ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής

ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗ

Μεταπτυχιακή Διατριβή

ΚΑΘΗΓΗΤΕΣ: **Βίρβου Μαρία**

Τίτλος Διατριβής	***Δημιουργία παιχνιδιού 3D με χρήση της πλατφόρμας Unity***
Όνομα φοιτητή – Αρ. Μητρώου	**Μαργαρίτης Καμήτσιος ΜΠΠΛ 13026**
Ημερομηνία παράδοσης	**Οκτώβριος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο	Όνομα Επώνυμο	Όνομα Επώνυμο
Βαθμίδα	Βαθμίδα	Βαθμίδα

Περίληψη

Το παιχνίδι μας δημιουργήθηκε στην πλατφόρμα Unity 2017.1.0f3. Στο κεφάλαιο 1, κάνουμε αναφορά για το VR στην εκπαίδευση, στο κεφάλαιο 2, κάνουμε ανασκόπηση πεδίου, αναφέροντας την ιστορική εξέλιξη των παιχνιδιών και αναλύοντας γενικότερα την βιομηχανία των παιχνιδιών. Στο κεφάλαιο 3, περιγράφουμε το σενάριο και τον σκοπό του παίχτη. Στο κεφάλαιο 4, κάνουμε ανάλυση, σχεδιασμό και υλοποίηση με διαγράμματα Uml, συγκεκριμένα με Use Cases και με Class diagrams. Στο κεφάλαιο 5, περιγράφουμε αναλυτικά όλες τις λειτουργίες του παιχνιδιού χρησιμοποιώντας εικόνες και αναλύοντας τον κώδικα που χρησιμοποιήσαμε. Τέλος, στο κεφάλαιο 6, περιγράφουμε τα συμπεράσματα και τις μελλοντικές επεκτάσεις του παιχνιδιού, συγκεκριμένα σε gaming, σε εκπαιδευτικό επίπεδο αλλά και σε πολιτιστικό επίπεδο.

Abstract

Our game was created on the Unity 2017.1.0f3 platform. In Chapter 1, we report on VR in education, in Chapter 2, we are reviewing the field, referring to the historical evolution of games and analyzing the game industry in general. In Chapter 3, we describe the player's scenario and purpose. In Chapter 4, we analyze, design and implement UML diagrams, namely Use Cases and Class Diagrams. In Chapter 5, we describe in detail all the game's functions using images and analyzing the code we used. Finally, in Chapter 6, we outline the conclusions and future extensions of the game, namely in gaming, at the educational level as well as at the cultural level.

Στους γονείς μου και στη σύζυγό μου,
για την αμέριστη στήριξη και συμπαράστασή τους μέχρι και σήμερα

Πίνακας Περιεχομένων

Κεφάλαιο 1ο - Εισαγωγή

Το παιχνίδι μας δημιουργήθηκε στην πλατφόρμα Unity 2017.1.0f3. Για την δημιουργία του παιχνιδιού αυτού έγινε αγορά και χρήση πακέτων από το κατάστημα unity asset store. Η αρχιτεκτονική των πακέτων που χρησιμοποιήθηκαν παρουσιάζεται με την μορφή διαγράμματος στο κεφάλαιο 4.2. Ας δώσουμε όμως μια σύντομη περιγραφή του κάθε κεφαλαίου.

Στο κεφάλαιο 2, κάνουμε ανασκόπηση πεδίου. Αναφέρουμε λοιπόν την ιστορική εξέλιξη των παιχνιδιών και δίνουμε τροφή για σκέψη για την μορφή του gaming στο μέλλον. Έπειτα αναφερόμαστε γενικά στην βιομηχανία των βιντεοπαιχνιδιών, μέσω διαγραμμάτων. Στα διαγράμματα αυτά βλέπουμε τους τζίρους των βιντεοπαιχνιδιών, τα target group στα οποία στοχεύουν, λίστα με τις σημαντικότερες πλατφόρμες και τάσεις των παιχτών. Έπειτα παρουσιάζουμε το συνολικό κόστος παραγωγής ενός βιντεοπαιχνιδιού αλλά και την ταχύτητα απόσβεσης κεφαλαίου. Κάνουμε σύγκριση στις σημαντικότερες πλατφόρμες δημιουργίας βιντεοπαιχνιδιών καθώς επίσης και στις σημαντικότερες πλατφόρμες ψηφιακής διάθεσης βιντεοπαιχνιδιών.

Στο κεφάλαιο 3, περιγράφουμε το σενάριο με ιστορική αναδρομή από το παρελθόν στο παρόν αλλά και τον σκοπό του παίχτη στο παιχνίδι μας.

Στο κεφάλαιο 4, κάνουμε ανάλυση, σχεδιασμό και υλοποίηση με διαγράμματα Uml. Συγκεκριμένα χρησιμοποιώντας διαγράμματα Use Cases αναλύουμε την γενική επαφή του χρήστη με το παιχνίδι μας, τις κινήσεις που μπορεί να κάνει ο χαρακτήρας αλλά και τις βασικές λειτουργίες του παίχτη μας. Έπειτα δείχνουμε με εικόνες την κατηγοριοποίηση που καναμε στις περιοχές του παιχνιδιού, έτσι ώστε να γίνουν τα διαγράμματα κλάσης πιο κατανοητά από τον αναγνώστη. Τέλος με Class diagrams παρουσιάζουμε τις βασικές κλάσεις του παιχνιδιού μας, την σύνδεση βασικής μουσικής με κώδικα ανά περιοχή αλλά και την αρχιτεκτονική των πακέτων που χρησιμοποιήθηκαν ανά περιοχή παιχνιδιού.

Στο κεφάλαιο 5, περιγράφουμε αναλυτικά όλες τις λειτουργίες του παιχνιδιού χρησιμοποιώντας εικόνες και αναλύοντας τον κώδικα που χρησιμοποιήσαμε. Συγκεκριμένα αναλύουμε: 1ον την δημιουργία και τις λειτουργίες των απέθαντων εχθρών, 2ον την δημιουργία και λειτουργία παγίδων, 3ον την επεξεργασία παίχτη και τις νέες λειτουργίες του, 4ον την λειτουργία πυλών, κλειδιών και σεντουκιών, 5ον την λειτουργία medikit, blue potion και red potion, 6ον την λειτουργία canvas και panels, 7ον την λειτουργία startmenu, 8ον την λειτουργία επιχειρήσεων στο παιχνίδι, 9ον την λειτουργία βασικής και εφεδρικής μουσικής, 10ον την λειτουργία πτώσεων-σύνδεση με gameover και 11ον όλες τις άλλες λειτουργίες.

Τέλος, στο κεφάλαιο 6, περιγράφουμε τα συμπεράσματα και τις μελλοντικές επεκτάσεις του παιχνιδιού. Συγκεκριμένα αναλύουμε λεπτομερώς τις επεκτάσεις σε επίπεδο gaming, σε εκπαιδευτικό επίπεδο αλλά και σε πολιτιστικό επίπεδο.

Η χρήση τεχνολογίας εικονικής πραγματικότητας(VR) αποτελεί το μέλλον στην διαδικασία διδασκαλίας. Και αυτό συμβαίνει διότι:

<< 1. Παρέχει εξαιρετικές απεικονίσεις που δεν είναι δυνατές στην παραδοσιακή αίθουσα διδασκαλίας.

Η εικονική πραγματικότητα είναι θαυμάσια, διότι μας επιτρέπει να διερευνήσουμε διαφορετικές πραγματικότητες και να εναλλάξουμε τις εμπειρίες μας. Φορώντας ένα ακουστικό VR, αντιμετωπίζουμε οπτικοποιήσεις υψηλής ποιότητας που μπορούν να μας σημαδέψουν με θετικό τρόπο. Διότι οι εικόνες μπορούν πραγματικά να μας βοηθήσουν να μάθουμε καλύτερα.

Οι παραδοσιακές μέθοδοι διδασκαλίας δεν μπορούν ποτέ να επιτύχουν έναν τόσο αποτελεσματικό τρόπο έμφασης των πραγμάτων μέσω απεικονίσεων.

2. Δημιουργεί ενδιαφέρον.

Όποια ηλικία έχουν, οι μαθητές θα αγαπούν πάντα να κάθονται και να παρακολουθούν κάτι αντί να το διαβάζουν. Η τεχνολογία VR είναι πολύ ενδιαφέρουσα, καθώς μπορεί να δημιουργήσει εκπληκτικές εμπειρίες που δεν θα μπορούσαν ποτέ να "ζουν" στην πραγματική ζωή. Οι μαθητές θα αισθάνονται σίγουρα μεγαλύτερο κίνητρο να μάθουν με τη χρήση αυτής της τεχνολογίας.

3. Αυξάνει την αλληλεπίδραση των μαθητών.

Σήμερα, οι καθηγητές θεωρούν ότι είναι δύσκολο να δημιουργηθεί μια παραγωγική αλληλεπίδραση μέσα στην τάξη. Με την τεχνολογία εικονικής πραγματικότητας που υπάρχει στην εκπαίδευση, αυτή η πτυχή θα εξαφανιστεί για πάντα, καθώς οι περισσότεροι από τους μαθητές θα νιώσουν τον πειρασμό να μιλήσουν για τις εμπειρίες τους μέσα στην εικονική πραγματικότητά τους.

4. Δεν το αισθάνονται σαν εργασία.

Ας το παραδεχτούμε. Τοποθετώντας ένα ακουστικό στο κεφάλι μας και βλέποντας τα πράγματα να εξελίσσονται μπροστά στα μάτια μας, μαθαίνοντας νέες πληροφορίες μέσω βίντεο και εκπληκτικές απεικονίσεις, δεν μοιάζει με εργασία. Αν μπορούμε να κάνουμε την παιδεία διασκεδαστική, τα παιδιά θα λαχταρούν να μάθουν περισσότερα πράγματα και να είναι πιο φιλόδοξα.

Αυτός είναι βασικά ένας γενικός κανόνας. Όταν απολαμβάνουμε να κάνουμε κάτι, θα το κάνουμε με μεγαλύτερο ενδιαφέρον, θα το κάνουμε καλύτερα και δεν θα νιώσουμε σαν να κάνουμε κάτι οδυνηρό.

5. Βελτιώνει την ποιότητα της εκπαίδευσης σε διαφορετικούς τομείς.

Πάρτε το φάρμακο για παράδειγμα. Το 2016, οι πρωτοποριακοί γιατροί επωφελούνται από την τεχνολογία VR προκειμένου να διερευνήσουν νέες πτυχές της ιατρικής και να διδάξουν τους άλλους καλύτερα. Ένα άλλο παράδειγμα θα είναι το πεδίο γραφής και επεξεργασίας περιεχομένου. Η εικονική πραγματικότητα μπορεί συχνά να βοηθήσει στην εύρεση σφαλμάτων στο περιεχόμενο και να προσφέρει εξαιρετικές δυνατότητες επεξεργασίας.

6. Εξαλείφει το φράγμα γλωσσών.

Το γλωσσικό εμπόδιο είναι συχνά ένα μεγάλο πρόβλημα όταν πρόκειται για την εκπαίδευση. Αν θέλετε να σπουδάσετε σε διαφορετική χώρα, πρέπει να καταλάβετε και να μιλήσετε τη γλώσσα. Με την εικονική πραγματικότητα, κάθε δυνατή γλώσσα μπορεί να εφαρμοστεί μέσα στο λογισμικό. Επομένως, η γλώσσα δεν θα αποτελεί πλέον εμπόδιο για τα εκπαιδευτικά σχέδια του μαθητή>>[*]

Κεφάλαιο 2^{ο} - Ανασκόπηση Πεδίου

2.1 Παιχνίδια: Από το Παρελθόν στο Μέλλον

<<Η ιστορία των βιντεοπαιχνιδιών ξεκινάει ήδη από τις αρχές της δεκαετίας του 1950, όταν οι ακαδημαϊκοί επιστήμονες υπολογιστών άρχισαν να σχεδιάζουν απλά παιχνίδια και προσομοιώσεις στο πλαίσιο της έρευνάς τους. Το video gaming δεν έφθασε στη γενική δημοτικότητα μέχρι τη δεκαετία του 1970 και του 1980, όταν εισήχθησαν στο ευρύ κοινό τα arcade video games και κονσόλες παιχνιδιών που χρησιμοποιούν χειριστήρια, κουμπιά και άλλους controllers, καθώς και γραφικά σε οθόνες υπολογιστών και παιχνίδια στο σπίτι. Από τη δεκαετία του 1980, τα video games έχουν γίνει μια δημοφιλής μορφή διασκέδασης και ένα μέρος της σύγχρονης λαϊκής κουλτούρας στα περισσότερα μέρη του κόσμου. Ένα από τα πρώτα παιχνίδια ήταν το Spacewar, το οποίο αναπτύχθηκε από επιστήμονες υπολογιστών. Τα πρώτα arcade video games αναπτύχθηκαν από το 1972 έως το 1978. Κατά τη διάρκεια της δεκαετίας του 1970, προέκυψε η πρώτη γενιά κονσόλων στο σπίτι, συμπεριλαμβανομένου του δημοφιλούς παιχνιδιού Pong και διάφορων "κλώνων". Η δεκαετία του 1970 ήταν επίσης η εποχή των παιχνιδιών κεντρικών υπολογιστών(mainframe computer games). Η χρυσή εποχή των arcade video games ήταν από το 1978 έως το 1982. Video arcades με μεγάλα, διακοσμημένα με γραφικά μηχανήματα που λειτουργούσαν με κέρματα ήταν κοινά σε εμπορικά κέντρα και οι δημοφιλείς, προσιτές κονσόλες στο σπίτι όπως το Atari 2600 και το Intellivision επέτρεψαν στους ανθρώπους να παίζουν παιχνίδια στο σπίτι τους με την χρήση τηλεόρασης. Κατά τη διάρκεια της δεκαετίας του 1980, εμφανίστηκαν υπολογιστές gaming, πρώιμα online παιχνίδια και φορητές συσκευές LCD παιχνιδιών. Από το 1976 έως το 1992, εμφανίστηκε η δεύτερη γενιά κονσόλων βίντεο.

Η τρίτη γενιά κονσολών, που ήταν μονάδες 8 bit, εμφανίστηκε από το 1983 έως το 1995. Η τέταρτη γενιά κονσολών, που ήταν 16-bit μοντέλα, προέκυψε από το 1987 έως το 1999. Η δεκαετία του 1990 είδε την αναζωπύρωση και την παρακμή των arcades, την μετάβαση σε 3D video games, βελτιωμένα παιχνίδια χειρός και παιχνίδια για υπολογιστές. Η πέμπτη γενιά κονσολών, που ήταν μονάδες 32 και 64 bit, ήταν από το 1993 έως το 2006. Κατά τη διάρκεια αυτής της εποχής, εμφανίστηκε το mobile gaming. Κατά τη διάρκεια της δεκαετίας του 2000, προέκυψε η έκτη γενιά κονσολών (1998-2013). Κατά τη διάρκεια αυτής της περιόδου, τα παιχνίδια στο διαδίκτυο και τα mobile games έγιναν σημαντικές πτυχές της gaming κουλτούρας. Η έβδομη γενιά κονσολών ήταν από το 2005 έως το 2012. Αυτή η εποχή χαρακτηρίστηκε από τεράστιους αναπτυξιακούς προϋπολογισμούς για ορισμένα παιχνίδια, με μερικούς να έχουν κινηματογραφικά γραφικά(cinematics). Σημειώθηκε η έναρξη της κονσόλας Wii με τις κορυφαίες πωλήσεις, κονσόλα στην οποία ο χρήστης θα μπορούσε να ελέγχει τις ενέργειες του παιχνιδιού με την πραγματική κίνηση του controller. Σημειώθηκε η

άνοδος των περιστασιακών παιχνιδιών PC που διατίθενται στο εμπόριο σε μη-παικτες και η εμφάνιση του cloud computing στα βιντεοπαιχνίδια.

Το 2013 εμφανίστηκε η όγδοη γενιά κονσολών, συμπεριλαμβανομένων των Wii U και Nintendo 3DS της Nintendo, του Xbox One της Microsoft και του PlayStation 4 της Sony και του PlayStation Vita. Το PC gaming έχει κρατήσει μεγάλο μερίδιο αγοράς στην Ασία και την Ευρώπη για δεκαετίες και συνεχίζει να αυξάνεται λόγω της ψηφιακής διανομής. Από την ανάπτυξη και την ευρεία χρήση των smartphones από τους καταναλωτές, τα mobile παιχνίδια αποτελούν τον κινητήριο παράγοντα των παιχνιδιών, καθώς μπορούν να προσεγγίσουν ανθρώπους που δεν ενδιαφέρονται για παιχνίδια και εκείνους που δεν μπορούν να αντέξουν οικονομικά ή να υποστηρίξουν ειδικό hardware, όπως κονσόλες βιντεοπαιχνιδιών.>>[1]

Ας δούμε όμως και το μέλλον.

<<Ένας από τους μεγάλους περιορισμούς των σημερινών συσκευών εικονικής πραγματικότητας (VR) είναι ότι πρέπει να είναι συνδεδεμένες με καλώδιο σε υπολογιστές για να γίνεται όπως πρέπει η επεξεργασία που θα επιτρέπει την προβολή γραφικών υψηλής ανάλυσης. Ωστόσο, το καλώδιο μειώνει την ευκινησία και μπορεί να έχει ως αποτέλεσμα οι χρήστες να μπλεχτούν με αυτό, με δυσάρεστα αποτελέσματα.

Ερευνητές του CSAIL (Computer Science and Artificial Intelligence Laboratory) πρόσφατα αποκάλυψαν ένα πρωτότυπο σύστημα ονόματι MoVR το οποίο επιτρέπει στους gamers να χρησιμοποιούν οποιοδήποτε VR headset ασύρματα. Οι ερευνητές δοκίμασαν το σύστημα στο HTC Vive, αλλά υποστηρίζουν ότι μπορεί να λειτουργήσει με οποιαδήποτε συσκευή.

Όπως φάνηκε σε δοκιμές, το MoVR μπορεί να επιτρέψει ασύρματη επικοινωνία σε επίπεδο πολλαπλών Gbps, χρησιμοποιώντας ειδικά ραδιοσήματα υψηλής συχνότητας, τα αποκαλούμενα mmWaves (millimeter waves, κύματα χιλιοστών), που σύμφωνα με αρκετούς ερευνητές, κάποια ημέρα θα βοηθήσουν στην άφιξη ταχύτατων 5G smartphones.

Τα mmWaves υπόσχονται πολλά σε μια μεγάλη γκάμα εφαρμογών, από Ίντερνετ υψηλών ταχυτήτων μέχρι ιατρικές διαγνώσεις. Τα κύματα αυτά έχουν ένα μεγάλο πλεονέκτημα, ωστόσο: Δεν λειτουργούν καλά όταν υπάρχουν εμπόδια ή αντανακλάσεις, οπότε εάν σκοπός είναι η συνεχής σύνδεση για, πχ, ένα VR παιχνίδι, χρειάζεται πάντα οπτική επαφή ανάμεσα στον πομπό και τον δέκτη- ακόμα και ένα χέρι το οποίο θα παρεμβληθεί στιγμιαία μπορεί να δημιουργήσει πρόβλημα.

Αυτό που έκαναν οι ερευνητές του MIT ήταν να αναπτύξουν το MoVR ως έναν προγραμματιζόμενο καθρέφτη, που εντοπίζει την κατεύθυνση του εισερχόμενο mmWave κύματος και ρυθμίζεται από μόνο του για να το ανακλά προς την κατεύθυνση του δέκτη στη συσκευή VR. Το MoVR μπορεί να «μαθαίνει» τη σωστή κατεύθυνση του σήματος με ακρίβεια δύο μοιρών, έτσι ώστε να καθορίζονται σωστά οι γωνίες του. Το

MoVR αποτελείται από δύο κατευθυντικές κεραίες, που η καθεμία είναι μεγέθους μικρότερου του μισού μιας πιστωτικής κάρτας. Οι κεραίες χρησιμοποιούν «phased arrays» για να εστιάζουν τα σήματα σε στενές ακτίνες, που μπορούν να κατευθύνονται ηλεκτρονικά μέσα σε κλάσματα δευτερολέπτου.

Οι ερευνητές εκτιμούν πως οι μελλοντικές εκδόσεις του MoVR θα μπορούσαν να έχουν μέγεθος smartphone, επιτρέποντας την ταυτόχρονη χρήση πολλών συσκευών σε ένα δωμάτιο, και επιτρέποντας σε πολλά άτομα να παίξουν ταυτόχρονα ένα παιχνίδι, χωρίς το ένα να μπλοκάρει το σήμα του άλλου.>>[2]

<<Ερευνητές του University of Washington έκαναν ένα σημαντικό πρώτο βήμα στην κατεύθυνση του χειρισμού σε ηλεκτρονικά παιχνίδια εικονικής πραγματικότητας μέσω απευθείας διέγερσης εγκεφάλου- φέρνοντας στο νου το τεχνητό σύμπαν του «Matrix», όπου οι πρωταγωνιστές συνδέονταν και δρούσαν σε έναν εικονικό κόσμο, ενώ στον πραγματικό φαίνονταν σαν να κοιμούνται.

Σε paper που δημοσιεύτηκε online στις 16 Νοεμβρίου στο Robotics and AI, οι ερευνητές περιγράφουν την πρώτη επιτυχή επίδειξη όπου άνθρωποι παίζουν ένα απλό, δισδιάστατο παιχνίδι στον υπολογιστή, μόνο μέσω διέγερσης εγκεφάλου, χωρίς να απαιτούνται άλλες αισθήσεις, όπως όραση, ακοή και αφή.

Οι παίκτες έπρεπε να κινηθούν σε 21 διαφορετικούς λαβυρίνθους, με δύο επιλογές: Να πάνε μπροστά ή κάτω, απόφαση που είχε να κάνει με ένα αντικείμενο οπτικής διέγερσης (φωσφώνιο), που εκλαμβανόταν ως κύκλος ή μπάρα φωτός. Για να υποδείξει πού έπρεπε να κινηθεί, οι ερευνητές παρήγαγαν ένα φωσφώνιο μέσω διακρανιακής μαγνητικής διέγερσης- μιας τεχνικής που χρησιμοποιεί ένα μαγνητικό πηνίο κοντά στο κρανίο για την άμεση και μη επεμβατική διέγερση μιας συγκεκριμένης περιοχής του εγκεφάλου.

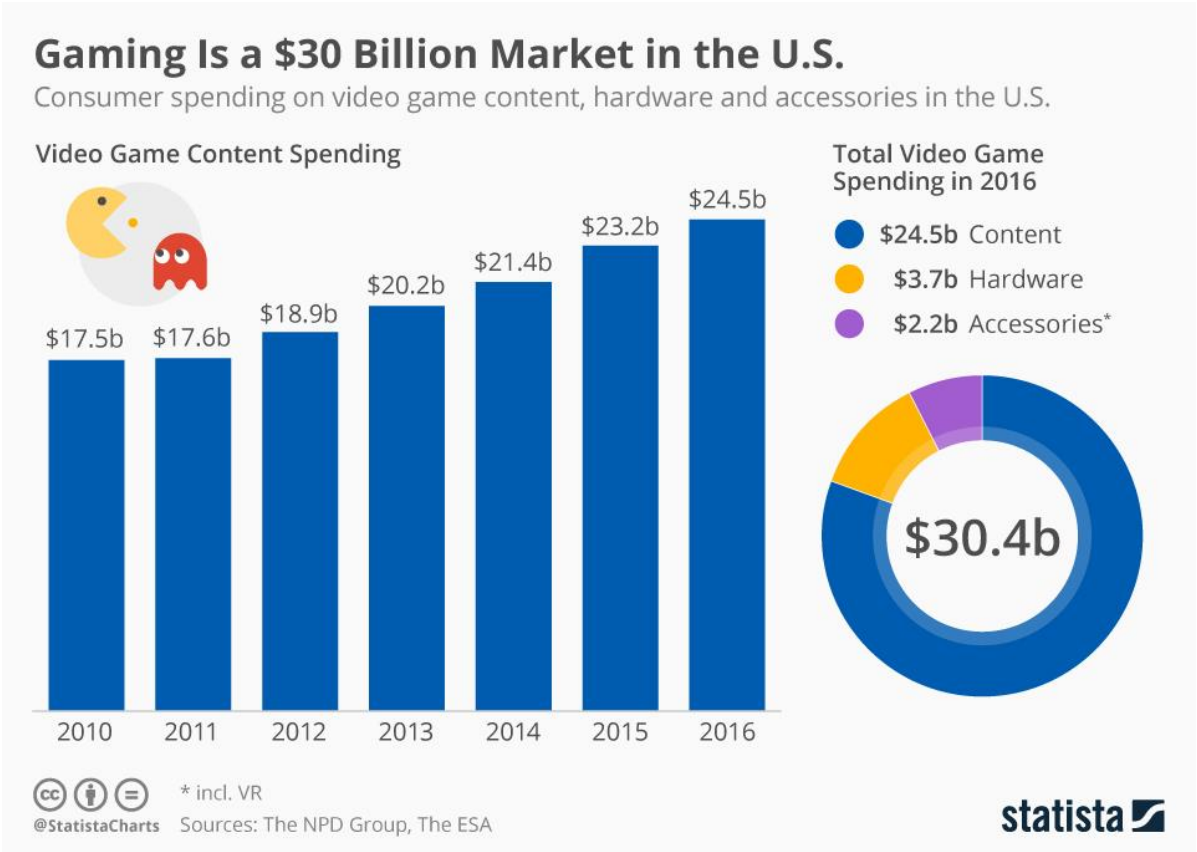
«Η εικονική πραγματικότητα σήμερα λαμβάνει χώρα μέσω οθονών, headsets και διοπτρών, αλλά είναι ο εγκέφαλός σου που δημιουργεί την πραγματικότητά σου» λέει ο Ρατζές Ράο, καθηγητής επιστήμης υπολογιστών και μηχανολογίας του UW. «Το θεμελιώδες ερώτημα που θέλαμε να απαντήσουμε ήταν: Μπορεί ο εγκέφαλος να αξιοποιήσει τεχνητές πληροφορίες που δεν έχει δει ποτέ, οι οποίες παρέχονταν απευθείας σε αυτόν, για να πλοηγηθεί σε έναν εικονικό κόσμο ή να κάνει εργασίες χωρίς δεδομένα από άλλες αισθήσεις; Και η απάντηση είναι “ναι”»

Τα πέντε άτομα που συμμετείχαν στη δοκιμή προέβησαν στις σωστές κινήσεις στους λαβυρίνθους στο 92% των περιπτώσεων, όταν λάμβαναν τις πληροφορίες μέσω άμεσης διέγερσης εγκεφάλου, εν συγκρίσει με το 15% που ίσχυε αντίστοιχα όταν δεν υπήρχε καθοδήγηση. Το απλό αυτό παιχνίδι επιδεικνύει έναν τρόπο με τον οποίο νέες πληροφορίες από τεχνητούς αισθητήρες ή παραγόμενους από υπολογιστή εικονικούς κόσμους μπορούν να κωδικοποιηθούν επιτυχώς και να αποσταλούν μη επεμβατικά στον ανθρώπινο εγκέφαλο, ώστε να γίνονται εργασίες. Επίσης, τα άτομα αυτά έγιναν πιο ικανά στην πλοήγηση με το πέρασμα του χρόνου, κάτι που υποδεικνύει ότι ήταν ικανά να μάθουν να αντιλαμβάνονται καλύτερα τα τεχνητά ερεθίσματα. «Στην ουσία

προσπαθούμε να δώσουμε στους ανθρώπους μια έκτη αίσθηση» λέει χαρακτηριστικά ο επικεφαλής συντάκτης της έρευνας, Ντάρμπι Λόσεϊ.>>[3]

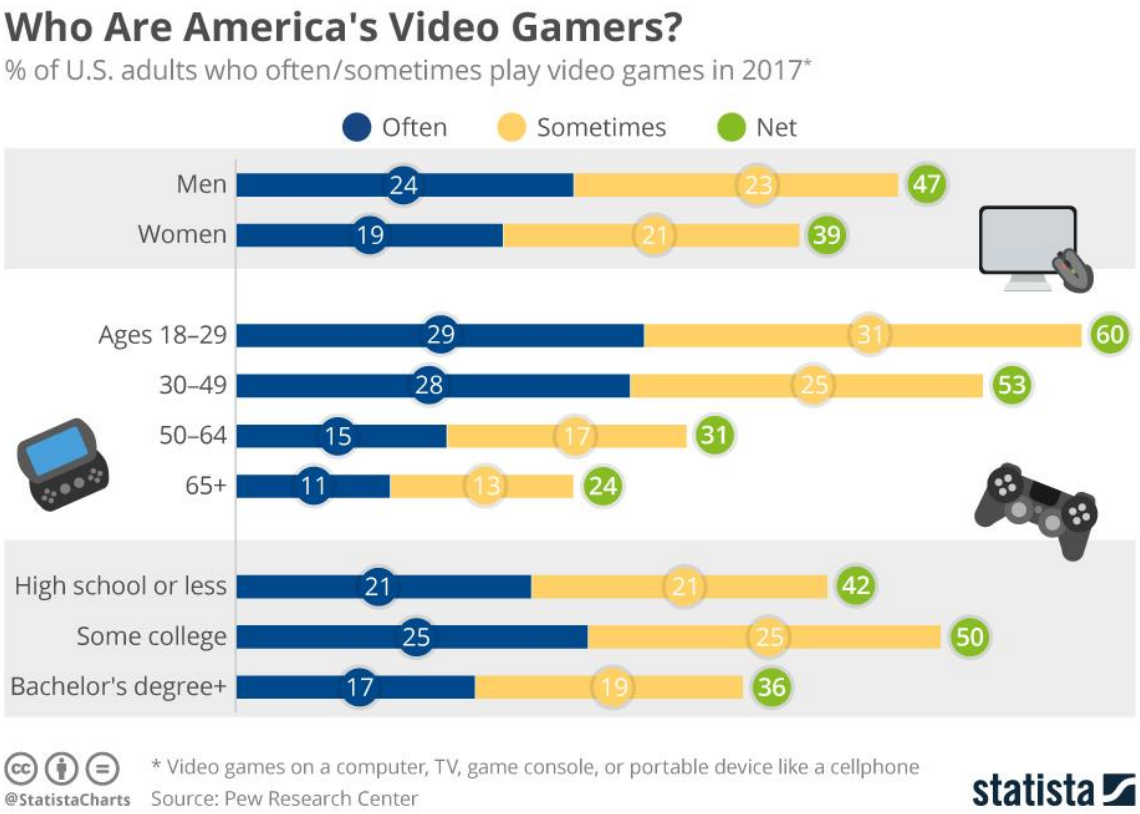
2.2 Η Βιομηχανία των Βιντεοπαιχνιδιών

Η βιομηχανία των βιντεοπαιχνιδιών αποτελεί πλέον έναν κολοσσό. <<Σύμφωνα με την Ένωση Λογισμικού Ψυχαγωγίας, οι Αμερικανοί ξόδεψαν το ποσό των 24,5 δισεκατομμυρίων δολαρίων για περιεχόμενο βιντεοπαιχνιδιών πέρυσι, σε σύγκριση με μόλις 3,7 δισεκατομμύρια δολάρια για το υλικό.>>[4]



Σε ποιους όμως απευθύνονται τα videogames; Ο κοινός μύθος που ήθελε τα videogames να απευθύνονται σε αντρικό κοινό νεαρής ηλικίας έχει καταρριφθεί. <<Σύμφωνα με την Pew Research, το 21% των γυναικών παίζουν παιχνίδια τουλάχιστον μερικές φορές, ενώ το 19% λένε ότι παίζουν συχνά. Οι ηλικιωμένοι Αμερικανοί απολαμβάνουν επίσης τα βιντεοπαιχνίδια με ένα τέταρτο των ατόμων ηλικίας 65 ετών και άνω να λένε ότι τα παίζουν τουλάχιστον μερικές φορές.

Το Gaming ποικίλλει επίσης ανά επίπεδο εκπαίδευσης. Οι απόφοιτοι κολλεγίων (36%) είναι πολύ λιγότερο πιθανό να παίξουν βιντεοπαιχνίδια από τους ενήλικες με κάποιο κολέγιο (50%). Στην έρευνα υπολογίστηκαν τα βιντεοπαιχνίδια που παίζονται σε υπολογιστή, τηλεόραση, κονσόλα παιχνιδιών ή φορητή συσκευή, όπως κινητό τηλέφωνο.>>[5]

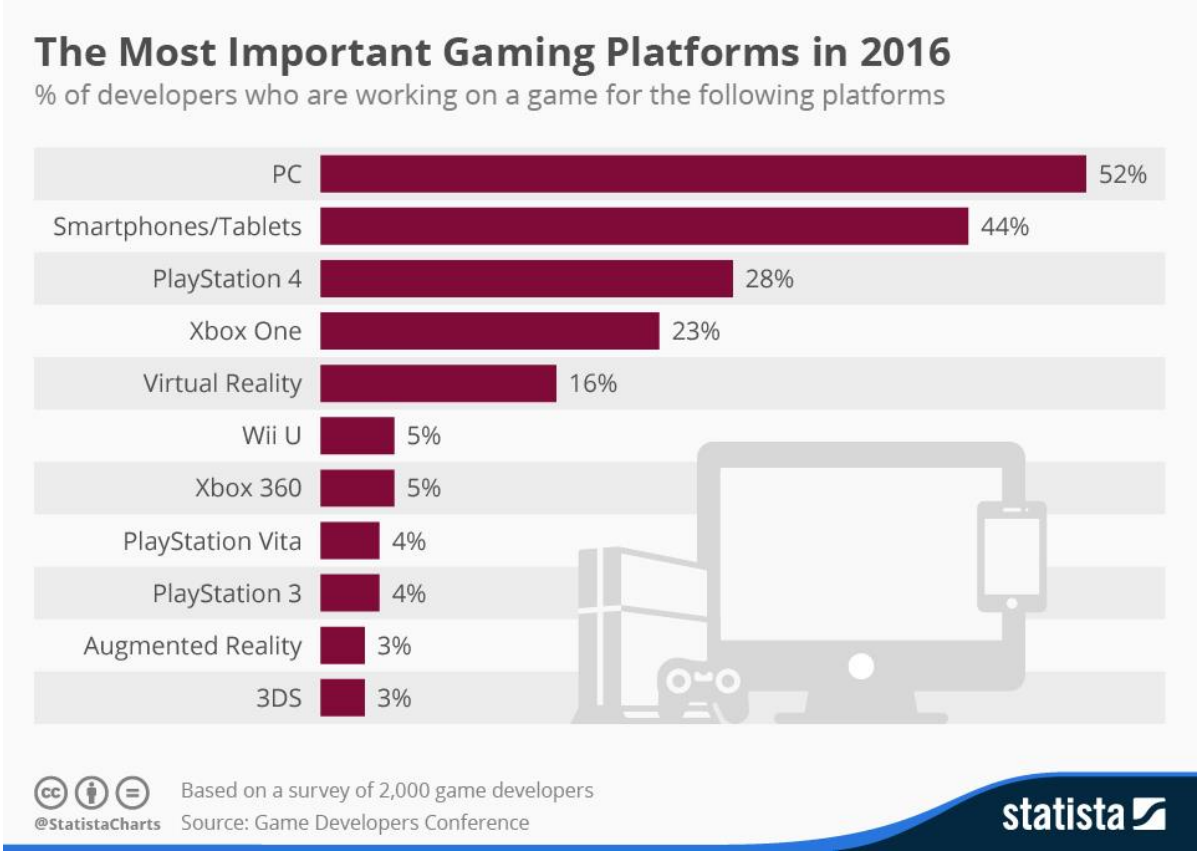


Ποια είναι όμως η πιο σημαντική πλατφόρμα gaming;

<<Για να είναι επιτυχημένη μια πλατφόρμα τυχερών παιχνιδιών, είναι απολύτως απαραίτητο να υποστηριχθεί από την κοινότητα των developers. Εξάλλου, πόσο καλό είναι το καλύτερο hardware εάν τα καλύτερα παιχνίδια δεν είναι διαθέσιμα σε αυτό;

Δεν αποτελεί έκπληξη το γεγονός ότι ο Η/Υ παραμένει πρωτοπόρος σε σχέση με την υποστήριξη των προγραμματιστών. Το 52% των ερωτηθέντων εργάζονται επί του παρόντος σε ένα παιχνίδι το οποίο θα κυκλοφορήσει σε υπολογιστές. Εν τω μεταξύ, οι κινητές συσκευές έχουν καθιερωθεί ως μια βιώσιμη πλατφόρμα βιντεοπαιχνιδιών. Τα smartphones είναι ιδιαίτερα ελκυστικά για τους προγραμματιστές λόγω του τεράστιου κοινού που μπορεί να επιτευχθεί. Η εικονική πραγματικότητα αναδύεται γρήγορα ως μια πλατφόρμα βιντεοπαιχνιδιών του μέλλοντος και έχει διπλασιάσει το επίπεδο της υποστήριξής της για προγραμματιστές σε σχέση με πέρυσι.

Μεταξύ της τρέχουσας γενιάς κονσολών, το PlayStation 4 είναι το πιο αγαπημένο από τους προγραμματιστές. Το PS4 προσπερνάει το Xbox One της Microsoft (28% έναντι 23%), ενώ το Wii U της Nintendo συνεχίζει να αγνοείται από μεγάλα τμήματα της κοινότητας των developers.>>[6]

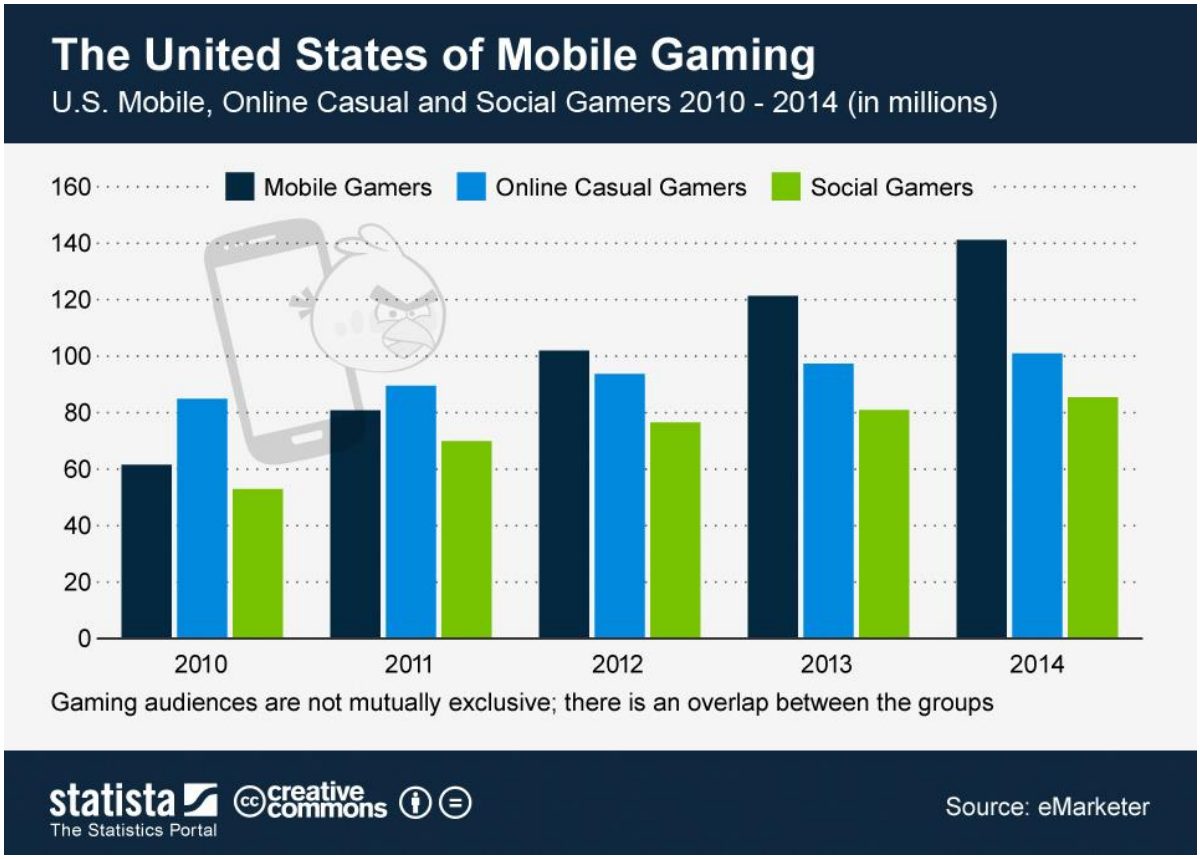


Οι τάσεις των παιχτών στρέφουν την βιομηχανία ηλεκτρονικών παιχνιδιών στο mobile gaming.

<<Η βιομηχανία που στήθηκε γύρω από τα ηλεκτρονικά παιχνίδια είναι κυριολεκτικά χρυσοφόρα, καθώς, όπως εκτιμάται, ο τζίρος του mobile gaming θα αγγίξει φέτος τα 36 δισ. δολάρια, νούμερο που θα αυξηθεί κατά τουλάχιστον 50% έως το 2019>>[7]

<<Σύμφωνα με το eMarketer, το 2013 θα υπάρχουν 121 εκατομμύρια Αμερικανοί mobile gamers. Για το 2014, ο αριθμός αυτός αναμένεται να φθάσει τα 141 εκατομμύρια, περίπου το 44% του πληθυσμού των ΗΠΑ. Αυτό καθιστά το mobile gaming την ταχύτερη ανάπτυξη όλων των κατηγοριών ψηφιακού gaming.

Η εξέλιξη του mobile gaming συνδέεται σαφώς με την ταχεία άνοδο των κινητών συσκευών. Τον Δεκέμβριο του 2012, 125,9 εκατομμύρια άνθρωποι στις ΗΠΑ κατείχαν smartphones, που ισοδυναμεί με ετήσια άνοδο 22%.>>[8]



Σε αυτό το σημείο, μπορούμε να δούμε και το συνολικό κόστος παραγωγής ενός videogame. Το συνολικό κόστος παραγωγής αποτελεί άθροισμα του κόστους δημιουργίας του, του κόστους μάρκετινγκ και των φόρων. Στην εικόνα παρακάτω παραθέτουμε με αναλυτικά στοιχεία την λίστα των 11 τίτλων videogames με το ακριβότερο συνολικό κόστος παραγωγής.[9]

List of most expensive video games to develop

From Wikipedia, the free encyclopedia

This list is incomplete; you can help by expanding it.

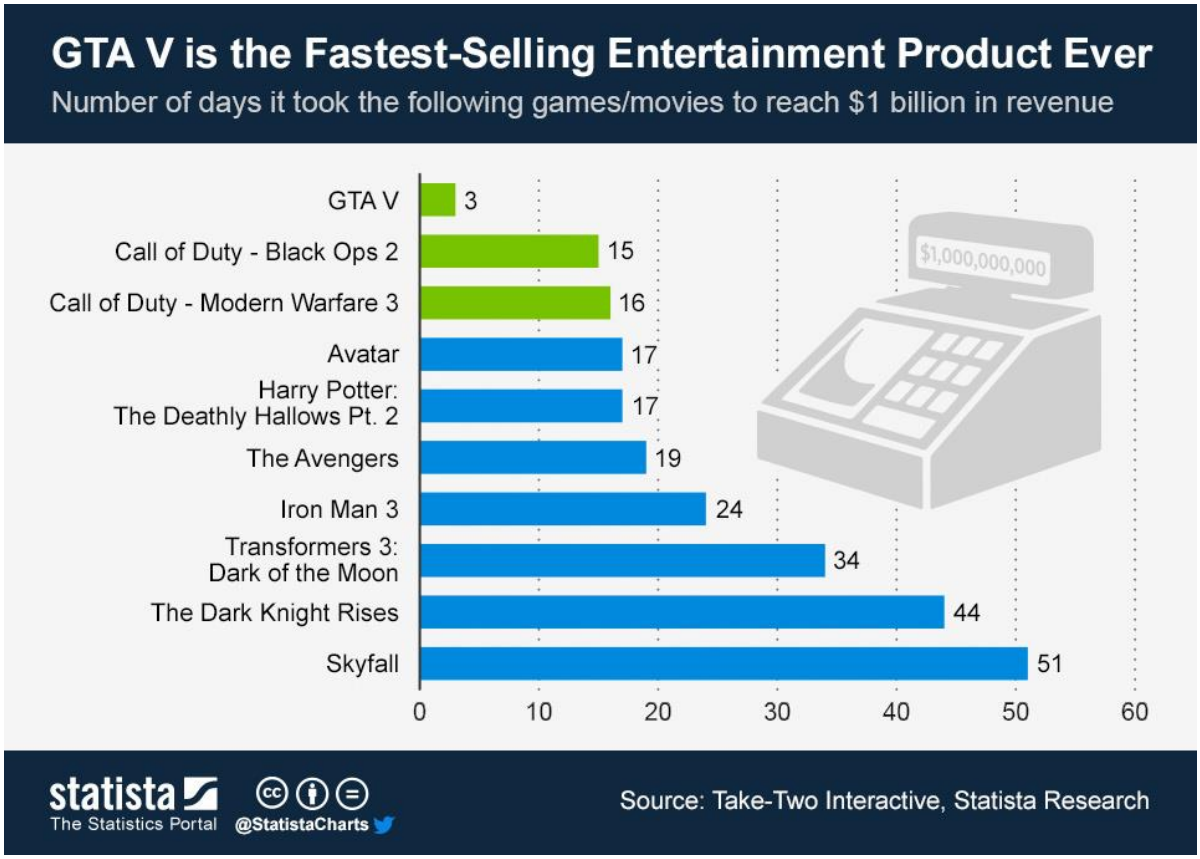
The following is a **list of** the **most expensive video games** ever developed, with a minimum total cost of US$50 million and sorted by the total cost adjusted for inflation.

Name	Year	Developer	Publisher	Platform	Development cost (million US$)	Marketing cost (million US$)	Total cost (million US$)	Total cost with 2017 inflation (million US$)
Call of Duty: Modern Warfare 2	2009	Infinity Ward	Activision	PC, PS3, Xbox 360	50	200	250[1]	279
Grand Theft Auto V	2013	Rockstar North	Rockstar Games	PC, PS3, PS4, Xbox 360, Xbox One	137[2]	128	265[3]	272
Final Fantasy VII	1997	Square	Square Enix, Sony Computer Entertainment (PS), Eidos Interactive (PC)	PS, PC	45[4]	100[5]	145	216
Star Wars: The Old Republic	2011	BioWare	Electronic Arts, LucasArts	PC	200[6]		200+[6]	213+
Destiny	2014	Bungie	Activision	PS3, PS4, Xbox 360, Xbox One	<140[7]	<140[7]	140[7][8][9]	142
Grand Theft Auto IV	2008	Rockstar North	Rockstar Games	PS3, Xbox 360, PC			100+[10]	111+
Too Human	2008	Silicon Knights	Microsoft Game Studios	Xbox 360	60-100[11]		100	111
APB: All Points Bulletin	2010	Realtime Worlds	Electronic Arts, Realtime Worlds, Deep Silver (PS4, XB1)	PC, PS4, Xbox One			100[12]	110
Max Payne 3	2012	Rockstar Studios	Rockstar Games	PC, Xbox 360, PS3	105[13]		105[13]	110
Red Dead Redemption	2010	Rockstar San Diego	Rockstar Games	PS3, Xbox 360	80-100[14]		100[15]	110
Deadpool	2013	High Moon Studios	Activision	PS3, PS4, Xbox 360, Xbox One, PC			100[16]	103

Ας δούμε όμως και την ταχύτητα με την οποία οι εταιρείες παιχνιδιών κάνουν απόσβεση χρημάτων.

<<Πριν από το πολυαναμενόμενο λανσάρισμα του παιχνιδιού GTA V, οι αναλυτές είχαν προβλέψει ότι το Take-Two Interactive θα μπορούσε πιθανώς να πουλήσει 15 με 20 εκατομμύρια μονάδες του παιχνιδιού και να κερδίσει έτσι ένα δισεκατομμύριο δολάρια μέχρι τον Μάρτιο του 2014. Η εκτίμηση αυτή αποδείχθηκε απλά συντηρητική: το παιχνίδι κέρδισε 800 εκατομμύρια δολάρια μέσα στις πρώτες 24 ώρες κυκλοφορίας του και μετά από λιγότερο από τρεις ημέρες, η Take-Two δημοσίευσε ένα δελτίο τύπου που ανέφερε ότι το παιχνίδι είχε μόλις περάσει πωλήσεις 1 δισεκατομμυρίου δολαρίων.

Όχι μόνο αυτό κάνει το GTA το ταχύτερο παιχνίδι για να φτάσει σε αυτό το ορόσημο, αλλά το ταχύτερα πωληθέν προϊόν ψυχαγωγίας όλων των εποχών. Οι δύο ταινίες με τα υψηλότερα κέρδη στην ιστορία, ο Τιτανικός και το Avatar του James Cameron, χρειάστηκαν πολύ περισσότερο για να τραβήξουν ένα δισεκατομμύριο. Η ταινία Avatar είναι στη τέταρτη θέση, ενώ η τελευταία ταινία του Χάρι Πότερ πέμπτη.>>[10]



2.3 Σύγκριση σε πλατφόρμες δημιουργίας παιχνιδιών

Οι πιο γνωστές πλατφόρμες δημιουργίας παιχνιδιών είναι η πλατφόρμα Unity, η Unreal Engine και το html 5 frameworks. Στην ανάλυση και σύγκρισή μας θα χρησιμοποιήσουμε τις εκδόσεις Unity5, Unreal Engine 4 και Html5 frameworks.

Το unity 5 μπορεί να χρησιμοποιηθεί για την δημιουργία μεγάλων έργων 3D αλλά και για μικρά 2D παιχνίδια. Πλεονεκτήματα του unity 5 είναι:

1^{ov} η γλώσσα προγραμματισμού C#, η οποία είναι γλώσσα υψηλού επιπέδου με πολλά στοιχεία και τεχνικές,

2^{ov} η μεταφορά του ίδιου κώδικα με μικροαλλαγές, σε διαφορετικές πλατφόρμες, όπως (Mac, Android, iOS, Web, WebGL και βιντεοκονσόλες),

3^{ov} Unity Community, όπου υπάρχει μια σαφής περιγραφή βασικών λειτουργιών της πλατφόρμας και προσωπικό το οποίο απαντάει σε ερωτήσεις

4ον Το Asset Store, ένα κατάστημα όπου μπορούμε να αναζητήσουμε οτιδήποτε, να αγοράσουμε πακέτα ή να προμηθευτούμε δωρεάν. Η εισαγωγή στην πλατφόρμα είναι άμεση.

5ον Η μηχανή αυτή υποστηρίζει assets από: 3ds Max, Maya, Softimage, CINEMA 4D, Blender.[12]

Η Unreal Engine 4 δημιουργήθηκε από την εταιρεία Epic Games και είναι ο διάδοχος του UDK. Είναι η πλατφόρμα δημιουργίας εκατοντάδων τίτλων. Πάρα πολλοί τίτλοι έγιναν εμπορικές επιτυχίες. Επιτυχίες όπως : Batman: Arkham Knight, Batman: Arkham Origins, Batman: Arkham City, Batman: Arkham Asylum, *Darksiders III,* Resident Evil 2: Reborn, Silent Hill: Downpour, Mortal Kombat Arcade Kollection HD, Mortal Kombat X, Unreal Tournament 2004, Duke Nukem Forever και πολλών άλλων. Μόνο τα ονόματα αυτών των τίτλων, μιλάνε μόνα τους για την δυναμική και την φήμη της πλατφόρμας. Προς το παρόν τα παινίδια της πλατφόρμας αυτής μπορούν να κυκλοφορήσουν σε PC, Mac, iOS, Android, Xbox One και PlayStation 4. Υπάρχουν πολλες ομοιότητες με την πλατφόρμα Unity5 αλλά η μεγαλύτερη τους διαφορά βρίσκεται στην γλώσσα προγραμματισμού. Η μηχανή Unreal Engine 4 χρησιμοποιεί την γλώσσα προγραμματισμού C++ .[13]

Η Html5 έχει τα εξής πλεονεκτήματα και μειονεκτήματα. Τα πλεονεκτήματα είναι τα εξής :

1ον Δεν χρειάζεται πρόσθετο λογισμικό, τρέχει απευθείας στο πρόγραμμα περιήγησης και απαιτεί λιγότερους πόρους

2ον Συμβατότητα.

3ον Τοπική αποθήκευση και Υποστήριξη για λειτουργία πολλαπλών χρηστών

4ον Κόστος. Είναι δωρεάν

5ον Χρησιμοποιεί λογισμικό ανοιχτού κώδικα

Μειονεκτήματα :

1ον Μερική και ασυνεπής εφαρμογή στα προγράμματα περιήγησης

2ον Έλλειψη συντονισμού μεταξύ των προγραμμάτων περιήγησης

3ον Κακή υποστήριξη

4ον Περιορισμένοι Πόροι[11]

2.4 Σύγκριση σε πλατφόρμες ψηφιακής διάθεσης παιχνιδιών

Η ψηφιακή διάθεση παιχνιδιών γίνεται online με την χρήση των πλατφόρμων διανομής steam, gog , origin. Τα πλεονεκτήματα της διαδικασίας αυτής είναι η άμεση αγορά του κάθε τίτλου online, χωρίς να χρειαζόμαστε dvd, καθώς και η μόνιμη ύπαρξή του στον λογαριασμό μας. Με τον τρόπο αυτό δεν κινδυνεύει το παιχνίδι να χαθεί ή να καταστραφεί. Μπορούμε ανά πάσα στιγμή να το ξανακατεβάσουμε, αφού θα υπάρχει μόνιμα στον λογαριασμό μας. Το σημαντικό μειονέκτημα των πλατφόρμων διανομής είναι η πολιτική διαμοιρασμού, εφόσον δεν επιτρέπεται η μεταπώληση. Βέβαια αυτό διαφέρει από εταιρεία σε εταιρεία.

Το steam είναι η πιο γνωστή και διαδεδομένη πλατφόρμα διαδικτυακού παιχνιδιού. Διαθέτει την μεγαλύτερη βιβλιοθήκη για την αγορά παιχνιδιών υπολογιστή. Είναι ιδανικό και για χρήστες Linux.

Το origin είναι μια πλατφόρμα διανομής από την εταιρεία EA. Περιέχει στην βιβλιοθήκη του πολύ γνωστούς τίτλους παιχνιδιών. Διαθέτει ανά καιρούς δωρεάν παλιότερους τίτλους παιχνιδιών στους χρήστες. Διαφοροποιείται από το steam, στο γεγονός ότι οι αγορές γίνονται κατευθείαν από τον browser, χωρίς την ανάγκη να κατεβάσουμε καποιον client.

Το gog αποτελεί μια αξιόπιστη εναλλακτική του steam καθώς διαθέτει πολύ μεγάλη βιβλιοθήκη με πολυ γνωστούς τίτλους παιχνιδιών. Με αυτή την πλατφόρμα μπορούμε να αγοράσουμε κάποιον τίτλο online, είτε με χρήση client, είτε απευθείας από τον browser.[14]

Κεφάλαιο 3ο - Σενάριο

3.1 Από το παρελθόν, στο παρόν

Η ιστορία μας ξεκινά τον μεσαίωνα, όπου τα κάστρα, οι βασιλιάδες, οι μάγισσες, τα μαγικά φίλτρα και οι κάθε λογής μαγείες ήταν όλα γνώριμα για την εποχή εκείνη. Την εποχή όπου από στόμα σε στόμα ζούσαν οι θρύλοι, οι μύθοι, οι κρυμμένοι θησαυροί, τα μαγικά χαλιά, οι απέθαντοι εχθροί και τόσα άλλα. Έτσι λοιπόν και στο παιχνίδι μας, η ιστορία, κατά την μεσαιωνική περίοδο, ξεκινά :

Κάποτε υπήρχε ένα βασίλειο, με έναν πολύ γενναίο και αγαπητό στον κόσμο βασιλιά, ο οποίος είχε τα πάντα, πλούτη, δόξα, πιστούς υπηκόους, αλλά και μια πανέμορφη βασίλισσα που του χάρισε δύο υγιέστατα παιδιά, δύο πανέμορφους πρίγκιπες. Ο βασιλιάς ήταν πολύ αγαπητός στον κόσμο, διότι άκουγε με προσοχή τον καθέναν που είχε να εκφράσει το παράπονό του ή να αναζητήσει λύση στο πρόβλημά του. Έσκυβε πάνω στο πρόβλημα του κάθε υπηκόου και αναζητούσε λύση άμεσα, δείχνοντας αμέριστη προσοχή, όρεξη για δίκαιη επίλυση αλλά και γενναιοδωρία. Έκανε το πρόβλημα δικό του, χωρίς ποτέ να δείξει ότι υπάρχει χαώδης διαφορά ανάμεσα στους απλούς υπηκόους και σε εκείνον. Ακριβώς αυτό, σε συνδυασμό με την γενναιότητά του στις μάχες, τον έκαναν τόσο διάσημο και ξακουστό.

Όμως δυστυχώς για τον άτυχο βασιλιά μας και την οικογένειά του, υπήρχε μια κακιά μάγισσα που παρακολουθούσε από το μαγικό της καθρέπτη την ζωή του. Εκείνη προσπαθούσε να τον σαμποτάρει με κάθε τρόπο, φοβίζοντας τους υπηκόους αλλά και διαδίδοντας φήμες που είχαν σκοπό να αμαυρώσουν το καλό όνομα του βασιλιά. Όμως, την στιγμή που ο βασιλιάς απέκτησε τους δύο, δίδυμους γιούς του, η μάγισσα ζήλεψε τόσο πολύ, που αποφάσισε να εξαπολύσει μια πανίσχυρη κατάρα. Μια κατάρα που θα μετέτρεπε τους πάντες σε απέθαντους, εκτός του βασιλιά. Μια κατάρα που θα εξάλειφε ολόκληρο το βασίλειο, αφήνοντάς τον μόνο εν ζωή άνθρωπο. Και έτσι έγινε. Από τη μια στιγμή στην άλλη, ο βασιλιάς έχασε τα πάντα, την οικογένειά του, τους υπηκόους του, αλλά και όποιον άλλον ήξερε. Και σαν να μην έφτανε αυτό, μετά από λίγο, οι δικοί του άνθρωποι μετατράπηκαν σε απέθαντους εχθρούς. Τρέχοντας για να σωθεί, πήγε στο μοναδικό μέρος που δεν κινδύνευε. Στο σημείο που είχε κρυμμένα όλα τα πλούτη του. Τα κοίταξε για τελευταία φορά και στα μάτια του αυτή τη φορά ο χρυσός δεν γυάλιζε. Παρατήρησε ότι δεν ήταν αυτά που του δίναν χαρά, αυτοπεποίθηση, ευτυχία αλλά οι δικοί του άνθρωποι και η οικογένειά του. Τα πλούτη δεν είχαν πλέον κανένα νόημα για εκείνον. Απελπισμένος και βυθισμένος σε πελάγη δυστυχίας ο βασιλιάς, αποφάσισε ότι δεν αξίζει να ζει σε μια τέτοια πραγματικότητα, διότι τίποτα δεν είχε νόημα πλέον. Πριν όμως δώσει τέλος στη ζωή του έκανε μια ευχή. Ευχήθηκε αυτό το μέρος να χαθεί από το χάρτη και ο αμύθητος θησαυρός του, να αποκαλυφθεί μόνο σε κάποιον σαν εκείνον, σε καποιον ενάρετο, σε κάποιον που θα έχει τις δικές του αρετές και τα δικά του σπάνια προτερήματα. Και ετσι έγινε.

Για πάρα πολλά χρόνια, κανείς δεν μπόρεσε να ανακαλύψει το ακριβές σημείο του βασιλείου, παρά τις αμέτρητες προσπάθειες. Κυκλοφορούσε από στόμα σε στόμα ο θρύλος του μοναχικού βασιλιά με τον αμύθητο θησαυρό, αλλά κανείς δεν πίστευε πραγματικά ότι πρόκειται για αληθινή ιστορία. Ο κόσμος νόμιζε ότι πρόκειται για μύθο. Μέχρι που το έτος 2000, μια ομάδα εξερεύνησης κοιτίδων πετρελαίου, ανακάλυψε το νησί, εντελώς τυχαία. Κανείς τους όμως δεν επιχείρησε περαιτέρω εξερεύνηση του νησιού. Ο επιχειρηματίας που χρηματοδοτούσε την αποστολή αποφάσισε ότι έπρεπε να επιστρέψουν με τα κατάλληλα μηχανήματα εξερεύνησης, προκειμένου να μαρκάρουν το σημείο στο χάρτη, αλλά και να ανακαλύψουν αν η περιοχή είναι κατοικήσιμη από κάποιον άγνωστο πολιτισμό. Μιας και θα ανακαλύπταν κάτι τόσο πρωτόγνωρο, ένα άγνωστο σημείο στον χάρτη, μια αρχαία γη, κατοικήσιμη ή μή, αναμενόμενο ήταν ότι, αργά ή γρήγορα, όλο και περισσότερος κόσμος θα ήθελε να επισκεφθεί το σημείο. Οπότε ο επιχειρηματίας αποφάσισε να μην χάσει χρόνο και να ρισκάρει χτίζοντας τέσσερις μικρές επιχειρήσεις για να αποκομίσει χρήματα από τους μελλοντικούς επισκέπτες. Όμως όσο και να έψαχναν, όσο και να προσπαθούσαν να θυμηθούν που ακριβώς ήταν το νησί, φαινόταν αδύνατον. Τους πήρε σχεδόν 7 χρόνια για να το ανακαλύψουν ξανά. Οπότε, δεν έχασαν χρόνο και έκτισαν τις τέσσερις επιχειρήσεις, σε ελάχιστο χρόνο μόλις βρήκαν το νησί. Παράλληλα επεδίωξαν να χαρτογραφήσουν το σημείο, αλλά αυτό φαινόταν μάταιο, διότι τα μηχανήματα δεν λειτουργούσαν ορθολογικά. Μέχρι και οι πυξίδες στριφογυρίζανε, κάνοντας τους ειδικούς να απορούν τι είχε συμβεί..

Ο χρόνος περνούσε και οι οικογένεια του επιχειρηματία ανησυχούσε. Είχαν όλοι χαθεί. Κανείς δεν είχε νέα, ούτε του επιχειρηματία, ούτε του συνεργείου. Και αυτό διότι αποφάσισαν να εξερευνήσουν περαιτέρω το νησί αλλά δεν περίμεναν τους απέθαντους εχθρούς. Οι εχθροί είχαν επιτεθεί. Και για να σιγουρέψουν ότι κανείς δεν θα επιστρέψει στο νησί, αφάνισαν όλους τους τεχνικούς, το προσωπικό αλλά και τον επιχειρηματία. Στην συνέχεια έκρυψαν και τα 4 κλειδιά των επιχειρήσεων αλλά και των πυλών, σε σημεία που φρουρούσαν.

Τα χρόνια περνούσαν και η οικογένεια του επιχειρηματία έχασε κάθε ελπίδα για την επιστροφή του. Πλέον ήταν σίγουροι ότι κάτι μοιραίο συνέβη. Το άτυχο αυτό γεγονός σε συνδυασμό με την πίεση των δικηγόρων όσων αφορά τις 4 επιχειρήσεις αγνώστου ύπαρξης και τοποθεσίας, οδήγησαν την οικογένεια να αποποιηθεί οποιοδήποτε δικαίωμα σε αυτές. Η οικογένεια, αποφάσισε να τις δωρίσει σε όποιον τις ανακαλύψει. Σε όποιον μπορέσει να εντοπίσει το σημείο και να γυρίσει πίσω σώος, με πληροφορίες για τον πατέρα τους... Εξάλλου τους θύμιζαν τόσο έντονα το άτυχο συμβάν.

Σε αυτό το σημείο θα αναφερθούμε στον Ethan, τον χαρακτήρα μας στο παιχνίδι. Ο Ethan είναι ένας 25χρονος επιχειρηματίας, ο οποίος διακρίνεται για την τόλμη του, το αλάνθαστο ένστικτό του αλλά και τις παράτολμες αποφάσεις του. Αποφάσεις σε επενδυτικό επίπεδο, οι οποίες είναι ριψοκίνδυνες μεν, αλλά πολύ προσοδοφόρες δε, αν αποδειχθούν σωστές με την πάροδο του χρόνου. Ο χαρακτήρας μας είχε ξεκινήσει να παρακολουθεί, από μικρή ηλικία, επενδυτικές κινήσεις μεγάλων εταιρειών, τρόπους επίτευξης κερδών, διαχείριση καινοτομιών, στρατηγικές πωλήσεων, κινήσεις μάρκετινγκ. Παρακολουθούσε συνέδρια επίτευξης στόχων, επικοινωνίας, μάρκετινγκ, πωλήσεων, στρατηγικών. Για έναν άνθρωπο τόσο παθιασμένο με τις επιχειρήσεις, η επιτυχία ήταν

αναμενόμενη, αργά ή γρήγορα. Και έτσι έγινε. Αφού βρήκε τους κατάλληλους συνεργάτες και το κατάλληλο κεφάλαιο, αποφάσισε να ρισκάρει ανοίγοντας την δικιά του επιχείρηση, βασισμένη σε τεχνολογικές καινοτομίες, στοχεύοντας στην δημιουργία και μεταπώληση τεχνολογικών πατέντων. Η επιχείρηση αυτή αποτέλεσε το πιο γρήγορα εξελίξιμο start up στην ιστορία και γιγαντώθηκε σε χρόνο ρεκόρ. Σε μερικά χρόνια ήταν παγκοσμίου βεληνεκούς και αξίας δισεκατομμυρίων ευρώ. Το πρόσωπο του επιχειρηματία αποτελούσε πλέον σύμβολο επιτυχίας. Είχε γίνει σε χρόνο ρεκόρ παράδειγμα προς μίμηση. Όμως παρά τον απότομο πλουτισμό του, δεν ξέχασε από που ξεκίνησε. Δεν έπαψε να βοηθάει τους μη έχοντες όχι μόνο χρηματικά αλλά και ψυχολογικά. Στεκόταν πάνω από το πρόβλημα του καθενός και έψαχνε λύσεις. Ακριβώς όπως και ο βασιλιάς. Έκτισε νοσοκομεία, σχολεία και τόσα άλλα. Φυσικά κάτι τέτοιο αποτελούσε πρόβλημα για το μετοχικό συμβούλιο, το οποίο πίστευε ότι ο ανθρωπισμός και η χρηματική βοήθεια προς τον συνάνθρωπο αποτελούσε λανθασμένη στρατηγική. Ο φθόνος των μετόχων προς τον πρόσωπο του Ethan σε συνδυασμό με την απόλυτη προσήλωσή τους στην κερδοφορία θα οδηγούσε αργά ή γρήγορα σε αδιέξοδο για τον επιχειρηματία. Έφτασε λοιπόν η στιγμή που οι μέτοχοι συνομώτησαν εναντίον του και ψήφισαν υπέρ της αλλαγής ηγεσίας. Ο Ethan βρέθηκε εξ απροόπτου. Από την μια στιγμή στην άλλη, έχασε τα πάντα. Και το χειρότερο ήταν ότι τον είχαν προδώσει οι δικοί του άνθρωποι.

Βλέποντας την περιουσία του να μειώνεται δραματικά, ο Ethan, ήταν σε κατάσταση κατάθλιψης. Σιγά σιγά αποκόπηκε από φίλους και συγγενείς και βυθίστηκε στις σκέψεις του. Προσπαθούσε να βρει λύση προκειμένου να επανέλθει στην κατάσταση που ήταν πριν χάσει την εταιρεία του, αλλά εκείνη την στιγμή, κάτι τέτοιο φάνταζε αδύνατο στα μάτια του. Το εντυπωσιακό όμως είναι, ότι ποτέ δεν σταμάτησε να βοηθάει τους ανθρώπους που είχαν ανάγκη, γνωρίζοντας πολύ καλά ότι αργά ή γρήγορα θα βρισκόταν και ο ίδιος στην θέση τους. Τότε ήταν που ένα όραμα τον ταρακούνησε. Μην γνωρίζοντας ο Ethan αν ονειρεύεται ή όχι, είδε δίπλα του μια βασιλική μορφή, περιτριγυρισμένη από χρυσάφι, να του δείχνει τον δρόμο. Να του δείχνει επίμονα ένα συγκεκριμένο σημείο στο χάρτη. Ο επιχειρηματίας αποφάσισε να το ψάξει και τότε ήλθε σε επαφή για πρώτη φορά με τον θρύλο του μοναχικού βασιλία. Το ένστικτό του έλεγε να ρισκάρει. Το όραμα που είδε τον γέμισε με αυτοπεποίθηση και ελπίδα ότι ο θρύλος ζει και μπορεί να τα καταφέρει. Εξάλλου τι είχε να χάσει πλέον;

Έτσι λοιπόν αποφάσισε να πουλήσει και τα τελευταία υπάρχοντά του για να να κάνει αυτό το όραμα πραγματικότητα. Αγόρασε ένα πλοίο, μηχανήματα για τον εντοπισμό αλλά και αρκετές προμήθειες για το ταξίδι. Και η περιπέτεια ξεκινά. Πλησιάζοντας ο Ethan στο σημείο του χάρτη που είχε δει στο όραμα, το μόνο που μπορούσε να δει ήταν ομίχλη. Οι πυξίδες στρυφογυρνούσαν και τα μηχανήματα απενεργοποιήθηκαν. Μαύρα σύννεφα κάλυψαν τον ουρανό και πελώρια κύματα εμφανίστηκαν ξαφνικά. Καθώς οι αναταράξεις ήταν δυνατές, ο Ethan χτύπησε το κεφάλι του και λιποθύμησε. Το πλοίο ακυβέρνητο πάλευε με τα κύματα που του προκαλούσαν ρωγμές. Αργά ή γρήγορα θα άρχιζε να βουλιάζει. Και έτσι έγινε. Σαν από θαύμα όμως ο ήρωάς μας ξεβράστηκε στο νησί. Ήταν άραγε το πνεύμα του μοναχικού βασιλιά που τον προστάτευε; Μόλις συνήλθε, δεν πίστευε στα μάτια του. Το όραμά του βγήκε πέρα για πέρα αληθινό. Το νησί υπήρχε!

3.2 Σκοπός του παίχτη

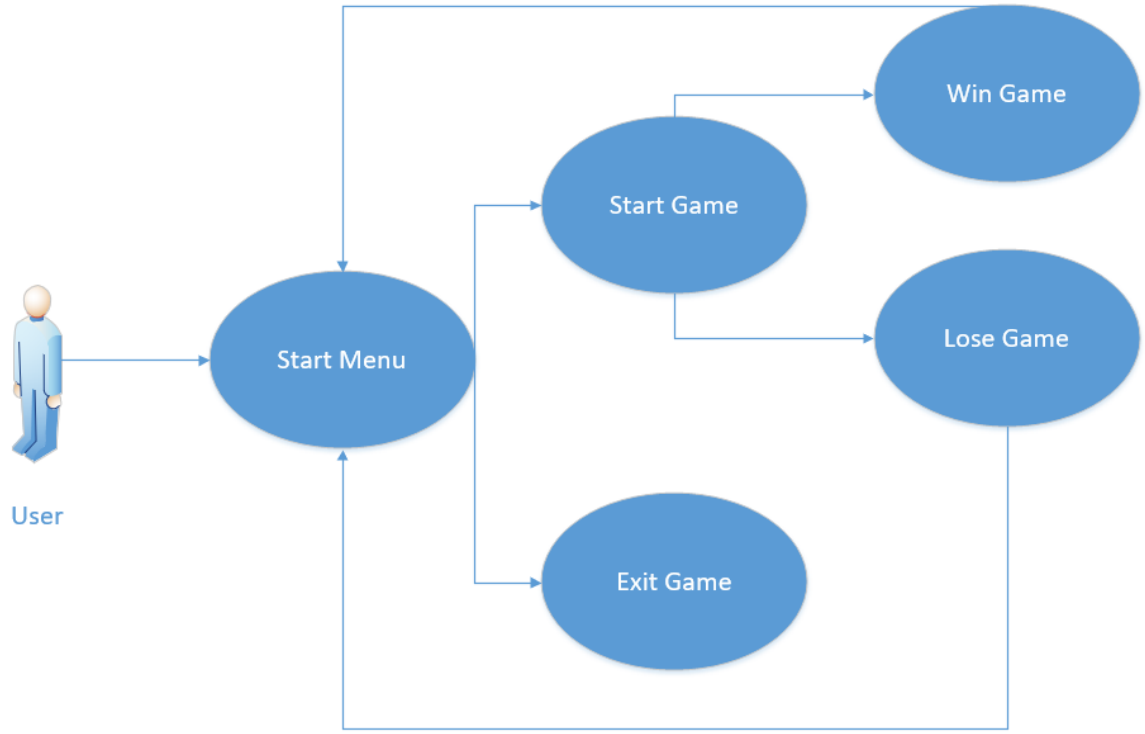
Ο σκοπός του ήρωά μας είναι να βρεί τα κρυμμένα κλειδιά των επιχειρήσεων και έπειτα να τις ανοίξει μία προς μία. Έτσι τις θέτει πάλι σε λειτουργία και γίνεται ο ιδιοκτήτης αυτών. Μόλις καταφέρει και τις ανοίξει όλες και εφόσον είναι ο κατάλληλος, σύμφωνα με την τελευταία ευχή του βασιλιά, ο αμύθητος θησαυρός θα αποκαλυφθεί μόνο σε αυτόν και η κατάρα που μάστιζε το βασίλειο επί τόσα χρόνια θα σπάσει. Όμως για να το κάνει ο ήρωάς μας αυτό, πρέπει πρώτα να βρει τα κλειδιά που ανοίγουν την κάθε πύλη στον λαβύρινθο. Και είναι κρυμμένα. Έπειτα πρέπει να βρει την έξοδο από τον λαβύρινθο. Έχει επίσης να αντιμετωπίσει τους απέθαντους εχθρούς οι οποίοι θα κάνουν τα πάντα για να τον αφανίσουν καθώς και τις θανατηφόρες παγίδες που ξεπροβάλλουν μπροστά του.

Κεφάλαιο 4ο - Ανάλυση,Σχεδιασμός,Υλοποίηση με Uml Diagrams

4.1 Use Case Diagrams

Ένα διάγραμμα περιπτώσεων χρήσης(Use case diagram), είναι μια αναπαράσταση της αλληλεπίδρασης ενός χρήστη με το σύστημα, η οποία δείχνει τη σχέση μεταξύ του χρήστη και των διαφόρων περιπτώσεων χρήσης στις οποίες εμπλέκεται. Ένα διάγραμμα περιπτώσεων χρήσης μπορεί να προσδιορίσει τους διαφορετικούς τύπους χρηστών ενός συστήματος και τις διαφορετικές περιπτώσεις χρήσης και συχνά συνοδεύεται και από άλλους τύπους διαγραμμάτων.

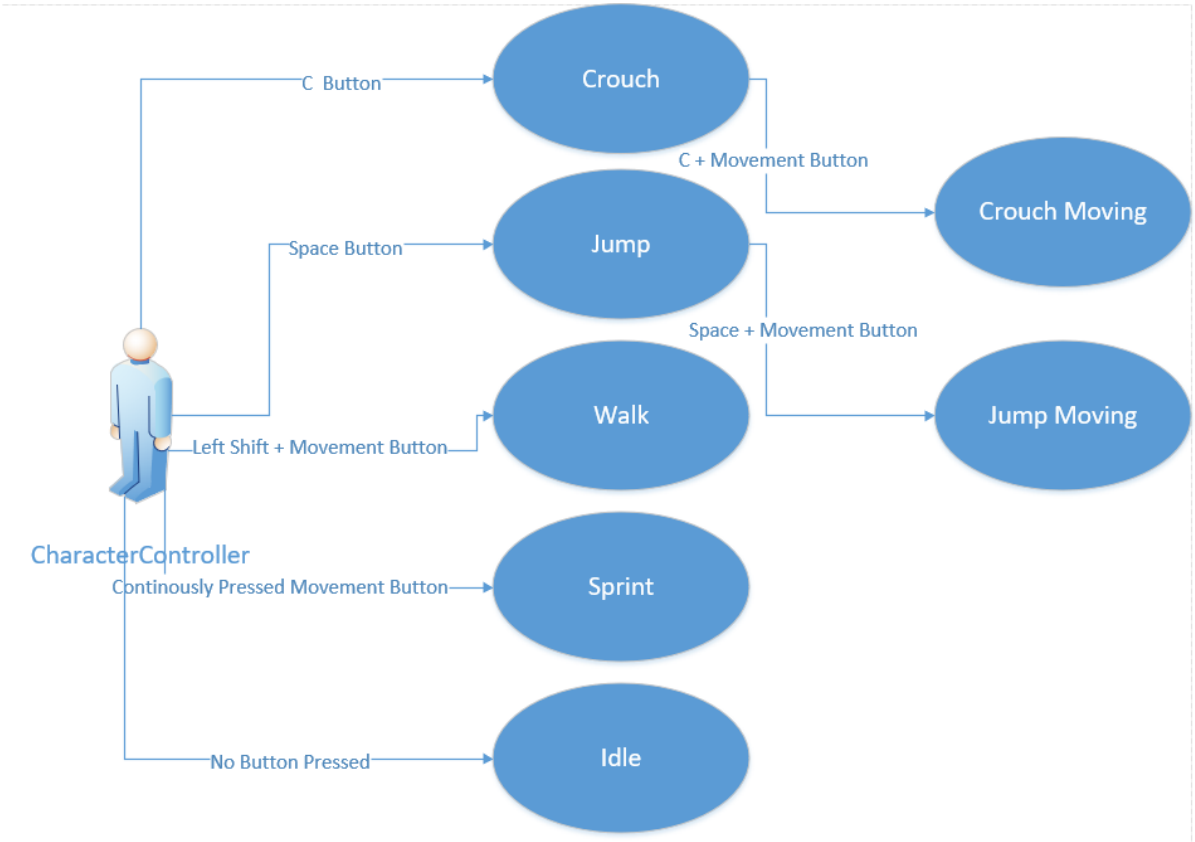
Αρχικά θα αναλύσουμε την γενική επαφή του χρήστη με το παιχνίδι μας, χρησιμοποιώντας το UserMenu Use Case Diagram.



Όπως βλέπουμε στο παραπάνω διάγραμμα, ο χρήστης αρχικά οδηγείται στο αρχικό μενού του παιχνιδιού. Έπειτα έχει την επιλογή, είτε να ξεκινήσει το παιχνίδι, πατώντας το κουμπί ¨START GAME¨, είτε να εξέλθει από αυτό, πατώντας στο κουμπί ¨EXIT GAME¨

και στην συνέχεια ¨YES¨ στο υπομενού που του εμφανίζεται. Εφόσον η επιλογή του είναι η εκκίνηση του παιχνιδιού, οι επιλογές του είναι δύο: είτε να κερδίσει, τερματίζοντας το παιχνίδι, είτε να χάσει. Όποιο και από τα δύο αυτά σενάρια συμβεί, ο χρήστης επανέρχεται στο αρχικό μενού.

Εφόσον η επιλογή του χρήστη είναι να ξεκινήσει το παιχνίδι, ο χρήστης αποκτά τον πλήρη έλεγχο του χαρακτήρα μας, ¨Ethan¨. Ο χαρακτήρας μας στο Unity, ελέγχεται από τον ¨ThirdPersonController¨. Ας δούμε το διάγραμμα CharacterController'sMovements Use Case Diagram.

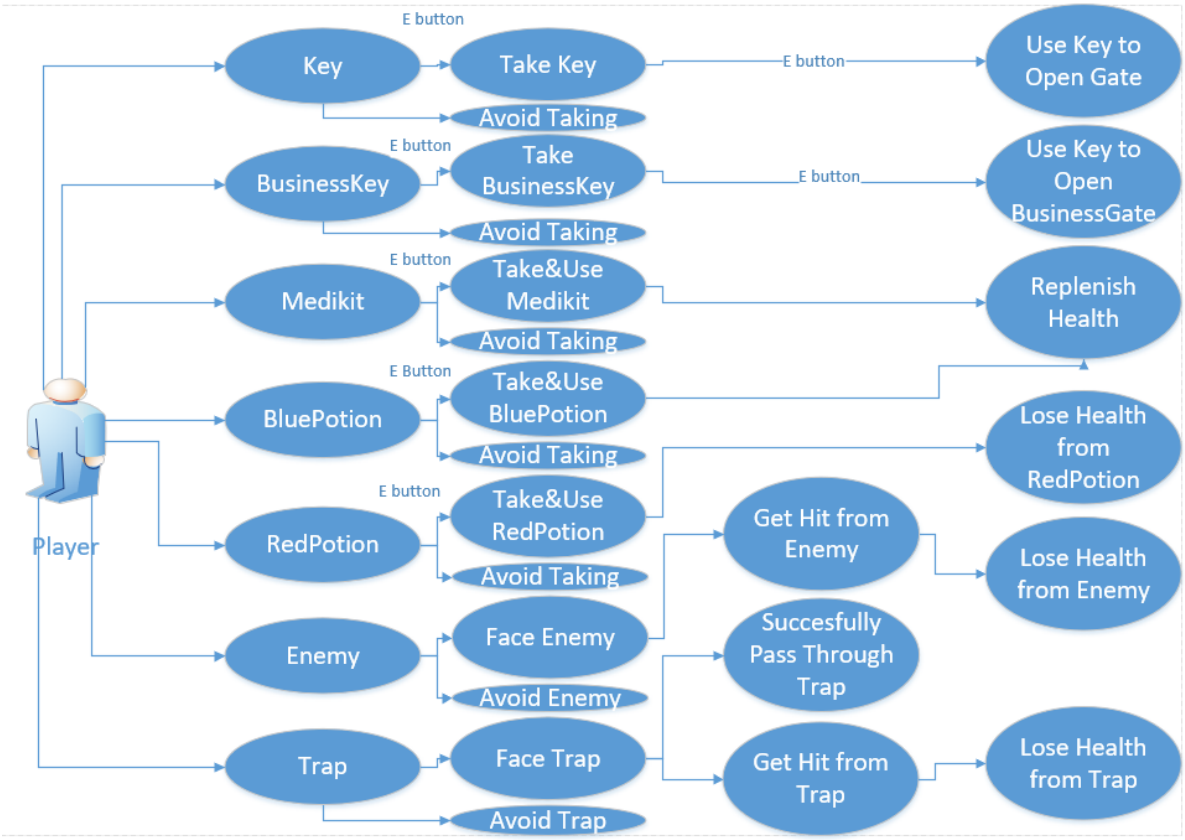


Όπως βλέπουμε στο διάγραμμα παραπάνω, ο παίχτης μπορεί να κάνει τις εξής κινήσεις:

1. Crouch, πατώντας στο πληκτρολόγιο το κουμπί ¨C¨
2. Crouch Moving, πατώντας στο πληκτρολόγιο το κουμπί ¨C¨ μαζί με οποιοδήποτε κουμπί κίνησης(¨W¨,¨A¨,¨S¨,¨D¨)
3. Jump, πατώντας στο πληκτρολόγιο το κουμπί ¨Space¨
4. Jump Moving, πατώντας στο πληκτρολόγιο το κουμπί ¨Space¨ μαζί με οποιοδήποτε κουμπί κίνησης

5. Walk, πατώντας στο πληκτρολόγιο το κουμπί ¨Left Shift¨ μαζί με οποιοδήποτε κουμπί κίνησης
6. Sprint, πατώντας στο πληκτρολόγιο οποιοδήποτε κουμπί κίνησης παρατεταμένα
7. Καθόλου κίνηση, μη πατώντας στο πληκτρολόγιο κανένα κουμπί κίνησης

Ας δούμε τώρα τις βασικές λειτουργίες του παίχτη μας χρησιμοποιώντας το AllBasicPlayer'sFunctions Use Case Diagram.



Στο διάγραμμα παραπάνω βλέπουμε τις εναλλακτικές επιλογές του παίχτη μας, όσον αφορά τις βασικές λειτουργίες του στο παιχνίδι. Βλέπουμε λοιπόν ότι μπορεί να πάρει στην κατοχή του ένα κλειδί, πατώντας στο πληκτρολόγιο το κουμπί ¨Ε¨, ή μπορεί να αγνοήσει το συγκεκριμένο κλειδί. Έχοντας όμως στην κατοχή του το συγκεκριμένο κλειδί, μπορεί να ανοίξει μια συγκεκριμένη πύλη, πατώντας πάλι στο πληκτρολόγιο το κουμπί ¨Ε¨. Το ίδιο ακριβώς συμβαίνει και με τα κλειδιά των επιχειρήσεων. Έτσι και με το Medikit, το BluePotion και το RedPotion, μπορεί πατώντας στο πληκτρολόγιο το κουμπί ¨Ε¨ να τα κατέχει και να τα χρησιμοποιήσει ή μπορεί να τα αγνοήσει. Βλέποντας ο παίχτης του εχθρούς, μπορεί είτε να τρέξει και να τους αποφύγει, είτε να τους πλησιάσει και να χάσει επίπεδο ζωής. Βλέποντας ο παίχτης τις παγίδες, μπορεί είτε να

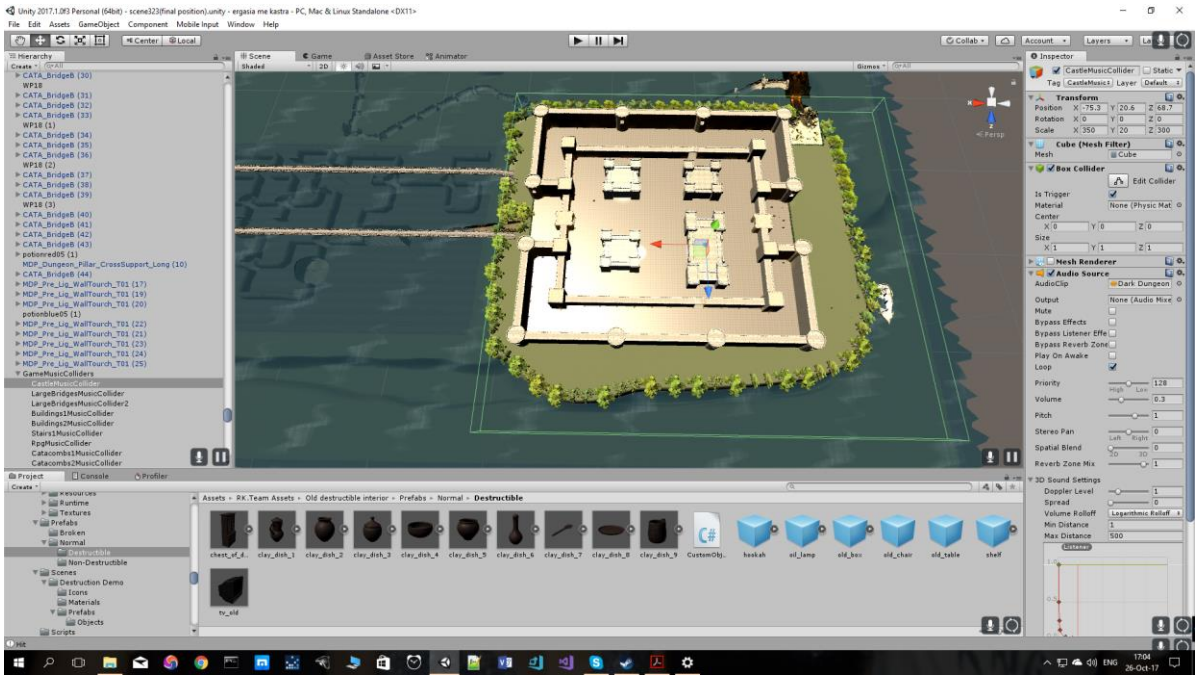
τις αποφύγει και να επιλέξει μια άλλη διαδρομή, είτε να προσπαθήσει να τις αντιμετωπίσει. Αντιμετωπίζοντάς τες λοιπόν, είτε θα τις περάσει επιτυχώς, είτε θα δεχτεί χτύπημα και θα χάσει επίπεδο ζωής.

4.2 Class Diagrams

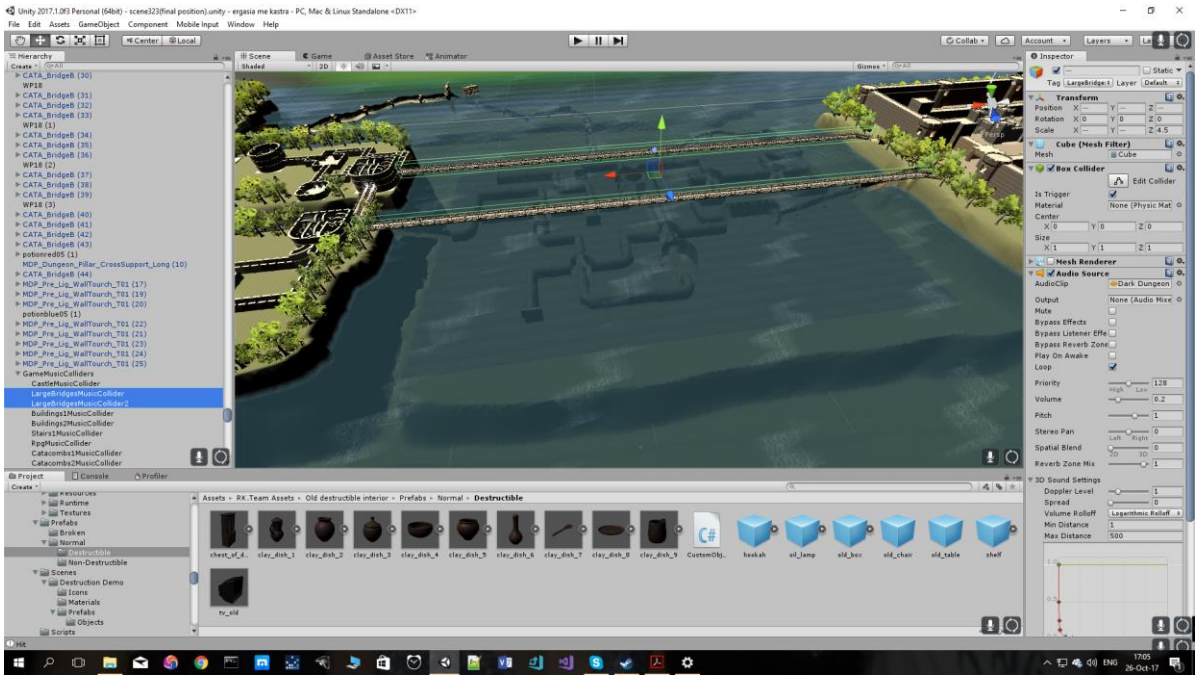
Στην τεχνολογία λογισμικού, ένα διάγραμμα κλάσης(Class Diagram), είναι ένας τύπος διαγράμματος στατικών δομών, το οποίο περιγράφει τη δομή ενός συστήματος, δείχνοντας τις κλάσεις του συστήματος, τα χαρακτηριστικά του, τις λειτουργίες (ή τις μεθόδους) και τις σχέσεις μεταξύ αντικειμένων.

Στα παρακάτω διαγράμματα κλάσης κάνουμε κατηγοριοποίηση ανά περιοχή παιχνιδιού. Την κάθε περιοχή, την ορίσαμε και την ονομάσαμε σύμφωνα με τους colliders μουσικής που χρησιμοποιήσαμε. Οπότε για να είναι κατανοητή η κατηγοριοποίηση αυτή από τον αναγνώστη, κρίνεται απαραίτητο να δείξουμε πρώτα με εικόνες τις συγκεκριμένες περιοχές:

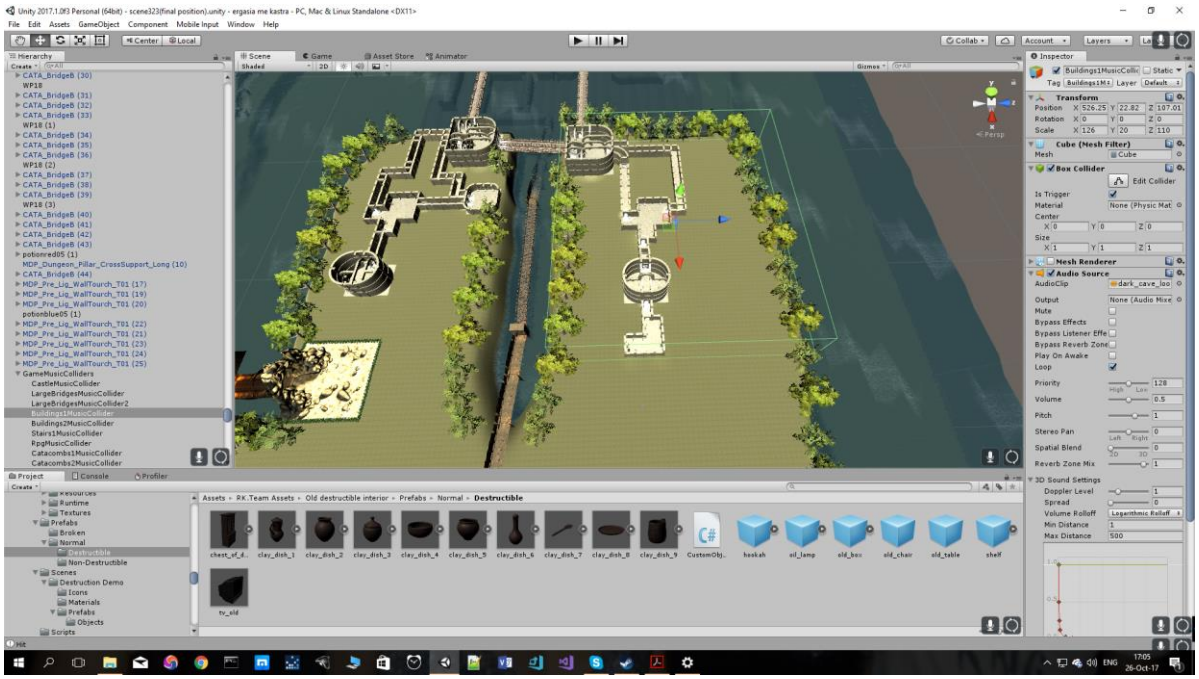
CastleMusicCollider:



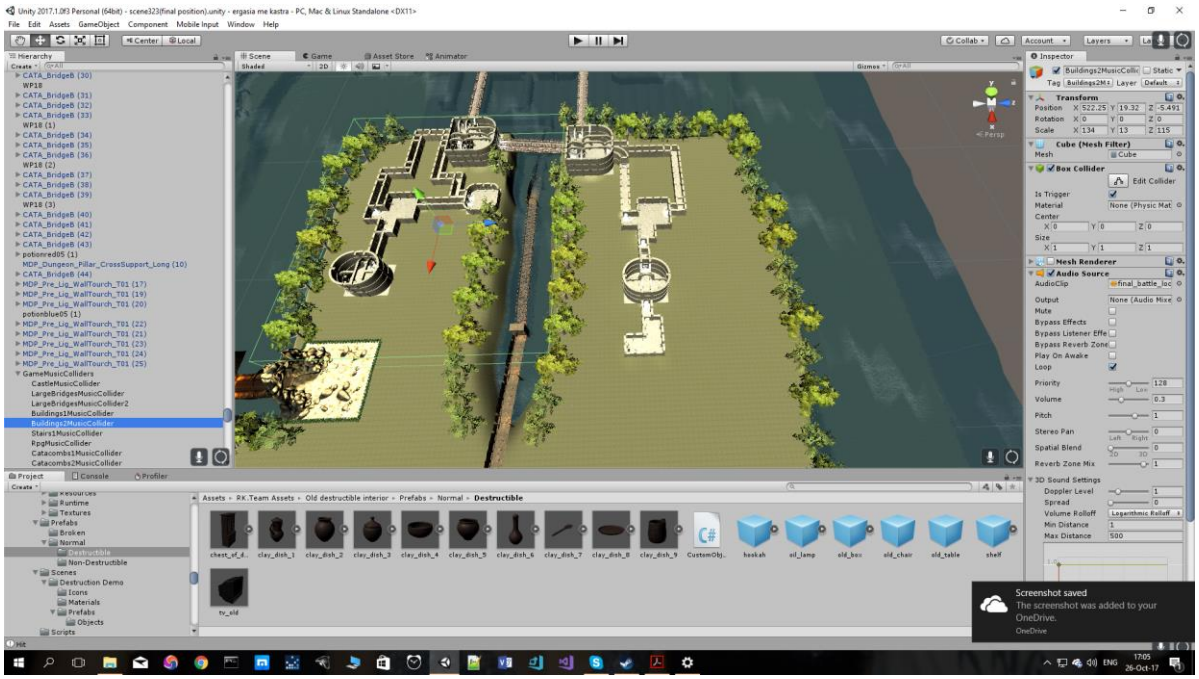
LargeBridgesMusicCollider:



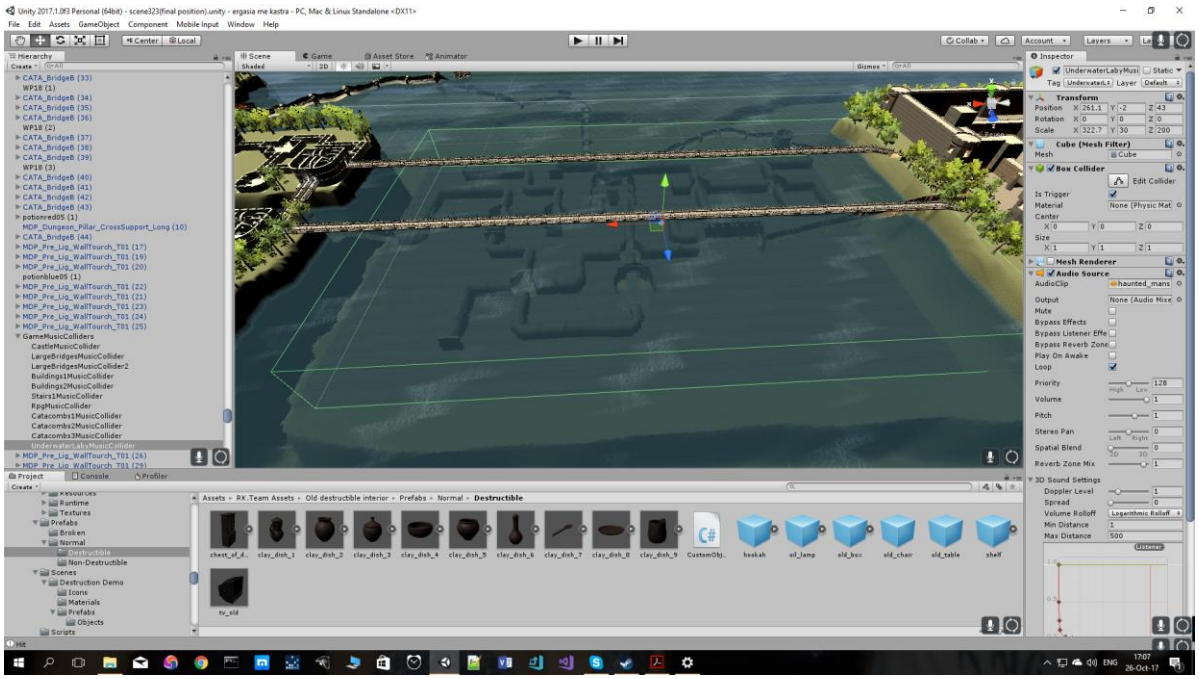
Buildings1MusicCollider:



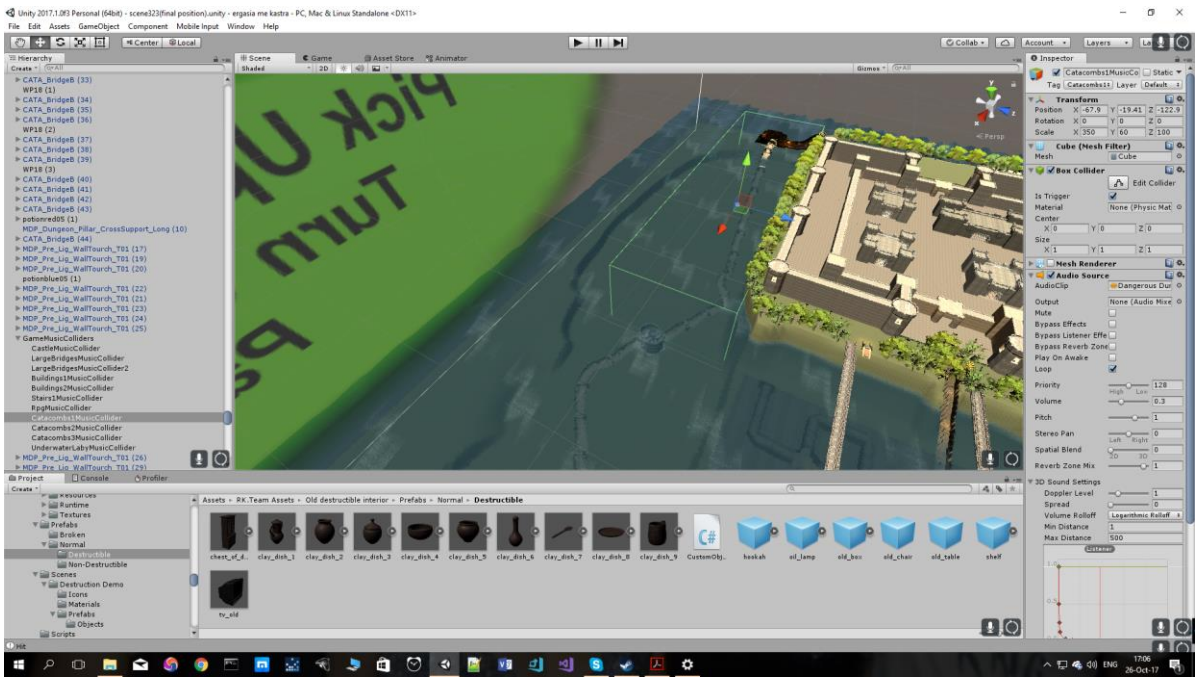
Buildings2MusicCollider:



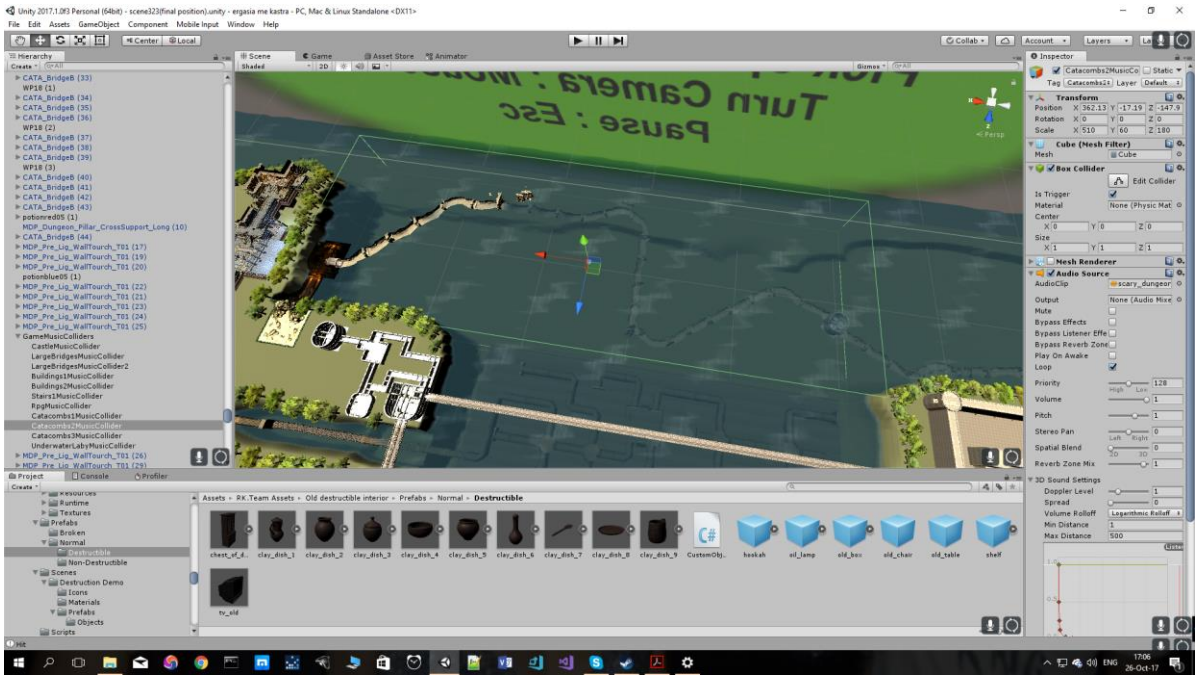
UnderwaterLabyMusicCollider:



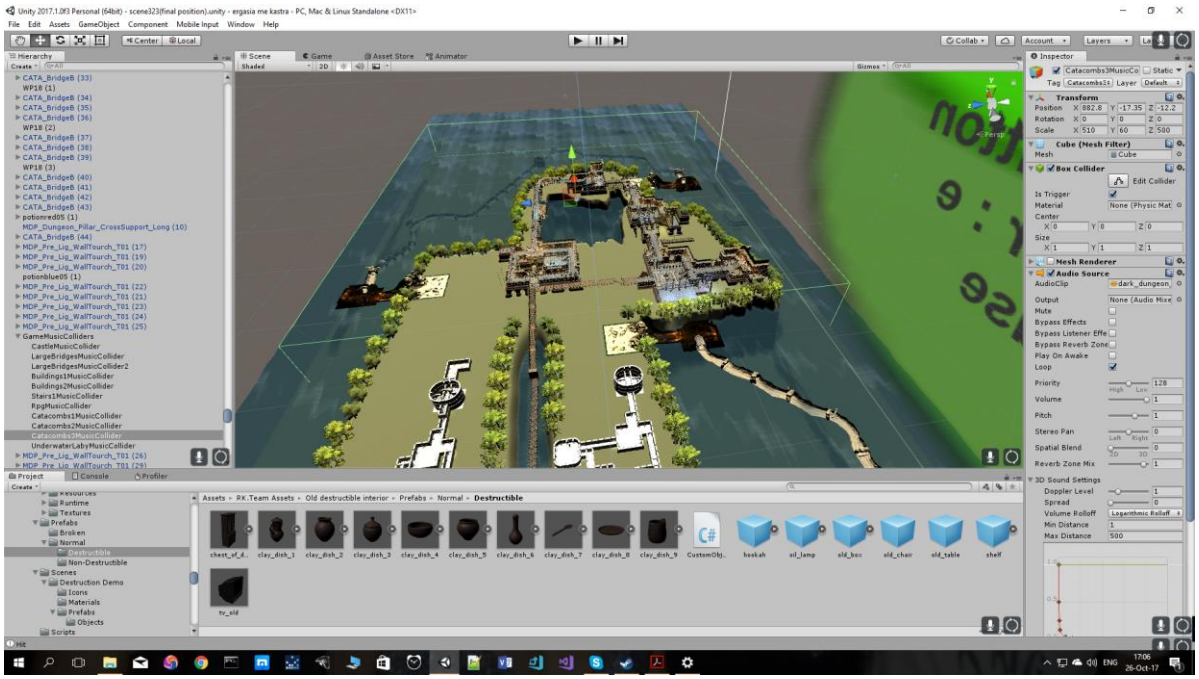
Catacombs1MusicCollider:



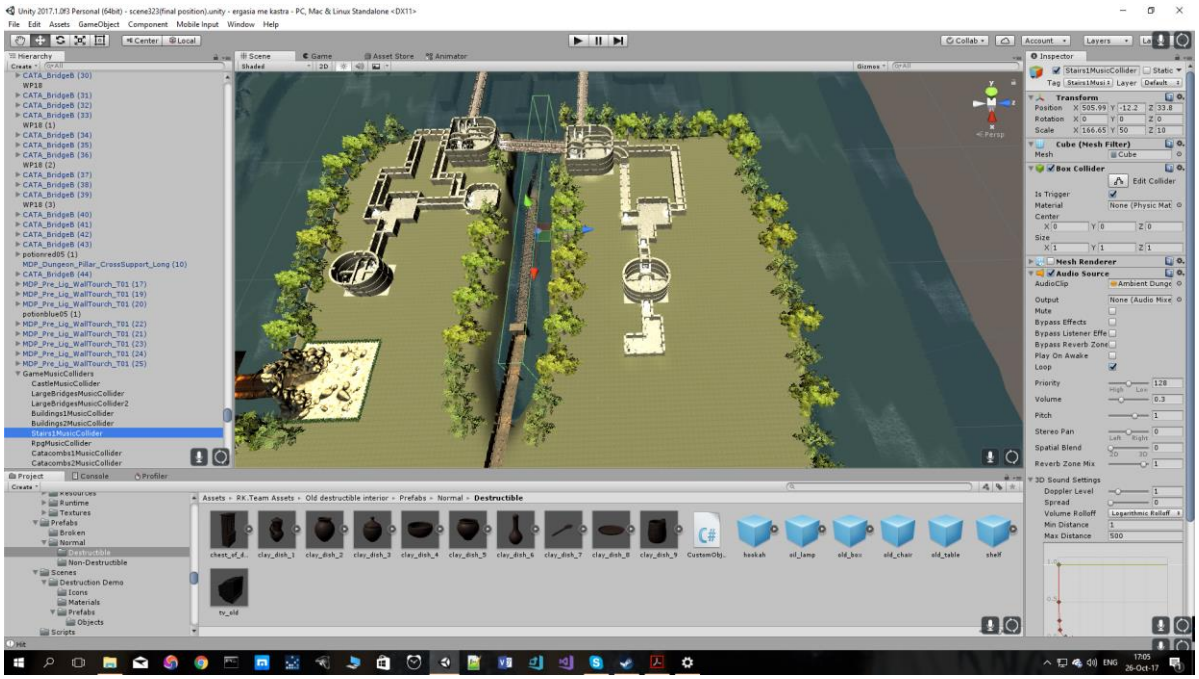
Catacombs2MusicCollider:



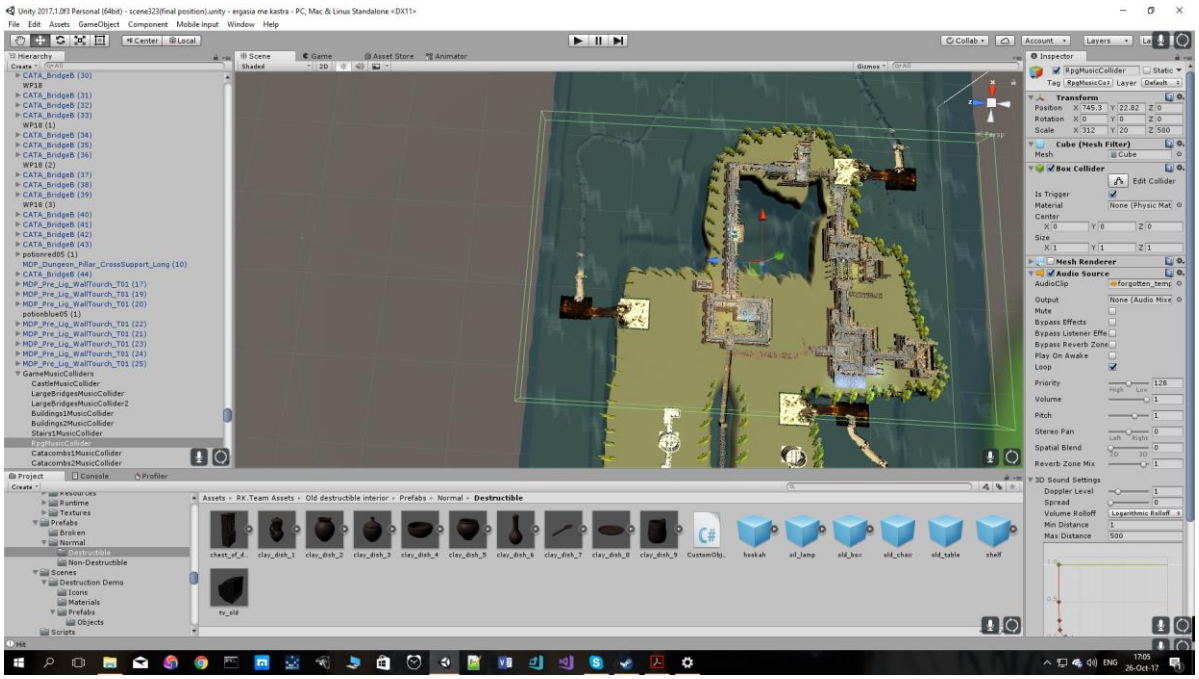
Catacombs3MusicCollider:



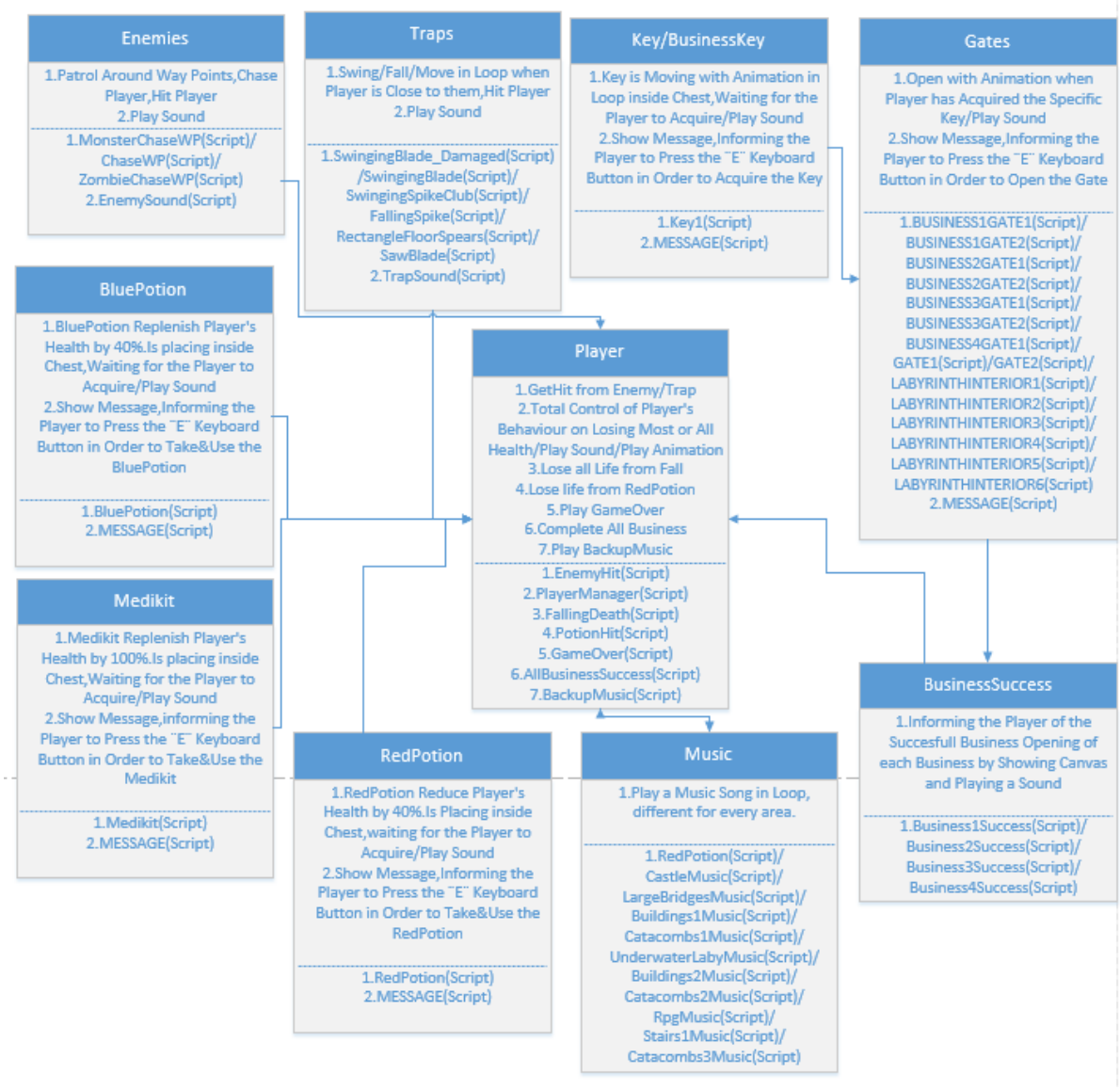
Stairs1MusicCollider:



RpgMusicCollider:

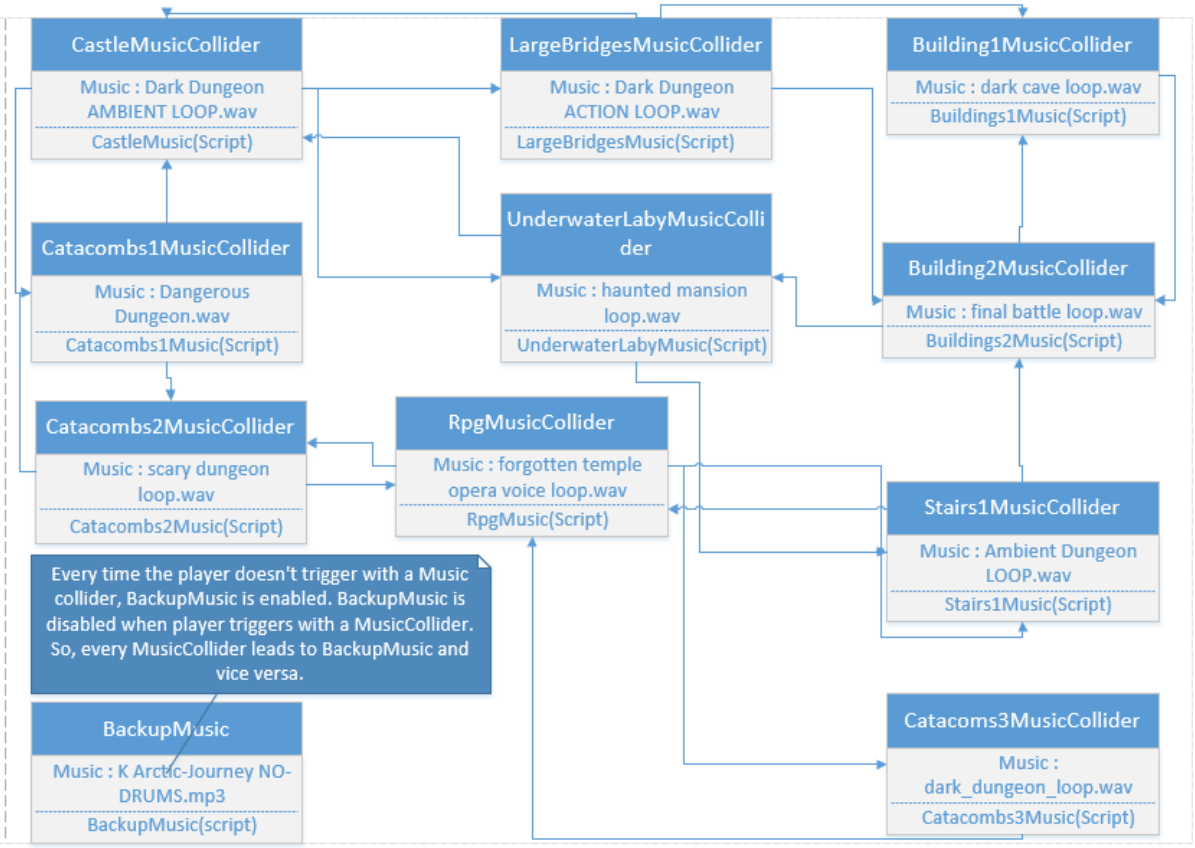


Αφού ολοκληρώσαμε την απεικόνιση, μπορούμε πλέον να δούμε τις βασικές κλάσεις του παιχνιδιού μας, τις λειτουργίες και τα χαρακτηριστικά τους, σε συνάρτηση με τον κώδικα που χρησιμοποιήσαμε για κάθε κλάση. Αυτό θα το επιτύχουμε αναλύοντας το διάγραμμα BasicFunction&Code Class Diagram.



Παρατηρούμε ότι το παραπάνω διάγραμμα μας δείχνει με μπλε χρώμα το όνομα της κάθε βασικής κλάσης στο παιχνίδι, έπειτα περιγράφει αριθμημένα τα χαρακτηριστικά και τις λειτουργίες της κλάσης αυτής και τέλος, ακολουθώντας την αρίθμηση, μας ενημερώνει για το κάθε script που χρησιμοποιήθηκε. Προσπαθώντας να γενικεύσουμε, φτιάξαμε μία κλάση για όλους τους εχθρούς, μία για όλες τις παγίδες, μία για όλα τα κλειδιά και μία για όλες τις πύλες. Αυτό μας έδωσε την δυνατότητα να παρατηρήσουμε την πληθώρα των παρόμοιων scripts. Έπειτα δείξαμε με connectors τις σχέσεις μεταξύ των κλάσεων, όσων αφορά τα scripts.

Σε αυτό το σημείο θα δούμε το BasicMusic&Code Class Diagram.



Στο διάγραμμα αυτό παρατηρούμε ότι έχουμε ταξινομήσει όλες τις περιοχές του παιχνιδιού σύμφωνα με τους colliders μουσικής που τοποθετήσαμε. Άρα στο όνομα της κάθε κλάσης έχουμε το όνομα του κάθε collider, στον οποίο έχουμε κάνει tag στο unity. Έπειτα δείχνουμε το μουσικό κομμάτι που ακούγεται, όταν ο παίχτης εισέρχεται στην κάθε περιοχή και τέλος, το script το οποίο είναι υπεύθυνο για την μουσική ανά περιοχή. Με connectors απεικονίζουμε όλες τις πιθανές μεταβάσεις που μπορεί να κάνει ο παίχτης μας από τον έναν collider στον άλλον. Λόγω αδυναμίας εμφάνισης των συνδέσεων μεταξύ της κλάσης BackupMusic με όλες τις υπόλοιπες, δημιουργήσαμε ένα σχόλιο στο οποίο αναφερόμαστε στις συνδέσεις αυτές.

Τέλος, θα θέλαμε να παρουσιάσουμε και να αναλύσουμε την αρχιτεκτονική των πακέτων που χρησιμοποιήθηκαν ανά περιοχή του παιχνιδιού. Έτσι έχουμε το Packages-Parts Per Area Class Diagram.

Το παραπάνω διάγραμμα χωρίστηκε σε δύο τμήματα προκειμένου να το απεικονίσουμε σωστά και να γίνει κατανοητό από τον αναγνώστη. Στην προσπάθειά μας αυτή, είχαμε στόχο να απεικονίσουμε το πλήθος των πακέτων αλλά και των parts από κάθε πακέτο, που χρησιμοποιήθηκαν ανά περιοχή του παιχνιδιού. Έτσι κατηγοριοποιούμε όλο το παιχνίδι ανά περιοχές και έπειτα εμφανίζουμε αριθμημένα την αντιστοιχία μεταξύ πακέτων-parts ανά πακέτο- που χρησιμοποιήσαμε ανά περιοχή. Το κάθε όνομα του part αναφέρεται μόνο μια φορά ανά περιοχή αλλά έχει χρησιμοποιηθεί παραπάνω από μια

φορές. Προκειμένου να παρουσιάσουμε ορθά την αρχιτεκτονική των πακέτων, συμπεριλάβαμε επίσης όλα τα πακέτα μουσικής και ήχων που χρησιμοποιήσαμε. Η κλάση General Location αφορά όλα τα πακέτα και parts που δεν έχουν ταξινομηθεί αυστηρά σε μία μόνο περιοχή, π.χ. οι εχθροί, τα σεντούκια, τα κλειδιά, αλλά γενικότερα σε όλο το παιχνίδι. Αξίζει να σημειωθεί ότι σε αυτό το διάγραμμα, χρησιμοποιήσαμε την κλάση Stairs για να περιγράψουμε όλες τις σκάλες του παιχνιδιού και όχι μόνο την περιοχή Stairs1 που δείξαμε στην εικόνα παραπάνω.

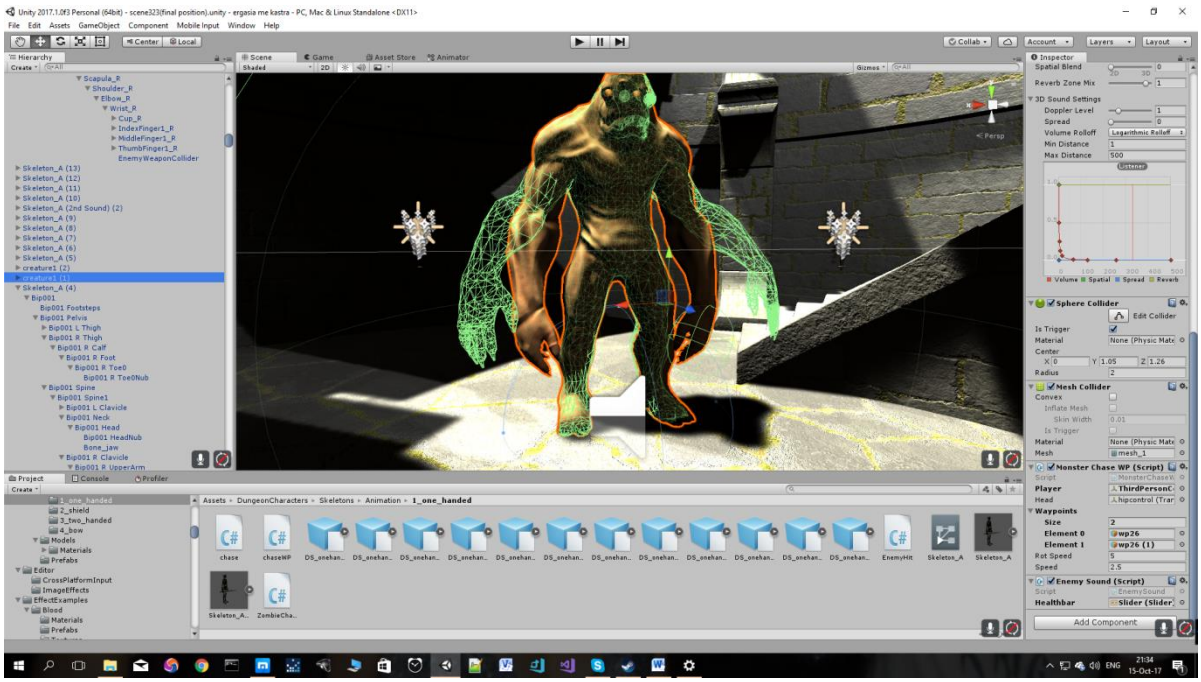
Κεφάλαιο 5° - Λειτουργίες

5.1 Δημιουργία και λειτουργίες απέθαντων εχθρών

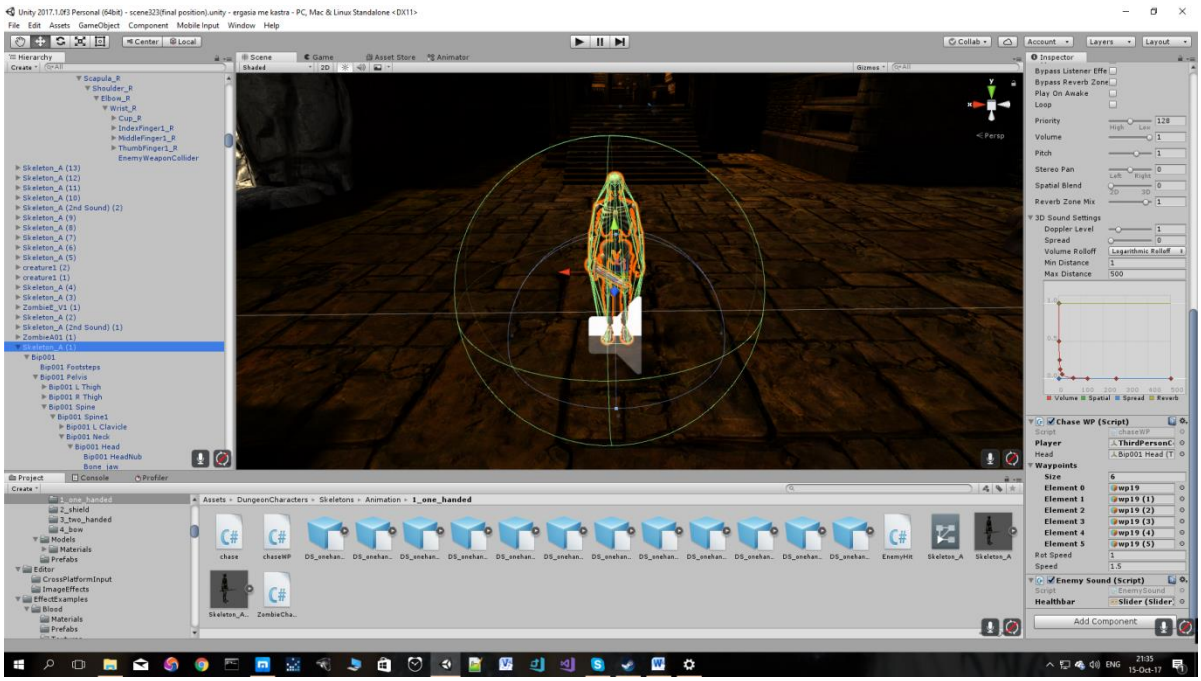
Στο παιχνίδι μας χρησιμοποιήσαμε 3 είδη διαφορετικών εχθρών, δημιουργήσαμε τον κώδικα τεχνητής νοημοσύνης που ορίζει τις κινήσεις τους όταν βρίσκεται κοντά ο παίχτης αλλά και όταν απουσιάζει. Προσαρμόσαμε τα animation που ταιριάζουν σε κάθε εχθρό χρησιμοποιώντας τον animator. Ορίσαμε τα waypoints για τον κάθε εχθρό ξεχωριστά καθώς επίσης δημιουργήσαμε collider που τον ονομάσαμε : “EnemyWeapon Collider”, υπεύθυνο για τα χτυπήματα που δέχεται ο παίχτης.

Ας δούμε λοιπόν τον κάθε εχθρό ξεχωστά με screenshots :

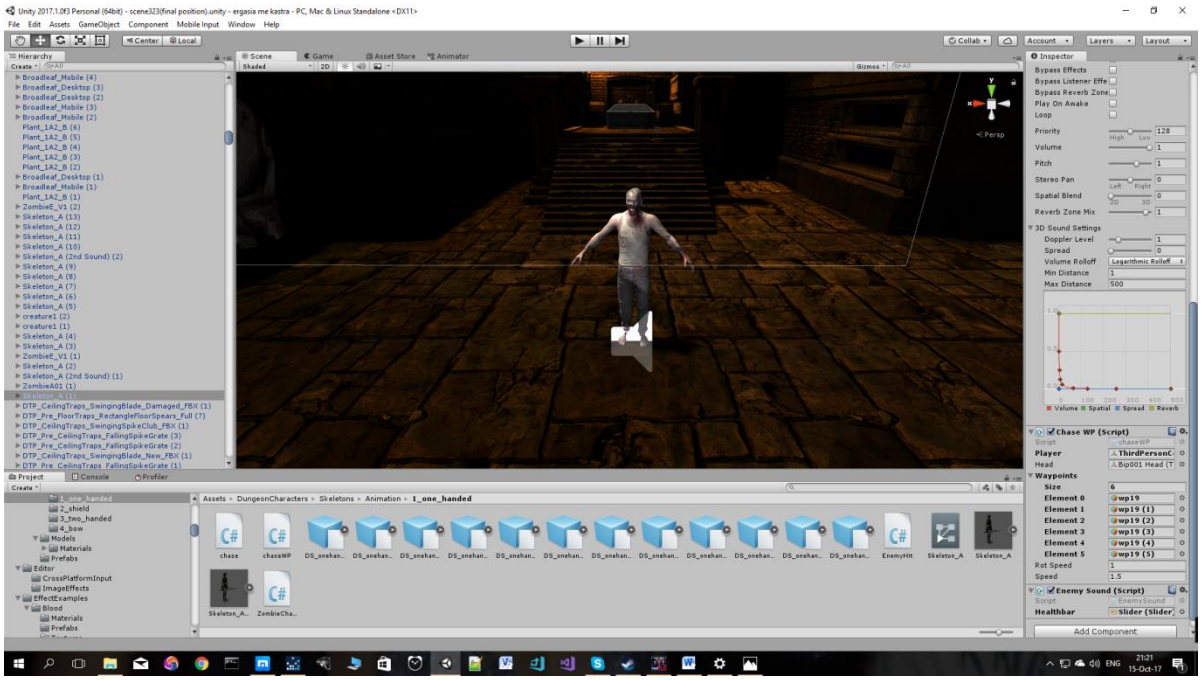
Εχθρός creature :



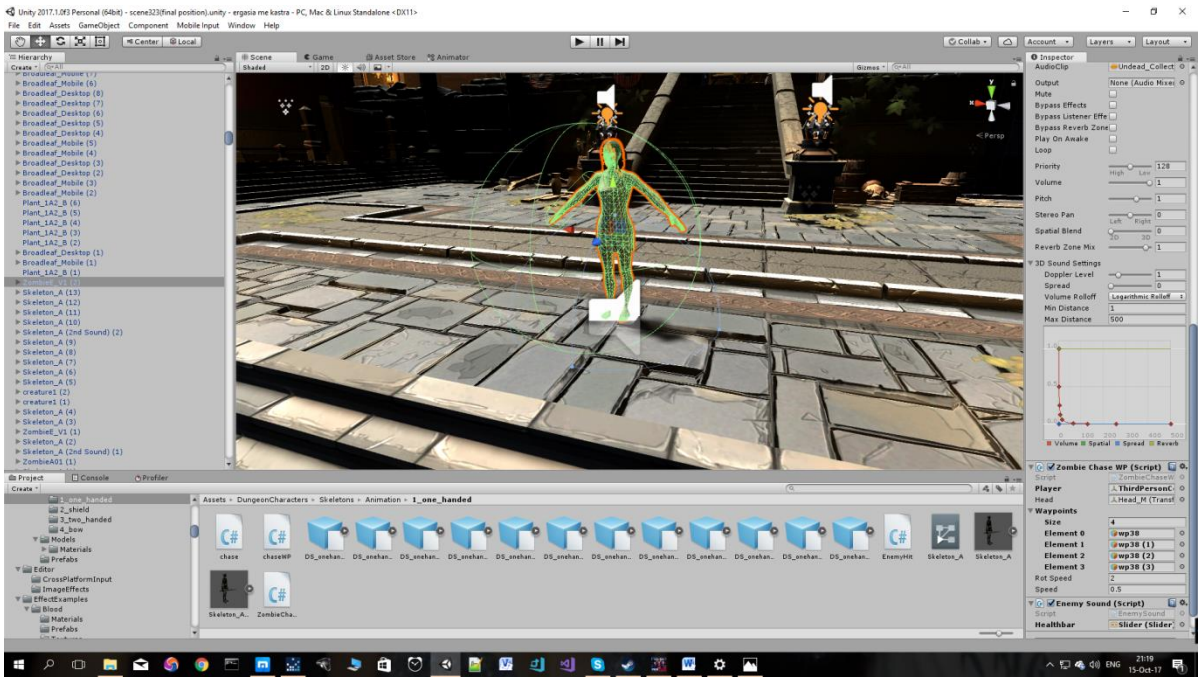
Εχθρός skeleton :



Εχθρός zombie male :



Εχθρός zombie female :



Παρατηρούμε σε κάθε screenshot παραπάνω οτι κάτω δεξιά χρησιμοποιήσαμε τον κώδικα τεχνητής νοημοσύνης ¨Chase WP¨, με ξεχωριστές τιμές για τον κάθε εχθρό. Οι τιμές αυτές προσαρμόζουν την ταχύτητα περιστοφής του, αλλα και την ταχύτητα κίνησής του. Θέλαμε να προσαρμόσουμε το animation ¨walking¨ του κάθε εχθρού με τον κώδικά μας προκειμένου να φαίνονται οι κινήσεις του όσο πιο αληθοφανής γίνεται.

Στο ίδιο script προσαρμόσαμε το κεφάλι του κάθε εχθρού σε συνδυασμό με την γωνία 30^{o} στον κώδικα. Έτσι με φυσικό τρόπο, ο εχθρός βλέπει τον παίχτη μόνο όταν το κεφάλι του είναι γυρισμένο προς αυτόν και τον πλησιάζει μόνο αν ο παίχτης βρίσκεται σε απόσταση μικρότερη των 10 μέτρων από αυτόν. Την απόσταση που ο εχθρός χρειάζεται να απέχει από τον παίχτη για να μεταβεί σε κατάσταση ¨attack¨ διαφέρει στον καθένα. Στο creature είναι 2.0, στον skeleton και στον zombie male ειναι 1.1 και στην zombie female ειναι 1.0. Είναι προσαρμοσμένο με ακρίβεια στο είδος του ¨attack animation¨ σε συνδυασμό με το ¨enemy weapon collider¨ προκειμένου να προκαλέσουν χτύπημα στον παίχτη. Στο ίδιο script βλέπουμε ότι ορίζουμε στους εχθρούς ποιος είναι ο παίχτης μας. Ένα δεύτερο script, βλέπουμε ότι είναι το ¨Enemy Sound¨ το οποίο ορίζει τον ήχο του κάθε εχθρού ξεχωριστά όταν ο παίχτης βρίσκεται σε κοντινή απόσταση. Το Healthbar και το Slider που βλέπουμε αφορά τον παίχτη και την ζωή του. Στον κώδικά μας θέλαμε να σταματάει ο ήχος των εχθρών όταν η ζωή του παίχτη μας είναι <= 0.

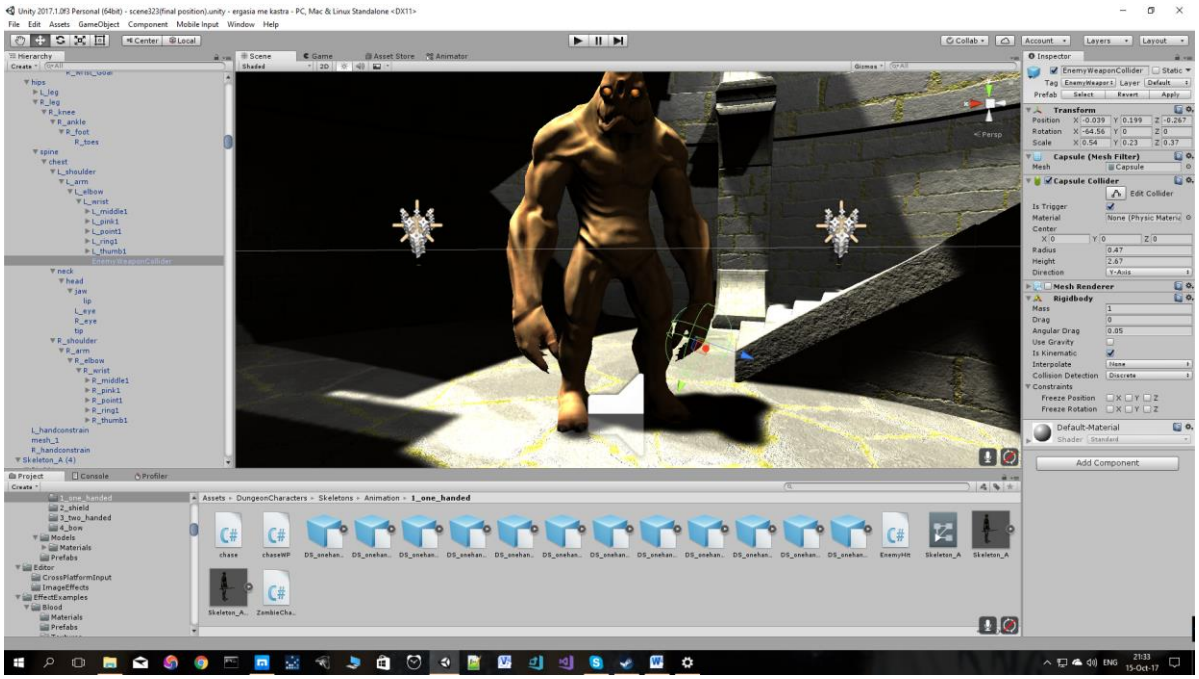
Παραθέτουμε τον κώδικα :

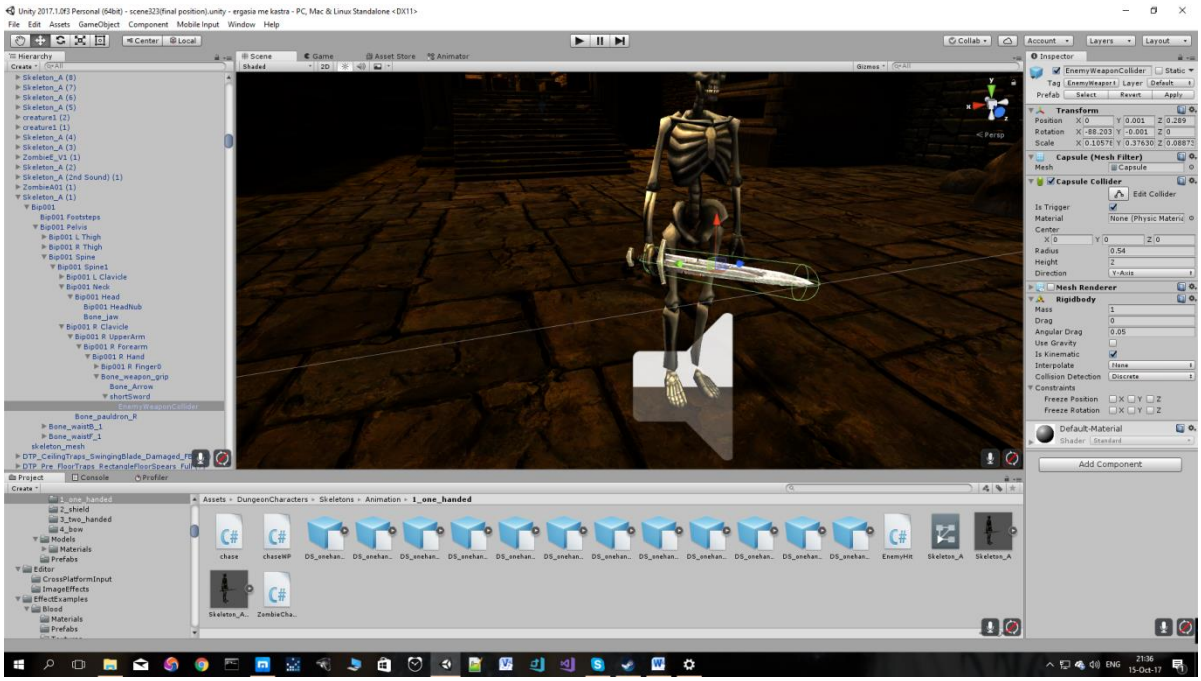
MonsterChaseWP : (Παράρτημα 5.1)

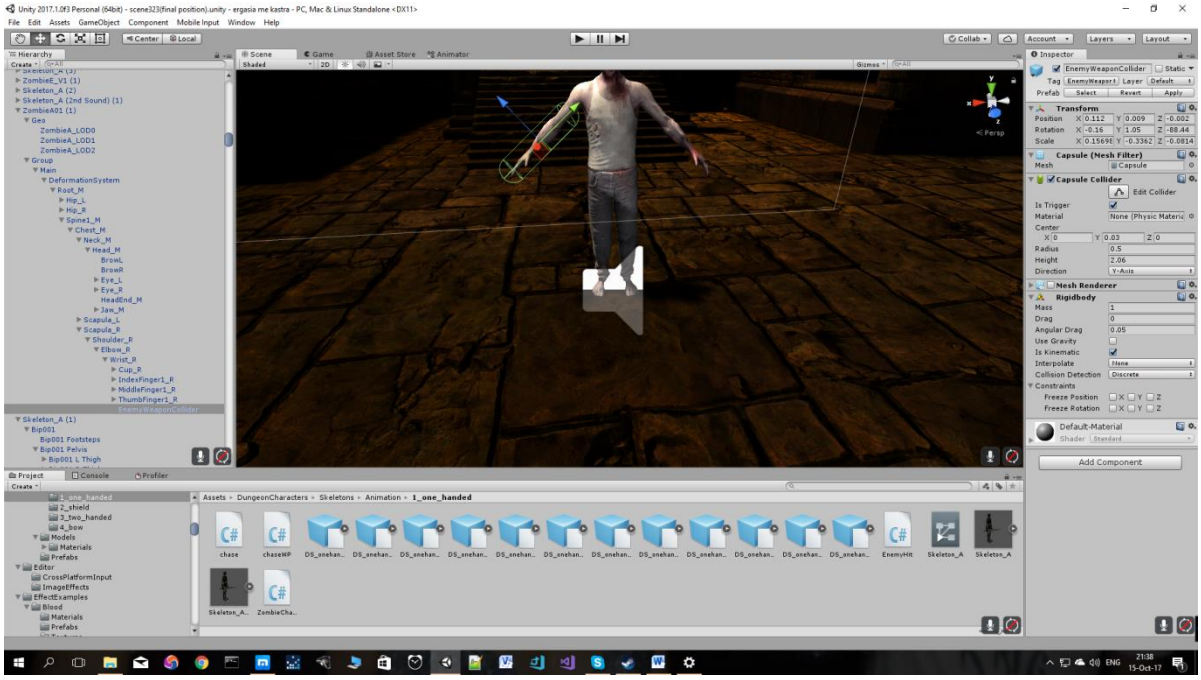
Παραθέτουμε τον κώδικα :

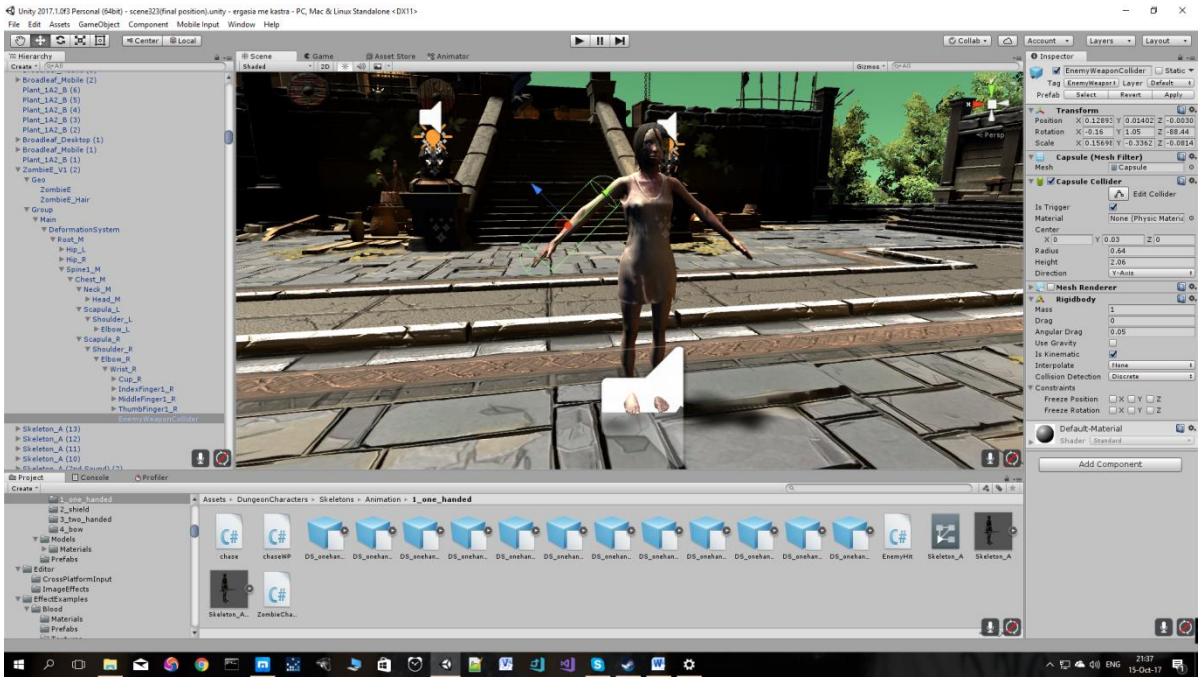
EnemySound : (Παράρτημα 5.1)

Έπειτα με screenshots για κάθε εχθρό θα δείξουμε που βρίσκεται ο capsule collider στον οποίο έγινε tag με την ονομασία ¨EnemyWeaponCollider¨ προκειμένου το script enemy hit που βρίσκεται τοποθετημένο στον παίχτη μας να εντοπίσει την επαφή εχθρού-παίχτη και να αφαιρέσει ζωή στον slider.





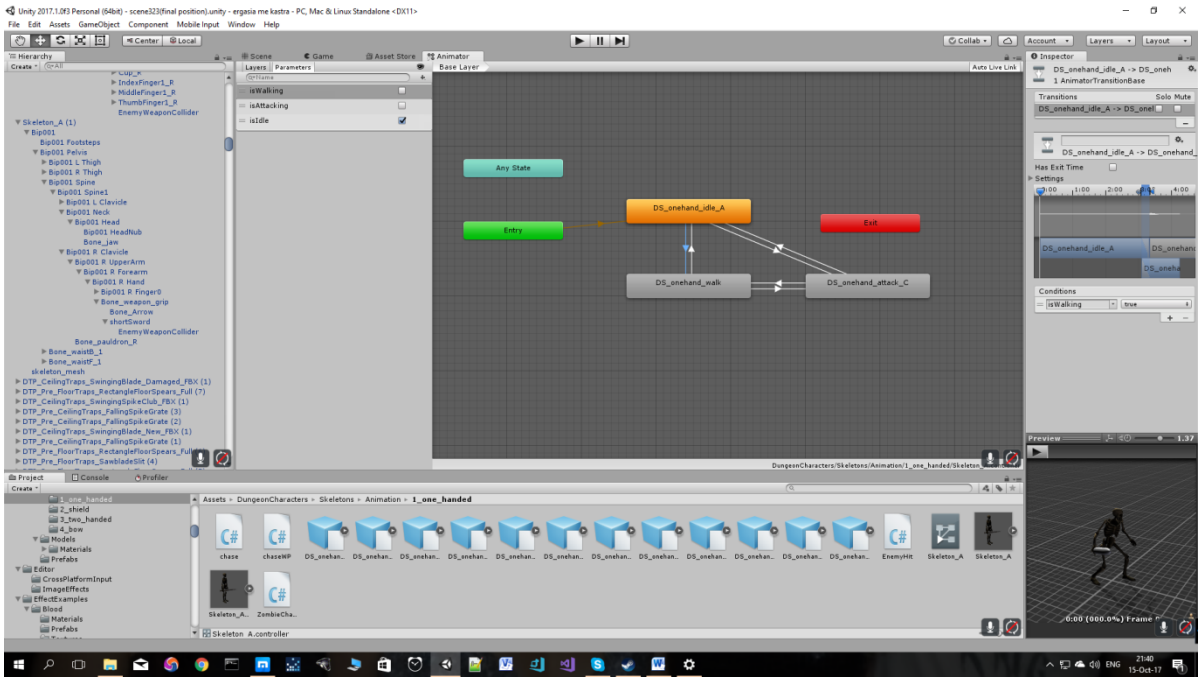




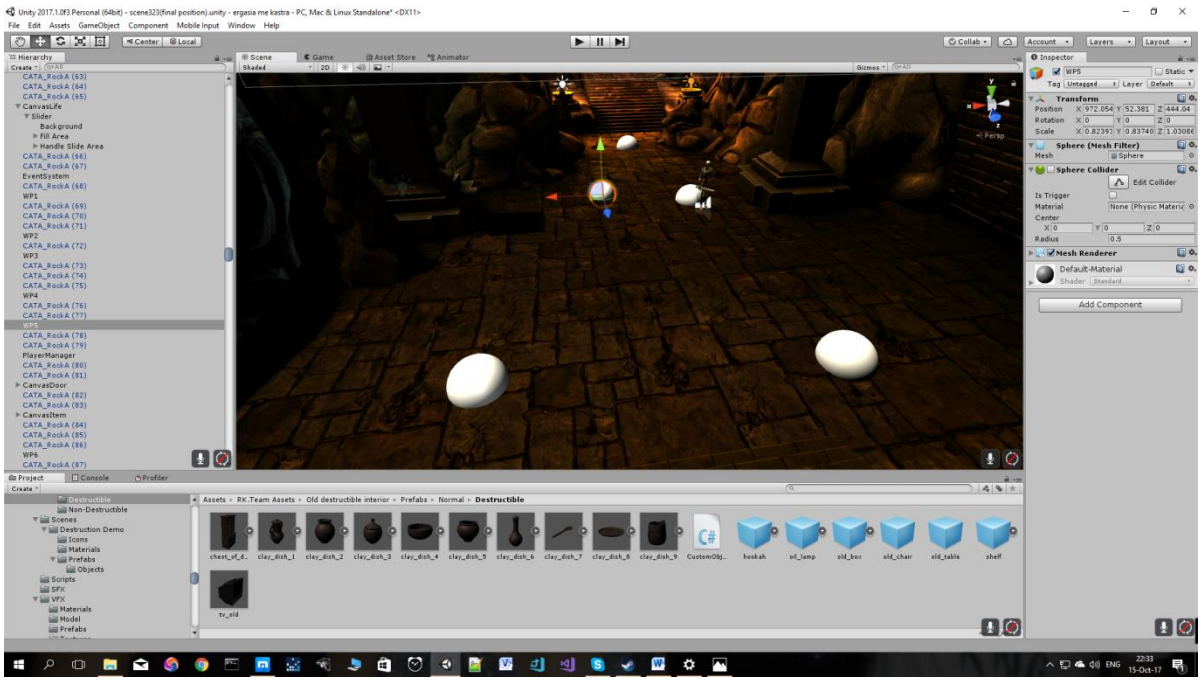
Σημαντικό είναι να αναφέρουμε ότι στους εχθρούς με ονομασία zombie και creature, χρησιμοποιήσαμε rigidbody με animation : Is Kinematic και επιλογή freeze position y, διότι ακολουθώντας ο εχθρός την περίπολο στα waypoints που έχουμε ορίσει -με απουσία παίχτη-, σιγά σιγά όσο κινούταν, τα πόδια βυθιζόντουσαν στο έδαφος. Τα πράγματα χειροτέρευαν όταν ο παίχτης ήταν παρών. Με αυτή την επιλογή λοιπόν, ο εχθρός μας παραμένει σταθερός στον άξονα των Y. Οσων αφορά το rigidbody που χρησιμοποιήσαμε ορίζοντας τον ¨EnemyWeaponCollider¨, έπρεπε πάλι να χρησιμοποιήσουμε την επιλογή στο animation : Is Kinematic, διότι αν δεν το κάναμε, ο capsule collider που ορίσαμε εκεί, δεν ακολουθούσε τα animation του χεριού ή του όπλου του εχθρού με αποτέλεσμα να μην υπάρχει επαφή με τον παίχτη.

Κάπου εδώ πρέπει να δείξουμε τον animator των εχθρών, ο οποίος αποτελείται από 3 καταστάσεις : isIdle, isWalking και isAttacking. Ας πάρουμε για παράδειγμα τον animator του εχθρού skeleton. Οι καταστάσεις αυτές αποτελούν Boolean (true - false) και ενεργοποιούνται από τον κώδικά μας. Όμως για να δουλέψει σωστά η μετάβαση στον animator, πρέπει να γίνουν οι αλληλοσυνδέσεις με βελάκια ανάμεσα στα 3 animation. Όπως βλέπουμε και στην εικόνα παρακάτω, για την μετάβαση από το animation DS_onehand_idle_A στο DS_onehand_walk, δηλαδή από την κατάσταση isIdle στην κατάσταση isWalking, πρέπει να ορίσουμε στο conditions κάτω δεξιά : isWalking = true. Για να συμβεί το ανάποδο, ορίζουμε στα conditions : isIdle = true. Επίσης στα δεξιά της εικόνας μας πρέπει να απενεργοποιήσουμε την επιλογή : Has Exit Time. Αν αυτή η επιλογή είναι ενεργή, ο εχθρός μας καθυστερεί προκειμένου να μεταβεί από την μια κατάσταση στην άλλη διότι περιμένει πρώτα να ολοκληρωθεί το

κάθε animation. Με τον ίδιο ακριβώς τρόπο ενεργοποιούνται και οι μεταβάσεις στον animator του παίχτη μας.



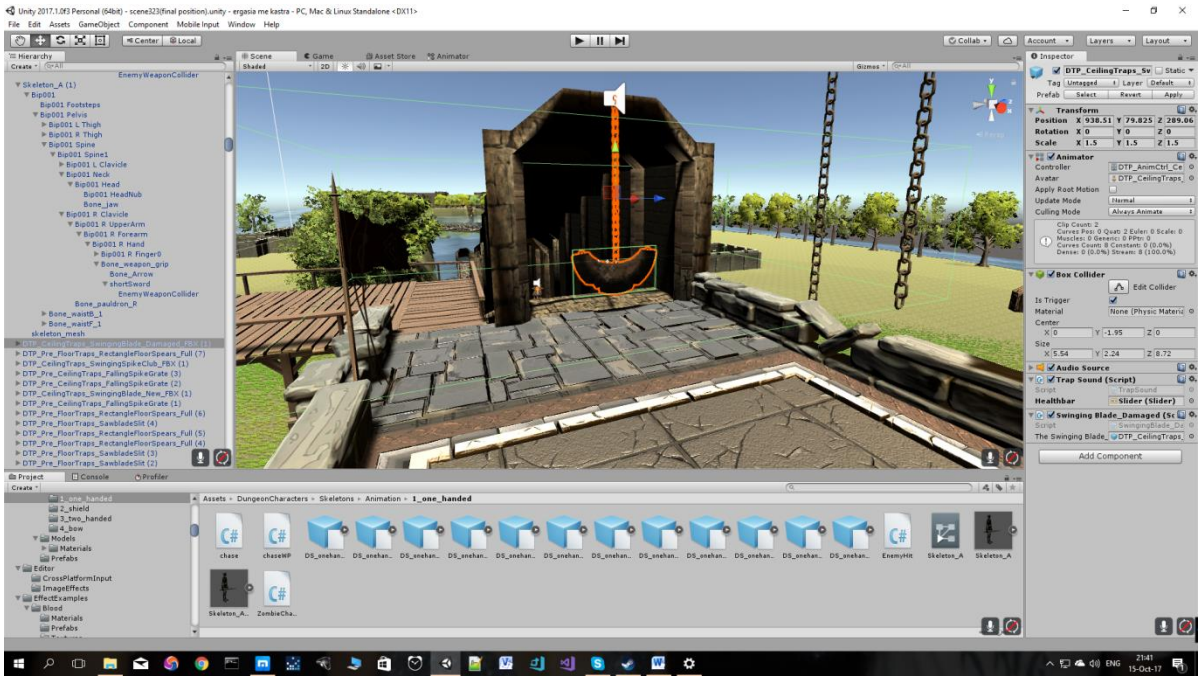
Κλείνοντας με τους εχθρούς, σημαντικό είναι να δείξουμε μια εικόνα των waypoints.



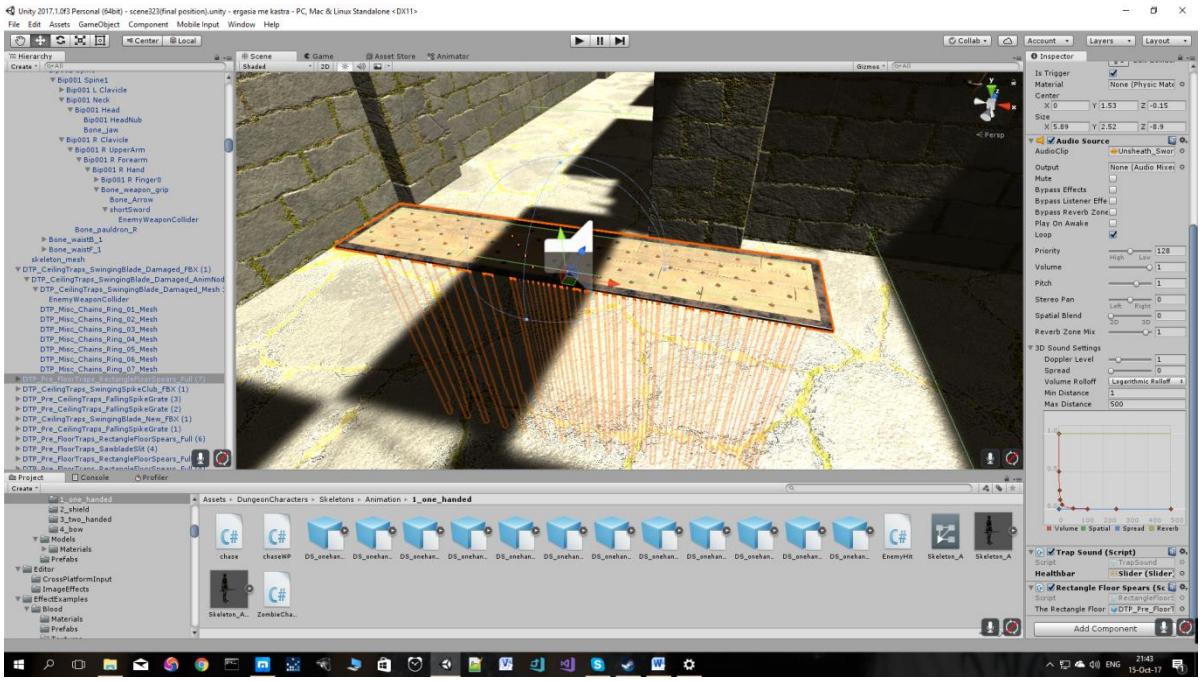
Τα waypoints αποτελούν sphere colliders, στους οποίους απενεργοποιήσαμε την υλική τους υπόσταση, κουτάκι sphere collider ανενεργό, αλλά και την οπτική τους προβολή, κουτάκι mesh renderer ανενεργό. Στην εικόνα μας το κρατάμε ενεργό απλά για να το δείξουμε. Τα waypoints έχουν τοποθετηθεί για κάθε εχθρό προκειμένου να καθορίσουν τα όρια κίνησής του όταν κάνει περιπολία και περιμένει τον παίχτη. Με τον κώδικα που αναφέραμε πιο πάνω, η μετάβαση από το ένα waypoint στο άλλο γίνεται με τυχαία συμπεριφορά. Όταν ο κάθε εχθρός εντοπίσει τον παίχτη, τον ακολουθεί και βγαίνει από τα waypoint του, έως ότου ο παίχτης απομακρυνθεί αρκετά από τον εχθρό. Μόλις συμβεί αυτό, ο εχθρός επιστρέφει στην αρχική του περιπολία.

5.2 Δημιουργία και λειτουργία παγίδων

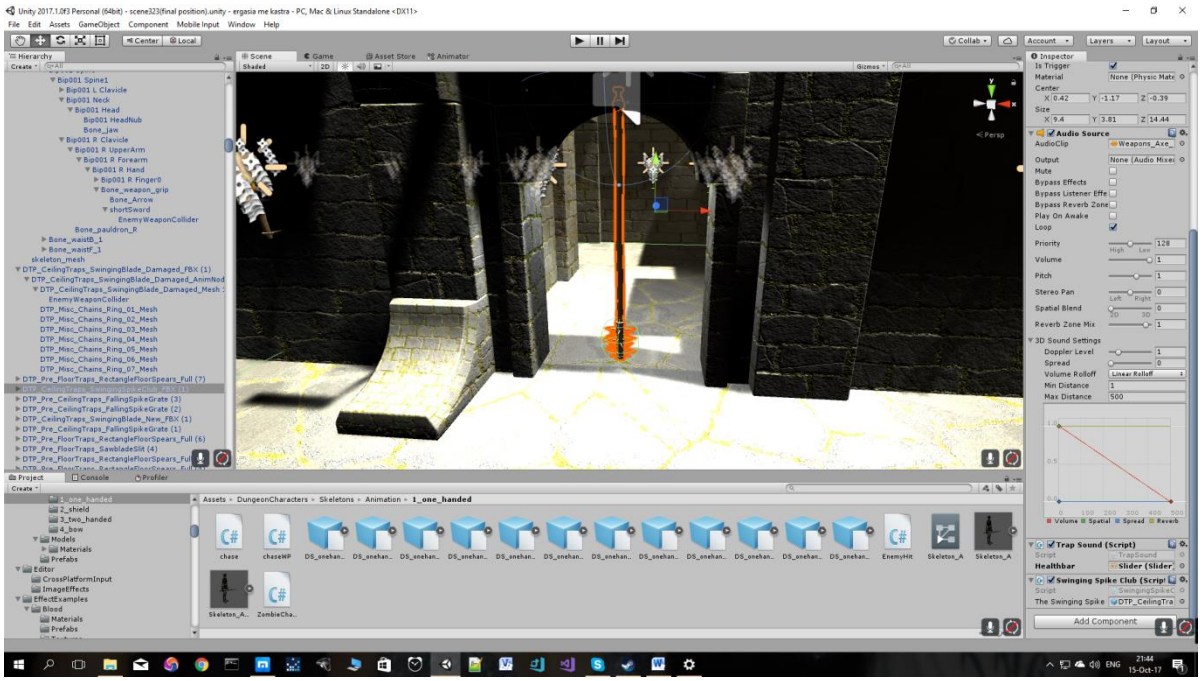
Ας δείξουμε πρώτα με εικόνες όλες τις παγίδες που χρησιμοποιήσαμε. Έχουμε πρώτον τις παγίδες ¨SwingingBlade¨ και την ¨SwingingBlade_Damaged¨, οι οποίες είναι ακριβώς οι ίδιες, με την μοναδική διαφορά ότι η 2^{η} φαίνεται χτυπημένη :



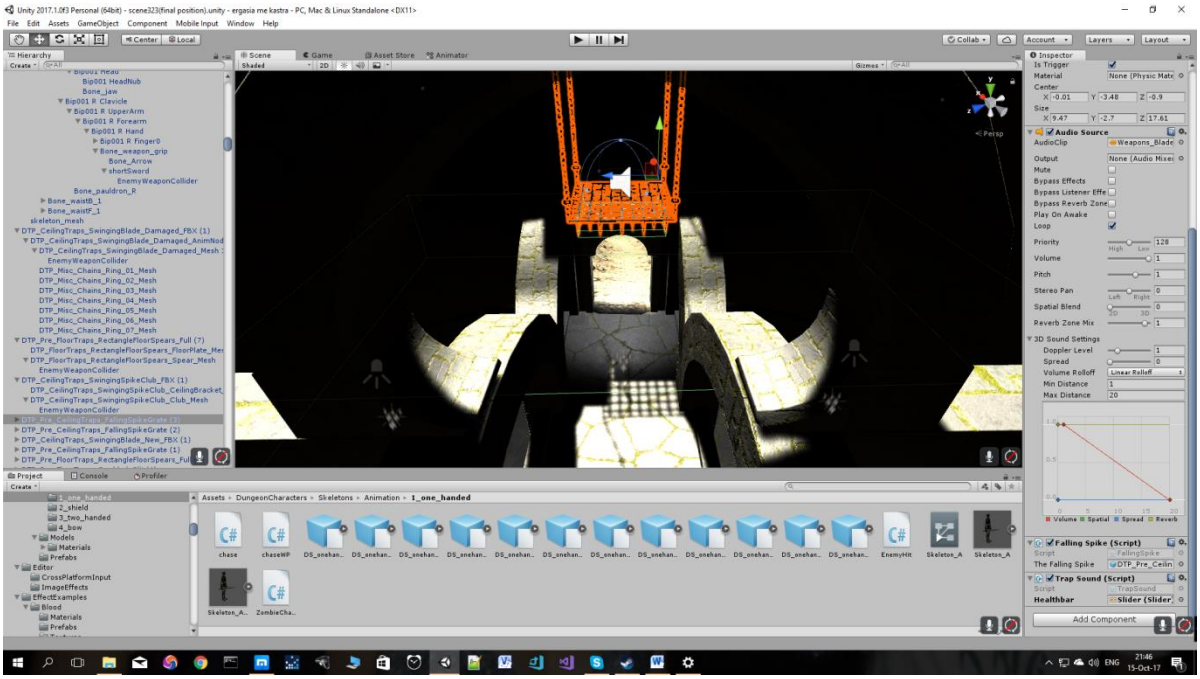
Έχουμε δεύτερον την παγίδα ¨**RextangleFloorSpears**¨ :



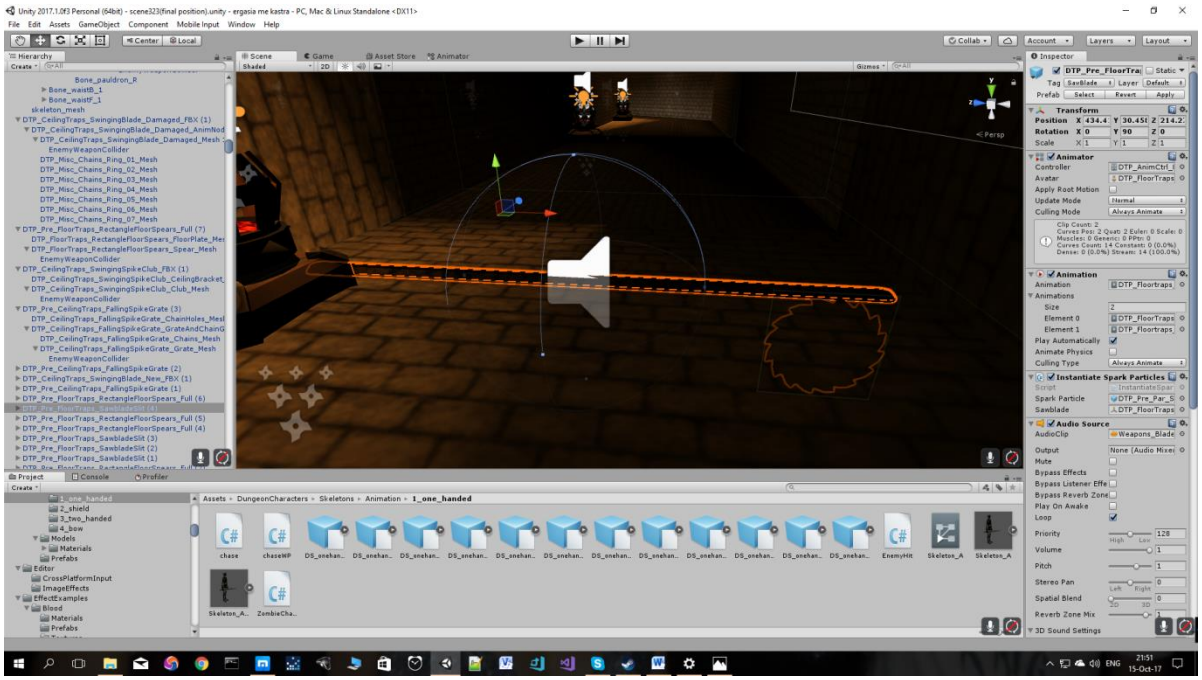
Έχουμε τρίτον την παγίδα ¨**SwingingSpikeClub**¨ :



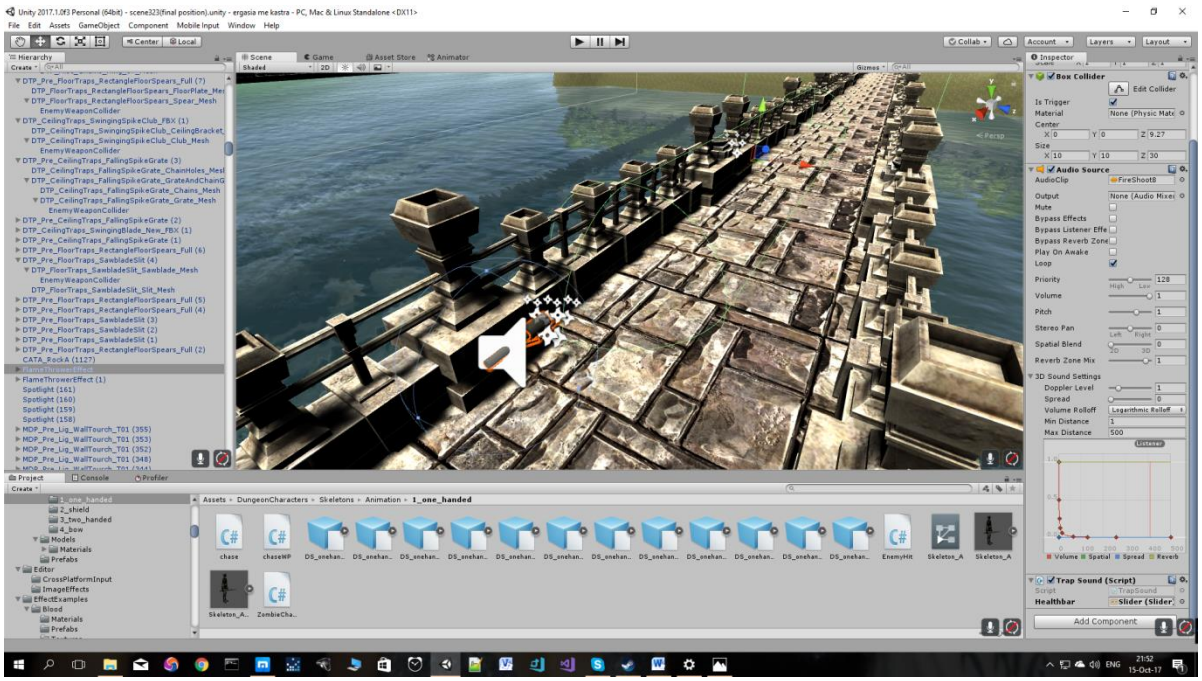
Έχουμε τέταρτον την παγίδα ¨**FallingSpikeGrate**¨ :



Έχουμε πέμπτον την παγίδα ¨**SawbladeSlit**¨ :



Έχουμε έκτον την παγίδα ¨**FlameThrowerEffect**¨ :

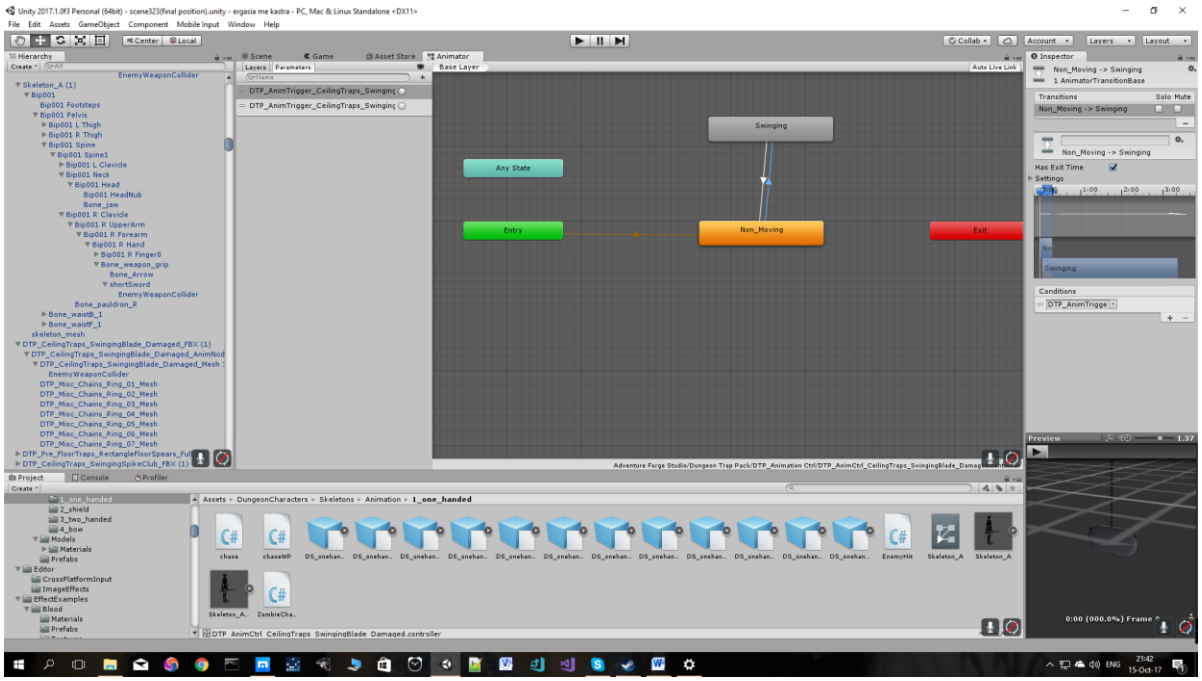


Όπως μπορούμε να δούμε από τις εικόνες παραπάνω, στην κάθε παγίδα ξεχωριστά χρησιμοποιήσαμε από δύο script, εκτός της τελευταίας που έχει μόνο ένα επειδή η κίνησή της καθορίζεται από ¨particle system¨. Το κοινό script λοιπόν σε όλες, ονομάζεται : ¨TrapSound¨

Το script αυτό απλά ενεργοποιεί τον ήχο της παγίδας όταν ο παίχτης βρίσκεται κοντά και τον απενεργοποιεί όταν ο παίχτης απομακρυνθεί. Και εδώ βλέπουμε την χρησιμοποίηση του ¨Slider¨ του παίχτη, έτσι ώστε όταν η ζωή του είναι <=0, η παγίδα σταματά να παράγει ήχο. Ο κώδικάς μας είναι ο εξής :

TrapSound(Script): (Παράρτημα 5.2)

Ο κώδικας που χρησιμοποιήσαμε για τις παγίδες είναι παρόμοιος, οπότε ας δούμε ενδεικτικά την πρώτη, ¨SwingingBlade_Damaged¨, μαζί με τον animator της :



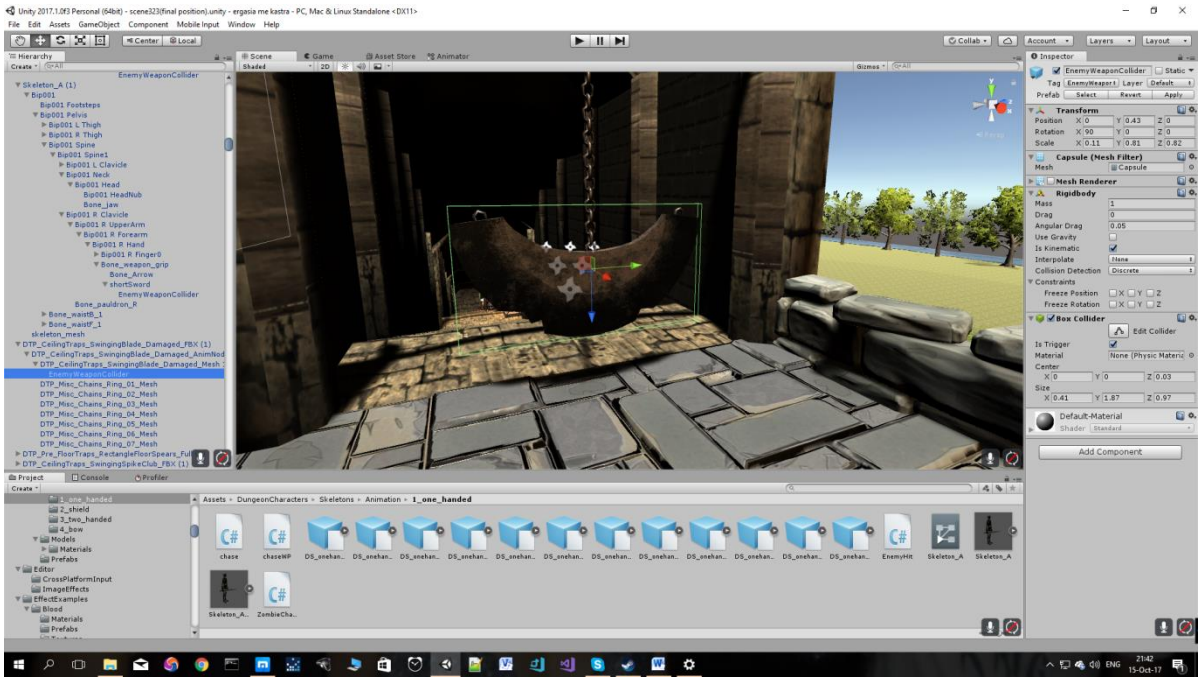
Διακρίνουμε στην εικόνα ότι στους παραμέτρους αριστερά, έχουμε : ¨trigger¨ και όχι ¨boolean¨. Στα δεξιά μας, στα conditions βλέπουμε τον ¨trigger¨ πλέον που καθορίζει την μετάβαση από την κατάσταση ¨Swinging¨ σε ¨Non_Moving¨ και αντίστροφα. Επίσης αυτή τη φορά στα δεξιά μας βλέπουμε την επιλογή ¨Has Exit Time¨ επιλεγμένη γιατί θέλουμε το κάθε animation να ολοκληρώνεται προτού μεταβεί στην επόμενη κατάσταση.

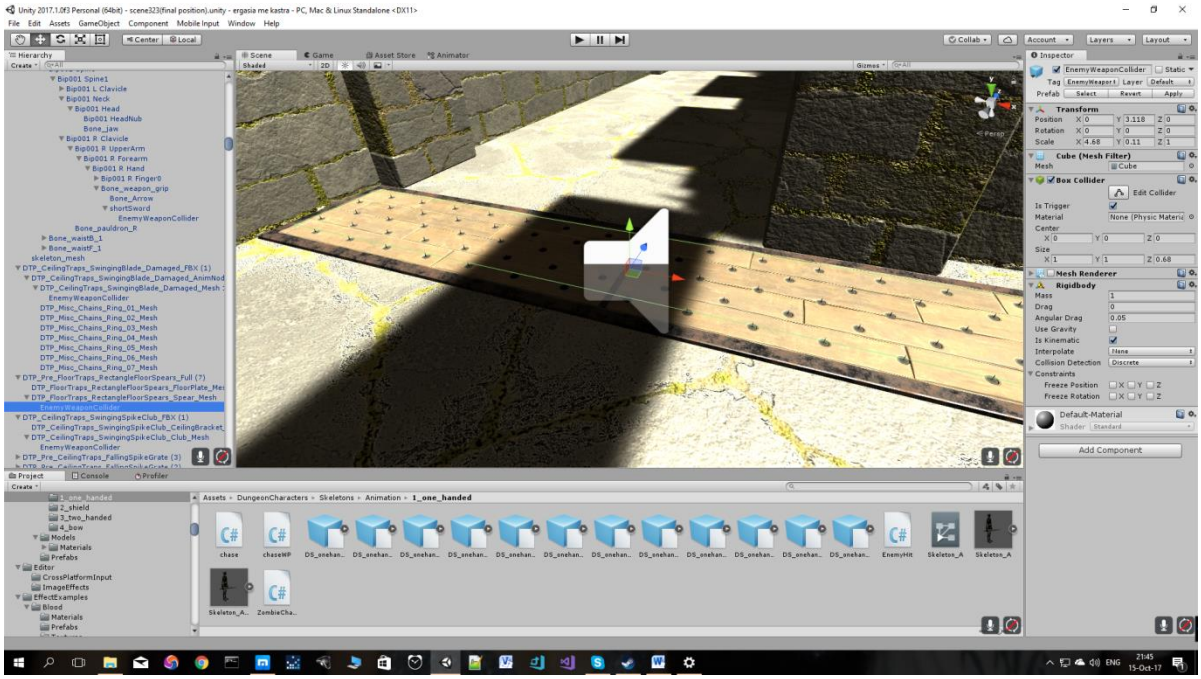
Ο κώδικάς μας είναι ο εξής :

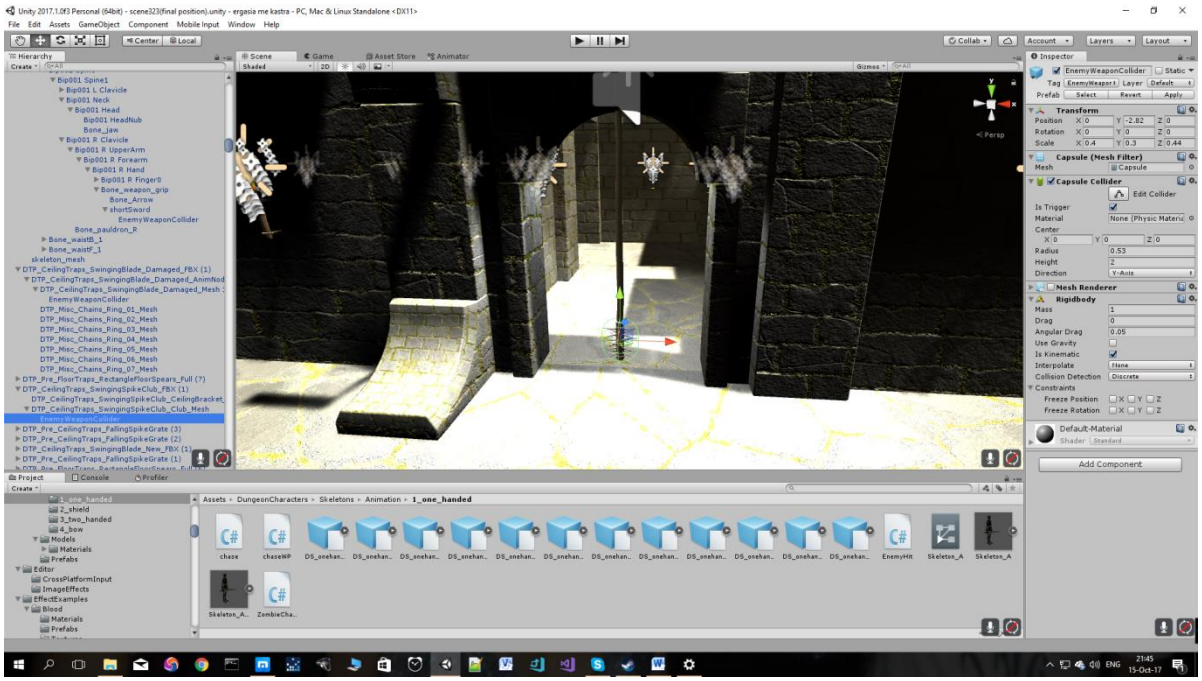
SwingingBlade_Damaged(Script): (Παράρτημα 5.2)

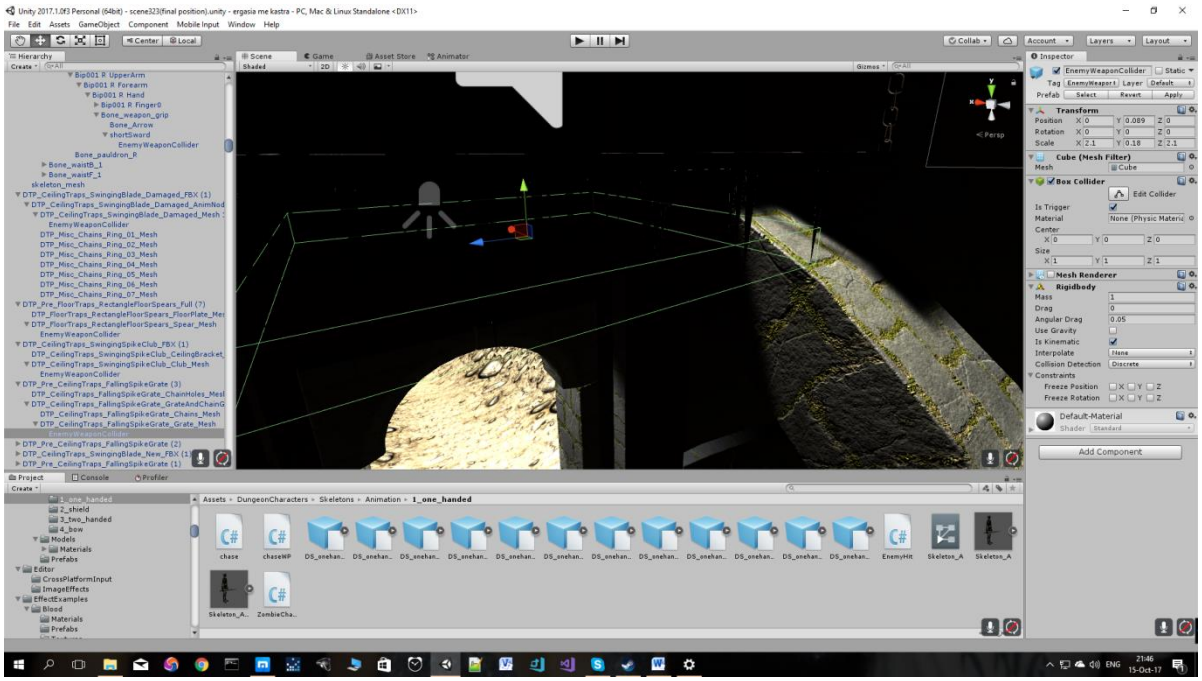
Σε όλες τις παγίδες μας τοποθετήσαμε colliders με tag name : ¨EnemyWeaponCollider¨, ακριβώς με τον ίδιο τρόπο που κάναμε και στους εχθρούς. Όταν το ¨EnemyWeaponCollider¨ έρθει σε επαφή με τον παίχτη μας, ο παίχτης θα δεχτεί χτύπημα και θα χάσει ζωή, όπως θα δούμε όλο τον μηχανισμό έπειτα, αναλύοντας τα script του παίχτη και συγκεκριμένα το ¨Enemy Hit (Script)¨.

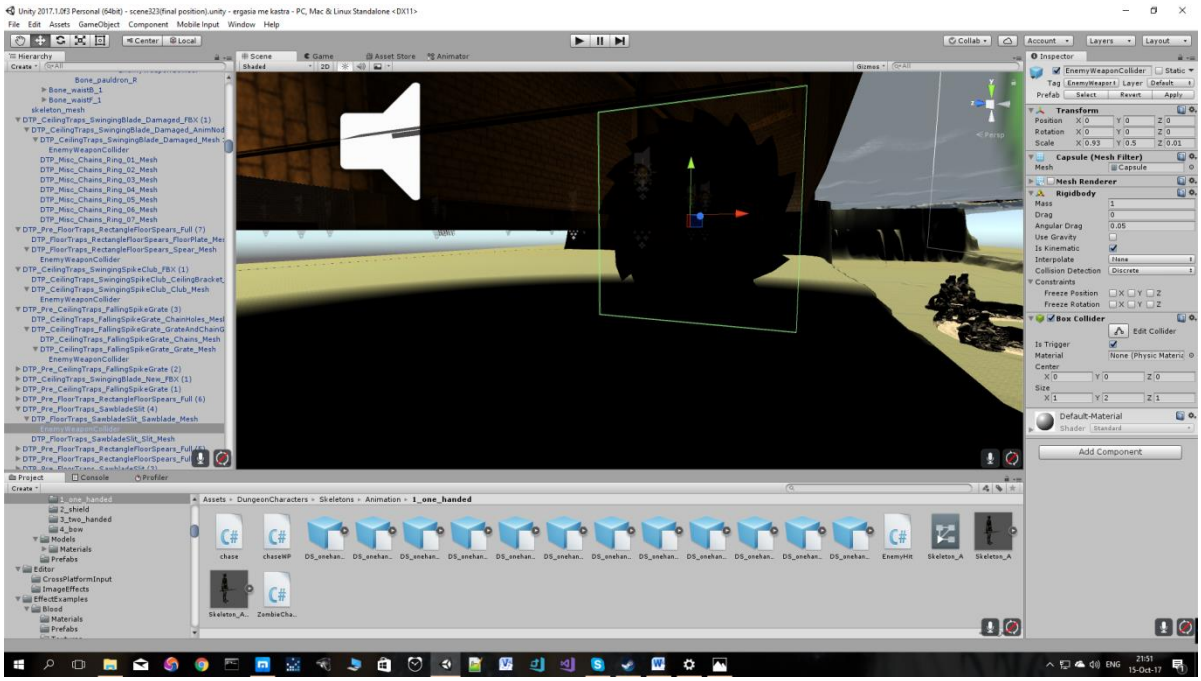
Ας δούμε τις εικόνες αναλυτικά για την κάθε παγίδα :

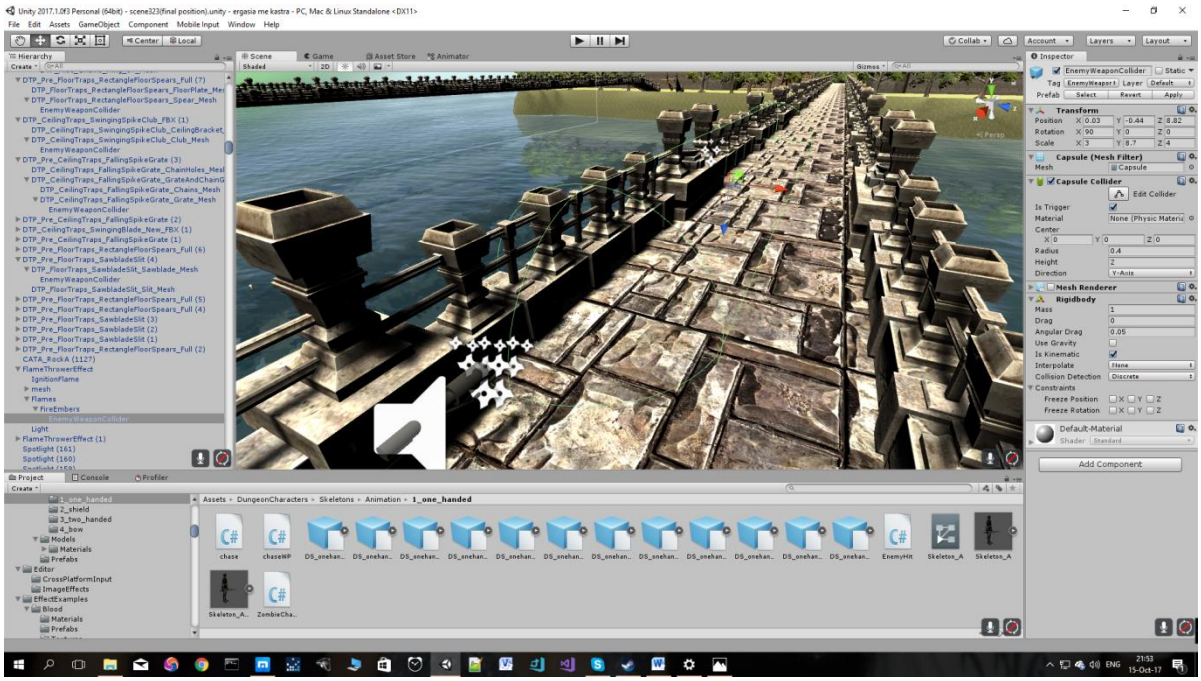










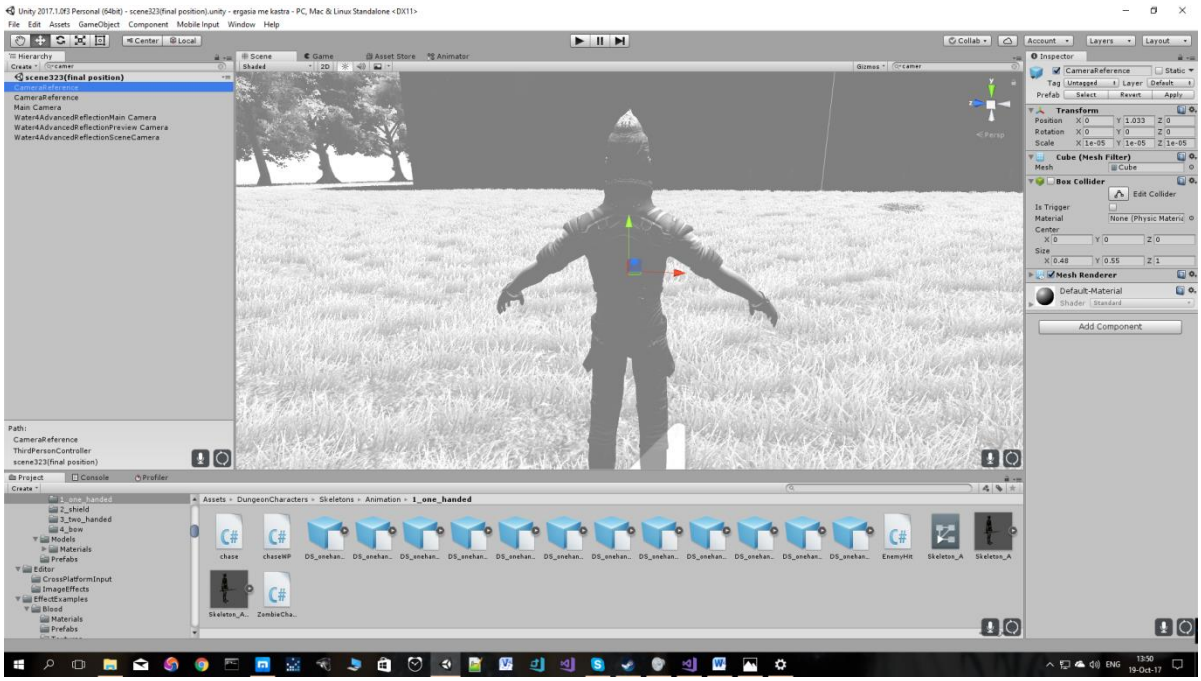


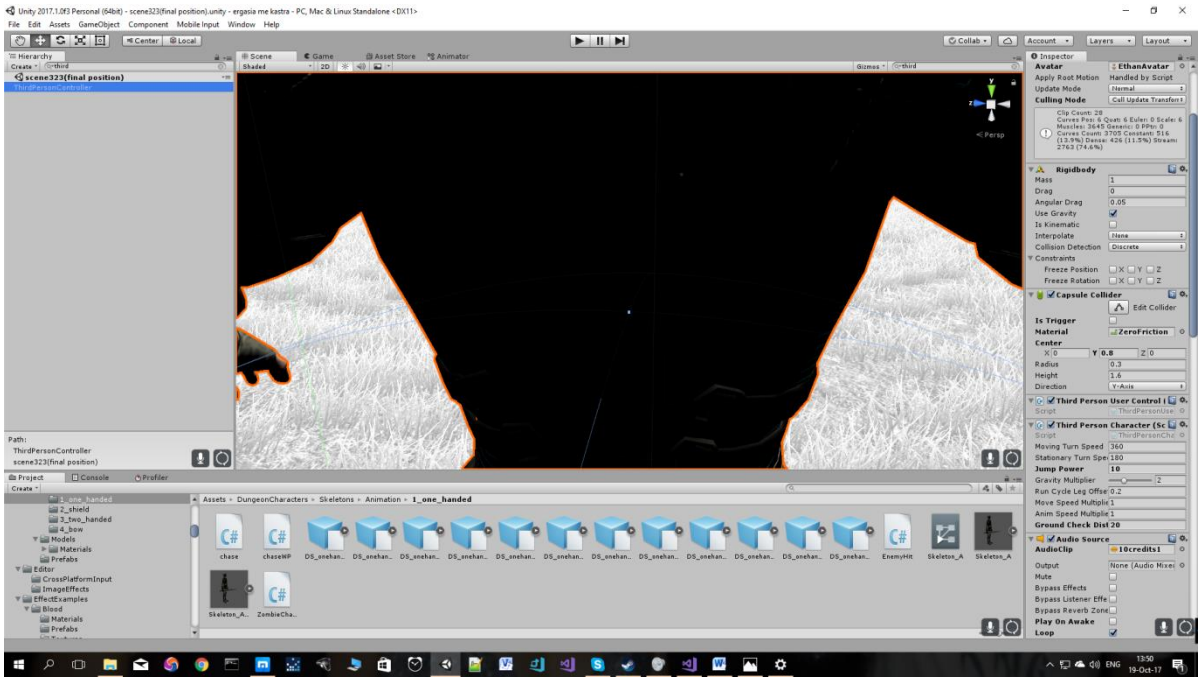
Οι κινήσεις που απαιτούνται από εμάς προκειμένου να δουλέψει όλος ο μηχανισμός σωστά, είναι στο ¨EnemyWeaponCollider¨ των παγίδων να απενεργοποιήσουμε το Mesh Renderer προκειμένου να μην είναι ορατό αλλά όχι τον Collider που κάνει επαφή. Και αυτό γιατί θέλουμε να έχει υλική υπόσταση το σημείο που προκαλεί χτύπημα στον παίχτη. Βέβαια και ο collider αυτός για να λειτουργήσει πρέπει να έχει ενεργή την επιλογή ¨Is Trigger¨. Όπως και στο όπλο του εχθρού, έτσι και εδώ χρειαζόμαστε το

rigidbody με την επιλογή ¨Is Kinematic¨ ενεργή, προκειμένου ο ¨EnemyWeaponCollider¨ να ακολουθεί την κίνηση του animation.

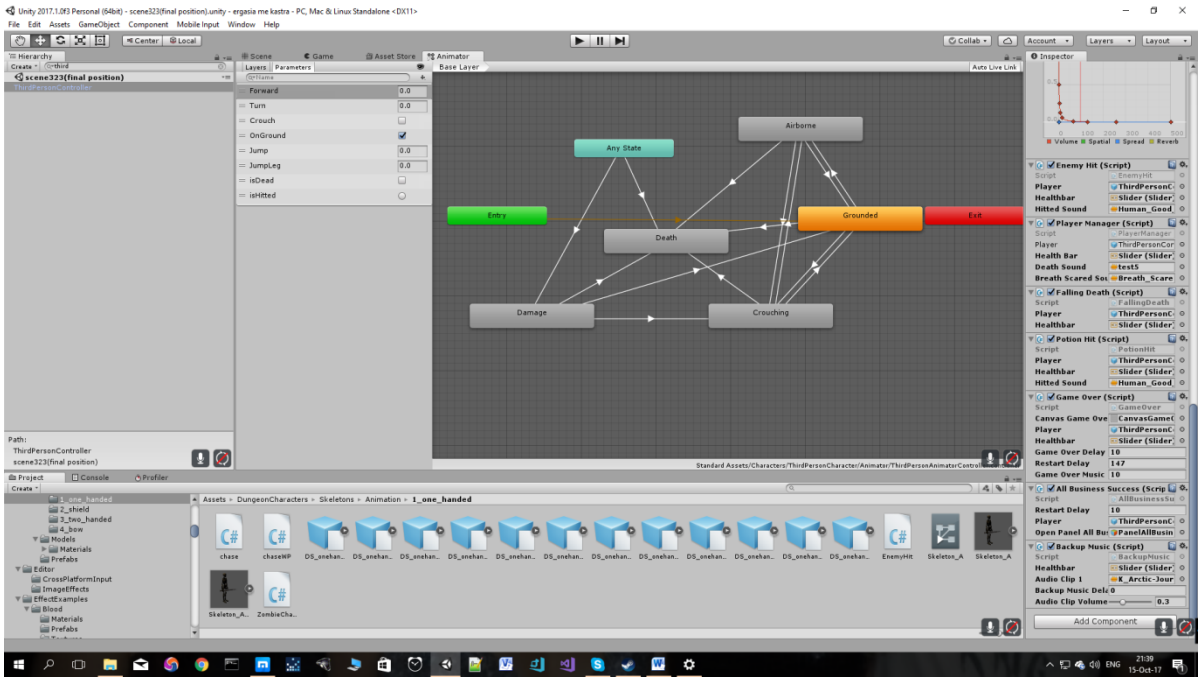
5.3 Επεξεργασία Παίχτη και νέες λειτουργίες

Ο παίχτης μας, ο Ethan είναι ο βασικός χαρακτήρας του unity. Ας παρουσιάσουμε με εικόνες την επεξεργασία που κάναμε σε αυτόν προσθέτοντάς του νέες λειτουργίες, νέα animation και script αλλά και μετατρέποντας το όλο project σε τρίτο πρόσωπο.

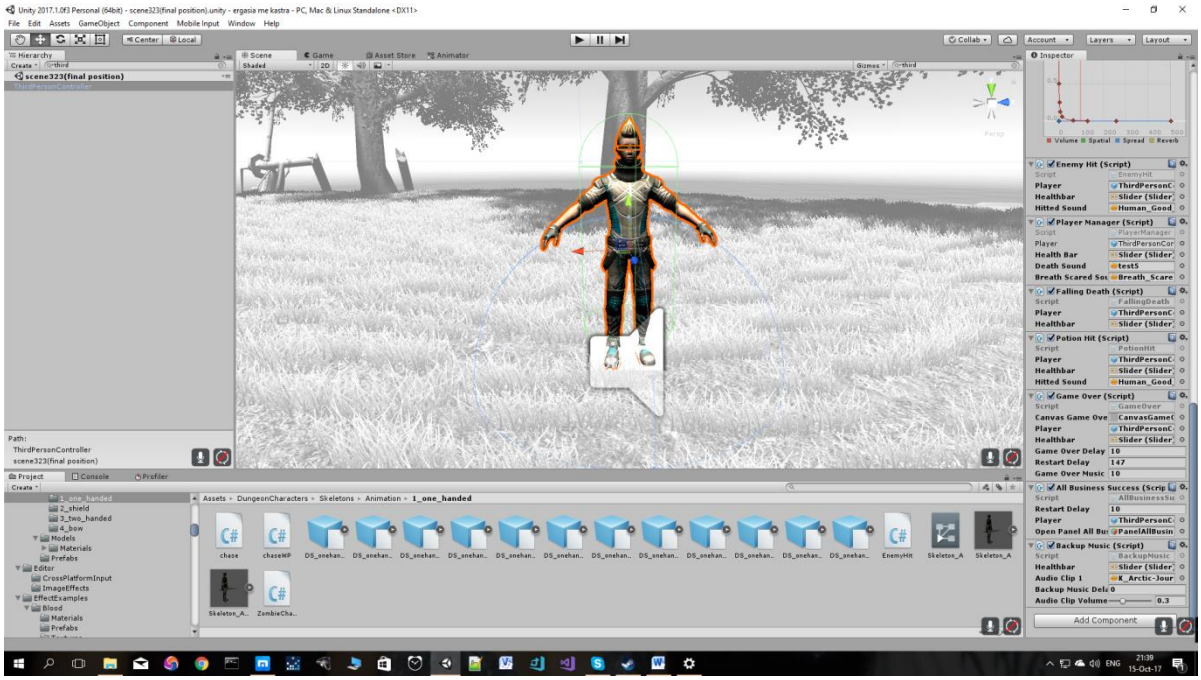




Στις παραπάνω εικόνες βλέπουμε αμυδρά, ένα πολύ μικρό BoxCollider που τοποθετήσαμε μέσα στον παίχτη μας. Αυτό το αμυδρά εμφανίσιμο κουτί, δεν είναι ορατό την ώρα του render και ο μόνος σκοπός του είναι να αποτελέσει σημείο αναφοράς για την κάμερά μας. Η κάμερα έχει τοποθετηθεί πίσω και πάνω από τον παίχτη μας, έτσι ώστε να επιτύχουμε την τρίτου προσώπου απεικόνιση. Επίσης, η κίνηση του ποντικιού μας, κάνει περιστροφή γύρω από αυτό κουτί, άρα και γύρω από τον παίχτη μας.



Παρατηρούμε στην παραπάνω εικόνα τον animator του παίχτη μας. Εμφανείς είναι οι καταστάσεις ¨isDead¨ και ¨isHitted¨ σαν παράμετροι στα αριστερά μας. Η πρώτη κατάσταση ενεργοποιείται με Boolean (true - false) και αφορά το animation Death και η δεύτερη με trigger και αφορά το animation Damage. Τα βελάκια έχουν τα conditions ακριβώς όπως αναλύσαμε και πριν σε παγίδες και εχθρούς αλλά με την επιλογή ¨Has Exit Time¨ επιλεγμένη μόνο στις μεταβάσεις από το animation Damage στο animation Grounded ή στο animation Crouching. Και αυτό γιατί θέλουμε το animation Damage να ολοκληρώνεται προτού μεταβεί στην επόμενη κατάσταση. Δεν θέλουμε να ολοκληρώνεται το συγκεκριμένο animation, όμως αν η ζωή του παίχτη είναι <=0 όπου ενεργοποιείται το animation Death. Όπως είναι λογικό, όλα τα βελάκια καταστάσεων που οδηγούν στο animation Death έχουν απενεργοποιημένη την επιλογή ¨Has Exit Time¨ προκειμένου το συγκεκριμένο animation να παίξει ακαριαία, χωρίς καθυστέρηση. Σημαντικό κρίνεται να αναφέρουμε ότι αυτά τα animations τα αντλήσαμε από ένα πακέτο εχθρών(Fantasy Monster-Skeleton) κάνοντας τις εξής ρυθμίσεις προκειμένου να προσαρμοστούν στον παίχτη μας : Στον inspector του animation, μετατροπή του animation type από generic σε humanoid και στην κατηγορία avatar definition, επιλογή : create from this model.



Παρατηρούμε στα δεξιά του παίχτη μας, ThirdPersonController, εφτά script με πολύ σημαντικές λειτουργίες. Τα script αυτά ειναι τα εξής : EnemyHit(Script), PlayerManager(Script), FallingDeath(Script), PotionHit(Script), GameOver(Script), AllBusinessSuccess(Script) και BackupMusic(Script). Σε αυτό το σημείο θα παρουσιάσουμε μόνο τα δύο πρώτα script, ενώ τα υπόλοιπα θα παρουσιαστούν στα επόμενα υποκεφάλαια, αφού πρώτα αναλύσουμε και άλλες λειτουργίες.

Οπότε έχουμε το

EnemyHit(Script): (Παράρτημα 5.3)

Σε αυτό το script βλέπουμε ότι όταν ο παίχτης έρθει σε επαφή με οτιδήποτε έχει tag : ¨EnemyWeaponCollider¨, που υπάρχει σε εχθρούς και σε παγίδες, όπως προαναφέραμε, τότε δίνεται η εντολή στο slider της ζωής του παίχτη να του αφαιρέσει το 20% της ζωής του, ενεργοποιείται το animation Damage μέσω του trigger (`"isHitted"`) αλλά και ενεργοποιείται ο ήχος ¨ HittedSound ¨ που εμείς έχουμε ορίσει κάνοντας public. Θέλουμε όμως αυτόν τον ήχο να παίξει μόνο μια φορά, γι'αυτό δίνουμε την εντολή στο AudioSource, PlayOneShot. Βέβαια επειδή είναι μέσα στο update του κώδικα, ο χειρισμός γίνεται με Boolean, το GetHit(true-false), αποφεύγοντας έτσι την αναπαραγωγή του animation και του ήχου ανά frame. Σημαντικό ήταν να ρυθμίσουμε όλο τον μηχανισμό αυτό να σταματάει όταν η ζωή του παίχτη είναι <=0, όπου ενεργοποιείται το GameOver(Script).

Οπότε έχουμε το

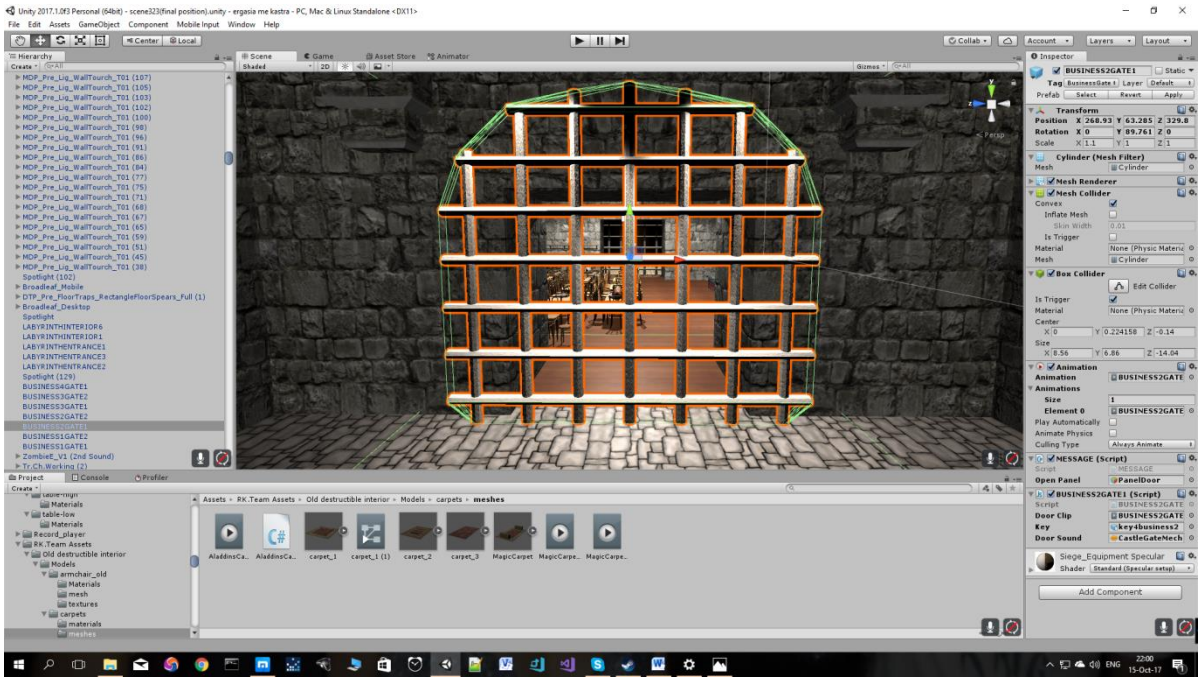
PlayerManager(Script): (Παράρτημα 5.3)

Αυτό το script αποτελεί το βασικό script του παίχτη. Βλέπουμε ότι το health το ορίζουμε στην αρχή ως static, κάτι που μας δίνει την δυνατότητα να το χρησιμοποιήσουμε σε όποιο άλλο script θέλουμε. Την function ¨ReduceHealth¨ την αφήνουμε ανενεργή προς το παρόν προκειμένου να χρησιμοποιηθεί ανά πάσα στιγμή χρειαστούμε να αυξήσουμε την δυσκολία του παιχνιδιού, κάνοντας τον παίχτη να χάνει ζωή με το πέρασμα του χρόνου. Ορίσαμε έναν ήχο τύπου ¨danger¨ που θα ειδοποιεί τον παίχτη ότι η ζωή του είναι σε χαμηλά επίπεδα και τέλος όταν η ζωή του είναι <=0 ενεργοποιούμε τον ήχο DeathSound που εμείς ορίσαμε, καθώς και το animation Death του παίχτη. Αφού παίξει επιτυχώς ο ήχος DeathSound και ολοκληρωθεί, τότε ενεργοποιείται το GameOver (Script). Το καταφέραμε αυτό βάζοντας δύο Delay στον ήχο και στις λειτουργίες του script, όπως θα δούμε στη συνέχεια.

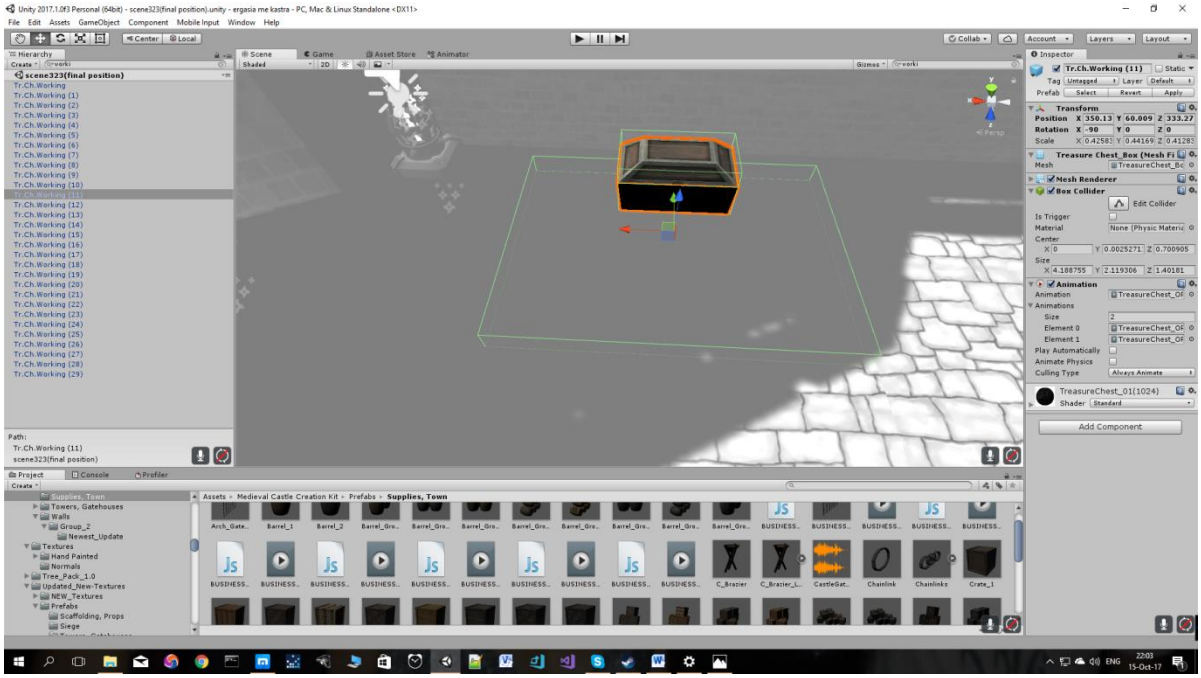
5.4 Λειτουργία Πυλών, κλειδιών και σεντουκιών

Αρχικά θα θέλαμε να παρουσιάσουμε με εικόνες τις πύλες (Gate), τα απλά κλειδιά που ανοίγουν τις πύλες στο κάστρο και στο λαβύρινθο (key), τα μεγαλύτερα κλειδιά που ανοίγουν τις πύλες των επιχειρήσεων (keybusiness) και τα σεντούκια μας που περιέχουν είτε τα 2 αυτά είδη κλειδιών, είτε το medikit, το bluepotion και το redpotion.

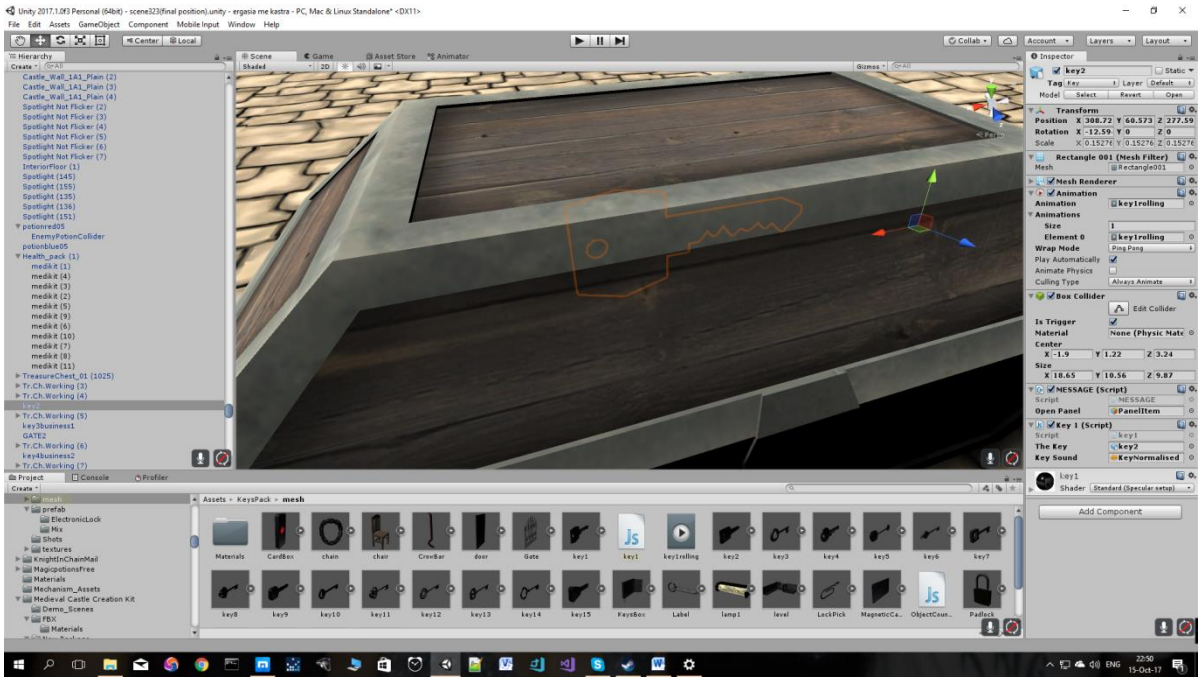
Gate :



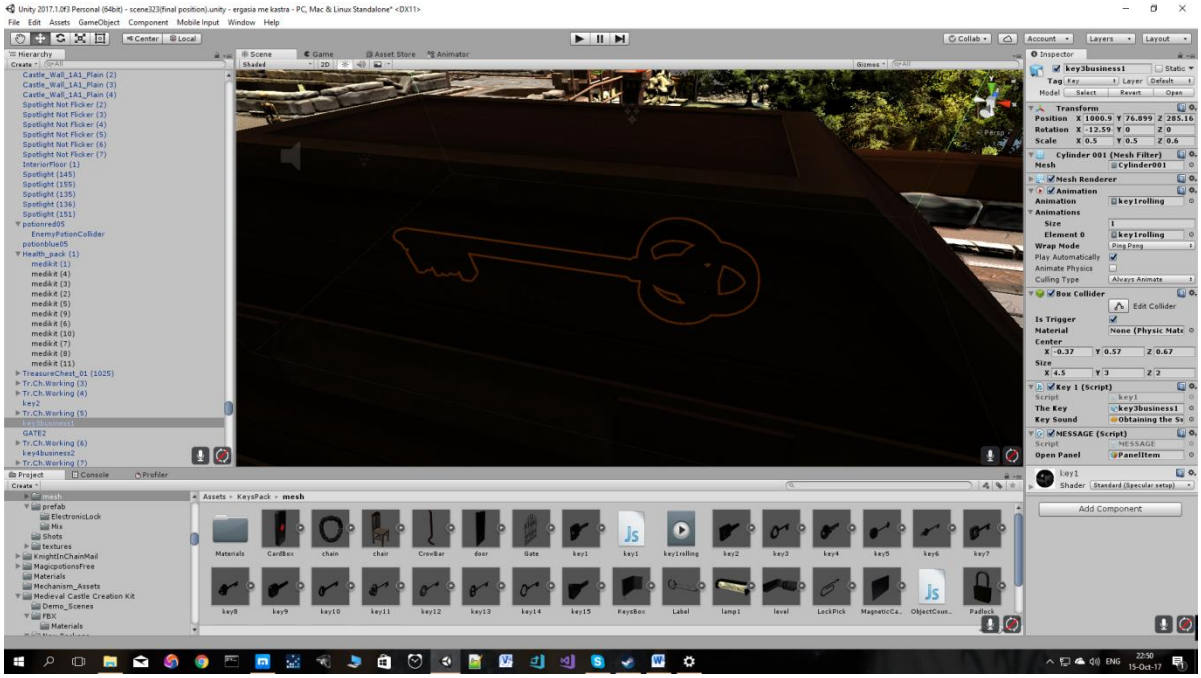
TreasureChest :



Key :



Keybusiness :



Οι Gates όπως και οι BusinessGates λειτουργούν ακριβώς με τον ίδιο τρόπο, χρησιμοποιώντας τα ίδια script. Τα keys όπως και τα keybusiness επίσης λειτουργούν με τον ίδιο τρόπο, χρησιμοποιώντας τα ίδια script. Το κάθε key ξεκλειδώνει μία και μόνο

Gate, στο κάστρο ή στον λαβύρινθο ενώ το κάθε keybusiness ξεκλειδώνει μία ή και δύο BusinessGates, της συγκεκριμένης επιχείρησης και μόνο, ανάλογα πόσες πύλες εισόδου έχει. Τα απλά key και τα keybusiness διαφέρουν οπτικά καθώς χρησιμοποιήσαμε άλλο τύπο κλειδιού. Και τα 2 κινούνται με animation το οποίο φαίνεται, όταν το σεντούκι ανοίγει. Από το script, με την χρήση var, κάνουμε πιο εύκολη την διαδικασία χρησιμοποίησης πολλών Gate, αφού μας επιτρέπει εκτός script να σύρουμε το animation που θέλουμε, τον ήχο που θέλουμε και το συγκεκριμένο κλειδί που θελουμε να ανοίγει την κάθε πόρτα. Το script των πυλών είναι αλληλένδετο με το script των κλειδιών. Ας δούμε και τον κώδικα :

BUSINESS2GATE1(Script) : (Παράρτημα 5.4)

Key1(Script): (Παράρτημα 5.4)

Παρατηρούμε λοιπόν από τον κώδικα κλειδιού, ότι όταν ο παίχτης είναι δίπλα στο κλειδί, δηλαδή μέσα στον collider του κλειδιού και πατήσει το κουμπί ¨E¨ στο πληκτρολόγιο, τότε και μόνο τότε θα πάρει το κλειδί, καθώς θα ακουστεί και ο ήχος που εμείς έχουμε ορίσει. Αν αυτό συμβεί το κλειδί παύει να είναι ενεργό μέσα στο σεντούκι και ανήκει στον παίχτη πλέον. Βλέπουμε ότι στο script της πύλης το gameobject είναι το key. Οπότε όταν έχουμε πάρει το συγκεκριμένο κλειδί που ανοίγει την συγκεκριμένη πόρτα, δηλαδή thekey.active=false –εξαφανίζεται– και παράλληλα βρισκόμαστε στον collider της πύλης και πατήσουμε και το κουμπί ¨E¨ στο πληκτρολόγιο, τότε θα ενεργοποιηθεί το animation που ανοίγει την πύλη μαζί με τον κατάλληλο ήχο. Στο script του κλειδιού, όπως και στης πόρτας, ορίζουμε εμείς μέσα από το unity ποιό θα είναι το εκάστοτε κλειδί για κάθε πόρτα, δίνοντάς μας την δυνατότητα να χρησιμοποιήσουμε το ίδιο script πολλές φορές.

Τα σεντούκια έχουν ρυθμιστεί να ανοίγουν με ένα απλό animation, όποτε ο παίχτης βρίσκεται κοντά τους.

Το MESSAGE(Script), το οποίο υπάρχει σε όλες τις πύλες, τα κλειδιά, τα medikit, τα bluepotion και τα redpotion, αποτελεί το μήνυμα που πληροφορεί τον παίχτη για την παρουσία πόρτας που μπορεί να ανοίξει ή Item που μπορεί να συλλέξει. Επίσης του δίνει την πληροφορία ότι κάθε πόρτα μπορεί να ανοίξει πατώντας το κουμπί ¨E¨ στο πληκτρολόγιο αλλά εφιστά την προσοχή στο γεγονός ότι η κάθε πόρτα ανοίγει με κλειδί. Όταν ο χρήστης δεν έχει το κατάλληλο κλειδί για την πόρτα, η πόρτα δεν θα ανοίξει και το μήνυμα θα εξαφανιστεί, δίνοντάς του έτσι να καταλάβει ότι πρέπει πρώτα να βρει το κατάλληλο κλειδί. Στην περίπτωση των item, η διαδικασία είναι πιο απλή διότι μόλις το μήνυμα εμφανιστεί και πατήσουμε το κουμπί ¨E¨ στο πληκτρολόγιο, θα έχουμε στην κατοχή μας το εκάστοτε κλειδί ή θα επέλθει η αυξομείωση της ζωής του παίχτη από τα medikit-potion. Σημαντικό είναι να αναφέρουμε ότι όταν ο παίχτης βγει από τον collider εμφάνισης μηνύματος που έχουμε ορίσει, το μήνυμα εξαφανίζεται και επανεμφανίζεται

πάλι όταν πλησιάσουμε. Όταν έχουμε χρησιμοποιήσει κάποιο item, το item εξαφανίζεται από το σεντούκι και το μήνυμα παύει πια να εμφανίζεται. Επίσης, όταν η πόρτα έχει ανοίξει επιτυχώς, το μήνυμα , παύει πια να εμφανίζεται.

Ας δούμε όμως και τον κώδικα :

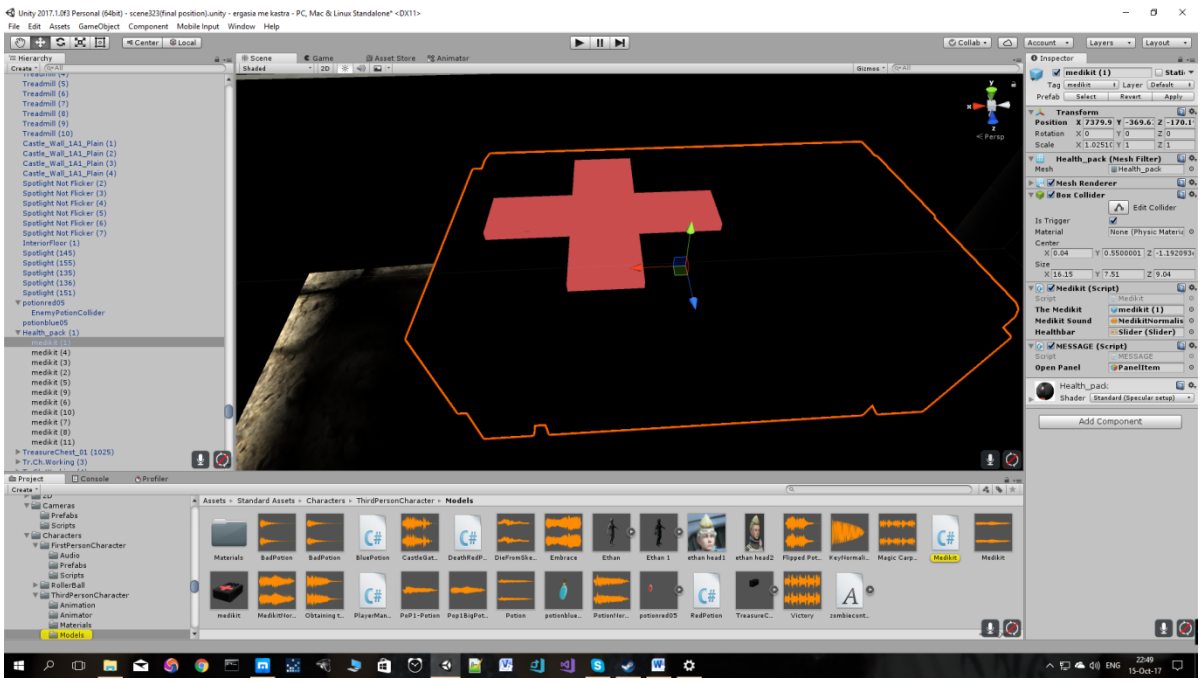
MESSAGE(Script): (Παράρτημα 5.4)

Βλέπουμε λοιπόν στον κώδικα την λειτουργία του μηνύματος. Το ¨openpanel¨ αναφέρεται στο panel που εμπεριέχεται στον canvas που έχουμε φτιάξει για το κάθε μήνυμα. Θα αναλύσουμε στην συνέχεια πως λειτουργούν τα canvas. Από τον κώδικά μας εδώ κρατάμε την εμφάνιση ή όχι του ¨openpanel¨ μηνύματος, ανάλογα αν ο παίχτης βρίσκεται μέσα στον collider ή όχι και η επιστροφή του στην ιεραρχία, αν ο παίχτης πατήσει το κουμπί ¨E¨ ή αν βγει εκτός collider.

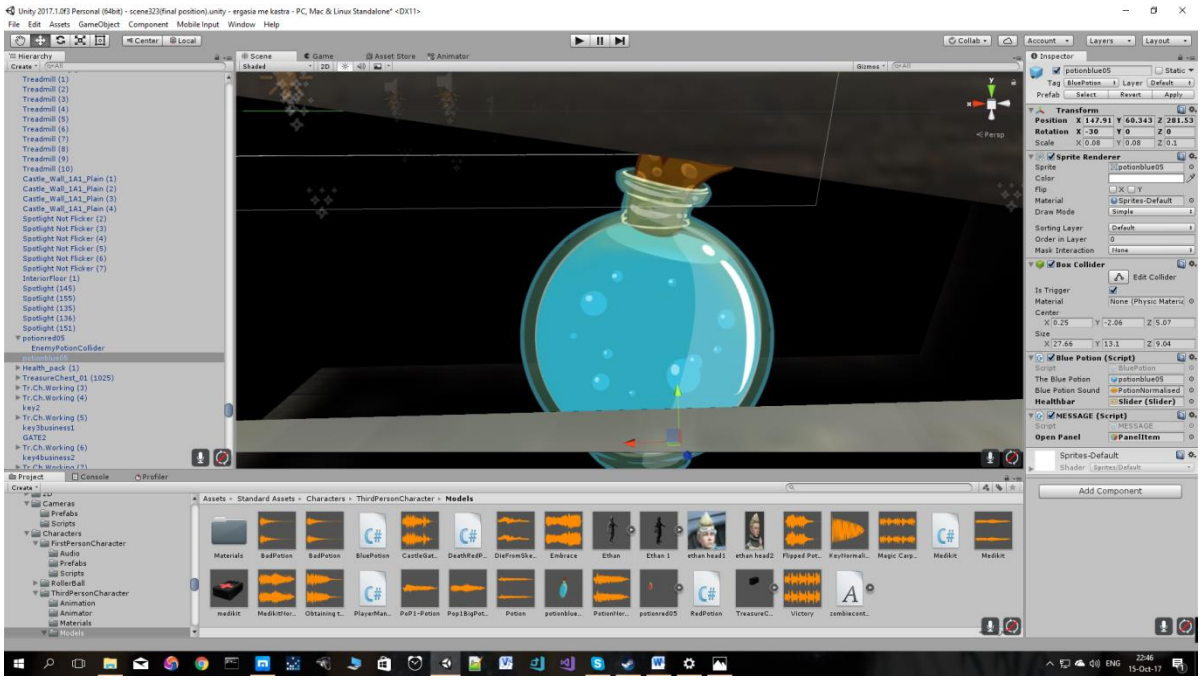
5.5 Λειτουργία Medikit, blue potion και red potion

Ας δούμε αρχικά με εικόνες αυτά τα 3 είδη αυξομοίωσης ζωής του παίχτη.

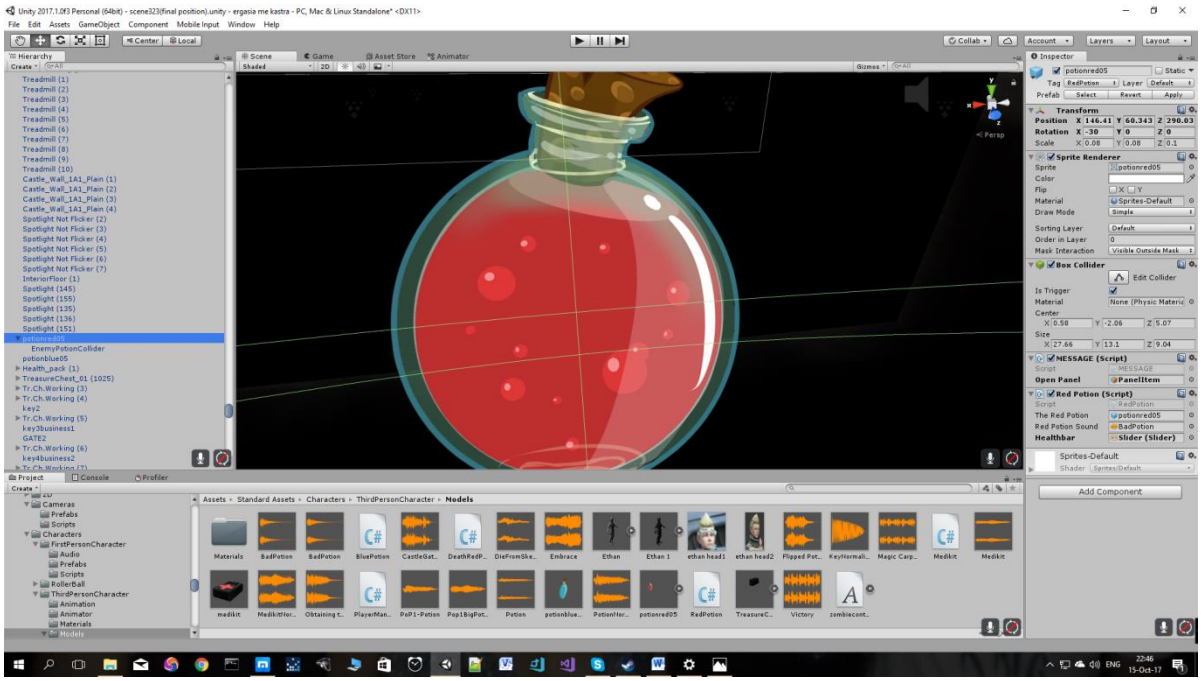
Medikit :



BluePotion:



RedPotion:



Παρατηρούμε ότι το μήνυμα MESSAGE(Script) υπάρχει σε όλα. Παρατηρούμε ότι και στα τρία υπάρχει το αντίστοιχο tag, που κατηγοριοποιεί το καθένα. Αλλά για μεγαλύτερη ανάλυση, πρέπει να δούμε τα script.

Medikit(Script): (Παράρτημα 5.5)

Βλέπουμε λοιπόν στον κώδικά μας ότι έχουμε ορίσει με Boolean την είσοδο στον collider ως : playerNextToMedikit. Οπότε όταν αυτό είναι αληθές και ταυτόχρονα πατήσουμε το κουμπί ¨E¨ στο πληκτρολόγιο, το medikit εξαφανίζεται, το μήνυμα εξαφανίζεται από το MESSAGE(Script), παίζει ο ήχος που έχουμε ορίσει μια φορά και τέλος η ζωή μας γεμίζει στο 100%, επειδή μπορούμε και τραβάμε το healthbar.value από το PlayerManager(Script).

Ακριβώς την ίδια λογική ακολουθεί ο κώδικας του bluepotion οπότε μπορούμε να τον παραλείψουμε στην ανάλυσή μας. Η μόνη διαφοροποίηση είναι ότι η ζωή που δίνει στον παίχτη είναι += 40.

Στην περίπτωση όμως του redpotion έχουμε διπλό script. Το πρώτο είναι ακριβώς σαν το script του medikit και του bluepotion αλλά το δεύτερο script είναι στις λειτουργίες του παίχτη, αναφερόμενο ως PotionHit(Script). Ας τα δούμε αναλυτικά:

RedPotion(Script): (Παράρτημα 5.5)

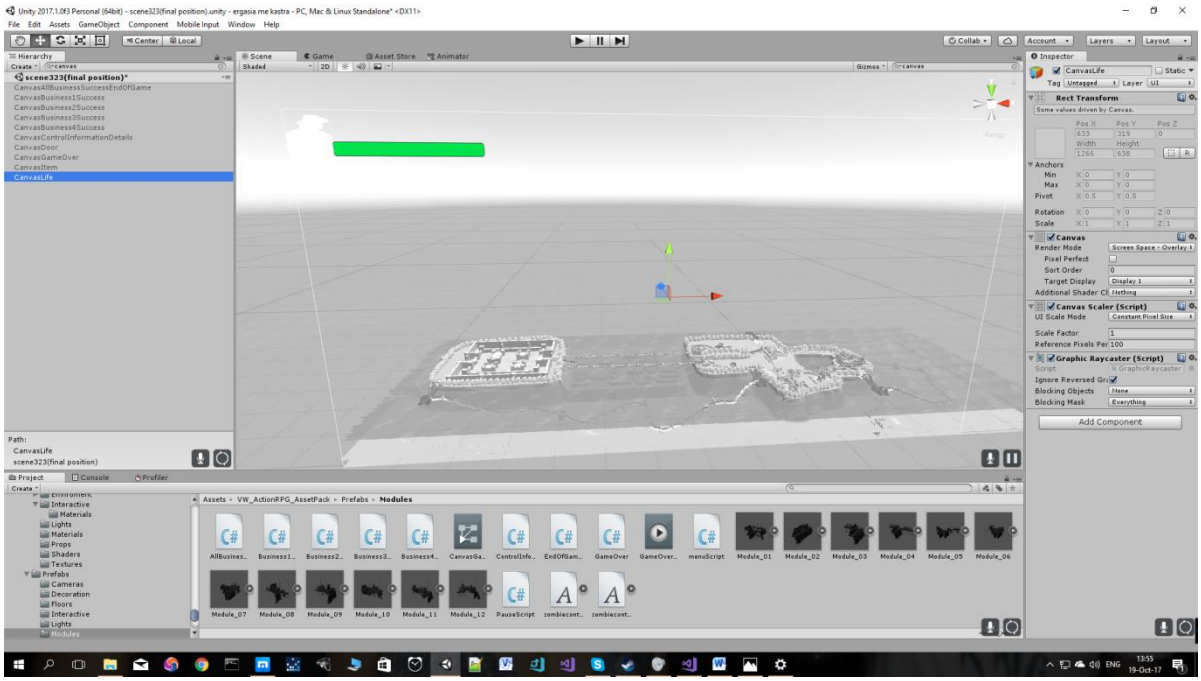
Θέλαμε εξαρχής τα redpotion να αποτελούν δηλητήριο για τον παίχτη και να του αφαιρούν το 40% της ζωής του. Αλλά παράλληλα θέλαμε όχι μόνο να φαίνεται οπτικά η αφαίρεση ζωής του αλλα να αντιδράει σαν να δέχεται χτύπημα. Επίσης, αν η ζωή του πριν πιεί το redpotion είναι μικρότερη του 40%, θέλαμε ο παίχτης να χάνει. Και όλα αυτά τα επιτύχαμε. Στο RedPotion(Script) εύκολα παρατηρούμε ότι η μεταβλητή απώλειας ζωής έχει ρυθμιστεί στο 20% ,`healthbar.value -= 20;` αλλά το υπόλοιπο 20% αφαιρείται ταυτόχρονα από το PotionHit(Script). Οπότε με αυτό τον τρόπο, η συνολική απώλεια ζωής του παίχτη είναι 40%.

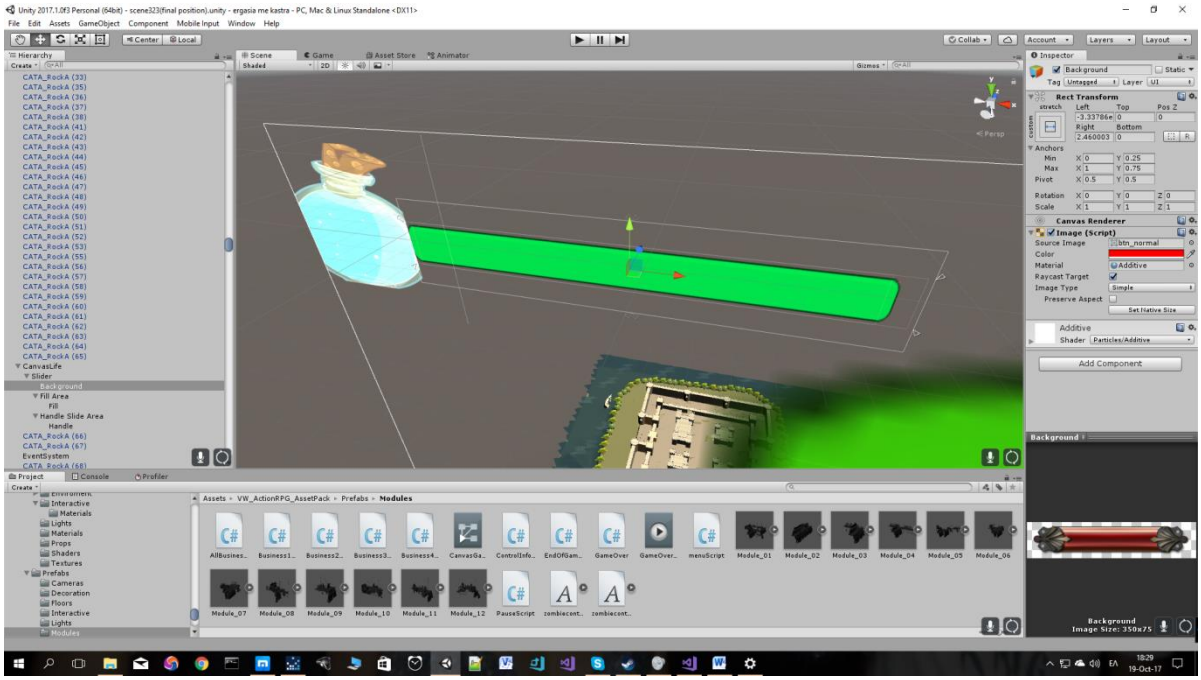
PotionHit(Script): (Παράρτημα 5.5)

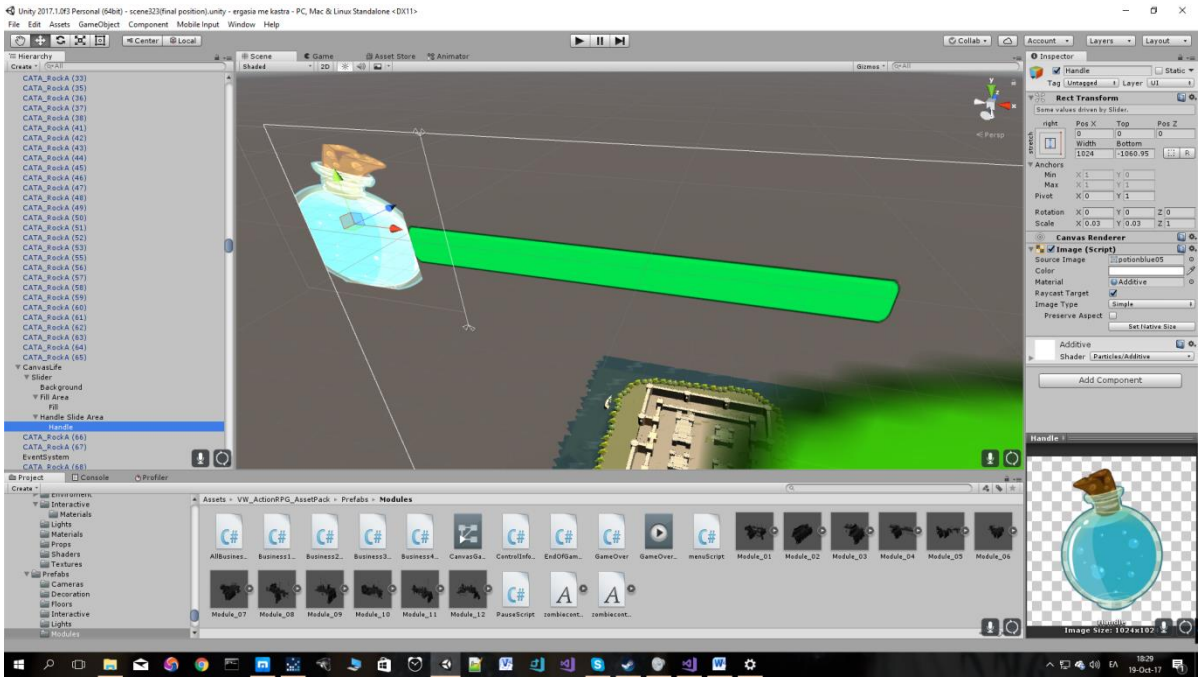
Βλέπουμε λοιπόν σε αυτό το script ότι λειτουργήσαμε με την λογική του ¨enemyweaponcollider¨ που περιγράψαμε παραπάνω, στις παγίδες και στους εχθρούς. Στην περίπτωσή μας εδώ δημιουργήσαμε έναν αποκλειστικό collider για το redpotion, με tag : ¨EnemyPotionCollider¨ ο οποίος ενεργοποιείται και αφαιρεί το υπόλοιπο 20% από τον παίχτη την στιγμή που θα πατήσει το πλήκτρο ¨E¨. Τότε ενεργοποιείται ο μηχανισμός του GetHit, παίζει το animation του χτυπήματος αλλά και τον κατάλληλο ήχο. Εάν η ζωή(slider) του παίχτη είναι μικρότερη απο 40% πριν κάνει χρήση του redpotion, τότε η ζωή του μεταβαίνει σε <0 και ενεργοποιείται το PlayerManager(Script), καθώς και το GameOver(Script), δηλαδή όλος ο μηχανισμός που δείχνει τον παίχτη να χάνει.

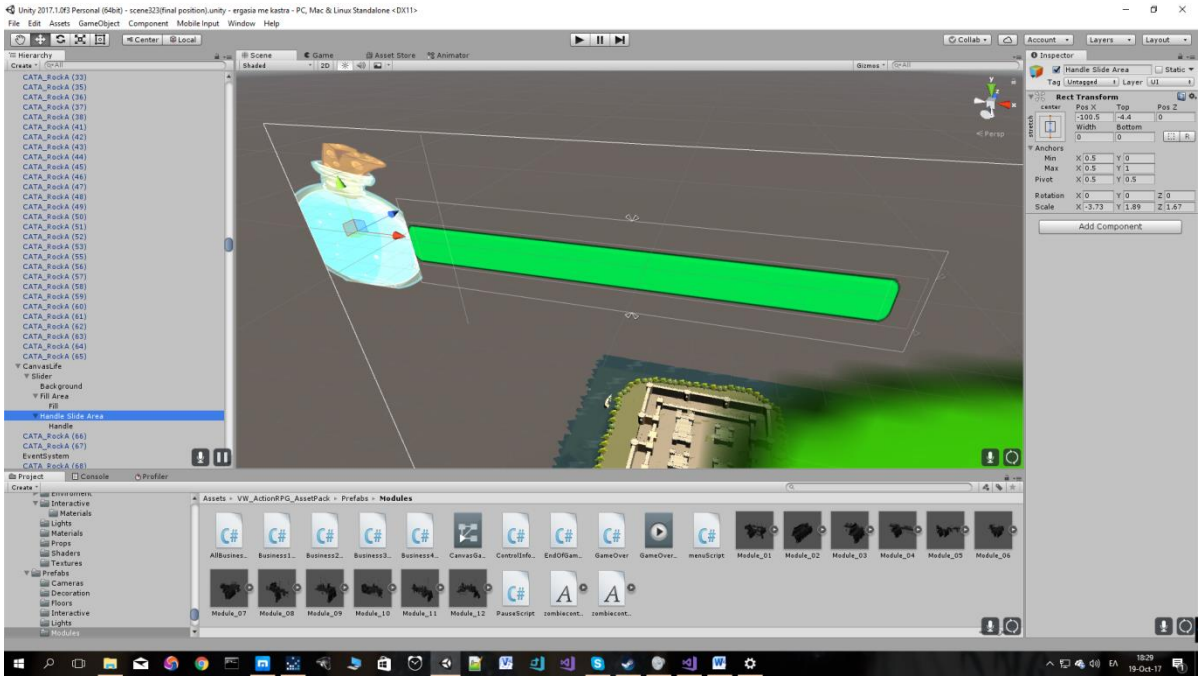
5.6 Λειτουργία Canvas και Panels

Ας δούμε τις εικόνες μία προς μία εξηγώντας την επεξεργασία και την λειτουργία των canvas, των panel και του text :







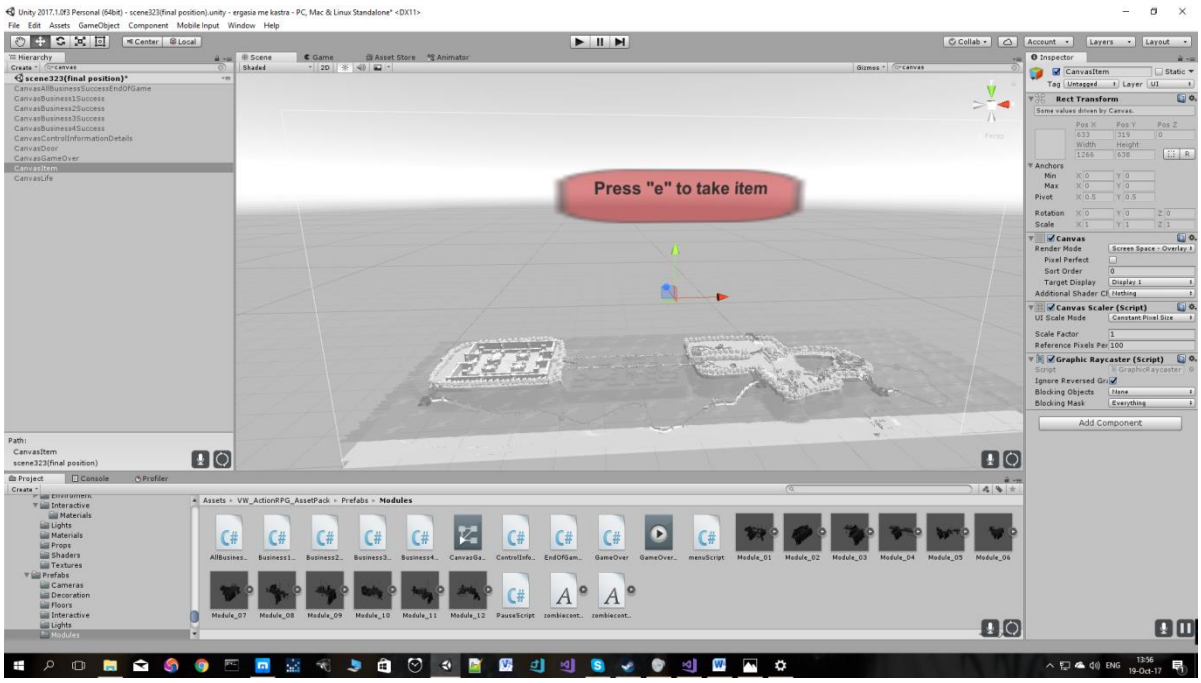


Το κάθε canvas εμπεριέχει σε υποφάκελο, το openpanel, ενώ μέσα σε αυτό εμπεριέχεται το text. Κάνοντας απλές ρυθμίσεις μπορούμε να προσαρμόσουμε την θέση του μηνύματος που θέλουμε να εμφανίσουμε αλλά και το είδος του μηνύματος. Αυτό το μήνυμα μπορεί να είναι text, μπορεί να είναι εικόνα ή μπορεί να είναι animation. Στις εικόνες μας παραπάνω βλέπουμε μια πιο πολύπλοκη διαδικασία δημιουργίας του CanvasLife του παίχτη. Εδώ χρησιμοποιήσαμε δύο εικόνες. Στην πρώτη

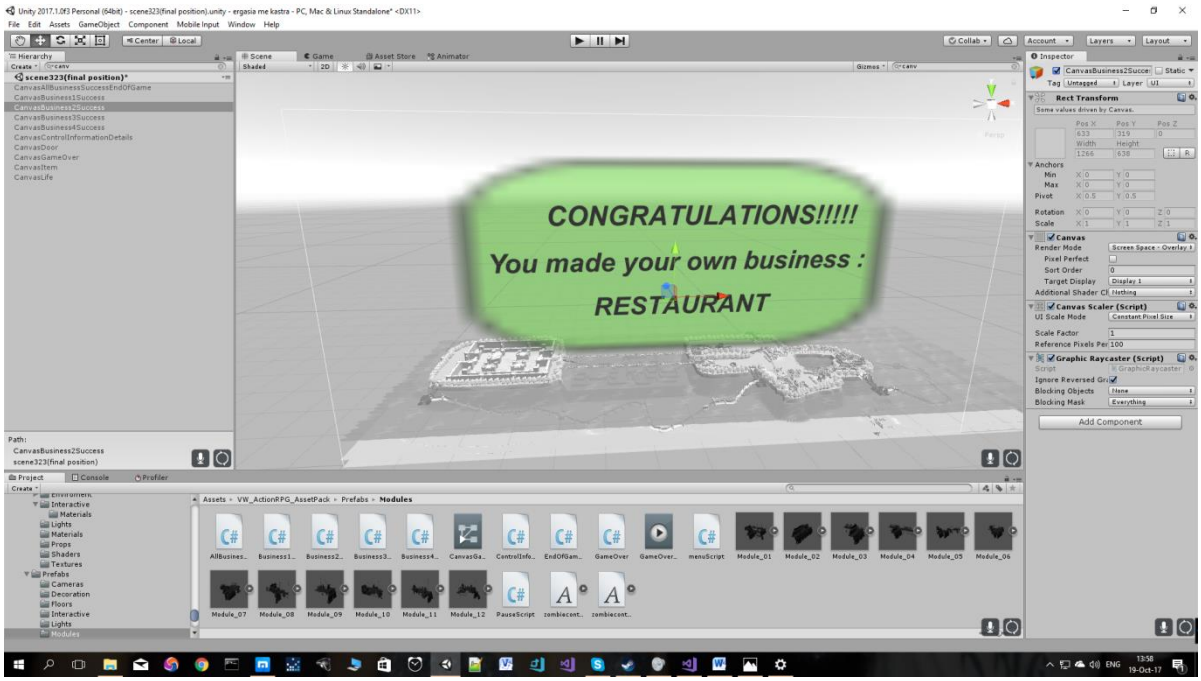
προσαρμόσαμε μια διάφανη εικόνα με κόκκινο background η οποία επικαλύπτεται με πράσινο χρώμα προκειμένου να μας οπτικοποιήσει την απώλεια ζωής. Στην δεύτερη, πάλι με διαφανή τρόπο, χρησιμοποιήσαμε την εικόνα του bluepotion, απλά σαν ένδειξη ζωής.

Με πιο απλό τρόπο δημιουργείται το canvas των πυλών-πορτών καθώς και των item. Εκεί απλά έχουμε μέσα στο canvas, το openpanel και μεσα σε αυτό το text. Με απλές ρυθμίσεις προσαρμόζουμε την θέση του μηνύματος, τα χρώματα και το υλικό που θα έχει ως background.

CanvasItem :

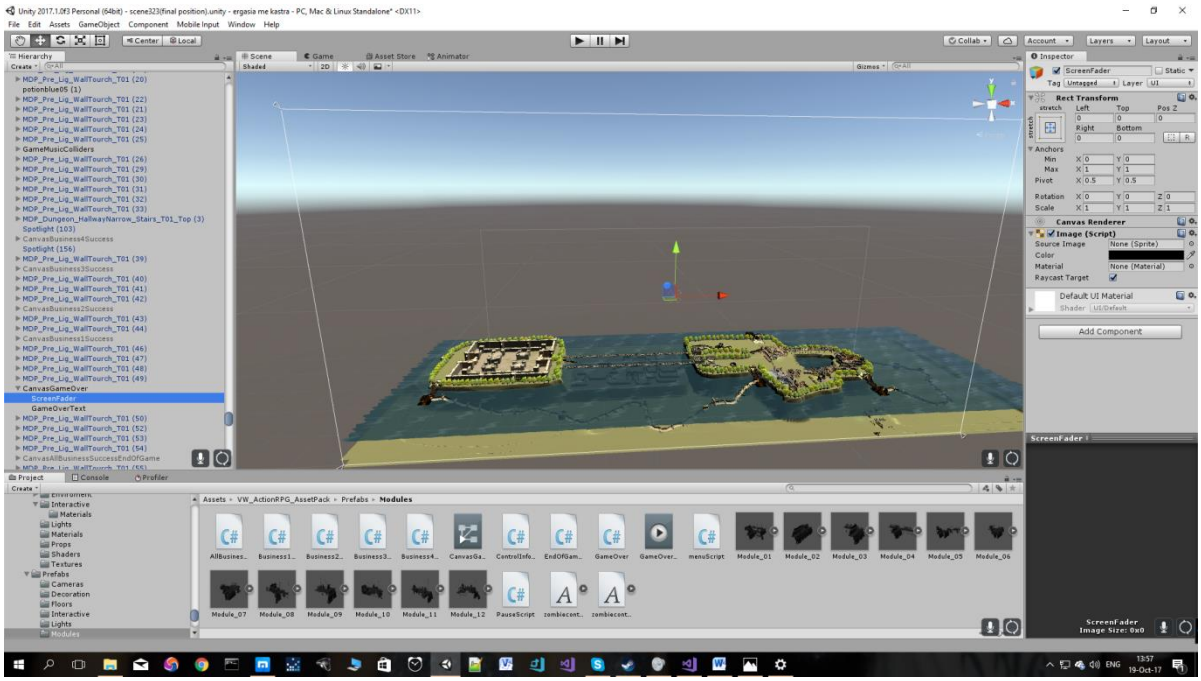


CanvasBusiness2Success :

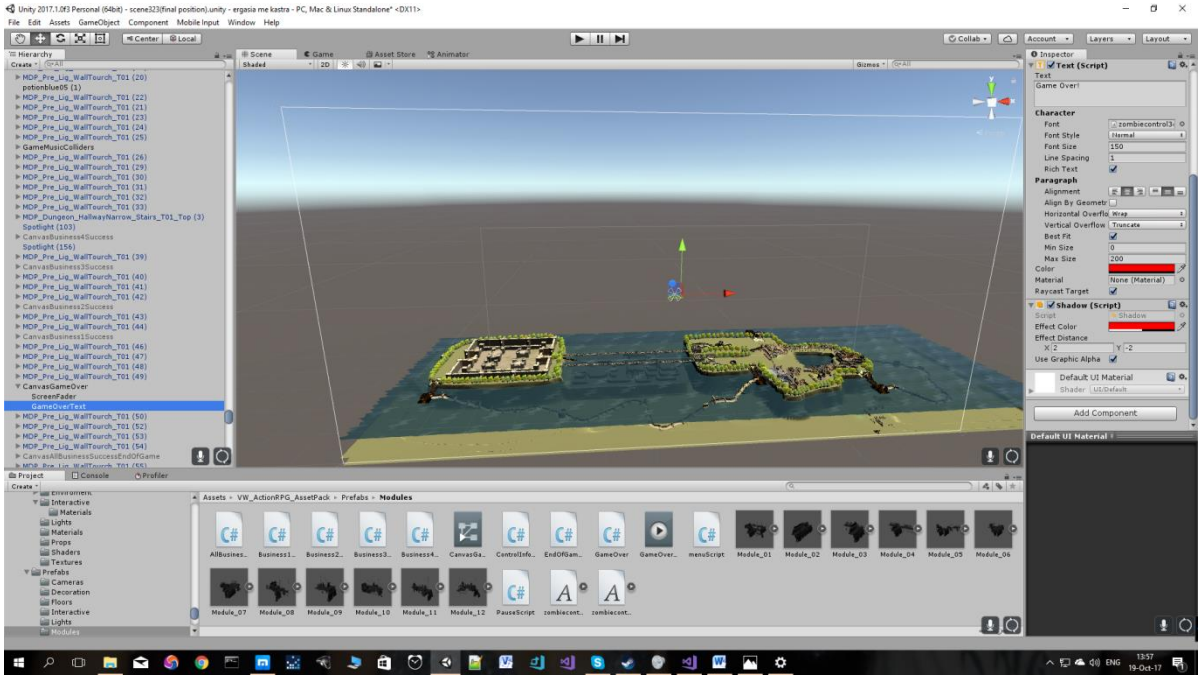


Σε αυτό το σημείο μπορούμε να δούμε την λειτουργία του GameOver που είχαμε προαναφέρει. Η διαφοροποίηση από τα προηγούμενα canvas έγκειται στο γεγονός ότι στην περίπτωση αυτή έχουμε animation. Βλέπουμε και από τις εικόνες παρακάτω ότι προσαρμόσαμε το canvas σε ολόκληρη την οθόνη, δίνοντάς του μαύρο φόντο και δώσαμε κόκκινο χρώμα στο text που μας ενημερώνει για το gameover. Εδώ χρησιμοποιήσαμε και το Shadow(Script) για να δώσουμε σκίαση.

CanvasGameOver-ScreenFader :



CanvasGameOver-GameOverText :



Ας δούμε τον κώδικα :

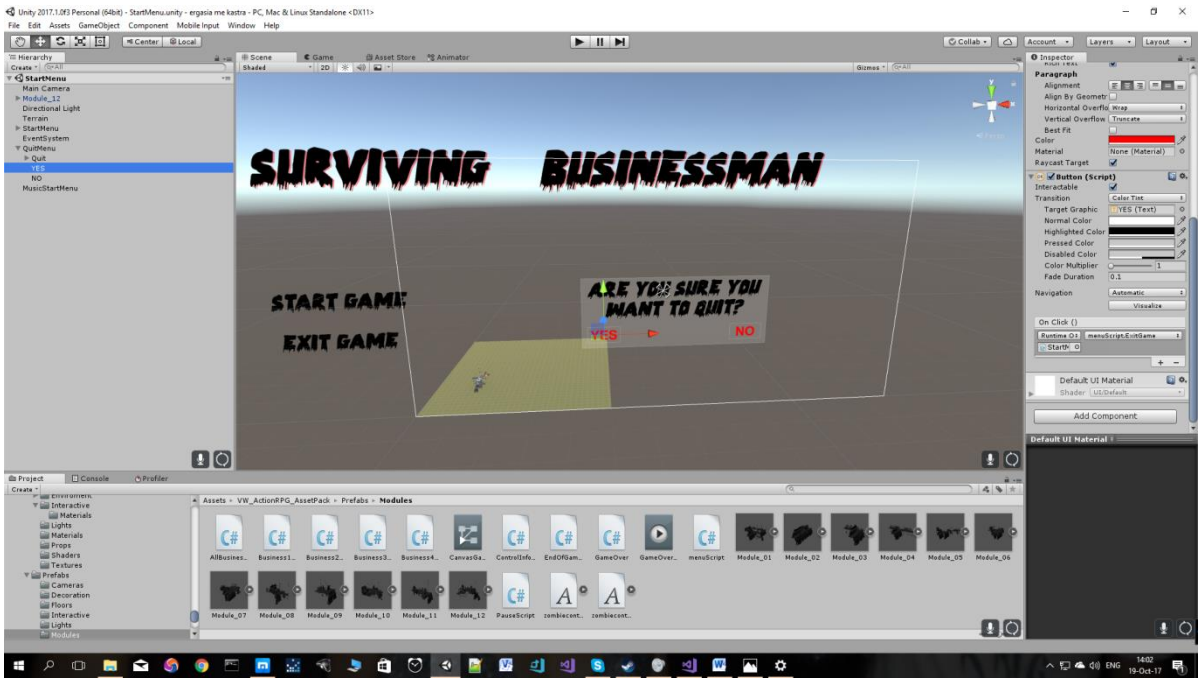
GameOver(Script): (Παράρτημα 5.6)

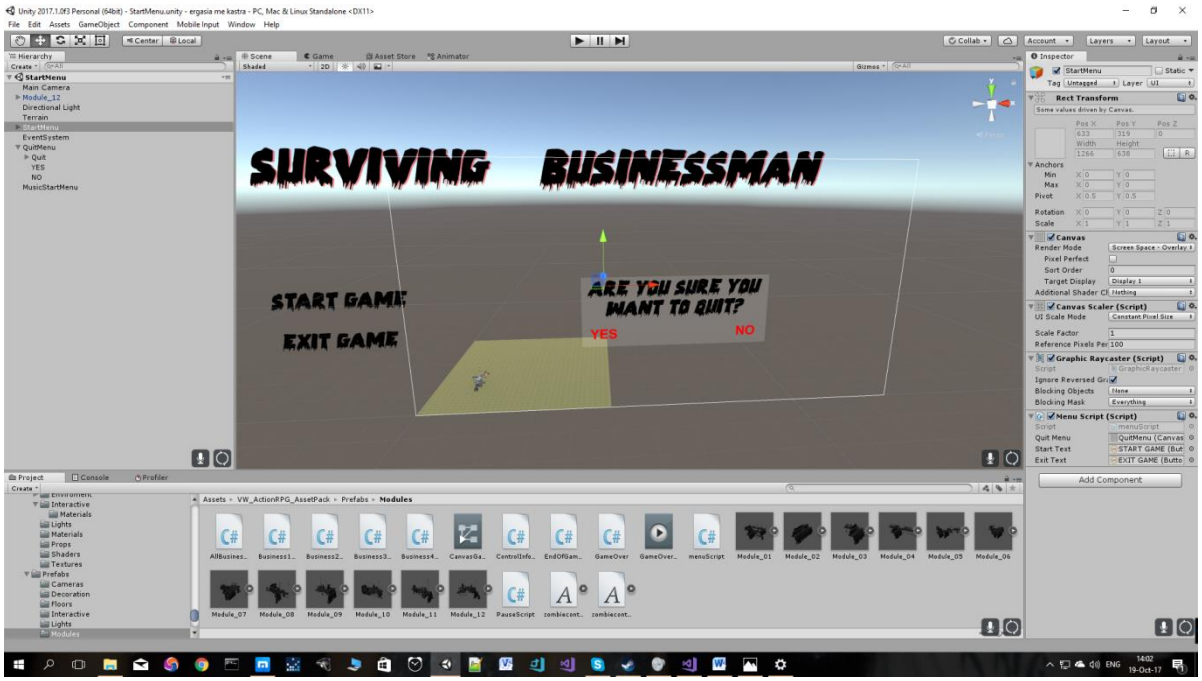
Πριν αναλύσουμε τον κώδικα, πρέπει να αναφέρουμε ότι μειώσαμε την ταχύτητα αναπαραγωγής του animation GameOver, καθώς και θέσαμε delay σε μουσική και animation. Ο λόγος είναι ότι θέλαμε πρώτα να παίξει το Death animation του παίχτη μαζί με την μουσική και ακριβώς στο τέλος της μουσικής αυτής, να ξεκινήσει όλος ο μηχανισμός του GameOver. Επίσης θέλαμε σχεδόν ελάχιστα πριν ολοκληρωθεί η μουσική του GameOver να γίνεται μετάβαση στο StartMenu και το επιτύχαμε αυτό με ακόμα ένα delay.

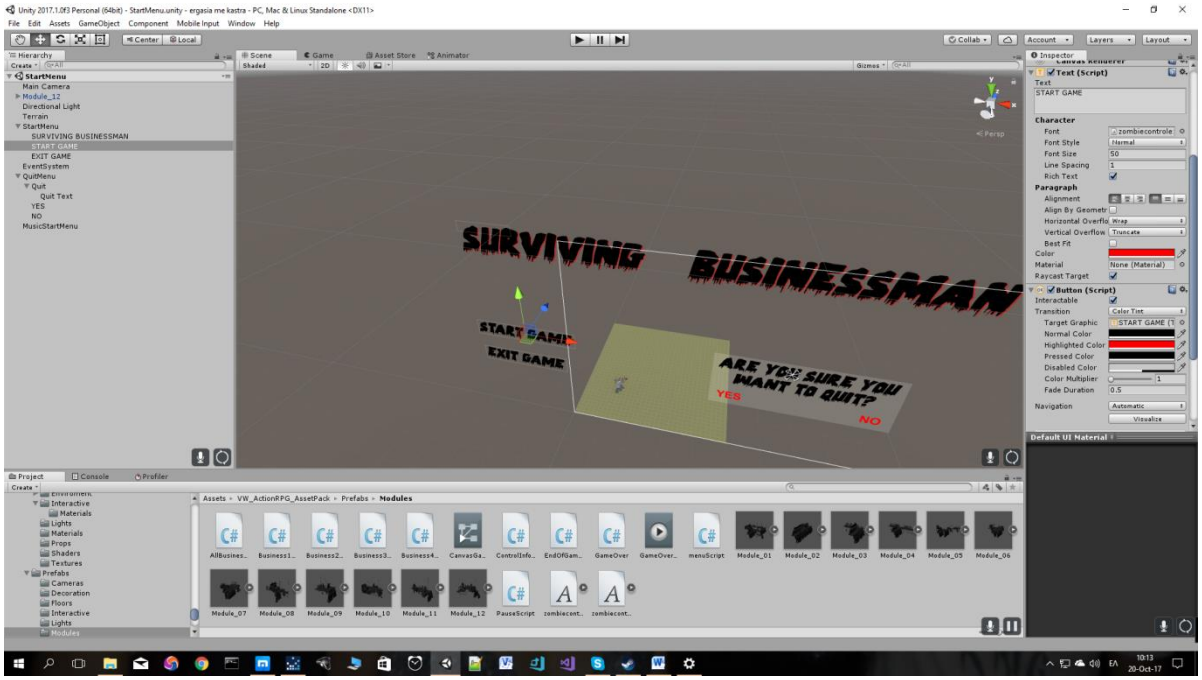
Στον κώδικά μας βλέπουμε ότι χάνοντας ο παίχτης μπαίνουνε σε λειτουργία και τα τρία delay και με if μέσα στο update ενεργοποιούμε την εμφάνιση του canvas, την εκκίνηση της μουσικής και την μετάβαση στο StartMenu. Στην εικόνα που πρωτοαναφέραμε το GameOver(Script) μπορούμε να διακρίνουμε τις τιμές που δώσαμε και στα τρία delay.

5.7 Λειτουργία StartMenu

Ας δούμε τις τρείς εικόνες εξαρχής :







Από τις παραπάνω εικόνες μπορούμε να δούμε την χρησιμοποίηση canvas για την δημιουργία του StartMenu και του QuitMenu, απλού Text για την δημιουργία των τίτλων ¨SURVIVING BUSINESSMAN¨ και ¨QuitText¨, ενώ Text και Button μαζί για την δημιουργία ¨START GAME¨, ¨EXIT GAME¨, ¨YES¨, ¨NO¨. Παρατηρούμε την χρησιμοποίηση του MenuScript (Script) το οποίο προσθέτει τις απαραίτητες λειτουργίες στο user interface μας και εκτελεί τις κατάλληλες εντολές μας προκειμένου να ξεκινήσουμε το παιχνίδι ή να κάνουμε έξοδο από αυτό. Στην εικόνα μας με το κουμπί

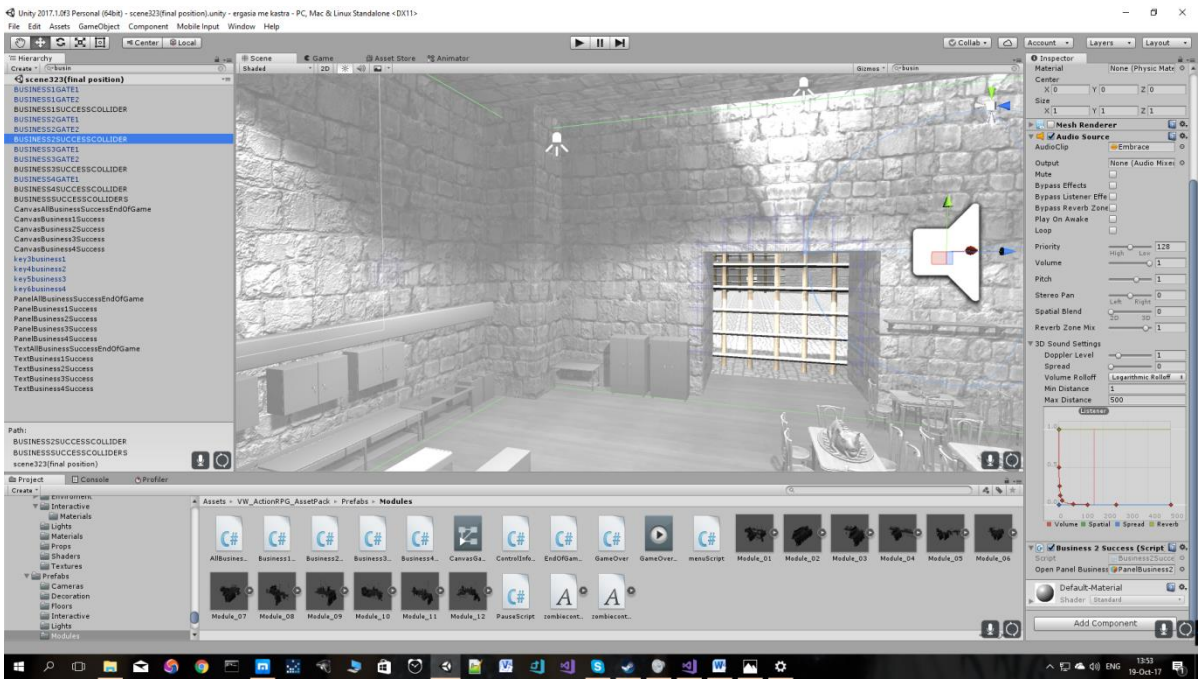
¨YES¨ παρατηρούμε στα δεξιά τις επιλογές που κάναμε για την κατάσταση ¨On Click¨. Επιλέξαμε την έξοδο από το παιχνίδι. Με παρόμοιο τρόπο ρυθμίσαμε και τα υπόλοιπα buttons. Ας δούμε και τον κώδικα :

MenuScript(Script): (Παράρτημα 5.7)

Στον κώδικά μας παρατηρούμε με απλή χρήση Boolean(true-false), την ενεργοποίηση ή απενεργοποίηση του υπομενού μας, την εκκίνηση του παιχνιδιού αλλά και την έξοδο από αυτό.

5.8 Λειτουργία Επιχειρήσεων στο παιχνίδι

Ας δούμε την εικόνα :



Από την εικόνα παραπάνω μπορούμε να διακρίνουμε τον collider μέσα σε κάθε επιχείρηση. Μόλις ο παίχτης μας εισέλθει στον collider της κάθε επιχείρησης τότε ενεργοποιείται ένα συγκεκριμένο script για την κάθε μία αλλά ταυτόχρονα ενημερώνεται και το ¨AllBusinessSuccess(Script)¨ του παίχτη μας. Αφού η εικόνα μας απεικονίζει την δεύτερη επιχείρηση, ας δούμε αναλυτικά τον κώδικά της :

Business2Success(Script): (Παράρτημα 5.8)

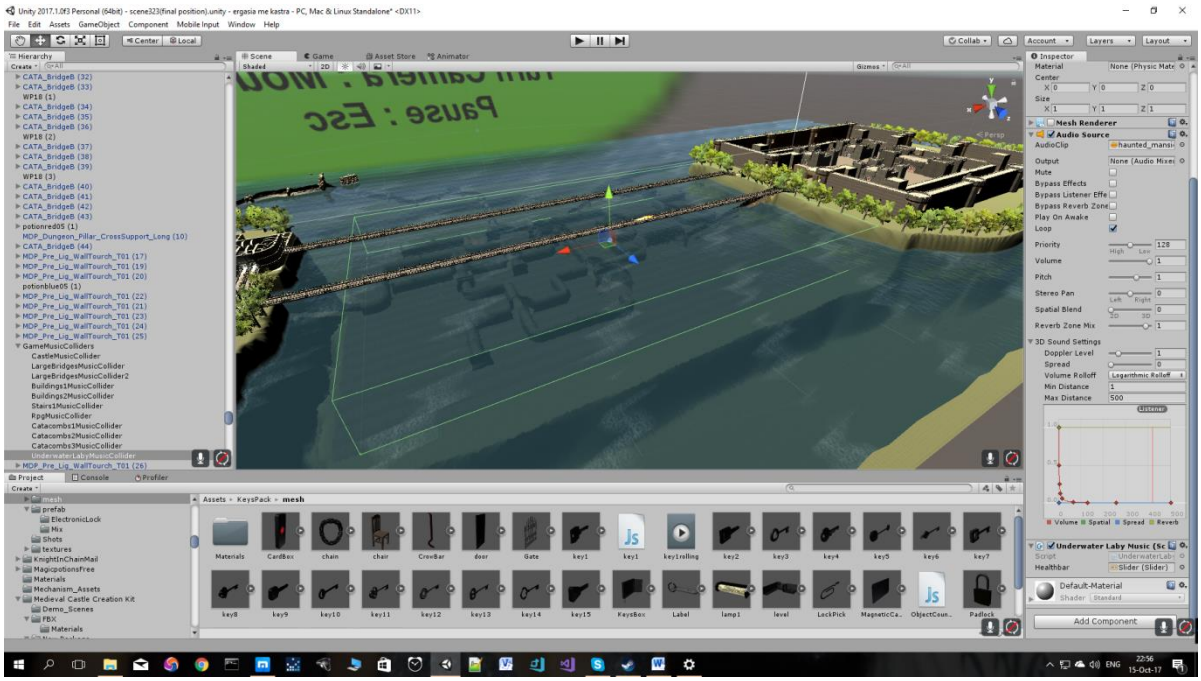
Βλέπουμε λοιπόν από τον κώδικά μας την εμφάνιση του panel που θέλουμε για την επιχείρησή μας, όταν ο παίχτης εισέρχεται με ταυτόχρονη έναρξη ήχου. Θέλαμε όταν ο παίχτης βγαίνει από τον collider και ξαναμπαίνει στην επιχείρηση να μην εμφανίζεται πια το μήνυμα, ούτε να ξαναπαίζει η μουσική. Γι αυτό, κατά την έξοδο του παίχτη από την επιχείρηση, κάνουμε Destroy(gameObject). Ας δούμε τι γίνεται με το ¨AllBusinessSuccess (Script)¨ του παίχτη :

AllBusinessSuccess(Script): (Παράρτημα 5.8)

Βλέπουμε λοιπόν από τον κώδικά μας ότι με την είσοδο του παίχτη μας στον collider με tag: `BUSINESS2SUCCESSCOLLIDER`, θεωρείται: `BUSINESS2HASOPENED = true;` . Έτσι ο κώδικάς μας καταλαβαίνει πότε έχουν ενεργοποιηθεί και οι τέσσερις επιχειρήσεις, ανεξαρτήτου σειράς. Ρυθμίσαμε τον κώδικα προκειμένου να δείχνει το μήνυμα επιτυχίας όλων των επιχειρήσεων, όταν ο παίχτης εξέρχεται από την τελευταία. Έπειτα χρησιμοποιούμε ένα delay, το οποίο σε χρόνο που εμείς επιλέγουμε, κάνει μετάβαση στην τελική σκηνή τερματισμού : `SceneManager.LoadScene(2);`

5.9 Λειτουργία Βασικής και Εφεδρικής Μουσικής

Στην παρακάτω εικόνα μπορούμε να διακρίνουμε τον collider που εμπεριέχει ολόκληρο τον λαβύρινθο. Δώσαμε συγκεκριμένο tag σε κάθε collider μουσικής έτσι ώστε μόλις ο παίχτης εισέρχεται, η συγκεκριμένη μουσική να ξεκινά.

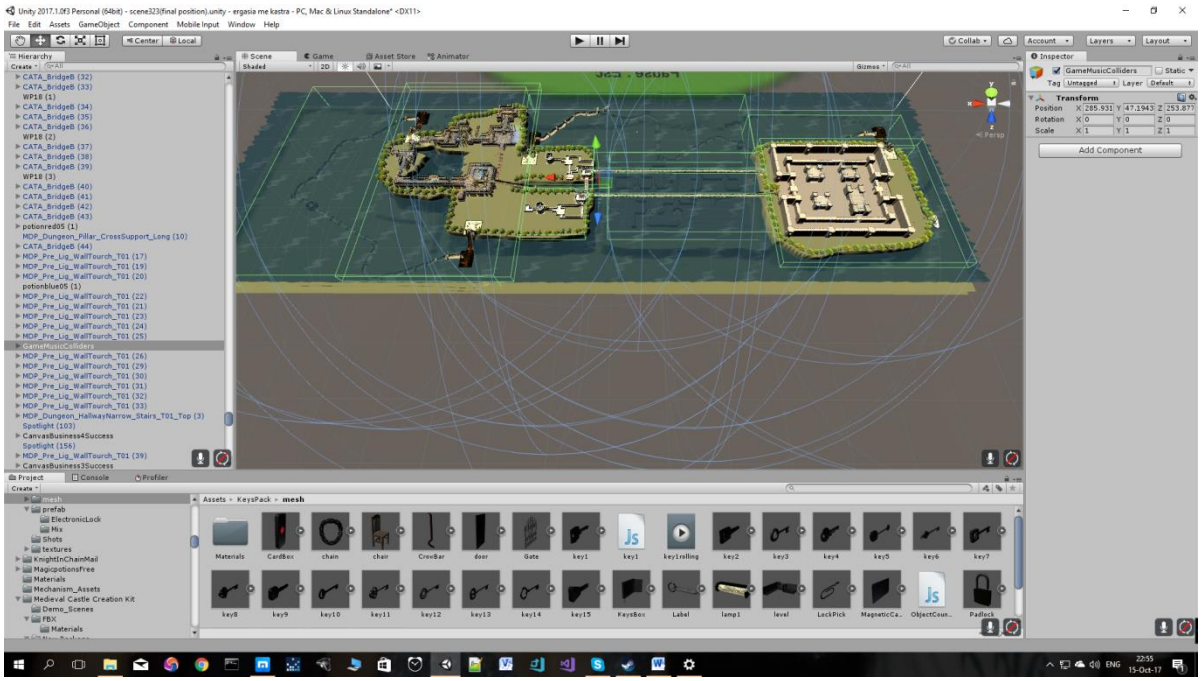


Ο κώδικάς μας για την μουσική του λαβύρινθου είναι ο εξής :

UnderwaterLabyMusic(Script): (Παράρτημα 5.9)

Βλέπουμε από τον κώδικά μας την εκκίνηση της μουσικής και την διακοπή της όποτε ο παίχτης εξέρχεται ή εισέρχεται στον συγκεκριμένο collider. Σημαντικό είναι να αναφέρουμε την χρησιμοποίηση του slider του παίχτη και εδώ. Θέλαμε η μουσική σε οποιοδήποτε σημείο του παιχνιδιού να σταματά όταν ο παίχτης χάνει, ακριβώς όπως κάναμε με την απενεργοποίηση των ήχων σε παγίδες και εχθρούς. Και αυτό διότι ενεργοποιούνται άλλα script με μουσική, όπως προαναφέραμε.

Ας δούμε όμως και όλους τους colliders μουσικής μαζί :



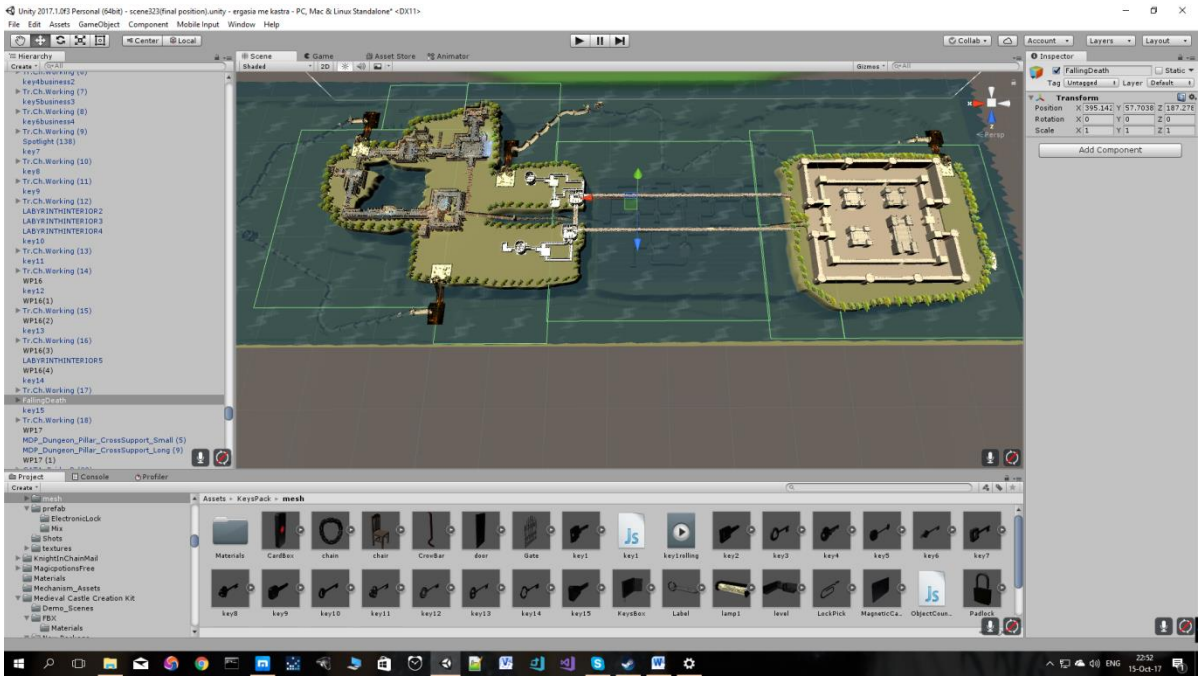
Με παρόμοιο τρόπο και παρόμοιο κώδικα εμπλουτίστηκαν και όλες οι άλλες περιοχές του παιχνιδιού. Κατά τον έλεγχο που κάναμε όσον αφορά την ορθή μετάβαση από collider μουσικής σε collider μουσικής, προσαρμόσαμε την ένταση του κάθε κομματιού, έτσι ώστε να ακούγεται ομαλά, χωρίς αυξομειώσεις στην ένταση. Στο σημείο αυτό μπορούμε να αναφέρουμε το BackupMusic(Script) του παίχτη, το οποίο λειτουργεί και ενημερώνεται παράλληλα με κάθε script μουσικής :

BackupMusic(Script): (Παράρτημα 5.9)

Βλέπουμε από τον κώδικά μας ότι οποιαδήποτε στιγμή ο παίχτης βρίσκεται μέσα σε collider μουσικής, τα υπόλοιπα script μουσικής αναλαμβάνουν την αναπαραγωγή και όχι αυτό. Γιαυτό έχουμε και το `audioSource.Stop();` .Όταν όμως το script μας εντοπίσει ότι ο παίχτης δεν βρίσκεται μέσα σε κάποιο collider μουσικής, ούτε έχει έρθει σε επαφή με collider πτώσης, τότε ενεργοποιεί την μουσική που εμείς επιλέγουμε, δίνοντάς μας την δυνατότητα επιλογής έντασης και delay προτού ενεργοποιηθεί. Όπως παρατηρούμε και από τον κώδικα, η μουσική αυτή θα παίζει με loop, ακριβώς όπως και όλες οι υπόλοιπες.

5.10 Λειτουργία Πτώσεων και Σύνδεση με GameOver

Στην εικόνα παρακάτω βλέπουμε όλους τους colliders μαζί, οι οποίοι καθορίζουν την πτώση του παίχτη στο κενό και ενεργοποιούν το GameOver.



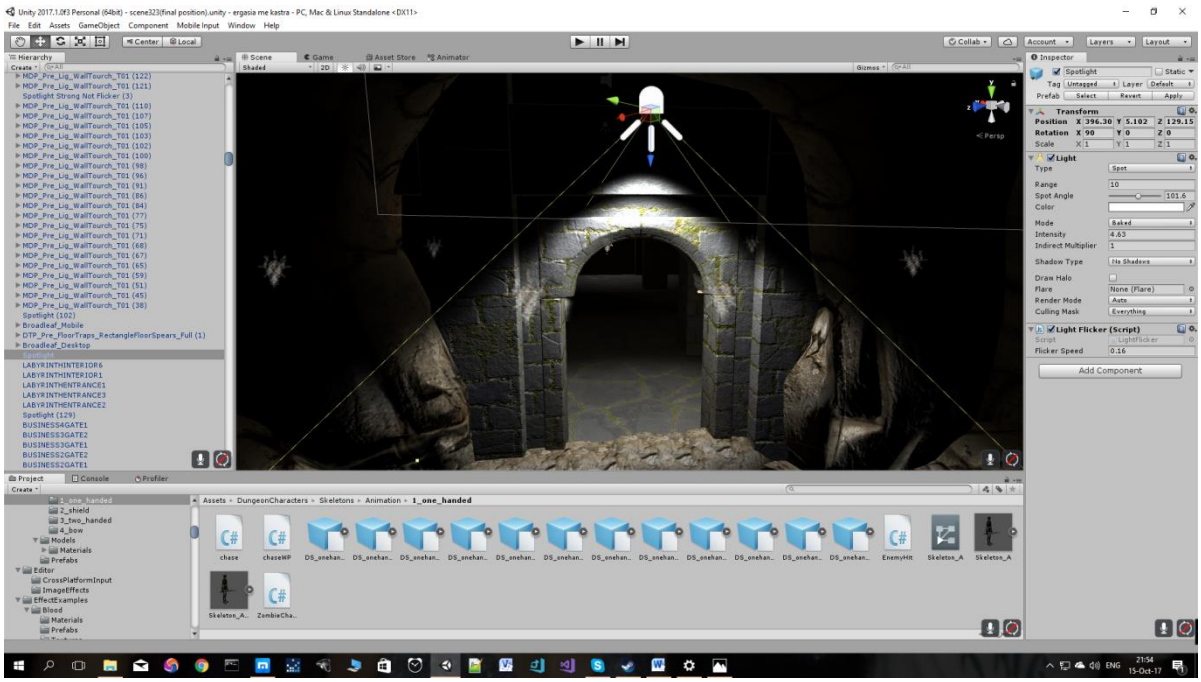
Σε αυτό το σημείο θα αναλύσουμε τον κώδικα FallingDeath(Script), που βρίσκεται στον στον παίχτη μας :

FallingDeath(Script): (Παράρτημα 5.10)

Από τον κώδικά μας βλέπουμε λοιπόν ότι οποιαδήποτε στιγμή ο παίχτης μας έρθει σε επαφή με κάποιον collider πτώσης, δηλαδή collider με tag : `FallingEdgeCollider` ,τότε του αφαιρείται 100% ζωής και ενεργοποιείται όλος ο μηχανισμός που δείχνει τον παίχτη να χάνει, όπως προαναφέραμε.

5.11 Άλλες Λειτουργίες

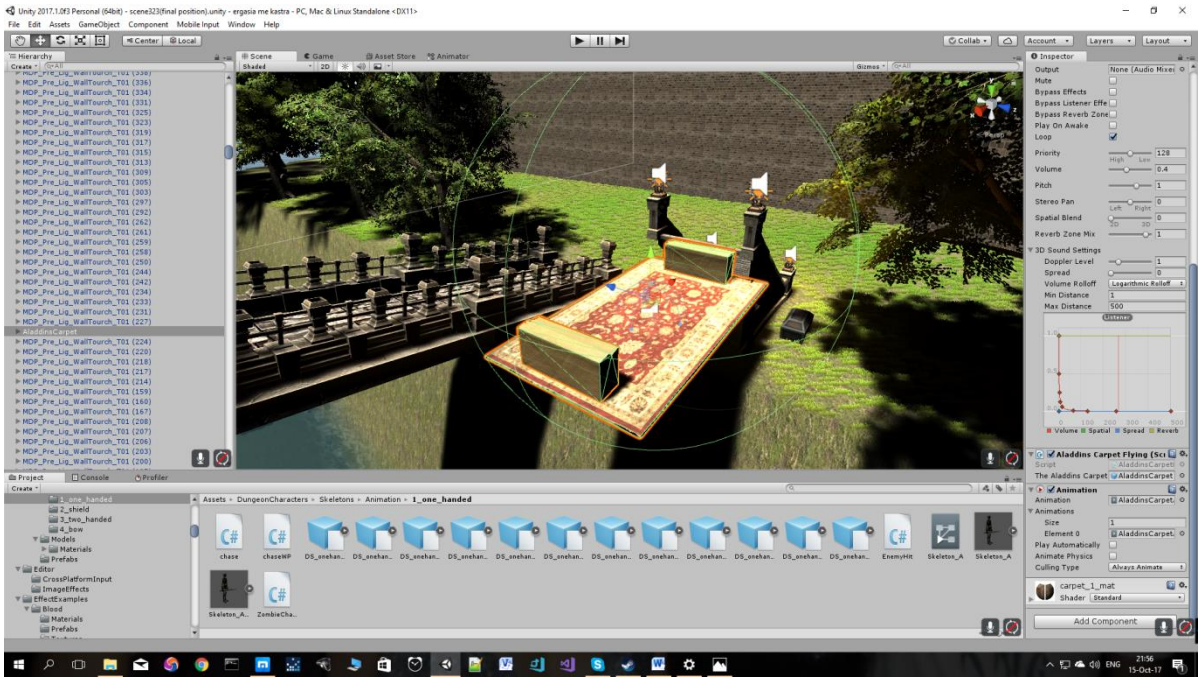
Flicker Spotlights :

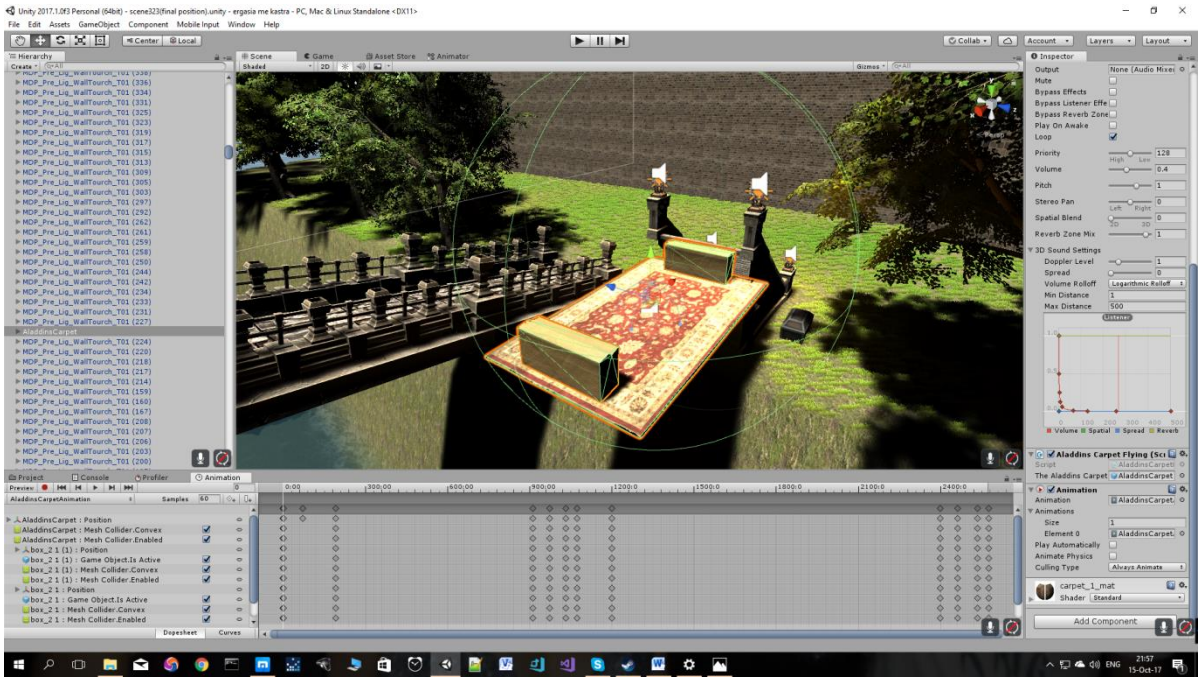


LightFlicker(Script): (Παράρτημα 5.11)

Με τον παραπάνω κώδικα, κάναμε τα spotlight να τρεμοπαίζουν.

Aladdin'sCarpet :



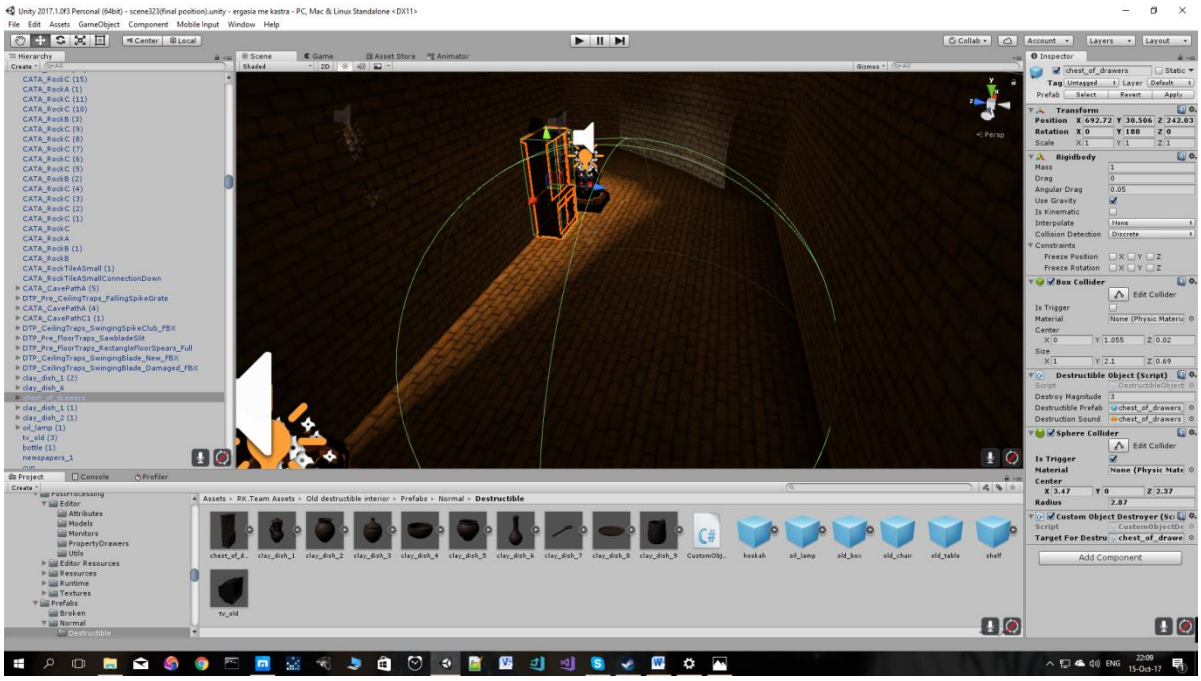


Στις παραπάνω εικόνες βλέπουμε το μαγικό χαλί και τον τρόπο που δημιουργήσαμε το animation.

AladdinsCarpetFlying(Script): (Παράρτημα 5.11)

Στο συγκεκριμένο script παρατηρούμε τον κώδικα που χρησιμοποιήθηκε για το μαγικό χαλί.

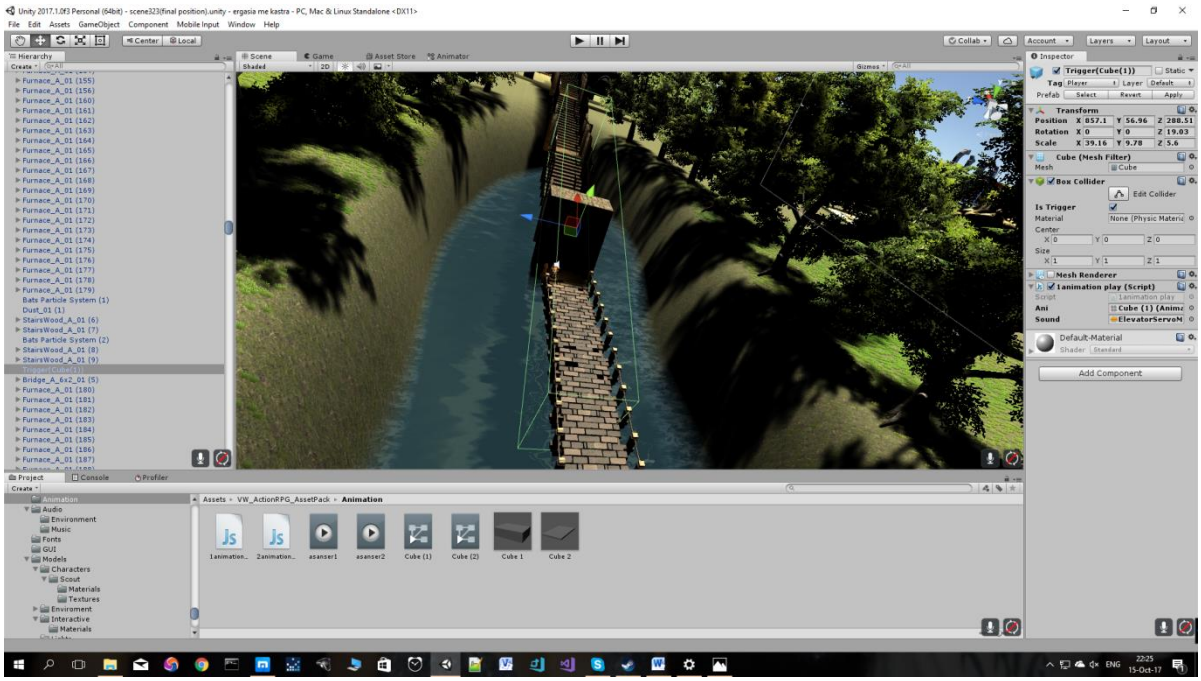
Destructible Objects :



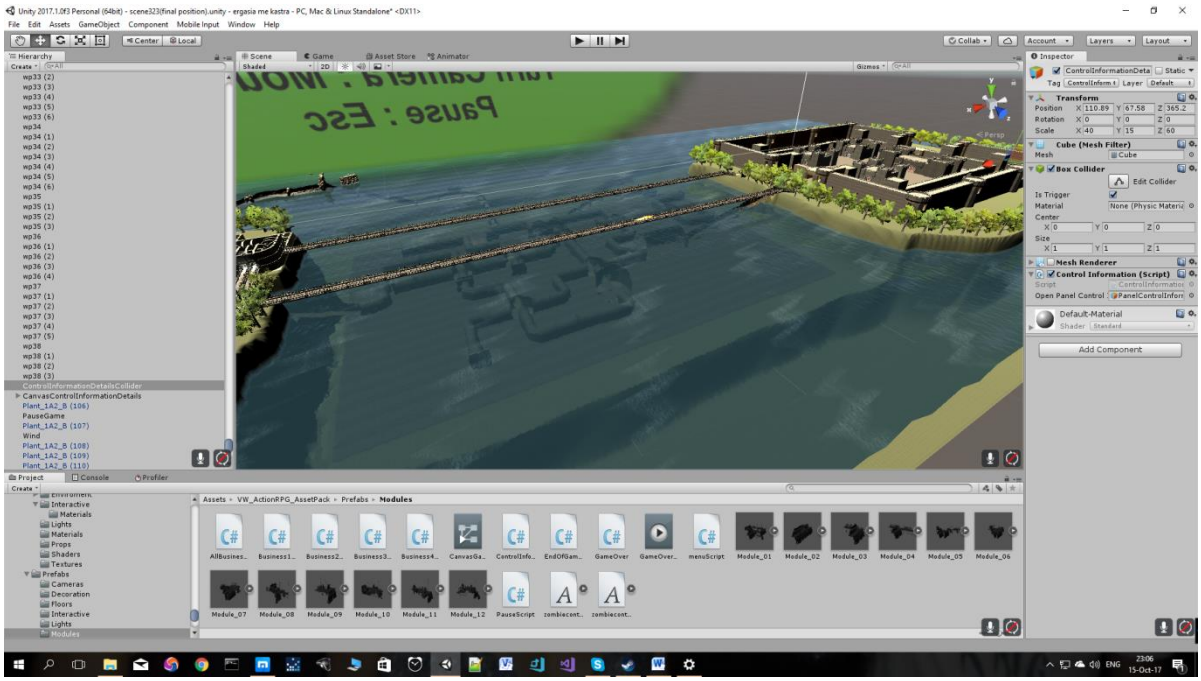
CustomObjectDestroyer(Script): (Παράρτημα 5.11)

Στο συγκεκριμένο script παρατηρούμε τον κώδικα που χρησιμοποιήθηκε για το αντικείμενα που σπάνε.

ElevatorCube :



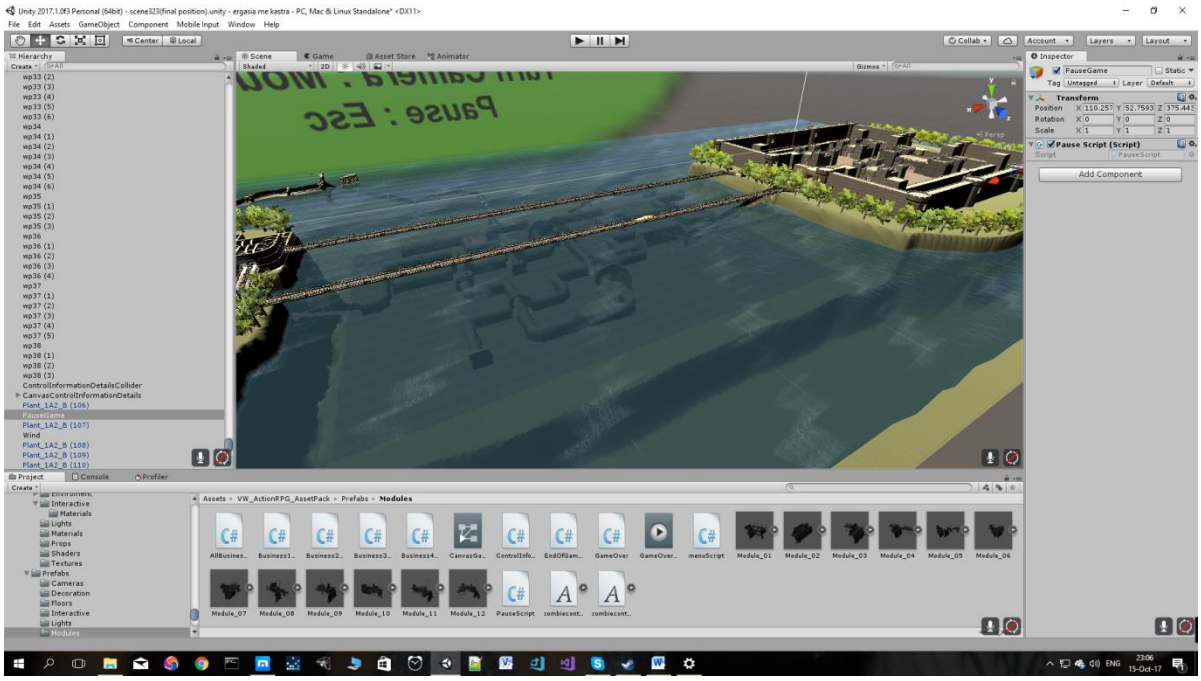
Controls Information :



ControlInformation(Script): (Παράρτημα 5.11)

Στο συγκεκριμένο script παρατηρούμε τον κώδικα που χρησιμοποιήθηκε για την εμφάνιση canvas, στην εκκίνηση του παιχνιδιού,ο οποίος ενημερώνει τον χρήστη για το κάθε κουμπί. Το canvas αυτό εξαφανίζεται καθώς ο παίχτης προχωράει.

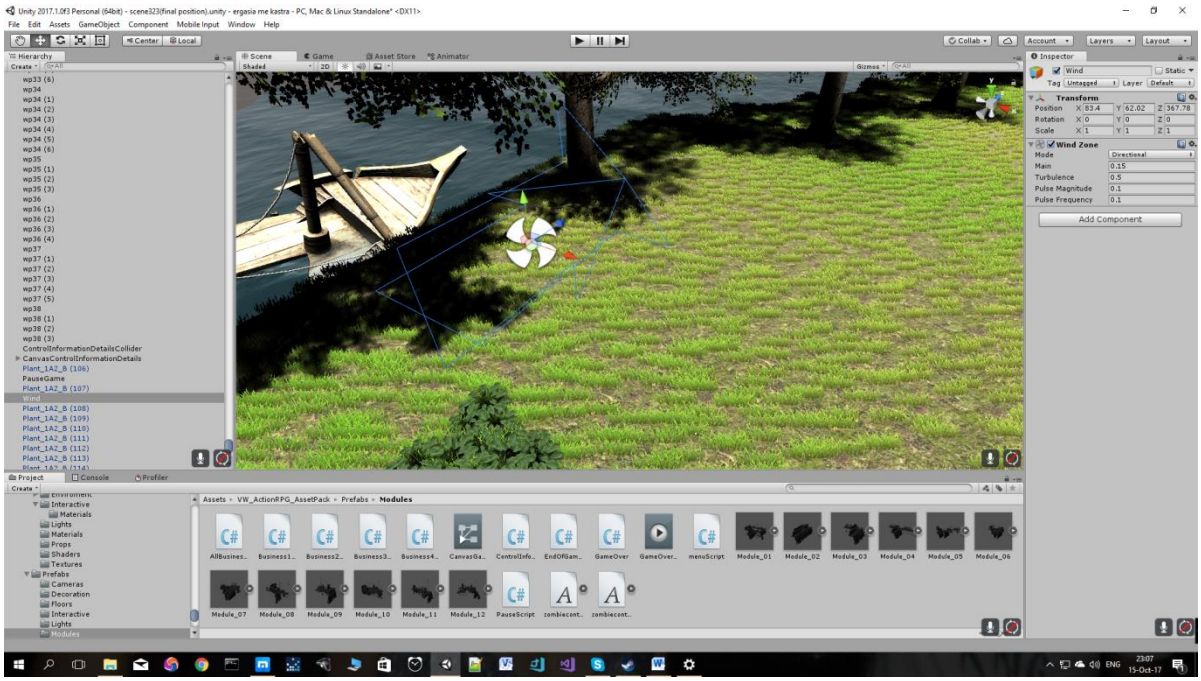
PauseGame:



PauseGame(Script): (Παράρτημα 5.11)

Στο συγκεκριμένο script παρατηρούμε τον κώδικα που χρησιμοποιήθηκε για το κουμπί παύσης του παιχνιδιού.

Wind Zone :



Τέλος, στην παραπάνω εικόνα παρατηρούμε τον τρόπο που τοποθετήθηκε ο άνεμος, την κατεύθυνση που του δώσαμε αλλά και τις ακριβείς ρυθμίσεις έντασης.

Κεφάλαιο 6^{ο} - Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Επεκτάσεις σε gaming

Στο παιχνίδι μας χρησιμοποιήσαμε τον βασικό χαρακτήρα του unity3d, τον ¨Ethan¨, τον οποίο παραμετροποιήσαμε δίνοντας του νέες λειτουργίες, ήχους και animations. Αλλά σε μελλοντικές επεκτάσεις στον τομέα του gaming, θα γίνει αλλαγή χαρακτήρα. Θα ενσωματώσουμε κάποιον έτοιμο χαρακτήρα με όπλο στο δεξί του χέρι, π.χ. ξίφος και ήδη έτοιμα animations για μάχες. Ο παίχτης μας αυτός θα μπορεί μέσω animation να χτυπήσει με το ξίφος τους εχθρούς και να τους αφαιρέσει ζωή. Χτυπώντας τους πολλές φορές θα μπορεί να τους νικήσει. Σίγουρα η παρέμβασή μας στον παίχτη αυτόν θα είναι σημαντική αφού θα του προσθέσουμε και νέες λειτουργίες. Για παράδειγμα, ενώ το ήδη υπάρχον animation του παίχτη θα ναι η κίνηση του ξίφους με στόχο τον εχθρό, θα παρέμβουμε δίνοντάς του animation για χρήση κονταριού μαγείας. Θα ΄ναι πιο αληθοφανές να δείχνουμε τον παίχτη να χτυπά το μαγικό κοντάρι στο έδαφος και να παράγει μαγεία που θα νικά τους γύρω εχθρούς. Θα υπάρχουν διαφορετικές μαγείες που θα αποκτά ο παίχτης και θα μπορεί να χρησιμοποιήσει. Κάθε μία θα έχει μοναδικό animation που θα φτιάξουμε. Για να μπορεί ο παίχτης να χρησιμοποιήσει μαγεία σε κάποιον εχθρό, θα πρέπει να χει πρώτον αποκτήσει το μαγικό όπλο και δεύτερον να χει γεμάτη ενέργεια. Άρα χρειαζόμαστε ένα δεύτερο slider, ο οποίος θα αντιπροσωπεύει την μαγική ενέργεια του παίχτη. Ο Slider αυτός θα χει μπλε χρώμα και θα βρίσκεται ακριβώς κάτω από τον πρώτο. Η ενέργεια αυτή θα γεμίζει από φίλτρα και από γρίφους που θα λύνουμε επιτυχώς. Μια πολύ καλή σκέψη είναι το inventory, όπου θα μπορούμε να αποθηκεύσουμε τα όπλα, τις ασπίδες, τα medikit και τα κλειδιά. Κάθε ένα από αυτά θα μπορούμε να το χρησιμοποιήσουμε όποτε χρειαστεί. Σκέψεις υπάρχουν και για περισσότερα όπλα άμεσης επαφής με εχθρούς, όπως τσεκούρι, μαχαίρι κ.τ.λ.

Ο χάρτης, θα τοποθετηθεί πάνω δεξιά και θα εμφανίζεται μόνο αν τον έχουμε αποκτήσει μέσα στο παιχνίδι. Κάθε περιοχή θα έχει τον δικό της χάρτη που θα μας βοηθά στον προσανατολισμό.

Οι εχθροί θα είναι πολύ περισσότεροι σε ποικιλία αλλά και σε πλήθος. Θα διαφέρουν σε επίπεδα δυσκολίας ανάλογα την ταχύτητα που κινούνται, το damage που μπορούν να προκαλέσουν στον παίχτη αλλά και την αντοχή τους στα χτυπήματά μας. Πάνω από κάθε εχθρό θα υπάρχει ένας slider που θα ενημερώνει τον χρήστη για το υπόλοιπο ζωής του κάθε εχθρού. Κάθε όπλο που χρησιμοποιούμε ή κάθε μαγεία θα επηρεάζει διαφορετικά τον κάθε αντίπαλο.

Γρίφοι θα υπάρχουν σε διάφορα σημεία του παιχνιδιού και ο παίχτης θα καλείται να τους λύσει. Λύνοντάς τους επιτυχώς θα κερδίζει πόντους εμπειρίας (xp). Τους πόντους αυτούς θα μπορεί να τους χρησιμοποιήσει για να αναβαθμίσει και να δυναμώσει τα όπλα του. Πιο εύκολοι γρίφοι θα του δίνουν σαν έπαθλο μαγική ενέργεια.

Cinematics θα ενεργοποιούνται την πρώτη φορά που ο παίχτης μας συναντά ένα καινούριο είδος εχθρού. Ένα πολύ ωραίο εφέ παιχνιδιών που ενημερώνει τον παίχτη για το καινούριο αυτό είδος και επιστά την προσοχή του.

Νέες πίστες θα τοποθετηθούν κάνοντας το παιχνίδι ακόμη πιο ελκυστικό, ενδιαφέρον και μεγάλο σε διάρκεια. Το επίπεδο υψηλής ανάλυσης γραφικών θα διατηρηθεί και στις νέες πίστες κάνοντας το παιχνίδι πιο αληθοφανές.

Ο χρήστης στο αρχικό μενού θα μπορεί να επιλέξει ανάμεσα σε καθαρό game ή παιχνίδι εκπαιδευτικού στόχου. Αν επιλέξει καθαρό game, τότε θα του δίνεται η επιλογή επιπέδου δυσκολίας. Θα υπάρχουν τέσσερα διαφορετικά επίπεδα.

Ο χρήστης θα έχει την δυνατότητα να κάνει skip σε cinematic και gameover.

Η δυνατότητα ¨save game¨ θα προστεθεί καθώς το παιχνίδι μεγαλώνει πλέον σε διάρκεια. Οπότε πλέον στην εκκίνηση, ο χρήστης θα μπορεί να φορτώσει το παιχνίδι από το σημείο που σταμάτησε την τελευταία φορά. Επίσης, checkpoints θα τοποθετηθούν ανά περιοχή. Με αυτό τον τρόπο, όταν ο χρήστης χάνει, θα ΄χει την δυνατότητα να ξεκινήσει πάλι το παιχνίδι από το τελευταίο checkpoint ή να κάνει load και να ξεκινήσει από το σημείο που έκανε τελευταία φορά save.

Το κάθε κλειδί θα έχει ένα συγκεκριμένο όνομα, έτσι ώστε να ξέρει ο χρήστης ποια πόρτα-πύλη ανοίγει. Η κάθε πόρτα-πύλη θα έχει επίσης συγκεκριμένο όνομα και θα αναφέρει με την χρήση canvas, το όνομα του συγκεκριμένου κλειδιού που απαιτείται για να ανοίξει.

Σε κάθε μετάβαση από περιοχή σε περιοχή, ο χρήστης θα ενημερώνεται για την τοποθεσία που βρίσκεται.

Σκέψη αποτελεί επίσης η χρησιμοποίηση πραγματικών ηχογραφημένων φωνών σε διάφορα σημεία του παιχνιδιού.

6.2 Εκπαιδευτικές επεκτάσεις

Στο εκπαιδευτικό κομμάτι, εφικτή είναι η είσοδος ασαφής λογικής στο παιχνίδι μας. Η ασαφής λογική αυτή, έχει δημιουργηθεί σε web και αποτελείται από σενάρια εκπαιδευτικού τύπου. Έτσι ο χρήστης καλείται να απαντήσει σε ερωτήσεις εκπαιδευτικού περιεχομένου και ανάλογα τις απαντήσεις που θα δώσει, επιλέγεται διαφορετικό σενάριο το οποίο είτε αυξάνει, είτε μειώνει την διάρκεια του παιχνιδιού. Άρα μπορούμε να φανταστούμε τα σενάρια αυτά σαν ένα δέντρο. Στην κορυφή του δέντρου βρίσκεται το σενάριο, όπου ο χρήστης απαντάει σε όλες τις ερωτήσεις σωστά. Εκεί το παιχνίδι μας θα παρουσιάσει τους ελάχιστους δυνατούς γρίφους, άρα και την ελάχιστη δυνατή διάρκεια παιχνιδιού. Στα ενδιάμεσα τμήματα του δέντρου βρίσκονται οι απαντήσεις που αξιολογούν τις γνώσεις του χρήστη σε μέτριο επίπεδο και στον πάτο του δέντρου σε κακό επίπεδο. Άρα όταν ο χρήστης απαντήσει λανθασμένα σε όλες τις

ερωτήσεις, τότε η διάρκεια του παιχνιδιού είναι η μέγιστη. Με αυτό τον πολύ όμορφο τρόπο, ο χρήστης μαθαίνει παίζοντας.

Όσον αφορά την ενσωμάτωση της ασαφής λογικής στο παιχνίδι μας, ο χρήστης θα μπορούσε να δυναμώσει τα όπλα του απαντώντας σωστά σε κάθε ερώτηση. Αυτό θα το επιτύγχανε κερδίζοντας πόντους εμπερίας (xp). Για να απαντήσει σωστά όμως ο χρήστης στην κάθε ερώτηση, κρίνεται απαραίτητη η εισαγωγή agents, οι οποίοι θα δίνουν τα δεδομένα στον χρήστη μαθαίνοντάς του κάθε φορά κάτι διαφορετικό, π.χ. διαφορετικές πληροφορίες που αφορούν τον προγραμματισμό με την γλώσσα C#. Η χρησιμοποίηση πολλών agents σε διαφορετικά σημεία του παιχνιδιού, αποτελεί ένα επιπλέον κίνητρο για τον χρήστη να περιηγηθεί και να εξερευνήσει όλα τα σημεία και τις περιοχές. Κάτι τέτοιο μεγαλώνει την διάρκεια του παιχνιδιού. Ο χρήστης, ξέροντας την σημαντικότητα των σωστών απαντήσεων δίνει πολύ μεγάλη έμφαση στην γνώση που προσφέρουν οι agents. Φυσικά η χρήση πολλών διαφορετικών agents, δίνει την δυνατότητα στον χρήστη να επιστρέψει στον ίδιο agent και να ξαναλάβει την συγκεκριμένη πληροφορία όσες φορές θέλει. Και αυτό είναι πολύ σημαντικό, διότι ενισχύει την μάθηση. Η επανάληψη θα οδηγήσει σε βαθύτερη γνώση και καλύτερη αφομοίωση. Σκέψη αποτελεί η χρησιμοποίηση αντίστροφης μέτρησης, που θα απεικονίζεται κάτω δεξιά, μέχρι τον επόμενο γρίφο.

Γρίφοι είναι δυνατόν να υπάρξουν και σε σημεία που ο παίχτης θα καλείται να ανοίξει μία πόρτα για να συνεχίσει την πορεία του ή πίσω από την πόρτα να υπάρχει ως έπαθλο ένα όπλο. Ένας όμορφος τρόπος απεικόνισης και επίλυσης γρίφου είναι ο εξής: Ο χρήστης να μπορεί να διαβάζει στον τοίχο του δωματίου στην ερώτηση ή καποιο κομμάτι πρότασης ή κώδικα που λείπει. Έπειτα να καλείται να βρει κάποιο κλειδί του δωματίου. Το κάθε κλειδί στο δωμάτιο θα αντιστοιχεί σε μια απάντηση. Μπορούμε να ΄χουμε αρκετά κλειδιά για κάθε γρίφο, έτσι ώστε να ελέγχουμε σε βάθος τις γνώσεις του χρήστη. Η σωστή απάντηση μόνο ανοίγει την πόρτα που θέλουμε. Η λάθος απάντηση ανοίγει μια κρύπτη όπου εμφανίζονται εχθροί και πρέπει ο παίχτης να τους αντιμετωπίσει πριν συνεχίσει με το επόμενο κλειδί. Αφού ο παίχτης εξουδετερώσει τους εχθρούς, μπορεί να επιλέξει άλλο κλειδί. Η κάθε όμως συνεχόμενα λανθασμένη απάντηση στο γρίφο, θα εμφανίζει κάθε φορά και περισσότερους εχθρούς. Κάτι τέτοιο αυξάνει το επίπεδο δυσκολίας του παιχνιδιού και παράλληλα δίνει έξτρα κίνητρο στον χρήστη να δώσει την σωστή απάντηση, επιλέγοντας το σωστό κλειδί. Με αυτό τον τρόπο εφιστούμε σε μέγιστο βαθμό την προσοχή του χρήστη στα δεδομένα που του δίνουν οι agents αλλά και στο τι ακριβώς ζητάει ο κάθε γρίφος. Ο χρήστης θα σκεφτεί αρκετή ώρα πριν επιλέξει την απάντηση μέσω κλειδιού. Κάτι τέτοιο θα τον ωθήσει να σκεφτεί αναλυτικά την κάθε απάντηση και να απορρίψει τις λανθασμένες, προτού επιλέξει. Δεν αποκλείεται ο χρήστης να σημειώνει σε χαρτί όλες τις πληροφορίες που λαμβάνει απο τους agents και να διαβάζει ξανά και ξανά την κάθε πληροφορία προτού επιλέξει. Με αυτό τον τρόπο, η αφομοίωση των πληροφοριών εκπαιδευτικού σκοπού μεγαλώνει.

6.3 Πολιτιστικές επεκτάσεις

Στο πολιτιστικό κομμάτι, το παιχνίδι μας θα μπορούσε να αποτελέσει εργαλείο προώθησης της Ελληνικής κληρονομίας. Θα μπορούσαμε να παρουσιάσουμε μνημεία και αρχαιολογικούς χώρους, π.χ. Ακρόπολη. Αυτό είναι εφικτό να το πετύχουμε με scenes και διαφορετικές πίστες. Παράδειγμα, στις κατακόμβες, θα ήταν εφικτό ο παίχτης να έχει την επιλογή τριών διαφορετικών μονοπατιών. Κάθε μονοπάτι θα οδηγεί σε διαφορετικό μνημείο. Άλλος τρόπος είναι με την χρήση μαγικών καθρεπτών, στις περιοχές buildings1 και buildings2. Περνώντας μέσα από κάθε μαγικό καθρέπτη, ο παίχτης θα εμφανίζεται σε διαφορετική πίστα όπου θα απεικονίζεται και διαφορετικό μνημείο. Στην οθόνη μας θα εμφανίζουμε κατά την εκκίνηση της πίστας, την τοποθεσία που βρισκόμαστε και την χρονολογία. Πλησιάζοντας το μνημείο, θα εμφανίζονται ιστορικά στοιχεία γι΄ αυτό, π.χ. χρονολογία δημιουργίας, χρησιμότητα στο παρελθόν κ.τ.λ. Με τον τρόπο αυτό, μέσω της ψυχαγωγίας που προσφέρει ένα παιχνίδι, προβάλλουμε την πολιτιστική κληρονομιά του τόπου μας. Σκέψη επίσης αποτελεί η απεικόνιση μουσείων με Ελληνικά έργα τέχνης, όπου θα μπορούμε να θαυμάσουμε το κάθε έργο σε τρισδιάσταση μορφή. Πλησιάζοντας το κάθε έργο τέχνης, ο χρήστης θα μπορεί να διαβάσει στην οθόνη του, ιστορικά στοιχεία γι΄ αυτό. Βέβαια για να επιτύχουμε κάτι τέτοιο πρέπει να μπορέσουμε να βρούμε 3D μοντέλα, δημιουργημένα από designers. Έπειτα είναι εύκολο να τα εισάγουμε στο unity με import, από όπου θα μπορούμε να τα επεξεργαστούμε κατάλληλα.

Το παιχνίδι μας επίσης θα μπορούσε να αποτελέσει εργαλείο προώθησης πολιτιστικής κληρονομιάς στο τομέα της μουσικής. Θα μπορούσαμε μέσω παιχνιδιού να προβάλλουμε την μουσική και το έργο, από μεγάλους Έλληνες μουσικοσυνθέτες. Ένα υποθετικό σενάριο είναι: Στις επιχειρήσεις, συγκεκριμένα στην επιχείρηση restaurant, θα μπορούσε να τοποθετηθεί μια προτομή, ενός μεγάλου Έλληνα μουσικοσυνθέτη. Πλησιάζοντας ο παίχτης την προτομή αυτή, πληροφορίες για την ζωή και το έργο του μουσικοσυνθέτη θα εμφανίζονται. Μια ορχήστρα θα υπάρχει στο εστιατόριο η οποία θα παίζει μόνο κομμάτια του συγκεκριμένου μουσικοσυνθέτη. Μια πολύ μεγάλη τηλεόραση θα είναι τοποθετημένη στον τοίχο, η οποία θα απεικονίζει με βίντεο, αποσπάσματα από την ζωή του. Επιλογή αποτελεί ο συγχρονισμός της μουσικής που ακούγεται στην επιχείρηση ανά πάσα στιγμή, με εικόνα βίντεο, εμφανιζόμενο στην οθόνη της τηλεόρασης. Ο χρήστης θα μπορεί να αλλάξει το μουσικό κομμάτι ανά πάσα στιγμή καθώς επίσης, θα μπορεί να βαθμολογήσει πόσο του αρέσει. Έτσι, με πλήθος τέτοιων επιχειρήσεων θα μπορούσαμε να προβάλλουμε τη ζωή, το έργο και την μουσική πολλών μουσικοσυνθετών. Ανάλογα την βαθμολογία που θα δώσει ο χρήστης για κάθε μουσικό κομμάτι, αυτόματα θα μπορούσε να δημιουργείται μια λίστα αγαπημένων του κομματιών, η οποία θα παίζει σε σημείο που εμείς θα επιλέγουμε. Μια άλλη σκέψη αποτελεί η οπτικοποίηση σεναριών από τραγούδια. Σε βαθμό που αυτό είναι εφικτό, να δημιουργούμε σενάρια για τον παίχτη μας, σενάρια τα οποία προκύπτουν από στίχους συγκεκριμένων τραγουδιών. Μόλις το σενάριο ενεργοποιηθεί και ο παίχτης βρεθεί στην συγκεκριμένη κατάσταση, σύμφωνα με τους στίχους, το συγκεκριμένο τραγούδι να ακούγεται.

Ιστογραφία – Δικτυότοποι

- *.eLearningINDUSTRY.The Pros And Cons Of Using Virtual Reality In The Classroom. Ανακτήθηκε Οκτώβριος 2017 https://elearningindustry.com/pros-cons-using-virtual-reality-in-the-classroom

- 1. WIKIPEDIA. History of video games. Ανακτήθηκε Οκτώβριος 2017 https://en.wikipedia.org/wiki/History_of_video_games

- 2. naftemporiki.gr. MIT: Ανοίγοντας τον δρόμο προς τα ασύρματα συστήματα εικονικής πραγματικότητας. Ανακτήθηκε Οκτώβριος 2017 http://www.naftemporiki.gr/story/1171641/mit-anoigontas-ton-dromo-pros-ta-asurmata-sustimata-eikonikis-pragmatikotitas

- 3. naftemporiki.gr. Χειρισμός σε ηλεκτρονικά παιχνίδια εικονικής πραγματικό-τητας μέσω εγκεφάλου. Ανακτήθηκε Οκτώβριος 2017 http://www.naftemporiki.gr/story/1181420/xeirismos-se-ilektronika-paixnidia-eikonikis-pragmatikotitas-meso-egkefalou

- 4. Statista-The Statistics Portal-. Gaming Is a $30 Billion Market in the U.S. Ανακτήθηκε Οκτώβριος 2017 https://www.statista.com/chart/9838/consumer-spending-on-video-games/

- 5. Statista-The Statistics Portal-. Who Are America's Video Gamers? Ανακτήθηκε Οκτώβριος 2017 https://www.statista.com/chart/11086/americas-diverse-pool-of-gamers/

- 6. Statista-The Statistics Portal-. The Most Important Gaming Platforms in 2016 Ανακτήθηκε Οκτώβριος 2017 https://www.statista.com/chart/4527/game-developers-platform-preferences/

- 7. ΠΡΩΤΟΘΕΜΑ.gr. Mobile Games: Παιχνίδια από... χρυσάφι. Ανακτήθηκε Οκτώβριος 2017 http://www.protothema.gr/technology/article/526099/paihnidia-apo-hrusafi/

- 8. Statista-The Statistics Portal-. The United States of Mobile Gaming Ανακτήθηκε Οκτώβριος 2017 https://www.statista.com/chart/995/gamers-in-the-the-united-states/

- 9. WIKIPEDIA. List of most expensive video games to develop.
  Ανακτήθηκε Οκτώβριος 2017
  https://en.wikipedia.org/wiki/List_of_most_expensive_video_games_to_develop

- 10. Statista-The Statistics Portal-. GTAV is the Fastest-Selling Entertainment Product Ever.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.statista.com/chart/1501/most-successful-entertainment-products/

- 11. Program ACE. Game Development: Unity vs HTML5
  Ανακτήθηκε Οκτώβριος 2017
  https://program-ace.com/press-room/articles/game-development-unity-vs-html5

- 12. GAME SPARKS. Game Engine Analysis and Comparison
  Ανακτήθηκε Οκτώβριος 2017
  https://www.gamesparks.com/blog/game-engine-analysis-and-comparison/

- 13. PLURAL SIGHT. Unity, Source 2, Unreal Engine 4, or CryENGINE - Which Game Engine Should I Choose?
  Ανακτήθηκε Οκτώβριος 2017
  https://www.pluralsight.com/blog/film-games/unity-udk-cryengine-game-engine-choose

- 14. PC STEPS. Αγορά Παιχνιδιών Υπολογιστή: Το Steam και οι Άλλοι
  Ανακτήθηκε Οκτώβριος 2017
  https://www.pcsteps.gr/119588-%CE%B1%CE%B3%CE%BF%CF%81%CE%AC-%CF%80%CE%B1%CE%B9%CF%87%CE%BD%CE%B9%CE%B4%CE%B9%CF%8E%CE%BD-%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CE%AE-steam/

- Youtube. Door that opens with a key! How to make a Horror Game 10 Unity 3D.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=gpgcuO5wEyo

- Unity/DOCUMENTATION. Animation.Play
  Ανακτήθηκε Οκτώβριος 2017
  https://docs.unity3d.com/ScriptReference/Animation.Play.html

- Youtube. Flickering Lights. How to make a Horror game 13 Unity 3D.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=kDUm7VoriFw&t=30s&list=PLF1JI43KNe-vDFakQe8_si5pAxJtXw18A&index=13

- Youtube. Enemy AI. How to make a Horror Game 7 Unity 3D.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=7vQ5ra53wU4&list=PLF1JI43KNe-vDFakQe8_si5pAxJtXw18A&index=7

- Youtube. Main Menu! How to make a Horror Game 6 Unity 3D.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=rJyqePmspVo&t=106s&list=PLF1JI43KNe-vDFakQe8_si5pAxJtXw18A&index=6

- Youtube. Jump Scare. How to make a Horror Game 3 Unity 3D.
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=fsWP2x10UD0&list=PLF1JI43KNe-vDFakQe8_si5pAxJtXw18A&index=3

- Youtube. A Simple GUI Inventory, Object Pickup and Respawn in Unity 5
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=CHUOprBocoY

- Youtube. Mechanim & Mixamo in Unity 5: The Basic Basics!
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=BEIaakl9vJE

- Youtube. Creating a Health Bar in Unity 5
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=S4wDjPk0tlQ

- Youtube. Pausing a Unity 5 Game
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=WV4ELhq9dBU

- Youtube. Basic Artificial Intelligence for a Non-Player Character with Unity 5
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=gXpi1czz5NA

- Youtube. Melee Combat with Unity 5
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=nt55pxA7Snk#t=3.639

- Youtube. Simple Waypoint Pathfinding with the Line of Sight AI
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=OmoKw1ikAi8#t=246.917248

- Youtube. Unity-Creating Prefabs
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=HECGcwwRM64

- Unity/DOCUMENTATION. AudioSource.Play
  Ανακτήθηκε Οκτώβριος 2017
  https://docs.unity3d.com/ScriptReference/AudioSource.Play.html

- Youtube. How To Destroy Or Delete Game Object In Unity 3D C# Tutorial Beginner
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=2v3YohbcpHo

- Youtube. Creating a Start Menu in Unity 5
  Ανακτήθηκε Οκτώβριος 2017
  https://www.youtube.com/watch?v=pT4uca2bSgc&t=324s

Παραρτήματα

Παράρτημα 5.1

MonsterChaseWP :

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


public class MonsterChaseWP : MonoBehaviour
{

    public Transform player;
    public Transform head;
    Animator anim;


    string state = "patrol";
    public GameObject[] waypoints;
    int currentWP = 0;
    public float rotSpeed = 0.2f;
    public float speed = 1.5f;
    float accuracyWP = 5.0f;

    //Use this for initialization
    void Start()
    {
        anim = GetComponent<Animator>();
    }
    //Update is called once per frame
    void Update()
    {

        Vector3 direction = player.position - this.transform.position;
        direction.y = 0;
        float angle = Vector3.Angle(direction, head.up);

        if (state == "patrol" && waypoints.Length > 0)
        {

            anim.SetBool("isIdle", false);
            anim.SetBool("isWalking", true);
            if (Vector3.Distance(waypoints[currentWP].transform.position,
            transform.position) < accuracyWP)
            {

                currentWP = Random.Range(0, waypoints.Length);
            }

            //rotate towards waypoint
```

```
            direction = waypoints[currentWP].transform.position - transform.position;
            this.transform.rotation = Quaternion.Slerp(transform.rotation,
                Quaternion.LookRotation(direction), rotSpeed * Time.deltaTime);
            this.transform.Translate(0, 0, Time.deltaTime * speed);
        }

        if (Vector3.Distance(player.position, this.transform.position) < 10 &&
            (angle   < 30 || state == "pursuing"))
        {

            state = "pursuing";
            this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
            Quaternion.LookRotation(direction), rotSpeed * Time.deltaTime);



            if (direction.magnitude > 2.0)
            {
                this.transform.Translate(0, 0, Time.deltaTime * speed);
                anim.SetBool("isWalking", true);
                anim.SetBool("isAttacking", false);
            }

            else
            {

                anim.SetBool("isAttacking", true);
                anim.SetBool("isWalking", false);
            }
        }
        else
        {

            anim.SetBool("isWalking", true);
            anim.SetBool("isAttacking", false);
            state = "patrol";
        }
    }
  }
```

EnemySound :

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



[RequireComponent(typeof(AudioSource))]
public class EnemySound : MonoBehaviour {
```

```
    public Slider healthbar;


    // Use this for initialization
    void Start ()
    {

    }


    void OnTriggerEnter(Collider TheCollider)
    {
        if (TheCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Play();
        }
    }

    void OnTriggerExit(Collider TheCollider)
    {
        if (TheCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }
    }


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

        if (healthbar.value <= 0)
        {
            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }

    }
}
```

Παράρτημα 5.2

TrapSound(Script):

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

```
using System;



[RequireComponent(typeof(AudioSource))]
public class TrapSound : MonoBehaviour
{

      public Slider healthbar;


    void Start()
   {

   }


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

        if (healthbar.value <= 0)
        {
            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }

    }

    void OnTriggerEnter(Collider TheCollider)
    {

        if (TheCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Play();
            Debug.Log("Trap");
        }
    }

    void OnTriggerExit(Collider TheCollider)
    {

        if (TheCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }

    }
   }
```

SwingingBlade_Damaged(Script):

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

public class SwingingBlade_Damaged : MonoBehaviour
{

    public GameObject TheSwingingBlade_Damaged;
    Animator anim;
    private bool playerNextToTrap = false;


    // Update is called once per frame
    void Update()
    {
        if (playerNextToTrap == true)
        {


GetComponent<Animator>().SetTrigger("DTP_AnimTrigger_CeilingTraps_SwingingBlades_Dama
ged_StartSwinging");

        }

        else
        {


GetComponent<Animator>().SetTrigger("DTP_AnimTrigger_CeilingTraps_SwingingBlades_Dama
ged_StopSwinging");
        }

    }


    // Use this for initialization
    void Start()
    {
        anim = GetComponent<Animator>();

    }


    void OnTriggerEnter(Collider SwingingBlade_DamagedCollider)
    {


        if (SwingingBlade_DamagedCollider.tag == "Player")
        {

            playerNextToTrap = true;
            Debug.Log("Trap");
        }
    }
```

```
    void OnTriggerExit(Collider SwingingBlade_DamagedCollider)
    {
        if (SwingingBlade_DamagedCollider.tag == "Player")
        {

            playerNextToTrap = false;
        }
    }

}
```

Παράρτημα 5.3

EnemyHit(Script):

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

public class EnemyHit : MonoBehaviour
{

    public GameObject player;
    public Slider healthbar;
    Animator anim;
    private bool GetHit = false;
    public AudioClip HittedSound;


    void OnTriggerEnter(Collider theCollider)
    {
        if (theCollider.tag == "EnemyWeaponCollider")
        {
            GetHit = true;
            healthbar.value -= 20;
            Debug.Log("Hit");


           if (healthbar.value <= 0)
              {

               GetHit = false;
              }
        }
    }


    // Use this for initialization
    void Start ()
    {
```

```
        anim = GetComponent<Animator>();
    }


    // Update is called once per frame
    void Update ()
    {
          if (GetHit == true )
              {

                    player.GetComponent<AudioSource>().PlayOneShot(HittedSound);
                    player.GetComponent<Animator>().SetTrigger("isHitted");
                    GetHit = false;
              }
    }

}
```

PlayerManager(Script):

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

public class PlayerManager : MonoBehaviour
{

    public static int health = 100;
    public GameObject player;
    public Slider healthBar;
    public AudioClip DeathSound;
    public AudioClip BreathScaredSound;
    private bool DeathSoundhasPlayed = false;
    private bool BreathScaredSoundhasPlayed = false;
    private AudioSource source;


    public void Awake ()
    {

        source = GetComponent<AudioSource>();
    }

    // Use this for initialization
    void Start ()
    {

        BreathScaredSoundhasPlayed = false;
    }

    void ReduceHealth()
```

```
    {
        health = health - 0;
        healthBar.value = health;
        if(health <= 0)
        {

        }

        if ((health <= 30) && (health >= 10))
        {

        }

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

        if ((healthBar.value <= 30) && (healthBar.value >= 10))
        {

            if (!BreathScaredSoundhasPlayed)
            {

                Debug.Log("BreathScaredSound");
                player.GetComponent<AudioSource>().PlayOneShot(BreathScaredSound);
                BreathScaredSoundhasPlayed = true;
            }

        }


        if (healthBar.value > 30)
        {

            BreathScaredSoundhasPlayed = false;
        }


        if (healthBar.value <= 0)
        {

            if (!DeathSoundhasPlayed)
               {

                  player.GetComponent<AudioSource>().PlayOneShot(DeathSound);
                  player.GetComponent<Animator>().SetTrigger("isDead");
                  Debug.Log("Dead");
                  DeathSoundhasPlayed = true;
               }
        }
    }
}
```

Παράρτημα 5.4

BUSINESS2GATE1(Script) :

```
#pragma strict


var doorClip : AnimationClip;
var Key : GameObject;
var doorSound : AudioClip;
private var Door = false;



function Start ()
{

}

function Update ()
{

	if (Input.GetKeyDown(KeyCode.E) && Door == true && Key.active == false)
	{
	AudioSource.PlayClipAtPoint(doorSound, transform.position);
	GameObject.Find("BUSINESS2GATE1").GetComponent.<Animation>().Play("BUSINESS2GA
TE1OPEN");

	}
}

function OnTriggerEnter (theCollider : Collider)
{
	if (theCollider.tag == "Player")
	{
		Door = true;
	}
}

function OnTriggerExit (theCollider : Collider)
{
	if (theCollider.tag == "Player")
	{
		Door = false;
	}
}
```

Key1(Script):

```
#pragma strict
```

```
var TheKey : GameObject;
var keySound : AudioClip;
private var playerNextToKey = false;

function Update ()
{
	if (Input.GetKeyDown(KeyCode.E) && playerNextToKey == true)
	{
	AudioSource.PlayClipAtPoint(keySound, transform.position);
	TheKey.active = false;
	}
}

function OnTriggerEnter (theCollider : Collider)
{
	if (theCollider.tag == "Player")
	{
		playerNextToKey = true;
	}
}

function OnTriggerExit (theCollider : Collider)
{
	if (theCollider.tag == "Player")
	{
		playerNextToKey = false;
	}
  }
```

MESSAGE(Script) :

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

public class MESSAGE : MonoBehaviour
{

    public GameObject OpenPanel = null;

    void Start()
    {
        OpenPanel.SetActive(false);
    }


    void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            OpenPanel.SetActive(true);
        }
    }
```

```
    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Player")
        {
            OpenPanel.SetActive(false);
        }
    }

    private bool IsOpenPanelActive
    {
        get
        {
            return OpenPanel.activeInHierarchy;
        }
    }

    void Update()
    {
      if(IsOpenPanelActive)
        {
            if(Input.GetKeyDown(KeyCode.E))
            {
                OpenPanel.SetActive(false);
            }
        }
    }
  }
```

Παράρτημα 5.5

Medikit(Script):

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

public class Medikit : MonoBehaviour

{

    public GameObject TheMedikit;
    public AudioClip medikitSound;
    private bool playerNextToMedikit = false;
    public Slider healthbar;

    // Update is called once per frame
    void Update()
```

```
    {
        if (Input.GetKeyDown(KeyCode.E) && playerNextToMedikit == true)
        {
            AudioSource.PlayClipAtPoint(medikitSound, transform.position);
            TheMedikit.active = false;
            healthbar.value += 100;
            Debug.Log("Medikit");
        }
    }


    // Use this for initialization
    void Start()
    {

    }


    void OnTriggerEnter(Collider theCollider)
    {

        if (theCollider.tag == "Player")
        {
            playerNextToMedikit = true;
        }
    }

    void OnTriggerExit(Collider theCollider)
    {
        if (theCollider.tag == "Player")
        {

            playerNextToMedikit = false;
        }
    }

}
```

RedPotion(Script):

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

public class RedPotion : MonoBehaviour

{

    public GameObject TheRedPotion;
    public AudioClip RedPotionSound;
    private bool playerNextToRedPotion = false;
    public Slider healthbar;

    // Update is called once per frame
```

```
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E) && playerNextToRedPotion == true)
        {
            AudioSource.PlayClipAtPoint(RedPotionSound, transform.position);
            TheRedPotion.active = false;
            healthbar.value -= 20;
            Debug.Log("RedPotion");


        }
    }


    // Use this for initialization
    void Start()
    {

    }


    void OnTriggerEnter(Collider theCollider)
    {

        if (theCollider.tag == "Player")
        {
            playerNextToRedPotion = true;
        }
    }

    void OnTriggerExit(Collider theCollider)
    {
        if (theCollider.tag == "Player")
        {

            playerNextToRedPotion = false;
        }
    }

}
```

PotionHit(Script):

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

public class PotionHit : MonoBehaviour
{

    public GameObject player;
    public Slider healthbar;
    Animator anim;
```

```
private bool GetHit = false;
public AudioClip HittedSound;
private bool playerNextToEnemyPotion = false;



void OnTriggerEnter(Collider theCollider)
{
    if (theCollider.tag == "EnemyPotionCollider")
    {

            playerNextToEnemyPotion = true;
            Debug.Log("EnemyPotionCollider");
    }
}


void OnTriggerExit(Collider theCollider)
{
    if (theCollider.tag == "EnemyPotionCollider")
    {

        playerNextToEnemyPotion = false;
    }
}



// Use this for initialization
void Start()
{
    anim = GetComponent<Animator>();
    playerNextToEnemyPotion = false;
}



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

    if (Input.GetKeyDown(KeyCode.E) && playerNextToEnemyPotion == true)
    {

        GetHit = true;
        healthbar.value -= 20;
        Debug.Log("Hit");
    }


    if (GetHit == true)
    {

        player.GetComponent<AudioSource>().PlayOneShot(HittedSound);
        player.GetComponent<Animator>().SetTrigger("isHitted");
        GetHit = false;
        playerNextToEnemyPotion = false;
    }
```

```
    }
}
```

Παράρτημα 5.6

GameOver(Script)

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



[RequireComponent(typeof(AudioSource))]
public class GameOver : MonoBehaviour
{

    public Canvas CanvasGameOver;
    public GameObject player;
    public Slider healthbar;
    public float GameOverDelay = 5f;
    float GameOverTimer;
    public float restartDelay = 5f;
    float restartTimer;
    private bool GameOverMusichasPlayed = false;
    public float GameOverMusicDelay = 5f;
    float GameOverMusicTimer;



    void Awake()
    {

    }


    void Start()
    {


        CanvasGameOver.enabled = false;
    }


    void Update()
    {

        if (healthbar.value <= 0)
        {

            GameOverTimer += Time.deltaTime;
            restartTimer += Time.deltaTime;
```

```
            GameOverMusicTimer += Time.deltaTime;

            if (!GameOverMusichasPlayed)
            {
                if (GameOverMusicTimer >= GameOverMusicDelay)
                {
                    AudioSource audio = GetComponent<AudioSource>();
                    audio.Play();
                    Debug.Log("GameOverMusic");
                    GameOverMusichasPlayed = true;
                }
            }

            if (GameOverTimer >= GameOverDelay)
            {

                CanvasGameOver = CanvasGameOver.GetComponent<Canvas>();
                CanvasGameOver.enabled = true;
            }

            if (restartTimer >= restartDelay)
            {

                SceneManager.LoadScene(0);
            }
        }
    }
}
```

Παράρτημα 5.7

MenuScript(Script):

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

public class menuScript : MonoBehaviour
{

    public Canvas quitMenu;
    public Button startText;
    public Button exitText;

    void Start()

    {

        quitMenu = quitMenu.GetComponent<Canvas>();
```

```
        startText = startText.GetComponent<Button>();
        exitText = exitText.GetComponent<Button>();
        quitMenu.enabled = false;
    }

    public void ExitPress()

    {

        quitMenu.enabled = true;
        startText.enabled = false;
        exitText.enabled = false;
    }

    public void NoPress()

    {

        quitMenu.enabled = false;
        startText.enabled = true;
        exitText.enabled = true;
    }

    public void StartLevel()

    {

        SceneManager.LoadScene(1);
    }

    public void ExitGame()

    {

        Application.Quit();
    }

  }
```

Παράρτημα 5.8

Business2Success(Script):

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


[RequireComponent(typeof(AudioSource))]
public class Business2Success : MonoBehaviour
```

```
{

    public GameObject OpenPanelBusiness2;

    void Start()
    {

        OpenPanelBusiness2.SetActive(false);
    }


    void OnTriggerEnter(Collider BUSINESS2SUCCESSCOLLIDER)
    {
        if (BUSINESS2SUCCESSCOLLIDER.tag == "Player")
        {

            OpenPanelBusiness2.SetActive(true);
            AudioSource audio = GetComponent<AudioSource>();
            audio.Play();
        }
    }

    void OnTriggerExit(Collider BUSINESS2SUCCESSCOLLIDER)
    {
        if (BUSINESS2SUCCESSCOLLIDER.tag == "Player")
        {

            OpenPanelBusiness2.SetActive(false);
            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
            Destroy(gameObject);
        }
    }

    private bool IsOpenPanelBusiness2Active
    {
        get
        {

            return OpenPanelBusiness2.activeInHierarchy;
        }
    }

    void Update()
    {

    }
  }
```

AllBusinessSuccess(Script):

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

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


[RequireComponent(typeof(AudioSource))]
public class AllBusinessSuccess : MonoBehaviour
{

    private bool BUSINESS1HASOPENED = false;
    private bool BUSINESS2HASOPENED = false;
    private bool BUSINESS3HASOPENED = false;
    private bool BUSINESS4HASOPENED = false;
    private bool ENDOFGAME = false;
    public float restartDelay = 5f;
    float restartTimer;
    public GameObject player;
    public GameObject OpenPanelAllBusinessSuccess;

    void Start()
    {

        OpenPanelAllBusinessSuccess.SetActive(false);
    }


    void OnTriggerEnter(Collider Other)
    {
        if (Other.tag == "BUSINESS1SUCCESSCOLLIDER")
        {

            BUSINESS1HASOPENED = true;
            Debug.Log("BUSINESS1HASOPENED");
        }


        if (Other.tag == "BUSINESS2SUCCESSCOLLIDER")
        {

            BUSINESS2HASOPENED = true;
            Debug.Log("BUSINESS2HASOPENED");
        }

        if (Other.tag == "BUSINESS3SUCCESSCOLLIDER")
        {

            BUSINESS3HASOPENED = true;
            Debug.Log("BUSINESS3HASOPENED");
        }

        if (Other.tag == "BUSINESS4SUCCESSCOLLIDER")
        {

            BUSINESS4HASOPENED = true;
            Debug.Log("BUSINESS4HASOPENED");
        }

    }
```

```
    private bool IsOpenPanelAllBusinessSuccessActive
    {
        get
        {

            return OpenPanelAllBusinessSuccess.activeInHierarchy;
        }
    }

    void Update()
    {

        if (ENDOFGAME == true)
        {

            restartTimer += Time.deltaTime;


            if (restartTimer >= restartDelay)
            {

                SceneManager.LoadScene(2);
            }
        }
    }


    void OnTriggerExit(Collider Other)
    {
        if (Other.tag == "BUSINESS1SUCCESSCOLLIDER")

        {

            if ((BUSINESS2HASOPENED == true) && (BUSINESS3HASOPENED == true) &&
(BUSINESS4HASOPENED == true))

            {

                OpenPanelAllBusinessSuccess.SetActive(true);
                Debug.Log("ALLBUSINESSHASOPENED-COUNTING DOWN FOR ENDOFGAME
LEVEL!!!!");

                ENDOFGAME = true;
            }
        }


        if (Other.tag == "BUSINESS2SUCCESSCOLLIDER")

        {

            if ((BUSINESS1HASOPENED == true) && (BUSINESS3HASOPENED == true) &&
(BUSINESS4HASOPENED == true))

            {
```

```
                OpenPanelAllBusinessSuccess.SetActive(true);
                Debug.Log("ALLBUSINESSHASOPENED-COUNTING DOWN FOR ENDOFGAME
LEVEL!!!!");

                ENDOFGAME = true;
            }
        }

        if (Other.tag == "BUSINESS3SUCCESSCOLLIDER")

        {

            if ((BUSINESS1HASOPENED == true) && (BUSINESS2HASOPENED == true) &&
(BUSINESS4HASOPENED == true))

            {

                OpenPanelAllBusinessSuccess.SetActive(true);
                Debug.Log("ALLBUSINESSHASOPENED-COUNTING DOWN FOR ENDOFGAME
LEVEL!!!!");

                ENDOFGAME = true;
            }
        }

        if (Other.tag == "BUSINESS4SUCCESSCOLLIDER")

        {

            if ((BUSINESS1HASOPENED == true) && (BUSINESS2HASOPENED == true) &&
(BUSINESS3HASOPENED == true))

            {

                OpenPanelAllBusinessSuccess.SetActive(true);
                Debug.Log("ALLBUSINESSHASOPENED-COUNTING DOWN FOR ENDOFGAME
LEVEL!!!!");

                ENDOFGAME = true;
            }
        }
    }
  }
```

Παράρτημα 5.9

UnderwaterLabyMusic(Script):

```
using System.Collections;
```

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


[RequireComponent(typeof(AudioSource))]
public class UnderwaterLabyMusic : MonoBehaviour
{

     public Slider healthbar;


    void OnTriggerEnter(Collider UnderwaterLabyMusicCollider)
    {
        if (UnderwaterLabyMusicCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Play();
            Debug.Log("UnderwaterLabyMusic");
        }
    }


    void OnTriggerExit(Collider UnderwaterLabyMusicCollider)
    {
        if (UnderwaterLabyMusicCollider.tag == "Player")
        {

            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }
    }


    // Use this for initialization
    void Start()
    {

    }

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

    {

        if (healthbar.value <= 0)
        {
            AudioSource audio = GetComponent<AudioSource>();
            audio.Stop();
        }
    }
  }
```

BackupMusic(Script):

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


[RequireComponent(typeof(AudioSource))]
public class BackupMusic : MonoBehaviour {


    public Slider healthbar;
    public AudioClip audioClip1;
    public float BackupMusicDelay = 5f;
    float BackupMusicTimer;
    private bool CastleMusicActive = false;
    private bool LargeBridgesMusicActive = false;
    private bool Buildings1MusicActive = false;
    private bool Buildings2MusicActive = false;
    private bool AladdinsCarpetMusicActive = false;
    AudioSource audioSource;
    [Range(0, 1)] public float audioClipVolume1;
    private bool UnderwaterLabyMusicActive = false;
    private bool Stairs1MusicActive = false;
    private bool RpgMusicActive = false;
    private bool Catacombs1MusicActive = false;
    private bool Catacombs2MusicActive = false;
    private bool Catacombs3MusicActive = false;
    private bool Cascading = false;



    void Start()
    {

        audioSource = GetComponent<AudioSource>();
        audioSource.loop = true;
        Cascading = false;
    }


    void OnTriggerEnter(Collider Other)
    {
        if (Other.tag == "CastleMusicCollider")
        {

            CastleMusicActive = true;
            audioSource.Stop();
        }


        if (Other.tag == "LargeBridgesMusicCollider")
        {
```

```
            LargeBridgesMusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "Buildings1MusicCollider")
        {

            Buildings1MusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "Buildings2MusicCollider")
        {

            Buildings2MusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "AladdinsCarpetCollider")
        {

            AladdinsCarpetMusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "UnderwaterLabyMusicCollider")
        {

            UnderwaterLabyMusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "Stairs1MusicCollider")
        {

            Stairs1MusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "RpgMusicCollider")
        {

            RpgMusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "Catacombs1MusicCollider")
        {

            Catacombs1MusicActive = true;
            audioSource.Stop();
        }

        if (Other.tag == "Catacombs2MusicCollider")
        {

            Catacombs2MusicActive = true;
            audioSource.Stop();
```

```
        }

        if (Other.tag == "Catacombs3MusicCollider")
        {

            Catacombs3MusicActive = true;
            audioSource.Stop();
        }


        if (Other.tag == "FallingEdgeCollider")
        {

            Cascading = true;
            audioSource.Stop();
        }

    }

    void OnTriggerExit(Collider Other)
    {
        if (Other.tag == "CastleMusicCollider")
        {

            CastleMusicActive = false;
        }

        if (Other.tag == "LargeBridgesMusicCollider")
        {

            LargeBridgesMusicActive = false;
        }

        if (Other.tag == "Buildings1MusicCollider")
        {

            Buildings1MusicActive = false;
        }

        if (Other.tag == "Buildings2MusicCollider")
        {

            Buildings2MusicActive = false;
        }

        if (Other.tag == "AladdinsCarpetCollider")
        {

            AladdinsCarpetMusicActive = false;
        }


        if (Other.tag == "UnderwaterLabyMusicCollider")
        {

            UnderwaterLabyMusicActive = false;
        }
```

```
        if (Other.tag == "Stairs1MusicCollider")
        {

            Stairs1MusicActive = false;
        }

        if (Other.tag == "RpgMusicCollider")
        {

            RpgMusicActive = false;
        }

        if (Other.tag == "Catacombs1MusicCollider")
        {

            Catacombs1MusicActive = false;
        }

        if (Other.tag == "Catacombs2MusicCollider")
        {

            Catacombs2MusicActive = false;
        }

        if (Other.tag == "Catacombs3MusicCollider")
        {

            Catacombs3MusicActive = false;
        }

    }


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

        BackupMusicTimer += Time.deltaTime;



        if ((CastleMusicActive == false) && (LargeBridgesMusicActive == false) &&
(Buildings1MusicActive == false) && (Buildings2MusicActive == false) &&
(AladdinsCarpetMusicActive == false) && (UnderwaterLabyMusicActive == false) &&
(Stairs1MusicActive == false) && (RpgMusicActive == false) && (Catacombs1MusicActive
== false) && (Catacombs2MusicActive == false) && (Catacombs3MusicActive == false) &&
(Cascading == false))
        {

            if (!audioSource.isPlaying)
            {

                if (BackupMusicTimer >= BackupMusicDelay)
                {

                    audioSource.PlayOneShot(audioClip1, audioClipVolume1);
                    Debug.Log("BackupMusicEnabled!!!!");
                }
```

```
            }
        }
    }
}
```

Παράρτημα 5.10

FallingDeath(Script):

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

public class FallingDeath : MonoBehaviour
{

    public GameObject player;
    public Slider healthbar;
    Animator anim;
    private bool Cascading = false;


    void OnTriggerEnter(Collider theCollider)
    {
        if (theCollider.tag == "FallingEdgeCollider")
        {
            Cascading = true;
            healthbar.value -= 100;
            Debug.Log("Cascading");
            Cascading = false;
        }
    }


    // Use this for initialization
    void Start()
    {
        anim = GetComponent<Animator>();
        Cascading = false;
    }


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

    }
}
```

Παράρτημα 5.11

LightFlicker(Script):

```
#pragma strict

var flickerSpeed : float = 0.07;
private var randomizer : int = 0;
while (true) {
    if (randomizer == 0) {
        GetComponent.<Light>().enabled = true;
    }
    else GetComponent.<Light>().enabled = false;
    randomizer = Random.Range (0, 1.1);
    yield WaitForSeconds (flickerSpeed);

  }
```

AladdinsCarpetFlying(Script):

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


[RequireComponent(typeof(AudioSource))]
public class AladdinsCarpetFlying : MonoBehaviour {


    public GameObject TheAladdinsCarpet;


    // Use this for initialization
    void Start ()
    {

    }


    void Update()
    {

    }

    void OnTriggerEnter(Collider AladdinsCarpetCollider)
    {
```

```
        if (AladdinsCarpetCollider.tag == "Player")
        {

            TheAladdinsCarpet.GetComponent<Animation>().Play();
            AudioSource audio = GetComponent<AudioSource>();
            audio.Play();
            Debug.Log("AladdinsCarpet");
        }
    }

    void OnTriggerExit(Collider AladdinsCarpetCollider)
    {

          if (AladdinsCarpetCollider.tag == "Player")
          {
              AudioSource audio = GetComponent<AudioSource>();
              audio.Stop();
          }
    }
  }
```

CustomObjectDestroyer(Script):

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



    public class CustomObjectDestroyer : MonoBehaviour
{

    public DestructibleObject _targetForDestruction;


    private void Start()
    {

    }

    private void Update()
    {

    }


    void OnTriggerEnter(Collider theCollider)
    {


        if (theCollider.tag == "Player")
        {
```

```
            _targetForDestruction.SimpleDestroyObject();
        }
    }
}
```

ControlInformation(Script):

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


public class ControlInformation : MonoBehaviour
{

    public GameObject OpenPanelControlInformationDetails;

    void Start()
    {

        OpenPanelControlInformationDetails.SetActive(false);
    }


    void OnTriggerEnter(Collider ControlInformationDetailsCollider)
    {
        if (ControlInformationDetailsCollider.tag == "Player")
        {

            OpenPanelControlInformationDetails.SetActive(true);
        }
    }

    void OnTriggerExit(Collider ControlInformationDetailsCollider)
    {
        if (ControlInformationDetailsCollider.tag == "Player")
        {

            OpenPanelControlInformationDetails.SetActive(false);
            Destroy(gameObject);
        }
    }

    private bool IsOpenPanelBusiness1Active
    {
        get
        {

            return OpenPanelControlInformationDetails.activeInHierarchy;
        }
    }
```

```
    void Update()
    {

    }
  }
```

PauseGame(Script):

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

public class PauseScript : MonoBehaviour {


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

        if (Input.GetKeyDown(KeyCode.Escape))
        {

            if (Time.timeScale == 1)
            {

                Time.timeScale = 0;
                Debug.Log("Paused");
            }

            else
                Time.timeScale = 1;

        }
       }
}
```