να δει ο χρήστης πως μοιάζει το ζώο δίπλα σε κάποιον άνθρωπο.



Σχήμα 4.4.36: Ασκηση 5 Σύγκριση ταχύτητας

Σχήμα 4.4.37: Ασκηση 5 Σύγκριση ύψους

4.5 Επίλογος

Η επαναχρησιμοποίηση είναι πολύ σημαντική λόγω του ότι δημιουργείται υλικό εύκολα και γρήγορα, γι'αυτό δημιουργήθηκαν κάποιες λειτουργίες η οποίες χρησιμοποιούνται σε πολλές ασκήσεις καλύπτοντας ένα μέρος της εφαρμογής. Η εφαρμογή αποτελείται από το κεντρικό UI στο οποίο μπαίνει ο χρήστης με το άνοιγμα της εφαρμογής. Το UI αποτελείται από οδηγίες χρήσης και κουμπιά που οδηγούν στις ασκήσεις. Μπαίνοντας στην άσκηση ο χρήστης πρέπει να σαρώσει την κατάλληλη φωτογραφία για να εμφανιστεί το περιβάλλον της άσκησης μέσω της κάμερας του χρήστη. Οι πέντε ασκήσεις έχουν ως στόχο να διδάξουν τον χρήστη την σημασία των κανόνων, τους γεωγραφικούς όρους, τα σήματα οδικής κυκλοφορίας, την χλωρίδα και την πανίδα του τόπου μας.

Κεφάλαιο 5ο: Συμπεράσματα

Οι εφαρμογές επαυξημένης πραγματικότητας δημιουργούν την αίσθηση πως υπάρχουν τρισδιάστατα εικονικά αντικείμενα στον πραγματικό κόσμο. Αυτό επιτυγχάνεται μέσω κάμερας της πλατφόρμας στην οποία χρησιμοποιείται μια τέτοια εφαρμογή. Στην παρούσα διπλωματική έχει υλοποιηθεί μια εφαρμογή επαυξημένης πραγματικότητας για το μάθημα Μελέτη περιβάλλοντος της Γ' δημοτικού με προσέγγιση εκπαιδευτικού παιχνιδιού της οποίας ο στόχος είναι να διδάξει τους μαθητές της Γ' δημοτικού βασικές έννοιες και ιδιότητες του τόπου μας μέσω ερωτήσεων κουίζ και αλληλεπίδρασης με το εικονικό περιβάλλον, με έναν πιό διασκεδαστικό τρόπο. Η εφαρμογή έχει δημιουργηθεί μέσω της μηχανής παιχνιδιού Unity με την οποία ο προγραμματιστής μπορεί εύκολα να δημιουργήσει την δική του εφαρμογή που μπορεί να είναι κάποιο παιχνίδι ή άλλη εφαρμογή που χρησιμοποιείται σε άλλους τομείς όπως την αρχιτεκτονική ή στον κινηματογράφο, με την βοήθεια των λειτουργιών που παρέχονται γενικότερα για την υλοποίηση της εφαρμογής ή ειδικότερα για την ένταξη των Components, δηλαδή των μεμονωμένων λειτουργιών που δινουν ιδιότητες στα GameObjects. Στο Unity μπορούν να συνδεθούν εξωτερικά εργαλεία (SDK), όπως το Vuforia Engine με το οποίο ο προγραμματιστής μπορεί να δημιουργήσει εφαρμογές επαυξημένης πραγματικότητας.

Πάντα υπάρχουν περιθώρια βελτίωσης σε όλες τις εφαρμογές. Η συγκεκριμένη εφαρμογή μπορεί να βελτιωθεί αρκετά με την βελτίωση των Scripts που χρησιμοποιήθηκαν, αλλά και την δημιουργία νέων επαναχρησιμοποιήσιμων λειτουργιών οι οποίες μπορούν να αντικαταστήσουν τις ήδη υπάρχουσες μεμονωμένες λειτουργίες, για την αποφυγή πολλών αρχείων, για ευκολότερη αποσφαλμάτωση αλλά και για διευκόλυνση του προγραμματιστή σε μελλοντική χρήση. Μια εξίσου σημαντική μελλοντική βελτίωση είναι δημιουργία και η ένταξη περισσότερων ασκήσεων έτσι ώστε η εφαρμογή να καλύπτει παραπάνω υλικό και να διδάσκει στον χρήστη περισσότερες γνώσεις που είναι χρήσιμες για το μέλλον του.

Συμπερασματικά, η διαδικασία υλοποίησης μιας ολοκληρωμένης εφαρμογής επαυξημένης πραγματικότητας είναι πολύ χρήσιμη και ενδιαφέρουσα εμπειρία, διότι πέρα από την απόκτηση νέων γνώσεων, δόθηκε η ευκαιρία για την δημιουργία μιας χρήσιμης εφαρμογής από την οποία οι χρήστες της θα μάθουν επιπλέον πληροφορίες για τον κόσμο μας, και ειδικότερα όταν η εφαρμογή στοχεύει σε μαθητές της Γ' δημοτικού, επιτρέποντας τους να μάθουν παράλληλα με τον πιό ευχάριστο τρόπο, το παιχνίδι.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Unity Documentation [Online]. Διαθέσιμο στο: https://docs.unity3d.com/Manual/index.html. Accessed: 2 Ιαν. 2022

[2] Vuforia Engine [Online]. Διαθέσιμο στο: https://developer.vuforia.com. Accessed: 28 δεκ. 2022

[3] Grahn, I. (2017). The vuforia sdk and unity3d game engine: Evaluating performance on android devices.

[4] Unity RP [Online]. Διαθέσιμο στο: https://docs.unity3d.com/Manual/render-pipelines.html. Accessed: 7 Ιαν. 2023

[5] Yuen, S. C. Y., Yaoyuneyong, G., & Johnson, E. (2011). Augmented reality: An overview and five directions for AR in education. Journal of Educational Technology Development and Exchange (JETDE), 4(1), 11.

[6] Chen, Y., Wang, Q., Chen, H., Song, X., Tang, H., & Tian, M. (2019, June). An overview of augmented reality technology. In *Journal of Physics: Conference Series* (Vol. 1237, No. 2, p. 022082). IOP Publishing.

[7] Howard, M. C., & Davis, M. M. (2022). A meta-analysis and systematic literature review of mixed reality rehabilitation programs: Investigating design characteristics of augmented reality and augmented virtuality. Computers in Human Behavior, 107197.

[8] Bryson, S. (1995). Approaches to the successful design and implementation of VR applications. Virtual reality applications, 3-15.

[9] Kipper, G., & Rampolla, J. (2012). Augmented reality: An emerging technologies guide to AR. Elsevier.

[10] Sutherland, I. E. (1968, December). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I* (pp. 757-764).

[11] Piekarski, W., & Thomas, B. H. (2003). ARQuake-Modifications and hardware for outdoor augmented reality gaming (Doctoral dissertation, Linux Australia).

[12] Henrysson, A., Billinghurst, M., & Ollila, M. (2006). AR tennis. In ACM SIGGRAPH 2006 Emerging technologies (pp. 1-es).

[13] Althoff, T., White, R. W., & Horvitz, E. (2016). Influence of Pokémon Go on physical activity: study and implications. Journal of medical Internet research, 18(12), e6759.

[14] Burdea, G. C., & Coiffet, P. (2003). Virtual reality technology. John Wiley & Sons.

[15] Gonzalez, D. A. Z., Richards, D., & Bilgin, A. A. (2021). Making it real: a study of augmented virtuality on presence and enhanced benefits of study stress reduction sessions. International Journal of Human-Computer Studies, 147, 102579

ΠΑΡΑΡΤΗΜΑ Α **: DentraPoesTrigger.cs**

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

public class DentraPoesTrigger : MonoBehaviour
{
    [SerializeField] GameObject Sunflower;
    [SerializeField] GameObject bush;
    [SerializeField] GameObject garlic;
    [SerializeField] GameObject tree;
    [SerializeField] GameObject mainMenu;

    [SerializeField] QuestionController qController;
    [SerializeField] GameObject Askisi3;

    Vector3 startingPosition;

    private void Start()
    {
        Sunflower.active = true;
        bush.active = false;
        garlic.active = false;
        tree.active = false;
        startingPosition = Sunflower.transform.position;
    }
    private void OnTriggerEnter(Collider other)
    {
        if (gameObject.transform.name == "TriggerDentro")
        {
            if (other.transform.name == Sunflower.transform.name) { qController.WrongAnswer(); }
            if (other.transform.name == garlic.transform.name) { qController.WrongAnswer(); }
```

```
            if (other.transform.name == bush.transform.name) { qController.WrongAnswer(); }


            if (other.transform.name == tree.transform.name) { qController.CorrectAnswer(); qController.HideAllQuestions(); Askisi3.active = false; mainMenu.active = true; tree.active = false; qController.ShowQuestion10Time(0); }
        }
        else if (gameObject.transform.name == "TriggerPoes")
        {
            if (other.transform.name == bush.transform.name) { qController.WrongAnswer(); }
            if (other.transform.name == tree.transform.name) { qController.WrongAnswer(); }


            if (other.transform.name == Sunflower.transform.name) { qController.CorrectAnswer(); bush.active = true; Sunflower.active = false; qController.ShowQuestion10Time(0); }
            if (other.transform.name == garlic.transform.name) { qController.CorrectAnswer(); tree.active = true; garlic.active = false; qController.ShowQuestion10Time(0); }
        }
        else if (gameObject.transform.name == "TriggerThamnos")
        {
            if (other.transform.name == Sunflower.transform.name) { qController.WrongAnswer(); }
            if (other.transform.name == tree.transform.name) { qController.WrongAnswer(); }
            if (other.transform.name == garlic.transform.name) { qController.WrongAnswer(); }


            if (other.transform.name == bush.transform.name) { qController.CorrectAnswer(); garlic.active = true; bush.active = false; qController.ShowQuestion10Time(0); }
        }
    }
    public void ResetTreePositions()
    {
        Sunflower.transform.position = startingPosition;
        bush.transform.position = startingPosition;
        garlic.transform.position = startingPosition;
        tree.transform.position = startingPosition;
    }
}
```

ΠΑΡΑΡΤΗΜΑ Β : DisableButtonOnAnimation.cs

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

public class DisableButtonsOnAnimation : MonoBehaviour
{
    [SerializeField] AudioSource audioSourceWolf;
    [SerializeField] AudioSource audioSourceBear;
    [SerializeField] AudioSource audioSourceEagle;
    [SerializeField] AudioSource audioSourceFrog;

    [SerializeField] GameObject[] ButtonsWolf;
    [SerializeField] GameObject[] ButtonsBear;
    [SerializeField] GameObject[] ButtonsEagle;
    [SerializeField] GameObject[] ButtonsFrog;


    public void disableButtonsWolf() { foreach (GameObject button in ButtonsWolf) { button.active = false; } }
    public void enableButtonsWolfTime(float time) { Invoke("enableButtonsWolf", time); }
    void enableButtonsWolf() { foreach (GameObject button in ButtonsWolf) { button.active = true; } }

    public void disableButtonsBear() { foreach (GameObject button in ButtonsBear) { button.active = false; } }
    public void enableButtonsBearTime(float time) { Invoke("enableButtonsBear", time); }
    void enableButtonsBear(){ foreach (GameObject button in ButtonsBear) { button.active = true; } }

    public void disableButtonsEagle() { foreach (GameObject button in ButtonsEagle) { button.active = false; } }
    public void enableButtonsEagleTime(float time) { Invoke("enableButtonsEagle", time); }
```

```
    void enableButtonsEagle() { foreach (GameObject button in ButtonsEagle) { button.active = true; }
}


    public void disableButtonsFrog() { foreach (GameObject button in ButtonsFrog) { button.active =
false; } }
    public void enableButtonsFrogTime(float time) { Invoke("enableButtonsFrog", time); }
    void enableButtonsFrog() { foreach (GameObject button in ButtonsFrog) { button.active = true; } }
}
```

ΠΑΡΑΡΤΗΜΑ C **CheckIfParked.cs**

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

public class CheckIfParked : MonoBehaviour
{
    [SerializeField] GameObject movingCar;
    [SerializeField] GameObject fakeCar;
    [SerializeField] GameObject question6;
    [SerializeField] GameObject RedTipSquare;
    [SerializeField] AudioClip CorrectSound;

    private void OnTriggerEnter(Collider other)
    {
        movingCar.SetActive(false);
        fakeCar.SetActive(true);
        gameObject.active = false;
        Invoke("enableQuestion6", 1);
        AudioSource.PlayClipAtPoint(CorrectSound, fakeCar.transform.position);
    }
    void enableQuestion6()
    {
        question6.active = true;
    }
    private void Update()
    {
        if (!RedTipSquare.active) RedTipSquare.active = true;

    }
}
```