Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία Επιπέδου Α΄

Τίτλος: Παιχνίδι τύπου Open World Single Player Shooter RPG

Βλαχάκης Βασίλειος (ΤΠ3998)

Υπεύθυνος Καθηγητής: **Παχουλάκης Ιωάννης**

Ευχαριστίες

Θα ήθελα να εκφράσω τη βαθύτατη εκτίμηση και την απόλυτη ευγνωμοσύνη μου στον υπεύθυνό μου καθηγητή Παχουλάκη Ιωάννη. Εξαιρετικές επιστημονικές γνώσεις, εμπειρία και δημιουργική σκέψη ήταν πηγή έμπνευσης και κίνητρο για μένα, ενώ η υπομονή και η ενθάρρυνσή του ήταν καθοριστικές για την ολοκλήρωση αυτής της εργασίας.

Περίληψη

Η εργασία είναι ένα παιχνίδι τύπου Open World Single Player Shooter RPG και είναι διαθέσιμη σε Windows και Android. Δημιουργήθηκε στην παιχνιδομηχανή Unity3D με χρήση εργαλείων του Unity3D και την γλώσσα προγραμματισμού C# για το scripting κομμάτι. Στο παιχνίδι υπάρχουν τρεις διαφορετικές περιοχές.

Η πρώτη είναι το χωριό όπου ξεκινάει και ο παίχτης στο οποίο υπάρχουν πέντε NPCs. Ο κάθε NPC έχει και ένα quest που δεν είναι υποχρεωτικό αλλά βοηθάει τον παίχτη να τελειώσει το παιχνίδι ευκολότερα. Όταν ο παίχτης τελειώσει το quest με επιτυχία, τότε το παιχνίδι του δίνει την ανταμοιβή του που περιέχει XPs και ίσως κάποιο αντικείμενο για να γίνει πιο δυνατός. Αφού πάρει το quest, μπορεί να πάει προς την πόρτα εξόδου για να κάνει τα quests και να πάρει κι' άλλα XPs.

Η δεύτερη πίστα έχει τέσσερεις διαφορετικούς τύπους τεράτων και βοηθητικά σημεία για τον παίχτη. Σε αυτή την πίστα γίνονται τα quests και σκοτώνοντας τα τέρατα ο παίχτης λαμβάνει XP ανάλογα το επίπεδό τους (όσο πιο μεγάλο επίπεδο τόσο πιο πολλά XPs). Τα βοηθητικά σημεία της πίστας είναι τρία σημεία τηλεμεταφοράς σε σημεία της πίστας και ένα σημείο που γεμίζει την ζωή του παίχτη. Σκοπός της πίστας είναι να δυναμώσει ο παίχτης όσο πιο πολύ μπορεί ώστε να σκοτώσει το δυνατότερο τέρας της πίστας και να εμφανιστεί η είσοδος της σπηλιάς του Boss.

Η τρίτη πίστα είναι η σπηλιά του Boss. Το Boss είναι το δυνατότερο τέρας του παιχνιδιού. Όταν το σκοτώσει ο παίχτης ουσιαστικά τελειώνει το παιχνίδι όμως υπάρχει η δυνατότητα να συνεχίσει απλά να δυναμώνει σκοτώνοντας τέρατα. Σαν ''αποδεικτικό'' ότι ο παίχτης τερμάτισε το παιχνίδι, ο παίχτης έχει πάνω του ένα ιδιαίτερο αντικείμενο (παίρνοντας βέβαια το ανάλογο quest).

Κατά την έναρξη του παιχνιδιού υπάρχει ένα μενού με επιλογή έναρξης ή αποχώρησης από το παιχνίδι. Έπειτα, με την επιλογή έναρξη, ο παίχτης ξεκινάει από την πρώτη πίστα αν τρέξει πρώτη φορά το παιχνίδι αλλιώς πηγαίνει στην πίστα και στο σημείο που αποθήκευσε την προηγούμενη φορά που έπαιξε. Αν σκοτωθεί από αντίπαλο, διαιρούνται τα συνολικά XPs του παίχτη δια δύο (πολύ πιθανό να μειωθεί το επίπεδό του) και έπειτα του εμφανίζεται ένα μενού με τις επιλογές ξαναπροσπάθησε και έξοδος από το παιχνίδι. Στην επιλογή ξαναπροσπάθησε μεταφέρεται στο χωριό. Πάνω δεξιά της οθόνης υπάρχει ένα μενού όπου ο παίχτης μπορεί να αλλάξει όπλο, να αλλάξει την ποιότητα των γραφικών, να αποθηκεύσει την τοποθεσία του και να κάνει έξοδο του παιχνιδιού. Για λόγους δυσκολίας τα XPs αποθηκεύονται αυτόματα.

Τέλος, ο χρήστης χρειάζεται να έχει πληκτρολόγιο και ποντίκι για να μπορεί να έχει πλήρως τις λειτουργίες του παιχνιδιού. Στην έκδοση του Android χρειάζεται μόνο ένα κινητό με λειτουργικό Android για να μπορεί να παίξει. Η έκδοση του κινητού είναι παρόμοια με του υπολογιστή, δηλαδή έχει τέσσερα κουμπιά πάνω στην οθόνη για το περπάτημα και τα υπόλοιπα γίνονται με την αφή. Επίσης, η έκδοση αυτή έχει κάποιες λειτουργίες λιγότερες αλλά όχι τόσο σημαντικές για το παιχνίδι.

Abstract

The project is an Open World Single Player Shooter RPG and it is available on Windows and Android. It was created on the Unity3D game engine using Unity3D tools and the programming language C# for the scripting part. There are three different scenes in the game.

The first scene is the village, where the player starts, with five NPCs. Each NPC have a quest that is not required to pass but it helps the player to finish the game more easily. When the player finishes the quest successfully, then the game gives him his reward containing XP and maybe an object to become stronger. After player receives the quest, he can go to the exit door to complete the quests and get more XPs.

The second scene have four different types of monsters and helping points for the player. In this scene, the quests can be completed and if player is killing the monsters, he receives XP according to their level (the larger the level the more XPs he receives). The helping points of the scene are three points that teleports the player at points of the scene and one point that refills the player's life points. The goal of the scene is to boost the player with XPs as much as possible to kill the strongest monster of the scene and unlock the entrance to the Boss Cave.

The third scene is the Boss Cave. The Boss is the strongest monster of the whole game. When the player kills it the game ends, but there is the probability to continue the game and boost the player by killing monsters. As a "proof" that the player has finished the game, he will have on him a particular object (he must take the Boss quest).

At the beginning of the game there is a menu with an option to start or quit from the game. Then, with the start option, the player starts from the first scene if he runs the game for the first time, otherwise he goes to the scene where he saved the last time he played. If player killed by an opponent, his total XPs are divided by two (very possible to drop his level) and then game displays a menu with the retry button to try again and exit button to exit the game. At the option retry, player will be transferred to the village. At the top right of the screen there is a menu button where the player can change weapon, the quality of graphics, save his position and exit the game. For difficulty reasons, XPs are automatically saved.

In the end, the user needs a keyboard and mouse to be able to have all of the game functions. In the Android version, only one mobile with Android is required to play. The mobile version is like computer's version, it have four buttons on the screen for walking and the functions are done with touchscreen. Also, mobile version have some features less but not so important for the game.

Περιεχόμενα

Πίνακας Εικόνων

Ακρωνύμια

NPC	(Non Player Character) Χαρακτήρας του παιχνιδιού που δεν τον χειρίζεται ο παίχτης
Quest	Αποστολή με ανταμοιβή
XPs	Πόντοι εμπειρίας για ενδυνάμωση του παίχτη
Boss	Τελικός και δυνατότερος αντίπαλος
Android	Λειτουργικό σύστημα σε κινητά
C#	Αντικειμενοστρεφής γλώσσα προγραμματισμού
MMORPG	RPG με πολλούς παίχτες μέσω διαδικτύου
Arcade	Τύπος παιχνιδιού

1 Εισαγωγή

1.1 Κίνητρο για την διεξαγωγή της πτυχιακής εργασίας

Το κίνητρο για την διεξαγωγή της εργασίας αυτής ήταν η θέληση για την δημιουργία ενός βιντεοπαιχνιδιού το οποίο το είχα φανταστεί και ήθελα να το κάνω πραγματικότητα. Επίσης, επειδή ήταν δικιά μου σαν σκέψη, το σενάριο και η υλοποίηση της εργασίας γινόταν σχεδόν ταυτόχρονα μέχρι να καταλήξω στο τελικό αποτέλεσμα. Άλλο ένα κίνητρο ήταν οι δυνατότητες της παιχνιδομηχανής Unity που επιτρέπει στους χρήστες της να το διεξάγουν σε πολλές πλατφόρμες κάνοντας μόνο κάποιες ελάχιστες αλλαγές στην εργασία, όπως και έκανα για το λειτουργικό Android. Τέλος, ένα ακόμα κίνητρο ήταν ο πρωτότυπος τρόπος εύρεσης σφαλμάτων (σε σχέση με της άλλες γλώσσες προγραμματισμού) που πρέπει να παίξεις το παιχνίδι για να τα βρεις αν τυχόν υπάρχουν.

1.2 Σκοπός και Στόχοι Εργασίας

Σκοπός της εργασίας είναι η ψυχαγωγία, η κίνηση του ενδιαφέροντος και η ώθηση για στρατηγική σκέψη κατά την διάρκεια του παιχνιδιού. Επίσης, άλλος ένας σκοπός είναι η ανάδειξη των δυνατοτήτων της Unity για την δημιουργία ενός παιχνιδιού.

Στόχος της εργασίας ήταν η εξοικείωση και εκμάθηση της γλώσσας C# και την όλη λογική του τρισδιάστατου προγραμματισμού και σχεδιασμού.

1.3 Δομή Εργασίας

Η εργασία αυτή ξεκινάει κάνοντας μία εισαγωγή σε βασικές έννοιες που έχουν σχέση με τους τύπους παιχνιδιών που είναι η συγκεκριμένη εργασία.

Στην συνέχεια, στο επόμενο κεφάλαιο γίνεται αναφορά για τις βασικές έννοιες και τους βασικούς ορισμούς του παιχνιδιού ώστε να είναι κατανοητή η αναφορά στα επόμενα κεφάλαια.

Στην τρίτη ενότητα υπάρχει ανάλυση για την παιχνιδομηχανή Unity στο ρόλο της και στα βασικά πράγματα και εργαλεία που έχει και χρησιμοποιούνται κυρίως στην κατασκευή παιχνιδιού όπως τα είδη παραθύρων και των βασικών components για το παιχνίδι.

Στην τέταρτη ενότητα γίνεται αναφορά για τα κύρια αντικείμενα στο παιχνίδι καθώς και η συμπεριφορά τους.

Στην πέμπτη ενότητα γίνεται η ανάλυση του παιχνιδιού μαζί με τους C# κώδικες των scripts. Σε ότι γίνεται η ανάλυση είναι μηχανισμοί οι οποίοι είναι φτιαγμένοι από το μηδέν από εμένα.

Στην έκτη ενότητα αναφέρω τις διαφορές μεταξύ των δύο εκδόσεων του παιχνιδιού που δημιούργησα λόγο της ιδιαιτερότητας και των μέσων που διαθέτει η κάθε πλατφόρμα (Windows, Android).

Στην τελευταία ενότητα γίνεται ανάλυση στο gameplay και στις δυσκολίες κατά την υλοποίηση του παιχνιδιού καθώς επίσης και προτάσεις για βελτίωση του παιχνιδιού στο μέλλον.

2 Ανάλυση Εννοιών

2.1 Τι είναι Open World;

Το Open World είναι ένας όρος για βιντεοπαιχνίδια όπου ο παίκτης μπορεί να κινηθεί ελεύθερα μέσω ενός εικονικού κόσμου έχοντας μεγάλη ελευθερία όσον αφορά το πώς και πότε να εκτελέσει συγκεκριμένους στόχους σε αντίθεση με άλλα βιντεοπαιχνίδια που έχουν μια πιο συγκεκριμένη δομή. Επίσης μπορεί να συναντήσουμε παιχνίδι με τους όρους free roam ή sandbox (πιο σπάνια) όπου σημαίνουν το ίδιο.

Τα Open World βιντεοπαιχνίδια συνήθως στερούνται τους αόρατους τοίχους και τις οθόνες φόρτωσης που είναι κοινά σε σχέδια γραμμικού επιπέδου, όσο μεγαλύτερος είναι ο κόσμος τόσο πιο πολύ υπάρχει αυτή η στέρηση. Γενικά, τα Open World παιχνίδια εξακολουθούν να επιβάλλουν πολλούς περιορισμούς στο περιβάλλον του παιχνιδιού, είτε λόγω τεχνικών περιορισμών είτε λόγω περιορισμών εντός του παιχνιδιού που επιβάλλονται από τη πορεία του παιχνιδιού. Παραδείγματα υψηλού επιπέδου αυτονομίας στα παιχνίδια υπολογιστών μπορούν να βρεθούν σε τύπου MMORPG ή σε παιχνίδια Single Player που ακολουθούν την έννοια του ανοιχτού κόσμου, όπως η σειρά Fallout. Η κύρια ουσία του Open World παιχνιδιού είναι ότι παρέχουν μια προσομοιωμένη πραγματικότητα και επιτρέπουν στους παίκτες να αναπτύξουν οι παίχτες τον χαρακτήρα τους και τη συμπεριφορά τους προς την κατεύθυνση της επιλογής τους. Στις περιπτώσεις αυτές, συχνά δεν υπάρχει συγκεκριμένος στόχος ή τέλος στο παιχνίδι.

2.2 Τι είναι RPG (Παιχνίδι Ρόλων);

Ένα παιχνίδι ρόλων (που μερικές φορές περιγράφεται ως παιχνίδι ρόλων και συντομογραφία σε RPG) είναι ένα παιχνίδι στο οποίο οι παίκτες αναλαμβάνουν τους ρόλους των χαρακτήρων σε ένα φανταστικό περιβάλλον. Οι παίκτες αναλαμβάνουν την ευθύνη για την ανάληψη αυτών των ρόλων μέσα σε μια αφήγηση, είτε μέσω συγκεκριμένων ενεργειών είτε μέσω μιας διαδικασίας λήψης αποφάσεων ή ανάπτυξης του χαρακτήρα. Οι ενέργειες που λαμβάνονται σε πολλά παιχνίδια επιτυγχάνουν ή αποτυγχάνουν σύμφωνα με το σύστημα των κανόνων και σκοπών. Υπάρχουν πολλές διαφορετικές μορφές RPG.

2.3 Τι είναι παιχνίδι Shooter;

Τα Shooter παιχνίδια είναι μια υποκατηγορία των παιχνιδιών δράσης, τα οποία συχνά έχουν να κάνουν με την ταχύτητα και τον χρόνο αντίδρασης του παίκτη. Περιλαμβάνουν πολλά κοινά χαρακτηριστικά που επικεντρώνονται στις ενέργειες του παίχτη χρησιμοποιώντας κάποιο είδος όπλου. Συνήθως αυτό το είδος όπλου είναι ένα όπλο κοντινής ή μακριάς εμβέλειας. Ένας κοινό χαρακτηριστικό που υπάρχει σε πολλά παιχνίδια Shooter είναι τα πυρομαχικά (σφαίρες). Συνήθως, ο σκοπός ενός Shooter παιχνιδιού είναι να πυροβολάς τους αντιπάλους και να προχωράς σε αποστολές χωρίς να σκοτωθεί ή να πεθάνει ο χαρακτήρας του παίκτη. Ένα παιχνίδι Shooter είναι ένα είδος βιντεοπαιχνιδιού όπου ο παίκτης έχει περιορισμένο εδαφικό έλεγχο του χαρακτήρα του και η εστίαση είναι σχεδόν εξ' ολοκλήρου στην εξολόθρευση των εχθρών του χαρακτήρα χρησιμοποιώντας όπλα μεγάλης εμβέλειας.

2.4 Τι είναι παιχνίδι Single Player;

Ένα Single Player παιχνίδι είναι ουσιαστικά ένα παιχνίδι με μόνο έναν παίκτη που χειρίζεται το παιχνίδι καθ' όλη τη διάρκεια της περιόδου του. Σε κάποια παιχνίδια υπάρχει το "mode single-player" που είναι συνήθως ένας τρόπος παιχνιδιού που έχει σχεδιαστεί για να παιχτεί από έναν παίκτη, αν και το παιχνίδι περιέχει και λειτουργίες πολλαπλών παικτών.

Η συντριπτική πλειοψηφία των σύγχρονων παιχνιδιών κονσόλας και των παιχνιδιών arcade σχεδιάζονται έτσι ώστε να μπορούν να παιχτούν από έναν παίκτη. Αν και πολλά από αυτά τα παιχνίδια έχουν τρόπους που επιτρέπουν σε δύο ή περισσότερους παίκτες να παίζουν (όχι απαραίτητα ταυτόχρονα), πολύ λίγοι χρειάζονται περισσότερους από έναν παίκτες για να παίξουν το παιχνίδι. Η σειρά Unreal Tournament είναι ένα παράδειγμα τέτοιου είδους.

3 Unity

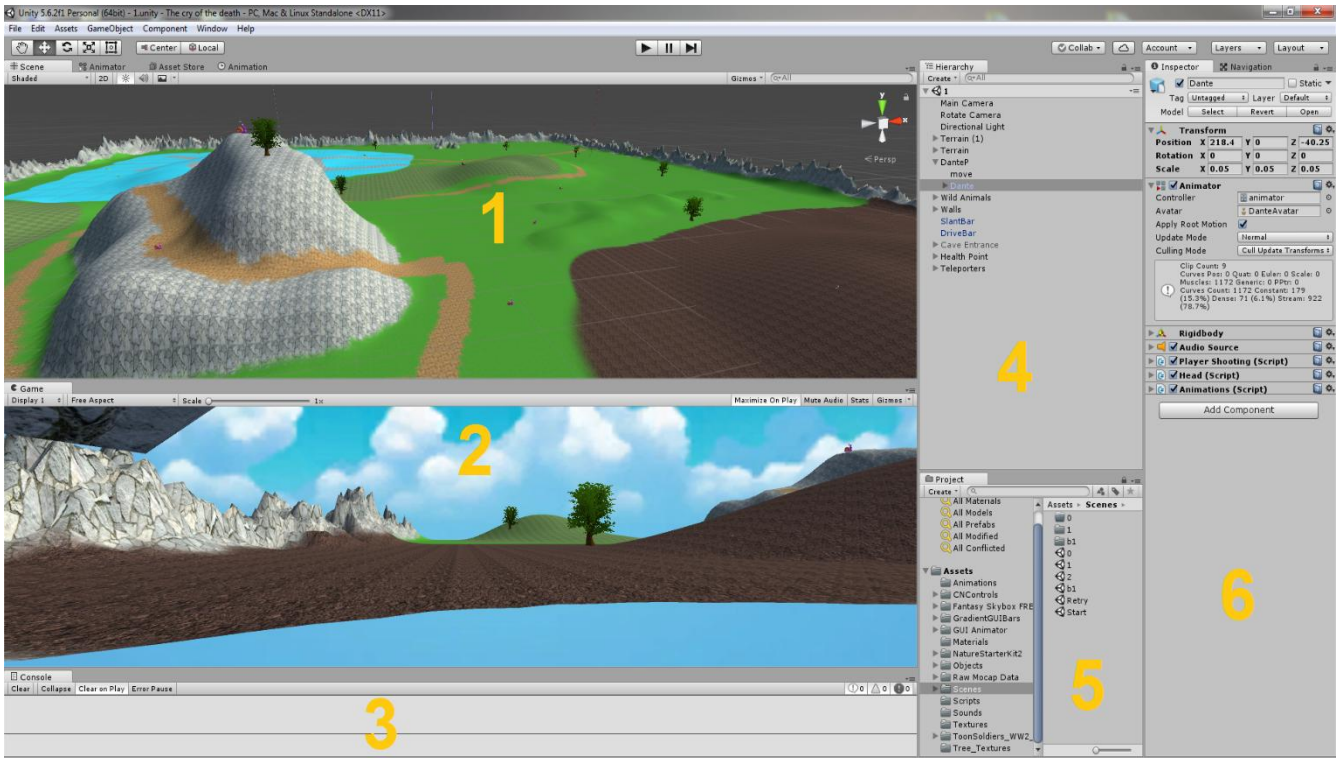
3.1 Τι είναι η Unity;

Η Unity είναι μια παιχνιδομηχανή πολλαπλών πλατφορμών που αναπτύχθηκε από την Unity Technologies, η οποία χρησιμοποιείται κυρίως για την ανάπτυξη βιντεοπαιχνιδιών και προσομοιώσεων για υπολογιστές, κονσόλες και κινητές συσκευές που υποστηρίζει γραφικά 2D και 3D, λειτουργία drag and drop και scripting μέσω C#. Δύο άλλες γλώσσες προγραμματισμού υποστηρίχθηκαν: Boo, η οποία απολύθηκε με την απελευθέρωση των Unity 5 και UnityScript η οποία καταργήθηκε τον Αύγουστο του 2017 μετά την απελευθέρωση της Unity 2017.1. Η UnityScript είναι μια ιδιόκτητη γλώσσα δέσμης ενεργειών, η οποία είναι συντακτικά παρόμοια με τη JavaScript. Ο κινητήρας στοχεύει τα ακόλουθα API γραφικών: Direct3D σε Windows και Xbox One. OpenGL σε Linux, MacOS και Windows. OpenGL ES σε Android και iOS. WebGL στον ιστό και ιδιόκτητων API στις κονσόλες βιντεοπαιχνιδιών. Επιπλέον, η Unity υποστηρίζει τα APIs χαμηλού επιπέδου σε iOS και macOS και Vulkan σε Android, Linux και Windows, καθώς και το Direct3D 12 στα Windows και το Xbox One. Μέσα στα 2D παιχνίδια, η Unity επιτρέπει την εισαγωγή των sprites και ενός προηγμένου 2D παγκόσμιου renderer. Για τα παιχνίδια 3D, η Unity επιτρέπει προδιαγραφές για τις ρυθμίσεις συμπίεσης και ανάλυσης render-to-texture για κάθε πλατφόρμα που υποστηρίζει ο μηχανισμός παιχνιδιών και παρέχει υποστήριξη για χαρτογράφηση συγκρούσεων, χαρτογράφηση ανάκλασης, χαρτογράφηση παράλλαξης, απομόνωση χώρου οθόνης SSAO, δυναμικές σκιές με χάρτες σκιάς, render-to-texture και full-screen post-processing effects. Η Unity προσφέρει επίσης υπηρεσίες σε προγραμματιστές, όπως: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity IAP, Unity Multiplayer, Unity Performance Reporting and Unity Collaborate.

Η Unity είναι αξιοσημείωτη για την ικανότητά της να στοχεύει παιχνίδια για πολλαπλές πλατφόρμες. Οι πλατφόρμες που υποστηρίζονται σήμερα είναι Android, Android TV, Facebook Gameroom, Fire OS, Gear VR, Google Cardboard, Google Daydream, HTC Vive, iOS, Linux, MacOS, Microsoft HoloLens, Nintendo 3DS family, Nintendo Switch, Oculus Rift, PlayStation 4, PlayStation Vita, PlayStation VR, Samsung Smart TV, Tizen, TVOS, WebGL, Wii U, Windows, Windows Phone, Windows Store και Xbox One. Η Unity παλαιότερα υποστήριξε 7 άλλες πλατφόρμες συμπεριλαμβανομένου του δικού της Unity Web Player. Το Unity Web Player ήταν ένα plugin του προγράμματος περιήγησης που υποστηριζόταν μόνο στα Windows και το OS X, το οποίο έχει καταργηθεί λόγο του WebGL.

3.2 Το περιβάλλον εργασίας της Unity;

Το περιβάλλον εργασίας της Unity είναι φιλικό προς τον χρήστη και προσφέρει πολλές δυνατότητες και ευελιξία στον χρήστη να το προσαρμόσει στα μέτρα του. Στην Εικόνα 1 φαίνονται τα συμαντικότερα υποπαράθυρα που υπάρχουν και εξηγούνται παρακάτω με την σειρά και τον αριθμό που έχει κάθε υποπαράθυρο.



Εικόνα 1 Το Περιβάλλον της Unity

1. Scene Window
2. Game Window
3. Console Window
4. Hierarchy Window
5. Project Window
6. Inspector

3.2.1 Scene Window

Το Scene Window (Παράθυρο Σκηνής) περιέχει όλη την επιλεγμένη πίστα. Ουσιατικά είναι το συνολικό στήσιμο των τρισδιάστατων αντικειμένων και του φωτός όπου δίνει το τελικό οπτικό αποτέλεσμα. Μέσω αυτού του παραθύρου υπάρχει και η δυνατότητα του στησίματος των αντικειμένων με την βοήθεια του ποντικιού.

3.2.2 Game Window

Το Game Window (Παράθυρο Παιχνιδιού) είναι το παράθυρο όπου ανοίγει όταν εκτελούμε να παίξει το παιχνίδι ώστε να κάνουμε αποσφαλμάτοση του παιχνιδιού. Στο παράθυρο αυτό έχουμε την δυνατότητα να κλείσουμε τον ήχο του παιχνιδιού (σε περίπτωση που ενοχλεί ο ήχος) και να ρυθμίσουμε το μέγεθος του παραθύρου για να δούμε πως τρέχει σε υψηλές και σε χαμηλές αναλύσεις σε περίπτωση που κάνουμε το παιχνίδι να είναι responsive.

3.2.3 Console Window

Το Console Window (Παράθυρο της Κονσόλας) είναι το παράθυρο όπου τυπώνει τα warnings, errors και hints της Unity τα οποία είναι μόνο για την ανοιγμένη σκηνή. Σε περίπτωση που τα warnings και errors είναι από κάποιο script, με την χρήση διπλού κλικ πηγαίνει στο αντίστοιχο script και στην συγκεκριμένη γραμμή που είναι το πρόβλημα. Επίσης χρησιμοποιώντας την εντολή Debug.Log(), όταν εκτελεστεί εμφανίζετε η τιμή επιστροφής της σαν hint στο Console Window.

3.2.4 Hierarchy Window

Στο Hierarchy Window (Παράθυρο Ιεραρχίας) υπάρχει η ιεραρχία όλων των αντικείμενων που υπάρχουν στην ανοιγμένη σκηνή. Κάνοντας κλικ σε ένα από τα αντικείμενα, φαίνονται τα περιεχόμενά τους στον Inspector.

3.2.5 Project Window

Το Project Window (Παράθυρο Εργασίας) ουσιαστικά περιέχει όλη την ιεραρχία του φακέλου Assets που είναι υποφάκελος της εργασίας. Δηλαδή, περιέχει όλα τα αντικείμενα, ήχους, εικόνες κ.α. που χρησιμοποιούνται στην εργασία. Επιπλέον, υπάρχει η δυνατότητα drag-n-drop των αρχείων στον Inspector και στο Scene Window καθώς και η προεπισκόπησή τους.

3.2.6 Inspector Window

Το Inspector Window (Παράθυρο Ελένχου) εμφανίζει ότι ιδιότητες και χαρακτηριστικά υπάρχουν σε ένα αντικείμενο που έχει επιλεχθεί από το Hierarchy. Επίσης, οποιαδήποτε ρύθμιση είναι δημοσίως ανοιχτή (public) μπορεί να ρυθμιστεί από τον χρήστη.

3.3 Βασικά εργαλεία και έννοιες στην Unity

Εδώ περιγράφονται οι βασικές και σημαντικές έννοιες που χρησιμοποιούνται ως ιδιότητες και τα χαρακτηριστικά σε περισσότερα αντικείμενα στα παιχνίδια της Unity.

3.3.1 Prefab

Prefab είναι ένα GameObject με ιδιότητες και χαρακτηριστικά. Το Prefab φτιάχνεται εισάγοντας ένα GameObject, μετά μέσω του Inspector του εισάγουμε τις ιδιότητες και χαρακτηριστικά που θέλουμε και τέλος το αποθηκεύουμε σαν αρχείο. Το Prefab αν χρησιμοποιείται σε πολλά Scenes, μπορούμε να επιλέξουμε το αρχείο του και μέσω του Inspector να το ρυθμίσουμε και να έχει τις ίδιες ρυθμίσεις σε όλα τα Scenes.

3.3.2 Components

Components είναι οι ιδιότητες και τα χαρακτηριστικά των GameObjects. Το GameObject από μόνο του είναι ένα στατικό αντικείμενο, βάζοντας όμως components το "ζωντανεύουν" και το κάνουν να έχει κάποιο χαρακτήρα, χρήσιμο προς το ίδιο το παιχνίδι και μερικές φορές είναι ευφυές. Τα πιο συνηθισμένα είναι τα transform, scripts και colliders.

3.3.3 GameObject

GameObject είναι αντικείμενο που υπάρχει μέσα στο παιχνίδι. Το GameObject δημιουργείται όταν εισάγουμε κάποιο αρχείο που αναγνωρίζει η Unity στο Hierarchy. Βάζοντάς του components αλλάζει η συμπεριφορά του μέσα στο παιχνίδι.

3.4 Βασικά Components στην Unity

Χωρίς τα components στην Unity δεν υπάρχει παιχνίδι. Χωρίς αυτά υπάρχει ένα απλό τρισδιάστατο περιβάλλον που δεν γίνεται τίποτα. Παρακάτω θα αναφέρω και θα αναλύσω τα σημαντικότερα από αυτά.

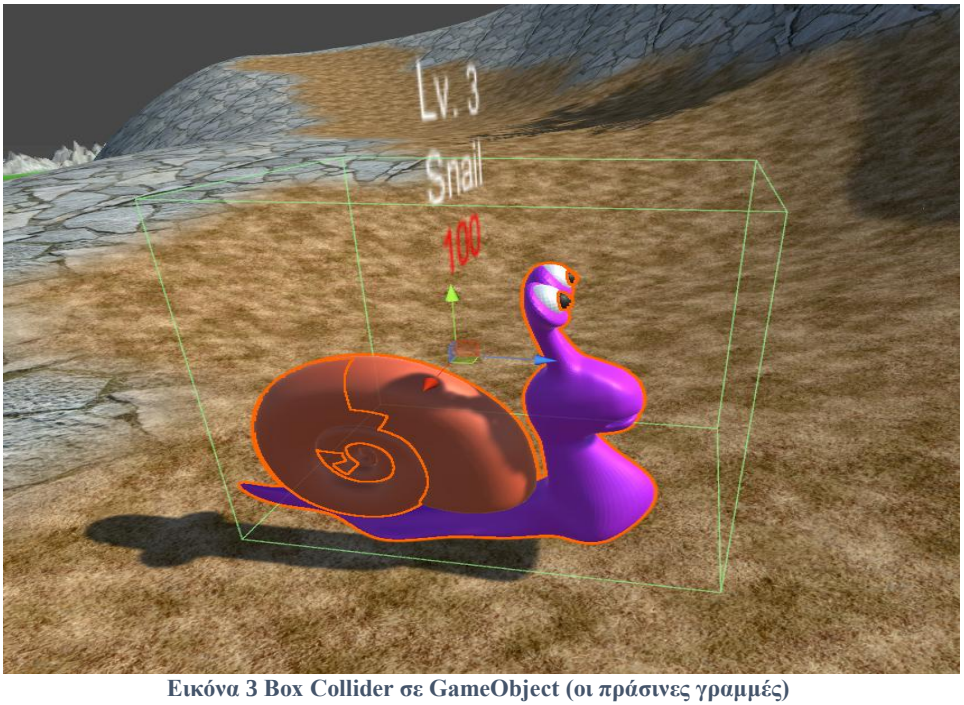
3.4.1 Audio Listener

Το Audio Listener είναι τα "αυτιά" του παιχνιδιού. Μέσω αυτού γίνεται αναπαραγωγή των ήχων που υπάρχουν στα σημεία του παιχνιδιού.

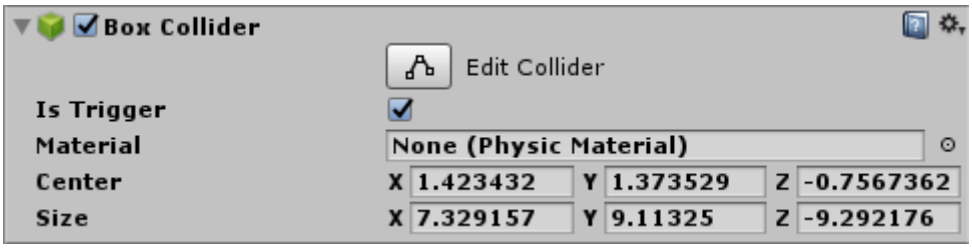
Εικόνα 2 Audio Listener

3.4.2 Collider

Ο Collider χρησιμοποιείται για ανίχνευση συγκρούσεων και αποτρέπει τον παίχτη να διαπεράσει μέσα από ένα άλλο αντικείμενο. Υπάρχει σχήμα στον collider που μπορούμε να διαλέξουμε ανάμεσα όμως σε βασικά σχήματα όπως σφαίρα, κύβος κ.α.. Επίσης, μπορούμε να κάνουμε τον collider τύπου triggered και σε περίπτωση σύγκρουσης να εκτελεστεί κάποιο script.



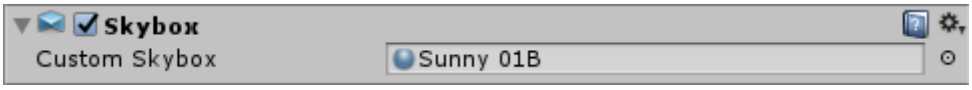
Εικόνα 3 Box Collider σε GameObject (οι πράσινες γραμμές)



Εικόνα 4 Box Collider Component

3.4.3 Skybox

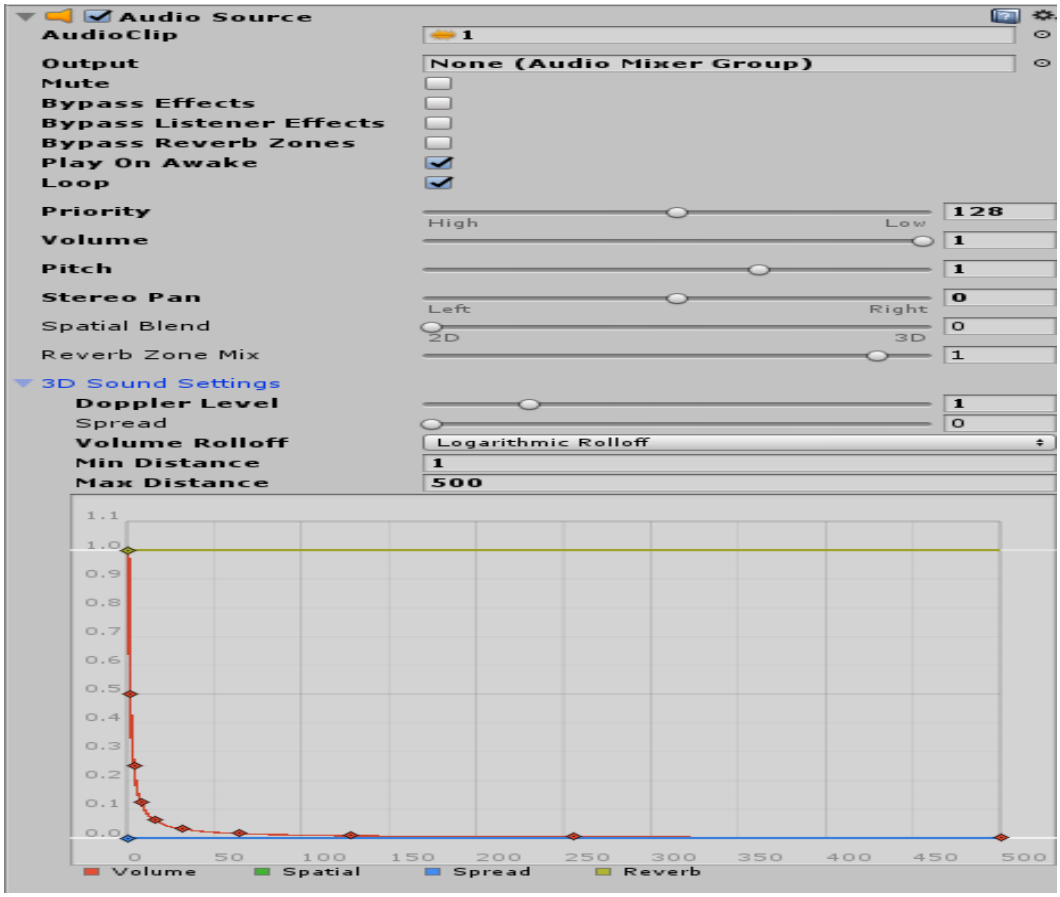
Skybox είναι μια σφαιρική εικόνα η οποία μπαίνει στην κενή ατμόσφαιρα του παιχνιδιού. Το component αυτό μπαίνει στην κάμερα και ουσιαστικά καλύπτει τα άχρωμα σημεία που βλέπει η κάμερα. Συνήθως σε ανοιχτούς χώρους το skybox είναι ο ουρανός.



Εικόνα 5 Skybox

3.4.4 Audio Source

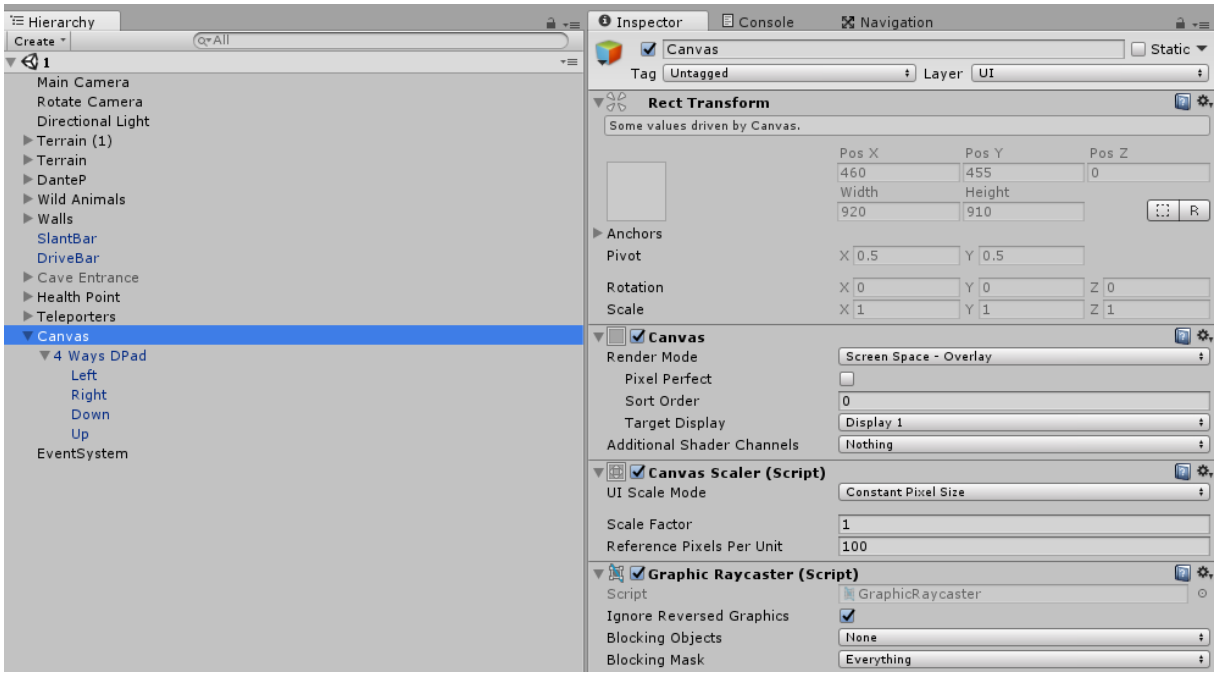
Το Audio Source είναι ένα σημείο που αναπαράγει ήχο. Βάζοντας ένα Audio Source σε κάποιο GameObject, είναι σαν να του τοποθετούμε ένα ηχείο που παράγει ένα συγκεκριμένο ήχο έχοντας σαν αποτέλεσμα να υπάρχει η δυνατότητα αναπαραγωγής διαφορετικού ήχου με βάση το σημείο που βρίσκεται ο παίχτης. Επίσης, υπάρχει η δυνατότητα για ρύθμιση του ήχου όπως ισοσταθμιστής, Left Right, ένταση και ρύθμιση έντασης στο χώρο.



Εικόνα 6 Audio Source

3.4.5 Canvas

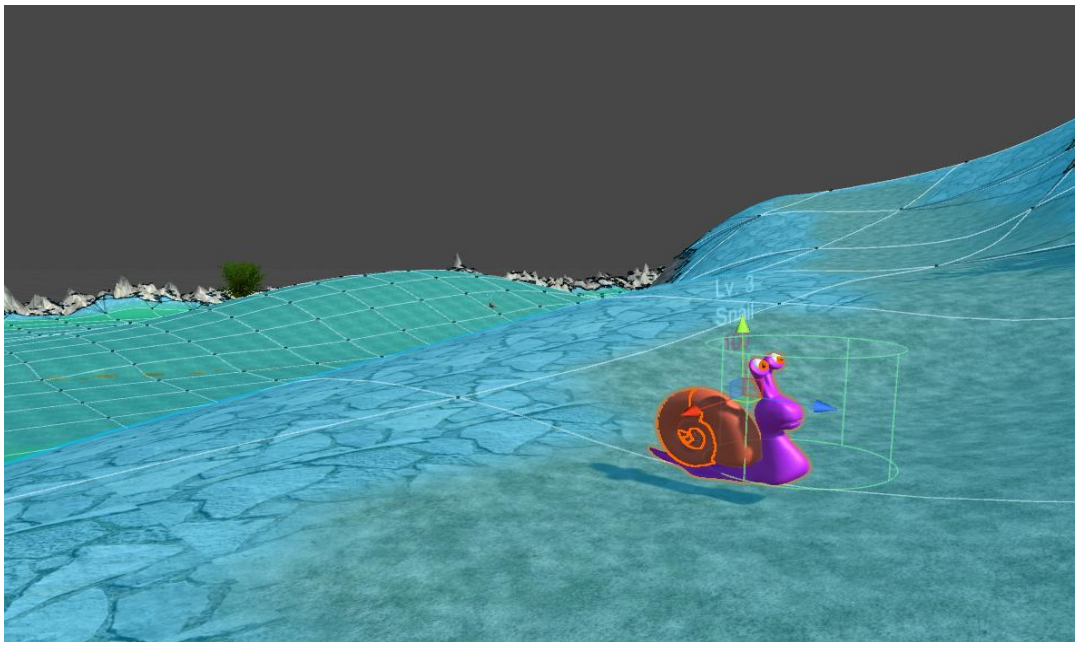
Canvas είναι ένα δισδιάστατο διαφανές αντικείμενο όπου έχει την ιδιότητα να είναι μόνιμα σε όλη την οθόνη του παίχτη. Στο Canvas συνήθως βάζουμε μπάρες, εικόνες και soft joystics, touch pads για τις εκδόσεις Android.



Εικόνα 7 Canvas

3.4.6 Nav Mesh Agent

Το Nav Mesh Agent δίνει την δυνατότητα σε ένα GameObject να ακολουθάει τον παίχτη ακριβώς στο σημείο που είναι. Το component αυτό χρησιμοποιεί τεχνητής νοημοσύνης για να βρει τον παίχτη και ταυτόχρονα μετακινεί και το GameObject που το έχει. Για να γίνει όλα αυτό το σύστημα πρέπει μέσω του Navigation Window και επιλεγμένο από τον Inspector το Terrain, να γίνει Bake ώστε να δημιουργηθεί μια μπλε τρισδιάστατη επιφάνεια που είναι ακριβώς η ίδια με την μορφολογία του Terrain.

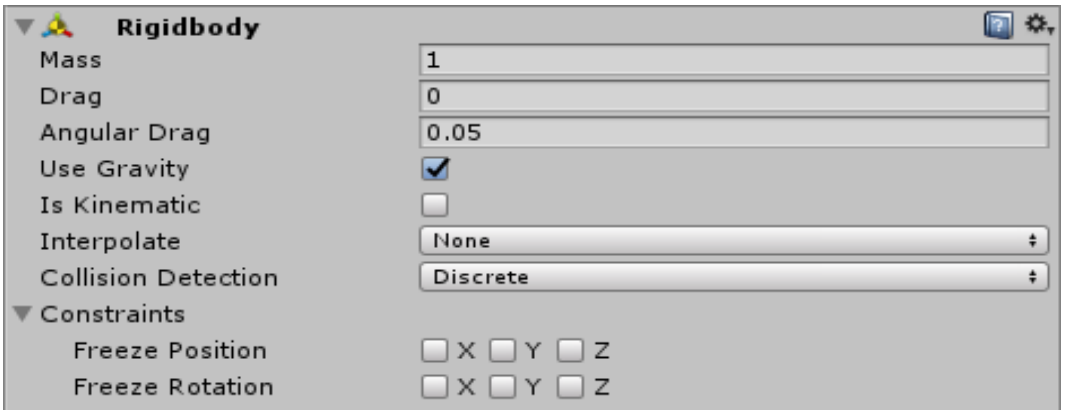


Εικόνα 8 Μπλε Επιφάνεια Navigator και Nav Mesh Agent σε GameObject

Εικόνα 9 Nav Mesh Agent

3.4.7 Rigidbody

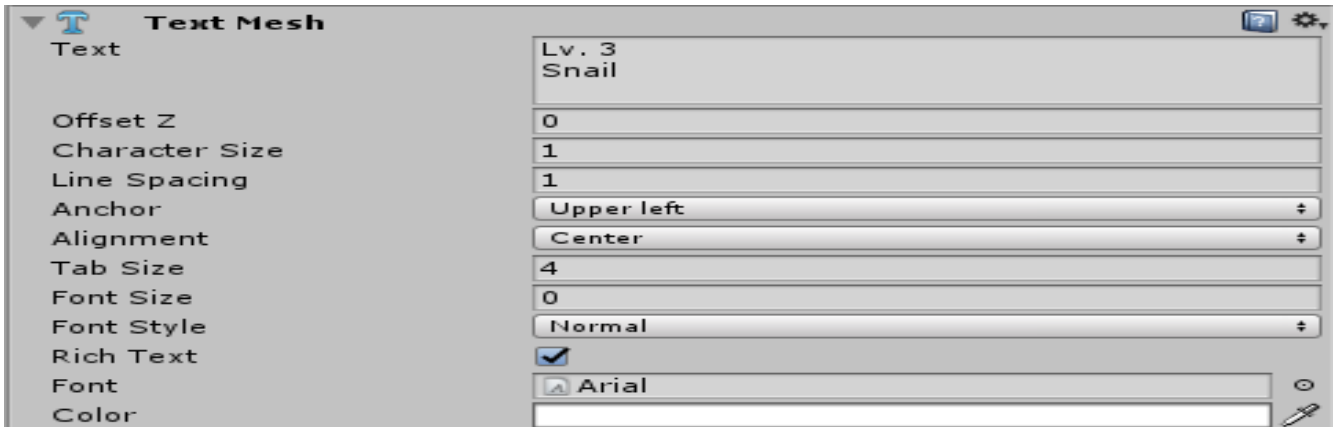
Το Rigidbody δίνει την ιδιότητα σε ένα GameObject να έχει και να δέχεται φυσικά φαινόμενα δυνάμεων, όμως χρειάζεται να έχει και ένα non-triggered collider. Βάζοντας Rigidbody στο GameObject, τότε αρχικά έχει τα βασικά φυσικά φαινόμενα όπως βαρύτητα, τριβή και αντίδραση όταν δέχεται δυνάμεις. Η Unity έχει κάποια συγκεκριμένα νούμερα για τις βασικές της δυνάμεις που μπορούν όμως να αλλαχτούν και να προγραμματιστούν με την βοήθεια κάποιου script.



Εικόνα 10 Rigidbody με τις βασικές τιμές

3.4.8 Text Mesh

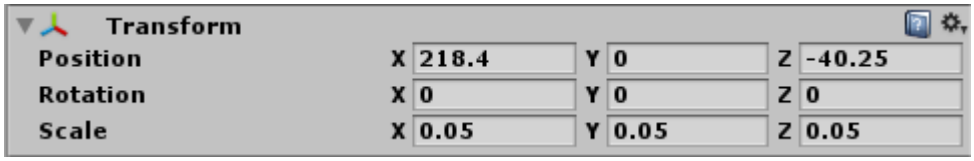
Text Mesh είναι ένα δισδιάστατο component με τρισδιάστατη παρουσία που περιέχει κάποια συμβολοσειρά. Υπάρχει και η δυνατότητα βασικής τροποποίησης της συμβολοσειράς.



Εικόνα 11 Text Mesh

3.4.9 Transform

Transform είναι η θέση, περιστροφή και κλίμακα ενός GameObject στο χώρο. Οτιδήποτε οπτικά αντιληπτό υπάρχει μέσα στο παιχνίδι έχει αναγκαστικά Transform. Στο Scene Window υπάρχει ένα εργαλείο με τρία βελάκια που αντιπροσωπεύει τους τρεις άξονες XYZ όπου μπορούμε τραβώντας το να κινήσουμε το GameObject μέσα στο χώρο αλλάζοντας αυτόματα τις αντίστοιχες τιμές του Transform.



Εικόνα 12 Transform

4 Εισαγωγή παιχνιδιού

Σε αυτό το κεφάλαιο θα αναφέρω όλα τα GameObjects και τον ρόλο τους μέσα στο παιχνίδι ανά κατηγορία. Θα αναφέρω τους NPCs, τον παίχτη με τα χαρακτηριστικά του, τις πίστες, τα λειτουργικά GameObject, μενού οθόνης και τους αντιπάλους.

4.1 Χαρακτήρας του παίχτη

Ο χαρακτήρας του παίχτη ονομάζεται Dante[21]. Επειδή το παιχνίδι είναι Shooter έχει τρεις διαφορετικούς τύπους όπλων τους machine gun, shotgun και sniper. Αναλόγως την κίνηση που κάνει, παίζει και το ανάλογο animation. Πάνω αριστερά στην οθόνη είναι η μπάρα με το ποσοστό της ζωής του παίχτη και στο τέλος της οθόνης είναι η μπάρα των XPs και το επίπεδο του παίχτη. Όταν κουνιέται η κάμερα δεξιά και αριστερά κουνιέται παράλληλα και ο παίχτης έτσι ώστε να είναι πιο εύχρηστο για τον χρήστη. Υπάρχει όμως και μία δεύτερη κάμερα που γυρνάει χωρίς να κουνιέται ο παίχτης. Με το αριστερό κλικ, ο παίχτης πυροβολεί ακριβώς στην κατεύθυνση που κοιτάει. Τα όπλα του είναι τρία και είναι machine gun, shotgun και sniper όπου μόνο με το sniper πατώντας δεξί κλικ, ο παίχτης σημαδεύει μέσα από τη δίοπτρα.

Εικόνα 13 Χαρακτήρας του παίχτη

4.2 Αντίπαλοι

Οι τύποι αντιπάλων είναι πέντε, τα worms[16], τα snails[17], τα birds[18], τα frogs[19] και ο boss[20]. Κάθε είδος αντιπάλου εκτός το boss, είναι από τρία μέχρι πέντε σε πλήθος συν τον βασιλιά τους που είναι μεγαλύτερος και σε ύψος και σε επίπεδο. Οι τέσσερεις αυτοί τύποι είναι στην δεύτερη πίστα διασκορπισμένοι μέσα στην πίστα. Ο boss είναι στην τρίτη πίστα όπου είναι και ο πιο δυνατός. Πυροβολώντας και πετυχαίνοντας έναν από τους αντιπάλους, τότε οι κοντινοί του και ίδιου τύπου αντίπαλοι επιτίθονται και σε περίπτωση που πυροβοληθεί ο βασιλιάς τότε όλοι οι αντίπαλοι ίδιου τύπου επιτίθονται. Παρακάτω είναι η περιγραφή των αντιπάλων:

Worms:

Worm:
Επίπεδο: 1
Ζωή: 100
Ταχύτητα: 10
XPs στον παίχτη: 25

Worm King:
Επίπεδο: 2
Ζωή: 100
Ταχύτητα: 10
XPs στον παίχτη: 50

Snails:

Snail:
Επίπεδο: 3
Ζωή: 100
Ταχύτητα: 22
XPs στον παίχτη: 100

King Snail:
Επίπεδο: 4
Ζωή: 100
Ταχύτητα: 22
XPs στον παίχτη: 200

Birds:

Bird:
Επίπεδο: 5
Ζωή: 100
Ταχύτητα: 20
XPs στον παίχτη: 400

King Bird:
Επίπεδο: 6
Ζωή: 100
Ταχύτητα: 20
XPs στον παίχτη: 800

Frogs:

Frog:
Επίπεδο: 7
Ζωή: 100
Ταχύτητα: 30
XPs στον παίχτη: 1600

King Frog:
Επίπεδο: 8
Ζωή: 100
Ταχύτητα: 30
XPs στον παίχτη: 3200

Boss:
Επίπεδο: 10
Ζωή: 100
Ταχύτητα: 1
XPs στον παίχτη: 12800

Εικόνα 14 Οι 4 αντίπαλοι της 2ης πίστας

Εικόνα 15 Το Boss

4.3 Όπλα

Τα τρία όπλα του παίχτη είναι machine gun, shotgun και sniper. Το καθένα από αυτά έχει μια μοναδική ιδιότητα. Το machine gun[22][8] πετάει γρήγορα τις σφαίρες(εικόνα[35]), το shotgun[23][10] πετάει τρεις σφαίρες σε τρεις διαφορετικές κατευθύνσεις και στο sniper[24][9] πατώντας δεξί κλικ σκοπεύει από το σκόπευτρό[42] του. Παρακάτω είναι η περιγραφή των όπλων:

Machine gun:
Χρόνος από σφαίρα σε σφαίρα: 0,25 δευτερόλεπτα
Ζημιά σε αντίπαλο: 6

Shotgun:
Χρόνος από σφαίρα σε σφαίρα: 1,25 δευτερόλεπτα
Ζημιά σε αντίπαλο: 20

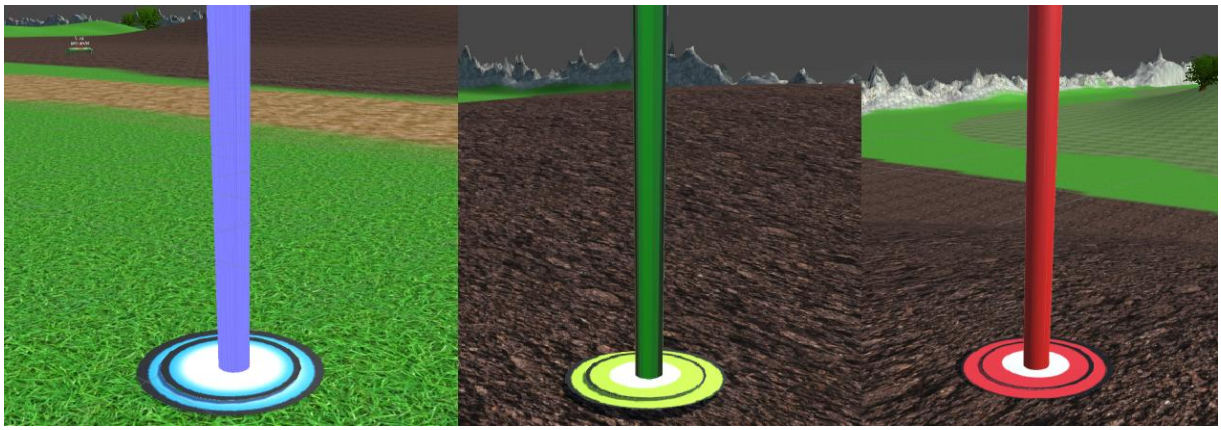
Sniper:
Χρόνος από σφαίρα σε σφαίρα: 2,5 δευτερόλεπτα
Ζημιά σε αντίπαλο: 50

Εικόνα 16 Όπλα

4.4 Σημαντικά GameObjects σκηνής

4.4.1 Teleporters

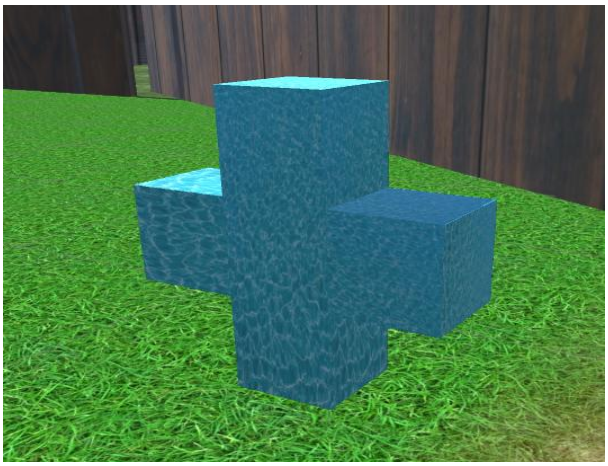
Ο teleporter είναι ένα σημείο όπου τηλεμεταφέρει τον παίχτη από ένα σημείο της πίστας σε ένα άλλο. Αυτό γίνεται επειδή η πίστα έχει μεγάλες αποστάσεις. Υπάρχουν τρεις διαφορετικοί teleporters όπου ο καθένας κάνει την τηλεμεταφορά στον αντίστοιχο του ίδιου χρώματος.



Εικόνα 17 Teleporters

4.4.2 Health Point

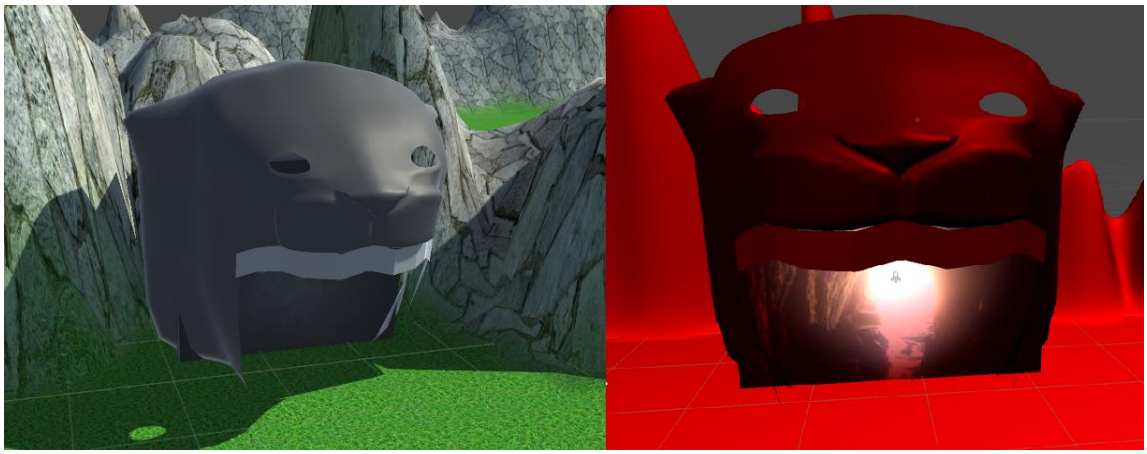
Στο health point (εικόνα[31]) όταν ακουμπάει πάνω ο παίχτης τότε γεμίζει την ζωή του στο 100%.



Εικόνα 18 Health Point

4.4.3 Boss Cave Entrance

Το Boss Cave Entrance[25](εικόνα[33]) είναι το σημείο που ο παίχτης πηγαίνει στην πίστα του boss. Στην αρχή δεν υπάρχει στην πίστα καθόλου, εμφανίζεται μόνο όταν σκοτωθεί ο King Frog που είναι ο πιο δυνατός αντίπαλος της δεύτερης πίστας. Επίσης, για να βγεις από την σπηλιά του boss πρέπει να σκοτώσεις το ίδιο το boss όπου μετά εμφανίζεται η έξοδος(εικόνα[37]).



Εικόνα 19 Είσοδος/Εξοδος Boss Cave

5 Ανάλυση παιχνιδιού

Σε αυτό το κεφάλαιο θα γίνει ανάλυση σε όλες τις λειτουργίες του κάθε GameObject και πως δημιουργήθηκε το οπτικό αποτέλεσμα. Επιπλέον, θα αναφερθεί και ο C# κώδικας στα αντίστοιχα κομμάτια.

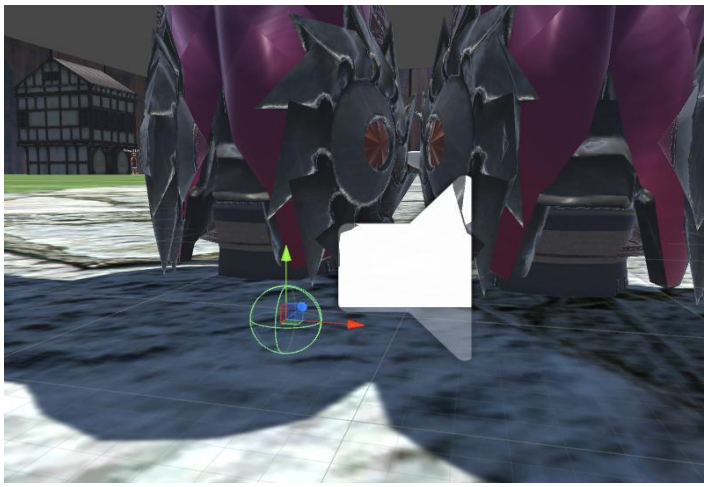
5.1 Λειτουργίες

5.1.1 Κίνηση Παίχτη

Ο παίχτης ουσιαστικά αποτελείται από δύο GameObjects, το GameObject του μοντέλου του παίχτη και ένα πολύ μικρό αόρατο μπαλάκι. Το έφτιαξα έτσι επειδή για να χρησιμοποιήσεις σωστά την βασική φυσική της Unity και να φαίνεται φυσιολογική η κίνηση του παίχτη πρέπει ένα αντικείμενο να έχει στον collider το isTriggered = false και στο rigidbody του το isKinematic = false. Αν το βάλουμε αυτό κατευθείαν στο GameObject του μοντέλου τότε ο παίχτης θα κάνει "κωλοτούμπες". Με την βοήθεια του script Head.cs, ουσιαστικά "ενώνουμε" το μπαλάκι με το μοντέλο του παίχτη και έτσι ο παίχτης για κάθε frame που αλλάζει προχωράει στην θέση που είναι το μπαλάκι χωρίς να κάνει "κωλοτούμπες".

Head.cs:

```
public class Head : MonoBehaviour {
   // this script synchronize the move object with the player as a result to make the player moving correctly

   public Transform body;
   public Transform head;

         void Update () {
      head.localPosition = body.localPosition;
         }
}
```



Εικόνα 20 Μπαλάκι Κίνησης

Άλλος ένας παράγοντας για την κίνηση του παίχτη είναι η περιστροφή της κάμερας. Η περιστροφή στον άξονα Y περιστρέφει ταυτόχρονα και τον παίχτη ώστε η κάμερα να είναι μόνιμα πίσω από τον παίχτη και με βάση το σημείο που κοιτάει η κάμερα προχωράει και ο παίχτης οριζόντια και κάθετα. Επίσης με το space απλά πηδάει προς τα πάνω.

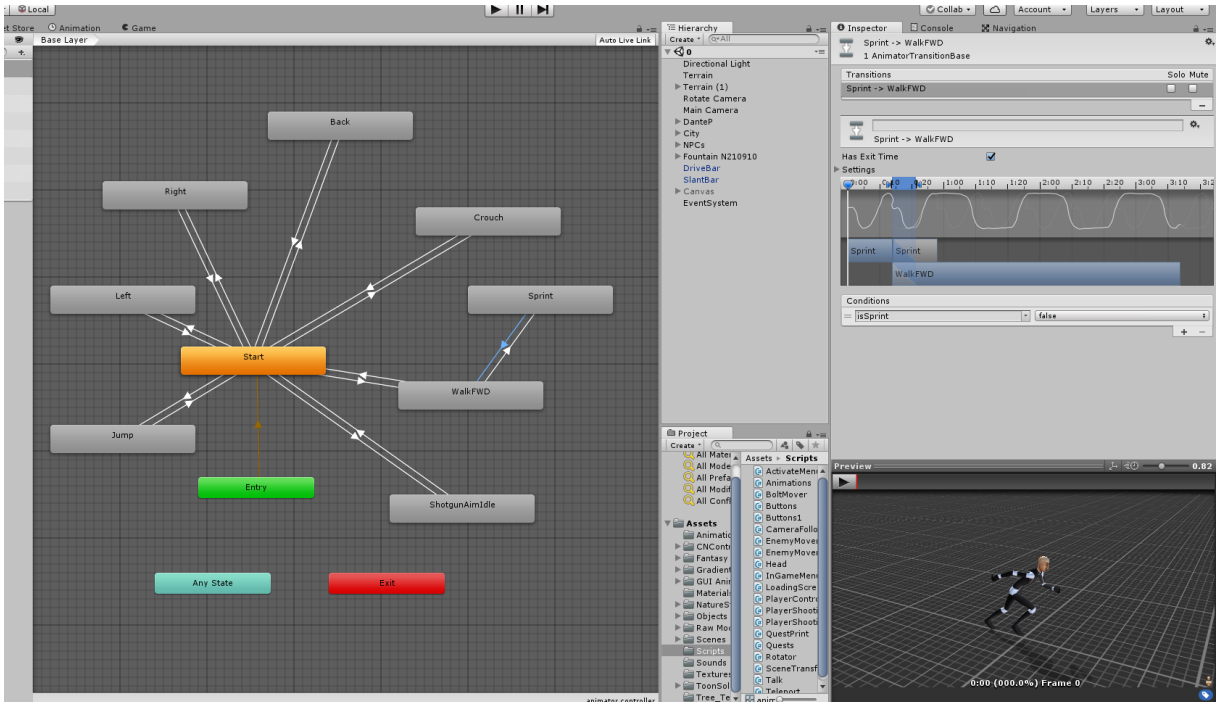
Κομμάτι του κώδικα κίνησης στο σημείο που βλέπει η κάμερα από το script PlayerController.cs

```
    void FixedUpdate() {
        // move
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        movement = new Vector3(moveHorizontal * 2, 0, moveVertical * 2);

        // jump if it is on the ground
        if (Input.GetKeyDown("space") && IsGrounded)
            rb.AddForce(Vector3.up * 5, ForceMode.Impulse);

        // moves the player based on camera rotation

        rb.AddForce(moveVertical * 2 * camera.transform.forward * speed);
        rb.AddForce(moveHorizontal * 2 * camera.transform.right * speed);
}
```

Κομμάτι του κώδικα όπου ο παίχτης περιστρέφεται ταυτόχρονα με την κάμερα από το script Animations.cs

```
void Update() {
  // rotates player with camera
        rotacja.eulerAngles = new Vector3(
            transform.rotation.eulerAngles.x,
            camera.transform.rotation.eulerAngles.y,
            transform.rotation.eulerAngles.z);
        transform.rotation = rotacja;
}
```

Αυτό το κομμάτι είναι για να κινείται απλά. Πρέπει να φαίνεται ότι κινείται φυσιολογικά, δηλαδή να κουνάει το σώμα του όπως το κάνει ένας κανονικός άνθρωπος όταν περπατάει. Οπότε πρέπει να ενσωματώσουμε στον παίχτη τα Animations[5][6]. Στη αρχή πρέπει να δημιουργήσουμε έναν animator που ουσιαστικά είναι ένα διάγραμμα οντοτήτων συσχετίσεων που μοιάζει με κατευθυνόμενο γράφο. Στο διάγραμμα αυτό παίρνουμε σαν οντότητα μια κίνηση του παίχτη πχ το animation που περπατάει ευθεία. Υπάρχουν όμως και οι βασικές οντότητες που είναι που είναι το entry όπου ξεκινάει το αρχικό animation, το start που είναι το αρχικό animation, το any state που χρησιμοποιήται για λόγους ευκολίας αλλαγής συσχέτισης και το exit που χρησιμοποιήται για στο τέλος όταν πχ πεθάνει ο παίχτης. Στις συνδέσεις μεταξύ των οντοτήτων χρησιμοποιούνται κατευθυνόμενες ακμές για να μεταφερόμαστε από το ένα animation στο άλλο. Στις ακμές αυτές μπαίνουν παράμετροι όπου αν έχουν την τιμή που τους ορίσαμε τότε αλλάζει το animation. Οι παράμετροι αυτοί λειτουργούν σαν μεταβλητές και μπορούν να είναι τύπου float, integer, boolean και triggered.



Εικόνα 21 Animator

5.1.2 Κίνηση/Επίθεση Αντιπάλων

Οι αντίπαλοι επιτίθονται όταν πάει κοντά τους ο παίχτης ή πυροβοληθούν από τον παίχτη. Μέσω του Nav Mesh Agent που έχουν οι αντίπαλοι βρίσκουν την ακριβής τοποθεσία του παίχτη και τον ακολουθούν ακόμα και εν κινήσει.

Όταν ο παίχτης βρίσκεται σε απόσταση μικρότερη του 50 από τον αντίπαλο τότε ο αντίπαλος από μόνος του επιτίθεται στον παίχτη. Αυτό το βρίσκουμε κάνοντας: απόσταση = $\sqrt{((\text{θέση\_αντιπάλου\_στο\_x} - \text{θέση\_παίχτη\_στο\_x})^2 + (\text{θέση\_αντιπάλου\_στο\_z} - \text{θέση\_παίχτη\_στο\_z})^2)}$ όπου είναι ο κλασικός τύπος της απόστασης δύο σημείων. Στην περίπτωση μας δεν χρειαζόμαστε τον άξονα Y επειδή είναι ίδιος σχεδόν παντού.

Ο αντίπαλος προκαλεί ζημιά στον παίχτη όταν φτάσει κοντά στον παίχτη και συγκρουστούν οι colliders τους. Η ζημιά που δέχεται ο παίχτης βγαίνει από τον τύπο ζημιά = επίπεδο_αντιπάλου * 0,22 / επίπεδο_παίχτη. Στον αντίποδα, όταν ο παίχτης κάνει ζημιά στον αντίπαλο τότε έχουμε τον τύπο ζημιά = ζημιά_όπλου * επίπεδο_παίχτη / επίπεδο_αντιπάλου και αφού χάσει ζωή τότε ανανεώνεται το κόκκινο νούμερο που αναγράφει την ζωή του αντιπάλου και επιτίθεται στον παίχτη ανεξάρτητα της απόστασης που έχει με .

Σε περίπτωση που σκοτωθεί ο αντίπαλος και είναι μέσα σε ενεργοποιημένο quest τότε ο αντίστοιχος μετρητής του ανεβαίνει κατά ένα.

Όλος ο κώδικας αυτός είναι στο EnemyMovement.cs:

```
using UnityEngine;

public class EnemyMovement : MonoBehaviour
{
    // follow player
    public Transform player,enemy;
    UnityEngine.AI.NavMeshAgent nav;
    private float distance;

    // health
    public TextMesh healthText;
    private float health = 100;
    public GameObject playerGO, Dante;
```

```
private PlayerController pc;
private PlayerShooting shooting;

public int enemyXP;
public int enemyLevel;

// unlock next stage
public bool stageking;

private bool hit = false;
private bool enemy_follow = false;

private Quests questController;

Animator anim;

public EnemyMovement [] follower;

void Start ()
{
   nav = GetComponent<UnityEngine.AI.NavMeshAgent> ();
   pc = playerGO.GetComponent<PlayerController>();
   questController = playerGO.GetComponent<Quests>();
   shooting = Dante.GetComponent<PlayerShooting>();
   anim = GetComponent<Animator>();
}


void Update ()
{
   // calculate the distance between enemy and player
   distance = Mathf.Sqrt(
            Mathf.Pow(enemy.position.x - player.position.x, 2) +
            Mathf.Pow(enemy.position.z - player.position.z, 2));

   // 50 is a logical distance to make enemy follow the player or it follow him if hit by bullet
   if (distance < 50 || hit || enemy_follow)
   {
      if (hit)
      {
         for (int i = 0; i < follower.Length; i++)
            follower[i].follow();
      }
      nav.SetDestination(player.position);
      if (anim)
      {
         anim.SetBool("isAttacking", true);
         anim.Play("attacking");
      }
   }
   else
   {
      if (anim)
      {
         anim.SetBool("isAttacking", false);
         anim.Play("Start");
      }
   }

}
void OnTriggerEnter(Collider other)
{
   // if hit by bullet
   if (other.tag == "Bullet")
```

```
    {
        // erase enemy's health with type 6*player level/enemy level
        if(shooting.get_active_ak())
            health -= 6 * pc.getLevel() / enemyLevel;

        if (shooting.get_active_spas())
            health -= 20 * pc.getLevel() / enemyLevel;

        if (shooting.get_active_sniper())
            health -= 50 * pc.getLevel() / enemyLevel;

        if (shooting.get_active_sniper2())
            health -= 50 * pc.getLevel() / enemyLevel;

        // enemy dies
        if (health <= 0)
        {
            // when the stage king dies, the gate of next stage must open
            if (stageking){pc.opencave();}

            Destroy(this.gameObject);
            // adds the amount of xp points that player must recieve
            pc.addXP(enemyXP);

            if(questController.getq1() == 1 && (enemyLevel == 1 || enemyLevel == 2))
            {
                questController.setcntq1();
            }

            if (questController.getq2() == 1 && (enemyLevel == 3 || enemyLevel == 4))
            {
                questController.setcntq2();
            }

            if (questController.getq3() == 1 && (enemyLevel == 5 || enemyLevel == 6))
            {
                questController.setcntq3();
            }

            if (questController.getq4() == 1 && (enemyLevel == 7 || enemyLevel == 8))
            {
                questController.setcntq4();
            }
        }

        // prints the new value after the enemy hit by bullet
        healthText.text = "" + health;

        // enables enemy to follow you
        hit = true;
    }

    // if enemy reach the player
    if (other.gameObject.tag == "Player")
    {
        // erase player health with type enemylevel*0.22/playerlevel
        pc.eraseHealth(enemyLevel * 0.22f / pc.getLevel());
    }

}

public void follow()
{
    enemy_follow = true;
```

```
    }

}
```

5.1.3 Σύστημα των quests

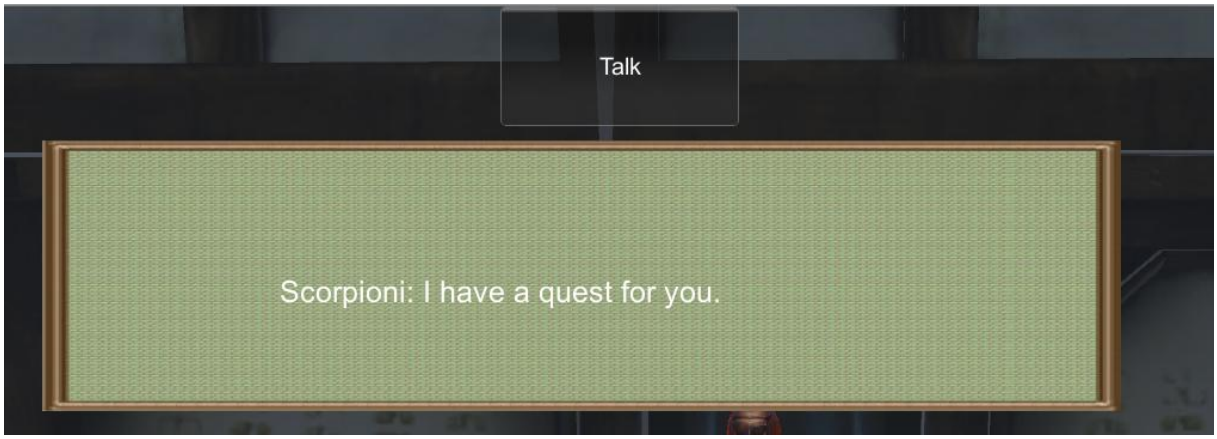
Τα quest λαμβάνονται από τους NPCs[11][12][13][14][15]. Πηγαίνοντας κοντά τους εμφανίζεται ενεργοποιείται ένας triggered collider και εκτελεί το script Talk.cs. Το script αυτό απενεργοποιεί την κίνηση της κάμερας, δεν αφήνει τον παίχτη να πυροβολεί και εμφανίζει τον κέρσορα. Έπειτα εμφανίζει το κείμενο που θέλει ο NPC να "πει" στον παίχτη μαζί με το πλαίσιό του από πίσω και το κουμπί Talk όπου προχωράει την συζήτηση.

Εικόνα 22 NPCs

Κώδικας του Talk.cs:

```
using UnityEngine;

public class Talk : MonoBehaviour
{
    public string[] conversation;

    private int convoIndex = 0;
    private string currentTalk = "";

    public GUIStyle fontsize;

    private bool stay = false;

    public GameObject playerGO, Dante, camer,camer2;
    private PlayerShooting gun_controller;
    private Animations anim;
    private ActivateMenu me;
    private CameraFollow cam_controll, cam_controll2;
    public int quest;

    public Texture2D backk;

    void Awake()
    {
        me = playerGO.GetComponent<ActivateMenu>();
        gun_controller = Dante.GetComponent<PlayerShooting>();
        anim = Dante.GetComponent<Animations>();
        cam_controll = camer.GetComponent<CameraFollow>();
        cam_controll2 = camer2.GetComponent<CameraFollow>();
    }

    void OnTriggerStay(Collider col)
```

```
    {
        if (col.gameObject.tag == "Player")
        {
            stay = true;
            gun_controller.set_fire(false);
            anim.set_talk(true);
            cam_controll.set_rotate(false);
            cam_controll2.set_rotate(false);
        }
    }

    void OnTriggerExit(Collider col)
    {
        if (col.gameObject.tag == "Player")
        {
            currentTalk = "";
            convoIndex = 0;
            stay = false;
            Cursor.visible = false;
            me.deactivate();
            gun_controller.set_fire(true);
            anim.set_talk(false);
            cam_controll.set_rotate(true);
            cam_controll2.set_rotate(true);
        }
    }

    void OnGUI()
    {
        if (stay)
        {
            Cursor.visible = true;
            GUI.skin.button.fontSize = 20;
            GUILayout.BeginArea(new Rect(Screen.width * 4 / 10, Screen.height * 0 / 7, Screen.width, Screen.height));
            if (GUILayout.Button("Talk", GUILayout.Width(200), GUILayout.Height(100)))
            {
                convoIndex++;
                if (convoIndex >= conversation.Length)
                {
                    me.activate(quest);
                    convoIndex = conversation.Length - 1;
                }
            }
            GUILayout.EndArea();
            currentTalk = conversation[convoIndex];

            GUI.Label(new Rect(Screen.width * (20 / 100f), Screen.height * (12 / 100f), backk.width * Screen.width/75,
backk.height * Screen.height /1000), backk);
            GUI.Label(new Rect(Screen.width * (40 / 100f), Screen.height * (25 / 100f), 0, 20), currentTalk, fontsize);
        }

    }

    public bool get_stay()
    {
        return stay;
    }
}
```



Εικόνα 23 Ομιλία με NPC

Όταν η συζήτηση φτάσει στο τέλος της που είναι πάντα μία ερώτηση για το αν θέλει να αποδεχθεί ο παίχτης το quest μαζί με το τι πρέπει να κάνει και την ανταμοιβή του. Έπειτα εμφανίζονται δύο κουμπιά Yes και No μέσω του script ActiveMenu.cs και γίνεται η αποδοχή ή όχι του quest.

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

public class ActivateMenu : MonoBehaviour {

   bool active = false;

   public GameObject playerGO;
   private Quests pc;
   private int quest;

   void Start()
   {
      pc = playerGO.GetComponent<Quests>();
   }

   void OnGUI() {
  if(quest != 0) {
      if (active)
      {
         Cursor.visible = true;
         GUILayout.BeginArea(new Rect(Screen.width * 4 / 10, Screen.height * 1 / 2f, Screen.width, Screen.height));
         if (GUILayout.Button("Yes", GUILayout.Width(200), GUILayout.Height(100)))
         {
            if(quest == 1)
               pc.setq1();
            if (quest == 2)
               pc.setq2();
            if (quest == 3)
               pc.setq3();
            if (quest == 4)
               pc.setq4();
            if (quest == 5)
               pc.setq5();

               active = false;
            Cursor.visible = false;
         }

         if (GUILayout.Button("No", GUILayout.Width(200), GUILayout.Height(100)))
         {
```

```
            active = false;
            Cursor.visible = false;
        }
        GUILayout.EndArea();
    }
    }
    else
    {
        active = false;
    }
}


public void activate(int qn)
{
    quest = qn;
    active = true;
}

public void deactivate()
{
    active = false;
}
}
```

Εικόνα 24 Αποδοχή/Απόρριψη quest

Αν δεν γίνει η αποδοχή δεν γίνεται απλά τίποτα. Αν την αποδεχθεί τότε το παιχνίδι αποθηκεύει το ότι το αντίστοιχο quest είναι ενεργό μέσω του script Quests.cs και γίνεται ανενεργό μόνο κατά την πραγματοποίησή του.

Κώδικας του Quests.cs

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


public class Quests : MonoBehaviour {

  private int q1, q2, q3, q4, q5;
  private int cntq1, cntq2, cntq3, cntq4, cntq5;
```

```
public GameObject playerGO;
private PlayerController pc;
private InGameMenu gun_active;


void Start () {
    q1 = PlayerPrefs.GetInt("q1");
    q2 = PlayerPrefs.GetInt("q2");
    q3 = PlayerPrefs.GetInt("q3");
    q4 = PlayerPrefs.GetInt("q4");
    q5 = PlayerPrefs.GetInt("q5");
    pc = playerGO.GetComponent<PlayerController>();
    gun_active = playerGO.GetComponent<InGameMenu>();
    cntq1 = PlayerPrefs.GetInt("cntq1");
    cntq2 = PlayerPrefs.GetInt("cntq2");
    cntq3 = PlayerPrefs.GetInt("cntq3");
    cntq4 = PlayerPrefs.GetInt("cntq4");
    cntq5 = PlayerPrefs.GetInt("cntq5");
}

        void Update () {
    if (cntq1 >= 2)
    {
        cntq1 = 0;
        q1 = 0;
        PlayerPrefs.SetInt("q1", 0);
        PlayerPrefs.SetInt("cntq1", 0);
        pc.addXP(200);
    }
    if (cntq2 >= 3)
    {
        cntq2 = 0;
        q2 = 0;
        PlayerPrefs.SetInt("q2", 0);
        PlayerPrefs.SetInt("cntq2", 0);
        pc.addXP(1000);
    }
    if (cntq3 >= 4)
    {
        cntq3 = 0;
        q3 = 0;
        PlayerPrefs.SetInt("q3", 0);
        PlayerPrefs.SetInt("cntq3", 0);
        pc.addXP(4000);
        gun_active.set_shotgun();
    }
    if (cntq4 >= 4)
    {
        cntq4 = 0;
        q4 = 0;
        PlayerPrefs.SetInt("q4", 0);
        PlayerPrefs.SetInt("cntq4", 0);
        pc.addXP(10000);
        gun_active.set_sniper();
    }
    if (cntq5 >= 1)
    {
        cntq5 = 0;
        q5 = 0;
        PlayerPrefs.SetInt("q5", 0);
        PlayerPrefs.SetInt("cntq5", 0);
        pc.addXP(40000);
        gun_active.set_sunglasses();
```

```
    }
}

public int getq1()
{
    return q1;
}

public int getq2()
{
    return q2;
}

public int getq3()
{
    return q3;
}

public int getq4()
{
    return q4;
}

public int getq5()
{
    return q5;
}

public void setq1()
{
    q1 = 1;
    PlayerPrefs.SetInt("q1",1);
}

public void setq2()
{
    q2 = 1;
    PlayerPrefs.SetInt("q2", 1);
}

public void setq3()
{
    q3 = 1;
    PlayerPrefs.SetInt("q3", 1);
}

public void setq4()
{
    q4 = 1;
    PlayerPrefs.SetInt("q4", 1);
}

public void setq5()
{
    q5 = 1;
    PlayerPrefs.SetInt("q5", 1);
}

public void setcntq1()
{
    cntq1++;
    PlayerPrefs.SetInt("cntq1", cntq1);
}
```

```
    public void setcntq2()
    {
        cntq2++;
        PlayerPrefs.SetInt("cntq2", cntq2);
    }

    public void setcntq3()
    {
        cntq3++;
        PlayerPrefs.SetInt("cntq3", cntq3);
    }

    public void setcntq4()
    {
        cntq4++;
        PlayerPrefs.SetInt("cntq4", cntq4);
    }

    public void setcntq5()
    {
        cntq5++;
        PlayerPrefs.SetInt("cntq5", cntq5);
    }

    public int getcntq1()
    {
        return cntq1;
    }

    public int getcntq2()
    {
        return cntq2;
    }

    public int getcntq3()
    {
        return cntq3;
    }

    public int getcntq4()
    {
        return cntq4;
    }

    public int getcntq5()
    {
        return cntq5;
    }
}
```

Αφού γίνει ενεργό, τότε μέσω του script QuestPrint.cs γίνεται εκτύπωση όλων των ενεργών quests σε σειρά από το ευκολότερο στο δυσκολότερο καθώς και την πρόοδό τους η οποία αποθηκεύεται με το που αλλάξει.

Κώδικας του QuestPrint.cs

```
using UnityEngine;
using System.Collections;

public class QuestPrint : MonoBehaviour
{
```

```
    private string currentTalk = "";
    public GUIStyle fontsize;

    public GameObject playerGO;
    private Quests pc;

    void Start()
    {
        pc = playerGO.GetComponent<Quests>();
    }

    void OnGUI()
    {
            fontsize.fontSize = 30;
            fontsize.normal.textColor = Color.white;
            currentTalk = "Quests:\n";
        if (pc.getq1() == 1)
            currentTalk = currentTalk + "Kill 2 worms ("+pc.getcntq1()+"/2)\n";
        if (pc.getq2() == 1)
            currentTalk = currentTalk + "Kill 3 snails (" + pc.getcntq2() + "/3)\n";
        if (pc.getq3() == 1)
            currentTalk = currentTalk + "Kill 4 birds (" + pc.getcntq3() + "/4)\n";
        if (pc.getq4() == 1)
            currentTalk = currentTalk + "Kill 4 frogs (" + pc.getcntq4() + "/4)\n";
        if (pc.getq5() == 1)
            currentTalk = currentTalk + "Kill the Boss (" + pc.getcntq5() + "/1)\n";
        GUI.Label(new Rect(Screen.width * (1 / 100f), Screen.height * (28 / 100f), 0, 20), currentTalk, fontsize);
    }
}
```

Εικόνα 25 Εκτύπωση quests

Τέλος όταν πραγματοποιήσει ο παίχτης κάποιο quest τότε λαμβάνει την ανταμοιβή του και δεν εμφανίζεται πλέον αριστερά της οθόνης το αντίστοιχο quest διατηρώντας όμως την ταξινόμιση της δυσκολίας.

5.1.4 Σύστημα του μενού

Το μενού[46] δημιουργείται από το script InGameMenu.cs και περιέχει τέσσερα κουμπιά με λειτουργίες όπως επιλογή όπλου, αλλαγή ποιότητας γραφικών, αποθήκευση τοποθεσίας παίχτη και έξοδος από το παιχνίδι. Το μενού ανοίγει είτε πατώντας το πλήκτρο “p” είτε κάνοντας κλικ πάνω του (το ποντίκι μπορεί να μην φαίνεται αλλά υπάρχει). Επίσης, το κουμπί του μενού είναι μόνιμα στην πάνω δεξιά άκρη της οθόνης.

Εικόνα 26 Μενού

Στην επιλογή όπλου (Guns), εμφανίζεται ένα δεύτερο υπομενού με κουμπιά των διαθέσιμων όπλων όπου επιλέγοντας ένα από αυτά γίνεται αλλαγή με αυτό που κρατάει και έπειτα κλίνει το υπομενού. Στην επιλογή αλλαγής ποιότητας γραφικών (Graphics), εμφανίζεται ένα δεύτερο υπομενού με κουμπιά Very Low, Low, Medium, High, Very High που αντιστοιχούν στην ποιότητα των γραφικών. Στην επιλογή αποθήκευσης της τοποθεσίας του παίχτη (Save), αποθηκεύει το transform με τις XYZ τιμές του και την πίστα που είναι . Στην επιλογή έξοδος από το παιχνίδι (Exit Game), σταματάει την λειτουργία της εφαρμογής.

Κώδικας InGameMenu.cs:

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

public class InGameMenu : MonoBehaviour {

    public bool active,guns, graphics;
    public Terrain terrain;

    public GameObject playerGO,Dante,camer, camer2;
    private PlayerController pc;
    private PlayerShooting gun_controller;
    private CameraFollow camera_controller, camera_controller2;

    public Texture button_image;

    private int shotgun, sniper, sunglasses;

    public GameObject sungl_act;

    // Use this for initialization
    void Start () {
        active = false;
        guns = false;
        graphics = false;
        pc = playerGO.GetComponent<PlayerController>();
        gun_controller = Dante.GetComponent<PlayerShooting>();
        camera_controller = camer.GetComponent<CameraFollow>();
        camera_controller2 = camer2.GetComponent<CameraFollow>();
        shotgun = PlayerPrefs.GetInt("spas");
        sniper = PlayerPrefs.GetInt("sniper");
        sunglasses = PlayerPrefs.GetInt("sunglasses");
    }
```

```
    void Update()
    {
        if (Input.GetKeyDown("p"))
        {
            if (active)
            {
                graphics = false;
                guns = false;
                active = false;
                gun_controller.set_fire(true);
                camera_controller.set_rotate(true);
                camera_controller2.set_rotate(true);
                Cursor.visible = false;
            }
            else
            {
                active = true;
                gun_controller.set_fire(false);
                camera_controller.set_rotate(false);
                camera_controller2.set_rotate(false);
            }
        }

        if(sunglasses == 1)
        {
            sungl_act.active = true;
        }
    }

    // Update is called once per frame
    void OnGUI () {

        if (GUI.Button(new Rect(Screen.width * (90 / 100f), Screen.height * (2 / 100f), Screen.width * (8 / 100f),
Screen.height * (8 / 100f)), button_image))
        {
            if (active)
            {
                graphics = false;
                guns = false;
                active = false;
                gun_controller.set_fire(true);
                camera_controller.set_rotate(true);
                camera_controller2.set_rotate(true);
                Cursor.visible = false;
            }
            else
            {
                active = true;
                gun_controller.set_fire(false);
                camera_controller.set_rotate(false);
                camera_controller2.set_rotate(false);
            }

        }
        if (active)
        {

            Cursor.visible = true;

            // main
            if (GUI.Button(new Rect(Screen.width * (90 / 100f), Screen.height * (10 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Guns"))
            {
```

```
            guns = true;
            graphics = false;
        }
        if (GUI.Button(new Rect(Screen.width * (90 / 100f), Screen.height * (15 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Graphics"))
        {
            graphics = true;
            guns = false;
        }

        if (GUI.Button(new Rect(Screen.width * (90 / 100f), Screen.height * (20 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Save"))
        {
            //Saving
            PlayerPrefs.SetFloat("PlayerX", pc.transform.position.x);
            PlayerPrefs.SetFloat("PlayerY", pc.transform.position.y);
            PlayerPrefs.SetFloat("PlayerZ", pc.transform.position.z);
            PlayerPrefs.SetString("Scene", SceneManager.GetActiveScene().name);
            active = false;
            gun_controller.set_fire(true);
            Cursor.visible = false;
        }
        if (GUI.Button(new Rect(Screen.width * (90 / 100f), Screen.height * (25 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Exit Game"))
        {
            Application.Quit();
            active = false;
            gun_controller.set_fire(true);
            Cursor.visible = false;
        }

        // guns
        if (guns) {
          if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (5 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Machine Gun"))
            {
                gun_controller.set_active_ak();
            guns = false;
            Cursor.visible = false;
            }
            if (shotgun == 1)
            {
                if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (10 / 100f), Screen.width * (8 /
100f), Screen.height * (5 / 100f)), "Shotgun"))
                {
                    gun_controller.set_active_spas();
                    guns = false;
                    Cursor.visible = false;
                }
            }
            if (sniper == 1)
            {
                if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (15 / 100f), Screen.width * (8 /
100f), Screen.height * (5 / 100f)), "Sniper"))
                {
                    gun_controller.set_active_sniper();
                    guns = false;
                    Cursor.visible = false;
                }
            }
        }

        // graphics
        if (graphics)
```

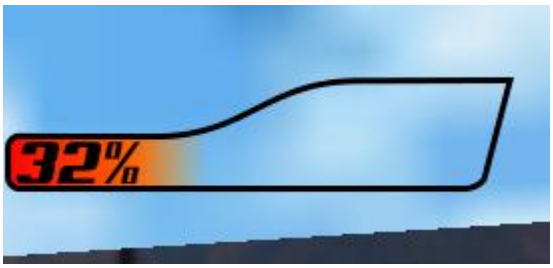
```
        {
            if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (5 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Very Low"))
            {
                terrain.heightmapPixelError = 200;
                terrain.basemapDistance = 0;
                graphics = false;
                Cursor.visible = false;
            }
            if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (10 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Low"))
            {
                terrain.heightmapPixelError = 150;
                terrain.basemapDistance = 500;
                graphics = false;
                Cursor.visible = false;
            }
            if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (15 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Medium"))
            {
                terrain.heightmapPixelError = 100;
                terrain.basemapDistance = 1000;
                graphics = false;
                Cursor.visible = false;
            }
            if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (20 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "High"))
            {
                terrain.heightmapPixelError = 50;
                terrain.basemapDistance = 1500;
                graphics = false;
                Cursor.visible = false;
            }
            if (GUI.Button(new Rect(Screen.width * (80 / 100f), Screen.height * (25 / 100f), Screen.width * (8 / 100f),
Screen.height * (5 / 100f)), "Very High"))
            {
                terrain.heightmapPixelError = 0;
                terrain.basemapDistance = 2000;
                graphics = false;
                Cursor.visible = false;
            }
        }
    }
        }

    public void set_shotgun()
    {
        PlayerPrefs.SetInt("shotgun", 1);
        shotgun = 1;
    }

    public void set_sniper()
    {
        PlayerPrefs.SetInt("sniper", 1);
        sniper = 1;
    }

    public void set_sunglasses()
    {
        PlayerPrefs.SetInt("sunglasses", 1);
        sunglasses = 1;
    }

    public bool get_active()
```

```
    {
        return active;
    }
}
```

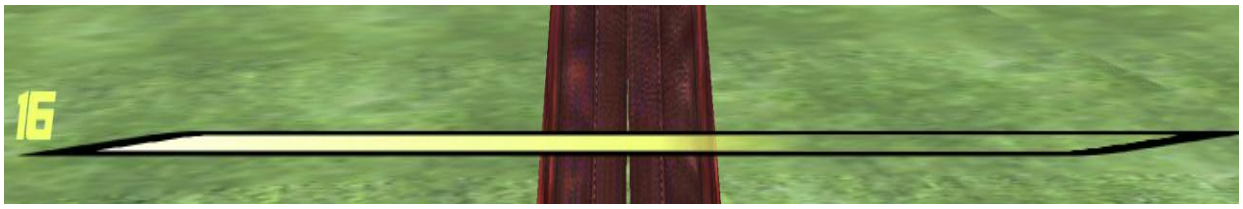
5.1.5 Σύστημα Απεικόνισης Ζωής/XPs

Το σύστημα απεικόνισης της μπάρας[3] ουσιαστικά είναι το script GUIBarScript.cs. Η διαφορά τους είναι μόνο οι εικόνες που είναι διαφορετικές και τα χρώματα. Το συγκεκριμένο είναι ένα asset από το asset store της unity που περιλαμβάνει το script και τις εικόνες. Η μπάρα της ζωής είναι πάντα στην πάνω αριστερά γωνία της οθόνης ενώ η μπάρα των XPs είναι κάτω στη οθόνη πιάνοντας όλο το μήκος της. Έπειτα για να είναι λειτουργικό πρέπει να το προσαρμόσουμε στον παίχτη και να γίνεται διαχείριση του κώδικα μέσω του PlayerController.cs με βάση τις ανάγκες του παιχνιδιού. Το script δέχεται τιμές πραγματικού αριθμού από το 0,0 έως 1,0. Για την μπάρα της ζωής, οι τιμές όπου αντιστοιχούν στο αναγραφόμενο ποσοστό της μπάρας 0% έως 100%.

Για την μπάρα των XPs είναι λίγο πιο σύνθετα τα πράγματα. Αρχικά έχει ένα επιπλέον αριθμό που αναγράφει το επίπεδο του αντιπάλου. Μετά, το επίπεδο βγαίνει με βάση το πόσα XPs έχει ο παίχτης αλλά δεν έχουν ισόποσες διαφορές. Η διαφορά μεταξύ των επιπέδων σε XPs είναι εκθετική και βγαίνει από τον τύπο XPs = $2^{\text{επίπεδο-1}} * 100$. Επειδή όμως χρησιμοποιώ XPs τότε για να βρω το επίπεδο πρέπει να αντιστρέψω τον τύπο των XPs όπου είναι επίπεδο = $\text{Log}_2(\text{XPs}/100) + 1$ και μετά παίρνω το ακέραιο μέρος του. Επειδή όμως η μπάρα δέχεται 0,0 έως 1,0 τότε πρέπει να πάρουμε την ανάλογη κλίμακα με τα XPs η οποία βγαίνει από τον τύπο κλίμακα = κλίμακαXPs / κλίμακαΕπιπέδου όπου κλίμακαΕπιπέδου = $2^{\text{επίπεδο}} * 100 - 2^{\text{επίπεδο-1}} * 100$ και κλίμακαXPs = XPs - $2^{\text{επίπεδο-1}} * 100$. Όλο αυτό γίνεται για να υπάρχει μια ισοστάθμιση στα επίπεδα του παίχτη δηλαδή πχ αν ο παίχτης είναι μεγαλύτερο επίπεδο από τον αντίπαλο που θα σκοτώσει τότε τα XPs που θα πάρει θα είναι λιγότερα σε αξία γι' αυτόν.



Εικόνα 27 Μπάρα Ζωής



Εικόνα 28 Μπάρα XPs

Κώδικας GUIBarScript.cs:

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

public class GUIBarScript : MonoBehaviour {

        //Declare variables

        //for the CurrentValue, and the Value it will be after the update
```

```
        private float CurrentValue;
        public float Value;

        //FadeValue is current amount the bar is faded
        private float FadeValue;
        //FadeFactor is a little complex, Open the ReadMe file to know more
        public float FadeFactor = 6f;

        //position and scale of the GUIBar on the screen
        public Vector2 Position;
        public float ScaleSize;

        //Font Variables
        public bool DisplayText = true;
        public string TextString;
        public Color TextColor;
        public bool OverRideTextColorWithGradient = false;
        public Font TextFont;
        public float TextSize;
        public Vector2 TextOffset;

        //Carries the .png images to display the GUIBar
        public Texture2D Background;
        public Texture2D Mask;
        public Texture2D ValueBar; //Each Bar needs it's own ValueBar Texture
        public Texture2D Foreground;

        //Carries the colors that the GUIbar will be
        public List<Color> GradientColors = new List<Color>();

        //These are used for redrawing the GUIBar
        private Gradient g = new Gradient();
        private GradientColorKey[] gck;
        private GradientAlphaKey[] gak;
        private Color[] MaskPixels;

    public GameObject playerGO;
    private PlayerController player;


    void Start()
    {
        player = playerGO.GetComponent<PlayerController>();
    }

    //Stanard OnGUI Method
    void OnGUI()
            {

                    //UpdateBar is a very large function so i'm only excuting it when i have to.
                    if (
                            Mathf.Round(CurrentValue * 100f) != Mathf.Round(Value * 100f)
                            )
                    {
                            UpdateBar();
                    }

                    //if you don't have a background texture i won't draw it
                    if (Background != null)
                    {
                            GUI.DrawTexture(new Rect(Screen.width*(0/100f),Screen.height * (93.5f / 100f),
Background.width * Screen.width/250,Background.height * 0.5f),Background);
                    }
```

```
		GUI.DrawTexture(new Rect(Screen.width * (0 / 100f), Screen.height * (93.5f / 100f),
ValueBar.width * Screen.width / 250, ValueBar.height * 0.5f),ValueBar);

		//if you don't have a foreground texture i won't draw it
		if (Foreground != null)
		{
			GUI.DrawTexture(new Rect(Screen.width * (0 / 100f), Screen.height * (93.5f / 100f),
Foreground.width * Screen.width / 250, Foreground.height * 0.5f),Foreground);
		}

		//if display text is enabled the display text will be drawn
		if (DisplayText)
		{

			GUIStyle LabelStyle =  new GUIStyle();

			if (OverRideTextColorWithGradient)
			{
				Color MaxGradientColor = new Color(g.Evaluate(Value *
0.99f).r,g.Evaluate(Value * 0.99f).g,g.Evaluate(Value * 0.99f).b,1.0f);
				LabelStyle.normal.textColor = MaxGradientColor;
			}
			else
			{
				LabelStyle.normal.textColor = TextColor;
			}
			LabelStyle.fontSize = (int)TextSize;
			LabelStyle.font = TextFont;

			TextString = ""+player.getLevel(); // xp

			GUI.Label(new Rect(Screen.width * (1 / 100f) + TextOffset.x, Screen.height * (93.5f /
100f) + TextOffset.y,ValueBar.width * ScaleSize,ValueBar.height * 0.5f),TextString,LabelStyle);

		}
	}

	//this method will redraw the bar
	private void UpdateBar()
	{
		//update the gradient
		UpdateGradient ();

		//error handling
		if (g == null)
		{
			return;
		}

		//for each pixle in the ValueBar, we will change the color to w/e it is in the gradient
		int y = 0;
		while (y < ValueBar.height)
		{
			int x = 0;
			float xf = 0f;
			while (x < ValueBar.width)
			{
				Color gC = g.Evaluate(xf/Mask.width);

				if (Mask.GetPixel(x,y).a > 0.1f)
	{
					ValueBar.SetPixel(x, y, gC);
	}
	x = x + 1;
```

```
            xf = xf + 1;
        }
        y = y + 1;
    }

                //set the new colors on the ValueBar
    ValueBar.Apply();
}

        //this method will update the gradient
        private void UpdateGradient ()
        {
                //error handling
                if (g == null)
                {
                        return;
                }

                //set the new value
                CurrentValue = Value;

    //the FadeFactor is used to set the FadeValue, see ReadMe document for more Info
                FadeValue = ((Mathf.Sin ((Value) * 3.14f))/FadeFactor) ;

                if (FadeFactor == 0)
                {
                        print ("FadeFactor = 0 does not produce a good gradient all the way through the bar");
                }

                //clamping values of variables
                FadeFactor = Mathf.Clamp(FadeFactor,-1f,20f);
                CurrentValue = Mathf.Clamp(CurrentValue,0f,1f);
                Value = Mathf.Clamp(Value,0f,1f);
                FadeValue = Mathf.Clamp(FadeValue,0.0001f,1f);

    //create variable to store the colors for the gradient
                gck = new GradientColorKey[GradientColors.Count];

                //add colors to gradient
                int i = 0;
                float f = 0f;
                while (i < GradientColors.Count)
                {
                        gck[i].color = GradientColors[i];
                        gck[i].time =  f/(GradientColors.Count - 1);
                        i++;
                        f++;
                }

                //if you do not want to use these colors you can hardcode them like so
                /*
                 * gck[0].color = [any color];
                 * gck[0].time =  [float number between 0, and 1];
                 *
                 * gck[1].color = [any color];
                 * gck[1].time =  [float number between 0, and 1];
                 *
                 * ...etc etc etc
                 *
                 */



                //set the alpha keys for the gradient
```

```
                gak = new GradientAlphaKey[3];
                gak[0].alpha = 1.0f;
                gak[0].time = 0.0f;

                gak[1].alpha = 1.0f;
                gak[1].time = CurrentValue - (FadeValue/2);

                gak[2].alpha = 0.00f;
                gak[2].time = CurrentValue + (FadeValue/2);

                //add keys to gradient
                g.SetKeys(gck,gak);

        }



        //The following methods can be used within other code do change how the GUIBar Looks
        public void AddNewColor(Color color, int Key)
        {
                GradientColors.Insert(Key,color);
        }

        public void ChangeColor(Color color, int Key)
        {
                GradientColors[Key] = color;
        }

        public void RemoveColor(int Key)
        {
                GradientColors.RemoveAt(Key);
        }

        public void SetNewValue(float V)
        {
                Value = V;
        }

public void SetNewCurrValue(float V)
{
   CurrentValue = V;
}

public void SetNewValue(double V)
        {
                Value = (float)V;
        }

        public void SetNewValue(float V, float MV)
        {
                Value = V/MV;
        }

        public void SetNewValue(double V, double MV)
        {
                Value = (float)V / (float)MV;
        }

        public void ForceUpdate()
        {
                UpdateBar();
        }
```

}

6 Διαφορές έκδοσης Android με Υπολογιστή

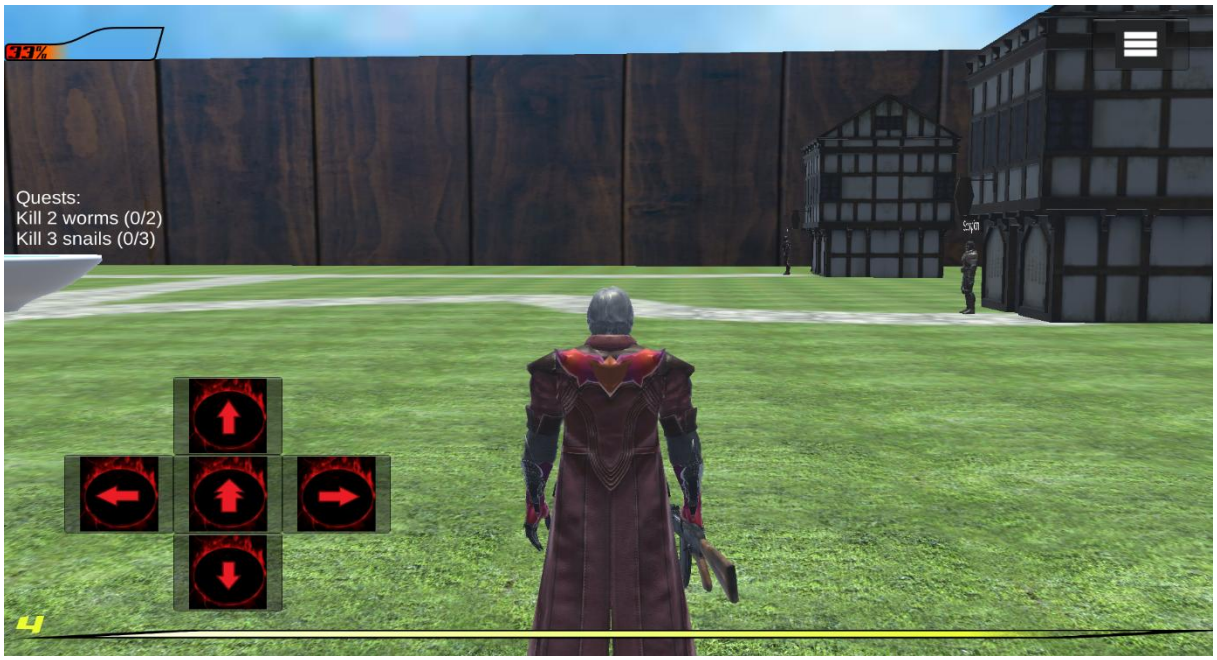
Η Unity έχει την δυνατότητα να παράγει αρχείο εγκατάστασης εφαρμογής Android (.apk). Χρειάζεται όμως να υπάρχει κατεβασμένο το Android SDK ώστε να δημιουργήσει το .apk αρχείο.

Η διαφορά είναι ότι δεν υπάρχουν κουμπιά πληκτρολογίου και ποντίκι ενσωματωμένα αλλά έχει οθόνη αφής και τους αισθητήρες σαν είσοδο. Άρα αλλάζει ο τρόπος που ο χρήστης θα αλληλοεπιδρά είναι διαφορετικός και πρέπει κάποιες λειτουργίες να αλλάξουν. Συγκεκριμένα, στην εργασία έβαλα εικονικά κουμπιά πάνω στην οθόνη για την κίνηση του παίχτη, η κάμερα περιστρέφεται κάνοντας ο χρήστης ολίσθηση με το δάκτυλό του πάνω στην οθόνη και τα κουμπιά του μενού μπορούν να πατηθούν απλά με την αφή πατώντας πάνω του.

Τα κουμπιά της κίνησης είναι τα απλά κουμπιά της Unity με μια εικόνα πάνω με βελάκι που αντιπροσωπεύει την κατεύθυνση. Συγκεκριμένα, είναι τα πάνω, κάτω, δεξιά, αριστερά και το μεσαίο που τρέχει μπροστά. Η κίνηση μαζί με τα Animations γίνεται από το script MobileMovement.cs.

Στην δεξιά μεριά της οθόνης υπάρχει ένα αόρατο touchpad[1] το οποίο ολισθαίνοντας το δάκτυλο πάνω του περιστρέφει την κάμερα (και ο παίχτης με την σειρά του ακολουθάει την κάμερα).

Τέλος, για να μην υπάρχουν πολλά κουμπιά στην οθόνη και για να μην γίνεται σύνθετο το gameplay του δεν έβαλα παραπάνω κουμπιά οπότε κάποιες λειτουργίες, όχι βασικές για το παιχνίδι, δεν είναι διαθέσιμες για την έκδοση του κινητού.



Εικόνα 29 Περιβάλλον στο κινητό

Κώδικας MobileMovement.cs:

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

public class MobileMovement : MonoBehaviour {

    Animator anim;
    private Rigidbody rb;
    public GameObject camera;
    public float speed;
    public Texture up,down,left,right,mid;
    public GameObject Dante;
    private bool movement;
    private PlayerShooting shoot;

    // Use this for initialization
    void Start () {
        anim = Dante.GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
        shoot = Dante.GetComponent<PlayerShooting>();
    }

    private void Update()
    {
        if (isMoving() == false)
        {
            shoot.set_fire(true);
            rb.velocity = Vector3.zero;
            rb.angularVelocity = Vector3.zero;
            if(shoot.get_fire())
                anim.Play("ShotgunAimIdle");
            else
                anim.Play("Start");
        }
    }

    void OnGUI () {
        movement = false;
        //kato
        if (GUI.RepeatButton(new Rect(Screen.width * (14 / 100f), Screen.height * (81 / 100f), Screen.width * (9 / 100f),
Screen.height * (12 / 100f)), down))
        {
            shoot.set_fire(false);
            rb.velocity = camera.transform.forward * -15;
            rb.AddForce(-2 * camera.transform.forward * speed);
            anim.SetBool("back", true);
            anim.Play("Back");
            movement = true;
        }
        //pano
        if (GUI.RepeatButton(new Rect(Screen.width * (14 / 100f), Screen.height * (57 / 100f), Screen.width * (9 / 100f),
Screen.height * (12 / 100f)), up))
        {
            shoot.set_fire(false);
            rb.velocity = camera.transform.forward * 15;
            rb.AddForce(2 * camera.transform.forward * speed);
            anim.SetBool("isWalking", true);
            anim.Play("WalkFWD");
            movement = true;
        }
```

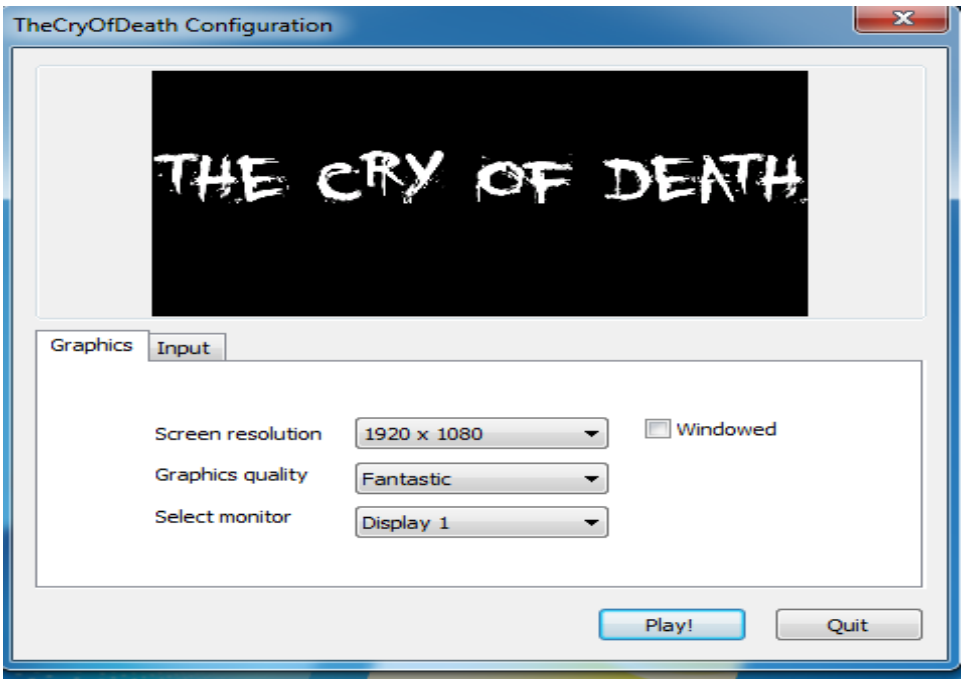
```
        // aristera
        if (GUI.RepeatButton(new Rect(Screen.width * (5 / 100f), Screen.height * (69 / 100f), Screen.width * (9 / 100f),
Screen.height * (12 / 100f)), left))
        {
            shoot.set_fire(false);
            rb.velocity = camera.transform.right * -15;
            rb.AddForce(-2 * camera.transform.right * speed);
            anim.SetBool("Left", true);
            anim.Play("Left");
            movement = true;
        }
        //deksia
        if (GUI.RepeatButton(new Rect(Screen.width * (23 / 100f), Screen.height * (69 / 100f), Screen.width * (9 / 100f),
Screen.height * (12 / 100f)), right))
        {
            shoot.set_fire(false);
            rb.velocity = camera.transform.right * 15;
            rb.AddForce(2 * camera.transform.right * speed);
            anim.SetBool("Right", true);
            anim.Play("Right");
            movement = true;
        }
        // treksimo
        if (GUI.RepeatButton(new Rect(Screen.width * (14 / 100f), Screen.height * (69 / 100f), Screen.width * (9 / 100f),
Screen.height * (12 / 100f)), mid))
        {
            shoot.set_fire(false);
            rb.velocity = camera.transform.forward * 45;
            rb.AddForce(2 * camera.transform.forward * speed);
            anim.SetBool("isSprint", true);
            anim.Play("Sprint");
            movement = true;
        }

    }

    private bool isMoving()
    {
        return movement;
    }
}
```

7 Το παιχνίδι

Το κεφάλαιο αυτό είναι το τελευταίο της αναφοράς και θα αναφέρω όλο τον χειρισμό του παιχνιδιού καθώς και δυσκολίες κατά την υλοποίηση και πιθανές βελτιώσεις.

7.1 Gameplay

Στην αρχή της εφαρμογής εμφανίζεται ένα παράθυρο βασικών ρυθμίσεων με επιλογές ανάλυσης οθόνης, ποιότητα γραφικών και οθόνη που θέλουμε να τρέξει το παιχνίδι (σε περίπτωση που έχουμε πάνω από ένα).

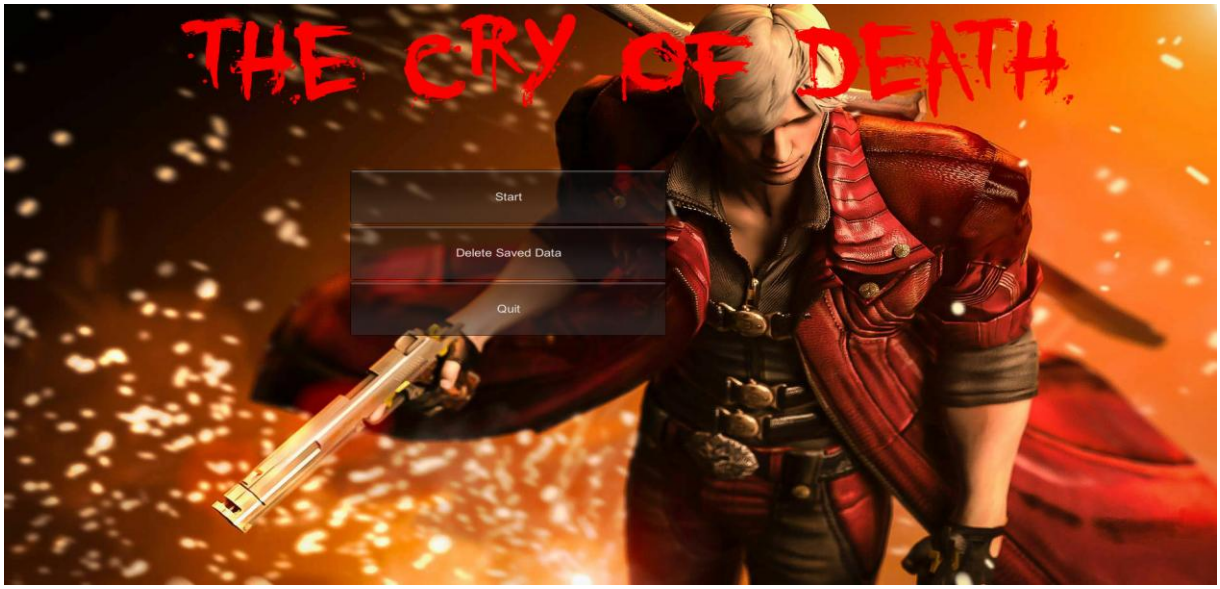


Εικόνα 30 Configuration Window

Έπειτα εμφανίζεται για τρία δευτερόλεπτα το Splash Screen με μια μικρή κίνηση.

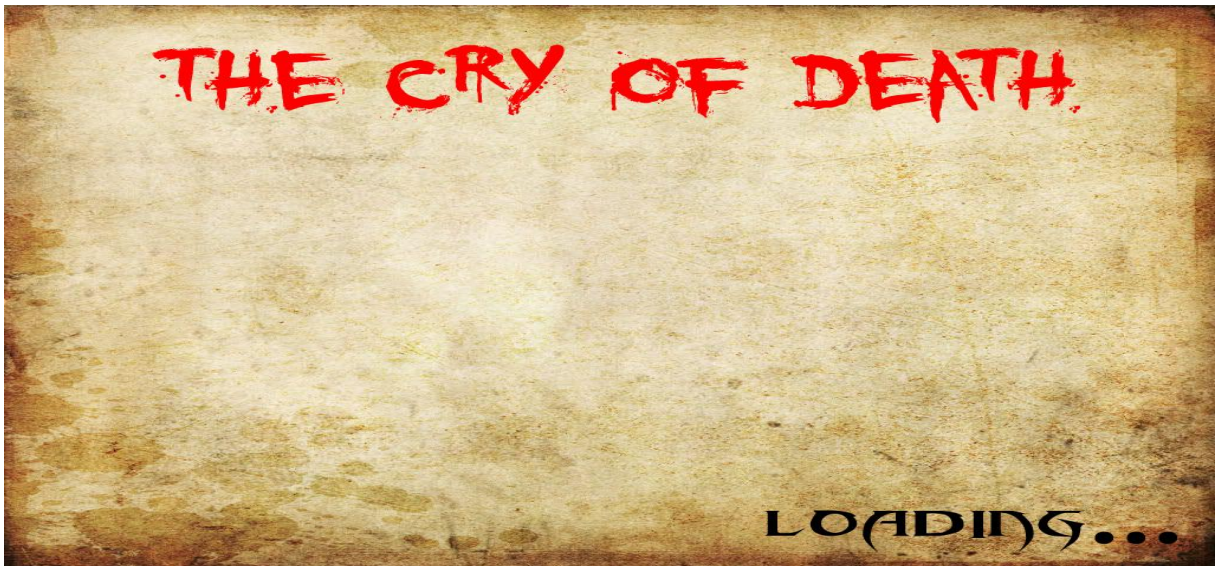
Εικόνα 31 Splash Screen

Και μετά το αρχικό μενού[38].



Εικόνα 32 Αρχικό Μενού

Πατώντας Start ξεκινάει το παιχνίδι από το σημείο που αποθηκεύτηκε την προηγούμενη φορά. Αν δεν υπάρχει κάτι αποθηκευμένο τότε ξεκινάει από ένα συγκεκριμένο σημείο. Πατώντας το κουμπί Delete Saved Data τότε γίνεται διαγραφή σε οποιαδήποτε αποθηκευμένη πληροφορία και μετά με την έναρξη ουσιαστικά ξεκινάει από την αρχή. Αφού ξεκινήσει, εμφανίζει αρχικά ένα loading screen[41] και μετά πάει στο ανάλογο σημείο.



Εικόνα 33 Loading Screen

Η πρώτη πίστα που βγάζει για πρώτη φορά περιέχει τους NPCs και την πύλη εξόδου που πηγαίνει στην δεύτερη πίστα με τους αντιπάλους (houses[27], skybox[2], fountain[28], grass[29], stone path[30], wall image[43], music[7]).

Εικόνα 34 Πρώτη Πίστα

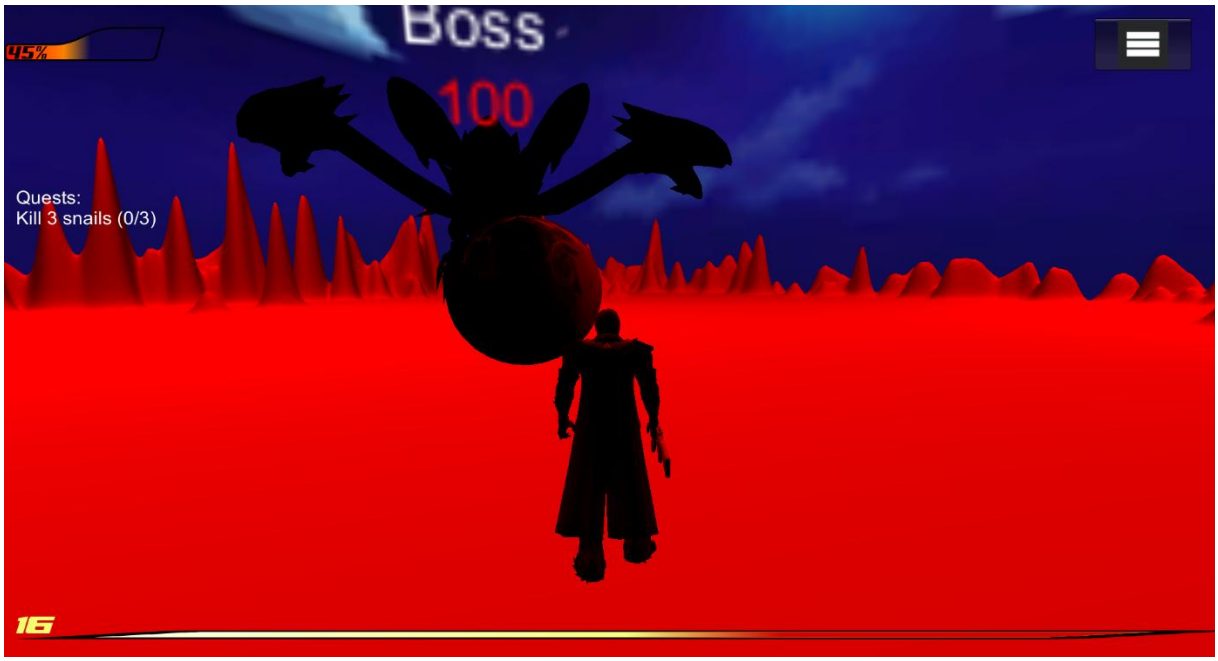
Όταν πάρει τα Quests ο παίχτης τότε δεν έχει να κάνει κάτι άλλο στην πρώτη πίστα, οπότε πάει στην δεύτερη για να ανεβάσει το επίπεδό του μέχρι να είναι έτοιμος να αντιμετωπίσει το Boss. Υπάρχει βέβαια η δυνατότητα να πάρει τα Quests πολλαπλές φορές(ground1[32], ground2[36]).

Εικόνα 35 Δεύτερη Πίστα

Για πιθανή αποφυγή αντιπάλου ή για στόχευση αντιπάλου στο βουνό υπάρχει δυνατότητα ο παίχτης να πετάει απλά στρέφοντας την κάμερα στον ουρανό και προχωρώντας μπροστά.

Εικόνα 36 Ο παίχτης όταν πετάει

Σκοτώνοντας τον Frog King τότε εμφανίζεται η είσοδος της σπηλιάς του Boss που είναι και η τρίτη πίστα (trees[4]).



Εικόνα 37 Τρίτη Πίστα

Και το παιχνίδι τελειώνει εδώ. Σαν αποδεικτικό ότι τέλειωσε το παιχνίδι ο παίχτης φοράει ένα ιδιαίτερο αντικείμενο[26].

Εικόνα 38 Αποδεικτικό Τερματισμού Παιχνιδιού

Υπάρχει η δυνατότητα όμως να παίζεται το παιχνίδι ανεβαίνοντας επίπεδο επ΄ άπειρον χωρίς κάποιο σκοπό. Αν η ζωή του παίχτη φτάσει στο 0% τότε πεθάνει και εμφανίζεται το Retry Μενού[40] όπου πατώντας Retry επιστρέφει στην πρώτη πίστα με τα μισά XPs και πατώντας Quit βγαίνει από το παιχνίδι.

Εικόνα 39 Retry Μενού

Τι είναι όμως ένα παιχνίδι χωρίς κάποιο cheat; Γι' αυτό λοιπόν πατώντας "o" ο παίχτης γεμίζει ζωή, ανεβαίνει στο δέκατο έκτο επίπεδο και ανοίγει την είσοδο της σπηλιάς του Boss.

7.2 Δυσκολίες υλοποίησης

Γενικά υπάρχουν δυσκολίες με την Unity όπως το ότι αλλάζει συχνά η έκδοση, τα καλά assets είναι επί πληρωμή, διαφορές στον κώδικα μεταξύ όμοιων συναρτήσεων και ότι πρέπει να έχει κάποιος ακριβό υπολογιστή για να τρέχουν όλα τέλεια (κυρίως κάρτα γραφικών).

Αλλάζοντας έκδοση της Unity είναι από την μία συμβατή σε παλαιότερες εκδόσεις αλλά υπάρχουν πολλά προβλήματα αν η έκδοση του project είναι μεγαλύτερη της εφαρμογής. Αν το project είναι παλιότερης έκδοσης από την εφαρμογή τότε δεν υπάρχει πρόβλημα και η εφαρμογή προσαρμόζει αυτόματα το project στην ίδια έκδοση ενώ αν συμβεί το ανάποδο θα ανοίξει το project αλλά θα δείχνει αλλόκοτα πράγματα. Το πρόβλημα αυτό κυρίως είναι όταν θέλει κάποιος να μεταφέρει ή να παρουσιάσει το project σε άλλο υπολογιστή απ' αυτών που δουλεύει.

Στον κώδικα υπάρχουν δύο παρόμοιες συναρτήσεις της C# που λέγονται Update και FixedUpdate. Με την κοινή λογική δεν διακρίνεται κάποια διαφορά. Στην περίπτωση την δικιά μου για τα animations της κίνησης του παίχτη χρησιμοποιούσα την FixedUpdate και ενώ δούλευε σωστά testing της Unity, δεν έτρεχε καθόλου σωστά όταν το έκανα εκτελέσιμο. Η διαφορά τους είναι πως η Update εκτελείτε μία φορά ανά frame ενώ η FixedUpdate εκτελείτε ανά Physic Step και επειδή τα animations είναι θέμα οπτικού αποτελέσματος, έπρεπε να χρησιμοποιήσω την Update. Επίσης επειδή είναι δύσκολο να προσεγγίσεις το πάτημα πολλαπλών κουμπιών και επειδή η Unity δεν έχει επεξεργασία των έτοιμων animation, δεν κατάφερα να κάνω τον παίχτη να πηγαίνει διαγώνια. Υπάρχει δωρεάν asset για την κίνηση του παίχτη αλλά προτίμησα να το φτιάξω από την αρχή.

7.3 Μελλοντικές βελτιώσεις

Σε γενικές γραμμές μπορούν να γίνουν βελτιώσεις στην κίνηση του παίχτη, υιοθετώντας περισσότερα στοιχεία από διάσημα RPG και Shooter παιχνίδια και μεγαλύτερο αριθμό αντικειμένων.

Η κίνηση μπορεί να βελτιωθεί δημιουργώντας την δυνατότητα κίνησης προς όλες τις κατευθύνσεις εναλλάσσοντας τα animations ομαλά και όχι απότομα. Επίσης και η κίνηση των αντιπάλων να γίνει πιο "έξυπνη" επειδή ο αντίπαλος κουνιέται μόνο αν έχει μεγάλη απόσταση με τον παίχτη.

Για την προσέγγιση προς τα RPG, μπορεί να γίνει βελτίωση να έχει περισσότερα αντικείμενα όπως όπλα, πανοπλία και χαρακτήρες. Επίσης θα μπορούν να προστεθούν οχήματα ή άλογα για να κινούνται πιο άνετα στις πίστες καθώς και περισσότερες πίστες με περισσότερα Boss και είδη αντιπάλων.

Για την προσέγγιση προς τα Shooter, μπορεί να αλλάξει ο τύπος της κάμερας σε first person δηλαδή να φαίνονται μόνο τα χέρια με το όπλο σαν να βλέπει ο χρήστης από τα "μάτια" του παίχτη. Επιπλέον, ο παίχτης να μπορεί να έχει την δυνατότητα να στοχεύει το όπλο και κάθετα.

Τέλος, χρειάζεται βελτίωση η έκδοση του Android ώστε να έχει ο παίχτης όλες της λειτουργίες που έχει και η έκδοση του υπολογιστή αλλά ταυτόχρονα να μην είναι η οθόνη γεμάτο κουμπιά ώστε να έχει και ωραία μορφή.

8 References

[1] CnControlls: https://www.assetstore.unity3d.com/en/#!/content/15233

[2] Fantasy SkyBox: https://www.assetstore.unity3d.com/en/#!/content/18353

[3] Gradient GUI Bars(life bar + XP bar): https://www.assetstore.unity3d.com/en/#!/content/19972

[4] Nature Starter Kit 2 (trees): https://www.assetstore.unity3d.com/en/#!/content/52977

[5] Raw Mocap Data (Animation of player): https://www.assetstore.unity3d.com/en/#!/content/5330

[6] Toon Soldier WW2 demo (Animation of player when shooting): https://www.assetstore.unity3d.com/en/#!/content/85702

[7] Scenes Music: https://www.youtube.com/watch?v=n0AY4ebPy4w

[8] Ak sound: https://www.youtube.com/watch?v=j7xRaJrEARE

[9] Sniper sound: https://www.youtube.com/watch?v=1BMyfnnkXfw

[10] Shotgun sound: https://www.youtube.com/watch?v=frLRIrLI7aA

[11] Ironman: https://free3d.com/3d-model/ironman-82411.html

[12] Catwoman: http://www.domawe.net/2015/03/catwoman-free-3d-model.html

[13] Luffy: https://free3d.com/3d-model/luffy-13431.html

[14] Scorpion: https://free3d.com/3d-model/scorpion-20143.html

[15] Gi: http://es.best-free-model.net/people-and-related-ware/people-and-body-parts/makoto-aihara-3d-model-40837/

[16] Worm: http://www.cadnav.com/3d-models/model-29067.html

[17] Snail: https://free3d.com/3d-model/snails-9980.html

[18] Bird: https://free3d.com/3d-model/cardinal-bird-2-34773.html

[19] Frog: https://free3d.com/3d-model/frog-3d-model-78864.html

[20] Dragon Boss: https://free3d.com/3d-model/blue-eyes-ultimate-dragon-28772.html

[21] Player: https://free3d.com/3d-model/dante-32520.html

[22] Machine Gun: https://free3d.com/3d-model/ak-47-79083.html

[23] Shotgun: https://free3d.com/3d-model/halo-4-shotgun-with-hd-textures-28534.html

[24] Sniper: https://free3d.com/3d-model/sniper-rifle-m200-36415.html

[25] Cave: https://sketchfab.com/models/d8662e3c5ce1462e81f5ed5563e1d08b

[26] Sunglasses: https://free3d.com/3d-model/raybanz-sunglasses-50410.html

[27] Houses: https://free3d.com/3d-model/environment-23742.html

[28] Fountain: https://free3d.com/3d-model/fountain-94350.html

[29] Grass path: http://images.torontofootysevens.com/footysevens/useruploaded/cropSmall/1.jpg

[30] Stone path: https://previews.123rf.com/images/thoth11/thoth110602/thoth11060200006/318748-Stone-Path-Stock-Photo-pathway.jpg

[31] Lake: https://previews.123rf.com/images/criminalatt/criminalatt1309/criminalatt130900237/22272032-Turquoise-sea-water-surface-seamless-background-texture-Stock-Photo.jpg

[32] Ground1: https://www.123rf.com/photo_20273905_soil-surface-background.html

[33] Cave Entrance: https://pre07.deviantart.net/166f/th/pre/f/2008/287/8/9/black_texture___ray_by_ethenyl.jpg

[34] Waterlily: https://gr.depositphotos.com/33771399/stock-photo-leaf-of-water-lily.html

[35] Fireball: http://www.wallpapersxl.com/wallpaper/1920x1080/llamas-de-pantalla-infierno-en-imagen-1383343.html

[36] Ground2: https://s-media-cache-ak0.pinimg.com/originals/ee/fa/55/eefa55bcafd2588ea776750fcd90e599.jpg

[37] Cave Exit: https://jakeofwinterhill.files.wordpress.com/2010/09/img01282.jpg?w=400&h=300

[38] Start Screen (edited): https://pre04.deviantart.net/423b/th/pre/f/2013/259/5/0/thedevilswork_by_lonewolf117-d6miewi.jpg

[39] Main grass: https://www.google.gr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwjS0Oanm97VAhWB2hoKHfpMAvwQjBwIBA&url=https%3A%2F%2Fforum.lumion3d.com%2Findex.php%3Faction%3Ddlattach%3Btopic%3D2882.0%3Battach%3D5150&psig=AFQjCNF1rbgqrxtPjKhjfhCSPCwupNQ37w&ust=1503057389894964

[40] Retry Screen (edited): https://steamuserimages-a.akamaihd.net/ugc/544132877198665147/3161E4BC40826D92530C3A2F38FE2EF7D8F62AA2/

[41] Loading Screen (edited): http://wallpapercave.com/wp/4c8xmGs.jpg

[42] SniperScope: http://vignette1.wikia.nocookie.net/callofduty/images/5/56/Default_sniper_scope_reticle.png/revision/latest?cb=20101008131509

[43] Village Wooden Wall (edited): https://3dw4cv24jbb7227rud1vdp6b-wpengine.netdna-ssl.com/wp-content/uploads/2012/09/Dusty-Wood-Texture-500x357.jpg

[44] Pointer: https://t3.rbxcdn.com/bf8435ab33773367722989c54f69c087

[45] ApplicationIcon: https://orig00.deviantart.net/8eb4/f/2013/047/e/1/devil_may_cry_4_icon_b_blank_by_them4cgodfather-d5v7dz0.png

[46] Menu Icon: https://sumy-1iwnfklpo3kun6wqra.netdna-ssl.com/wp-content/uploads/2015/08/hamb.png