ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

<<Δημιουργία παιχνιδιού με τη Unity>>

Του φοιτητή Ποτουρίδη Ιωάννη
Αρ. Μητρώου: 052904

Επιβλέπων Καθηγητής
Ράπτης Πασχάλης

Θεσσαλονίκη 2015

ΠΡΟΛΟΓΟΣ

Υπάρχουν πολλοί τύποι βιντεοπαιχνιδιών. Η πτυχιακή αυτή εστιάζει στα **Independent Video Games**, που συνήθως αναφέρονται ως **Indie Games**. Τα Indie Games είναι βιντεοπαιχνίδια που δημιουργούνται από άτομα ή μικρές ομάδες χωρίς να έχουν χρηματοδοτική στήριξη από κάποιον εκδότη βιντεοπαιχνιδιών. Συχνά εστιάζουν στην καινοτομία και βασίζονται στην ψηφιακή διανομή. Το Indie Gaming έχει αρχίσει να γίνεται δημοφιλές από το 2000 και έπειτα, κυρίως λόγω των νέων μεθόδων διανομής μέσω Internet και των εργαλείων ανάπτυξης. Σκοπός αυτής της πτυχιακής είναι η δημιουργία ενός Indie Game με τη χρήση του **Unity** σαν εργαλείο ανάπτυξης. Στις επόμενες σελίδες θα παρουσιαστεί αναλυτικά η ανάπτυξη ενός FPS (First Person Shooter) Indie Horror Game από το μηδέν και θα αντιμετωπιστούν διάφορα προβλήματα που παρουσιάζονται κατά την ανάπτυξη ενός τέτοιου παιχνιδιού.

ΠΕΡΙΛΗΨΗ

Ποιος θα είναι ο σκοπός του παιχνιδιού; Ποια **μηχανή γραφικών** θα χρησιμοποιηθεί; Προορίζεται για εμπορική χρήση ή όχι; Θέτονται πολλά ερωτήματα όταν κάποιος θέλει να ξεκινήσει την δημιουργία ενός βιντεοπαιχνιδιού και το τελικό παιχνίδι εξαρτάται σημαντικά από τις απαντήσεις. Η διαδικασία σχεδίασης και υλοποίησης του παιχνιδιού όμως στις περισσότερες περιπτώσεις είναι παρόμοια.

Όπως και στην δημιουργία του κόσμου, έτσι και στην δημιουργία ενός παιχνιδιού είναι καλό να ξεκινήσει κανείς με το περιβάλλον. Ξεκινάει με την δημιουργία γενικών και άχρωμων 3D σχημάτων τα οποία σιγά σιγά παίρνουν **μορφή**, **χρώμα** και **υφή** και συμπληρώνουν τις λεπτομέρειες στον εικονικό κόσμο.

Έπειτα, αφού ολοκληρωθεί το περιβάλλον μπαίνει ο κατάλληλος **φωτισμός** και αρχίζει να τοποθετείται η **ζωή** πάνω στον κόσμο. Η ζωή σε ένα παιχνίδι επιτυγχάνεται με την τοποθέτηση αντικειμένων που έχουν **συμπεριφορά,** μπορούν να **αλληλεπιδρούν** με άλλα αντικείμενα που βρίσκονται στον κόσμο και καμιά φορά μπορεί να **αντιδράσουν** με την συμπεριφορά άλλων αντικειμένων.

Τέλος, αφού τοποθετηθούν όλα στην θέση τους και το **αποτέλεσμα** είναι καλό δεν μένει τίποτα άλλο από το να παίξουμε. Βέβαια η ανάπτυξη του παιχνιδιού δεν σταματάει εκεί καθώς παίζοντας, **ελέγχεται** το παιχνίδι και έρχονται νέες ιδέες για **επέκταση** ή **βελτίωση** του.

ABSTRACT

What is the purpose of the game? Which **game engine** is going to be used? Is it intended for commercial use or not? Questions like these must concern new game developers and the final game considerably depends on the answers. The design and implementation process of the game though are similar in most cases.

As in the creation of our world, so in game development it is considered good start to begin with the environment. It begins with the creation of general and colorless 3D shapes which slowly passes **form**, **color** and **texture** and complete the details in the game's virtual world.

Then, after completing the environment and the appropriate **lighting** has been added, **life** is placed upon the world. In-game life can be achieved by placing objects with certain **behavior**, which can **interact** with other objects in the world and can sometimes **react** to other objects' behavior.

Finally, after everything is placed in position and the **result** is good in the eye, there's nothing left other than to play. The development of the game though does not stop there. While playing, the game is being **tested** for further **expansion** or **improvement**.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία με θέμα «Δημιουργία παιχνιδιού με τη Unity», πραγματοποιήθηκε, στο πλαίσιο της πτυχιακής εργασίας του τμήματος Μηχανικών Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης.

Στο σημείο αυτό αισθάνομαι την ανάγκη να εκφράσω τις ειλικρινείς και θερμές ευχαριστίες μου σε όσους συνέβαλαν στην ολοκλήρωση αυτής της προσπάθειας.

Θα ήθελα να ευχαριστίσω όλους αυτούς που μοιράζονται τις γνώσεις τους πάνω σε κάποια πράγματα, στην προκειμένη περίπτωση πάνω στο SketchUp, στο Unity και στη Javascript, μέσω βίντεο στο YouTube ή σε διάφορα Forums, χωρίς να έχουν κανένα όφελος, για την πολύτιμη βοήθεια τους σχετικά με το υλικό αλλά και με την διευθέτηση της πτυχιακής μου εργασίας και συνέβαλαν στην εκπλήρωση του στόχου μου, την ολοκλήρωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Αν θέλει κάποιος να αναπτύξει και να προωθήσει το δικό του παιχνίδι έξω στον κόσμο, υπάρχουν μερικά πολύ σημαντικά πράγματα που πρέπει να εξετάσει πριν από την έναρξη του σε αυτό το ταξίδι. Υπάρχουν πολλές δωρεάν μηχανές παιχνιδιού που έχουν τα χαρακτηριστικά που χρειάζεται κανείς για να δημιουργήσει το δικό του παιχνίδι, αλλά το ερώτημα είναι ποια θα πρέπει να επιλέξει;

Βιντεοπαιχνίδι είναι κάθε μορφή ηλεκτρονικού παιχνιδιού, με το οποίο μπορεί κάθε άνθρωπος να έρθει σε επικοινωνία με μια <<διεπαφή χρήστη>> και να διασκεδάσει/ψυχαγωγηθεί μέσω μιας ηλεκτρονικής συσκευής όπως η τηλεόραση ή ο ηλεκτρονικός υπολογιστής. Ο όρος <<βιντεοπαιχνίδι>> (Εικόνα 1), μας παραπέμπει σε κάθε ηλεκτρονική συσκευή η οποία δύναται να αναπαραστήσει δισδιάστατα ή τρισδιάστατα γραφικά.

Εικόνα 1 "Video Games"

Λίγα χρόνια αργότερα μετά την άφιξη των τηλεοράσεων στη ζωή μας, τα βιντεοπαιχνίδια έκαναν την πρώτη τους εμφάνιση σε κονσόλες φορητές και μη, "pinballs" και υπολογιστές, κάπου στα τέλη της δεκαετίας του '40. Επισήμως όμως τα πρώτα παιχνίδια που εμφανίστηκαν, ήταν τα "pinball" ή "flipper" (Εικόνα 2) γύρω στο '70 και αργότερα τα υπόλοιπα σε υπολογιστές και κονσόλες. Στις μέρες μας πλέον έχουν εγκατασταθεί και στα κινητά μας τηλέφωνα.

Εικόνα 2 "Pinball"

Από την πρώτη κιόλας επίσημη εμφάνιση των βιντεοπαιχνιδιών, τα πρωτεία κρατούσαν κυρίως ιαπωνικές εταιρείες όπως η Sony και η Nintendo. Για την ανάπτυξη ενός παιχνιδιού, κάθε εταιρεία κάνει χρήση μιας μηχανής δημιουργίας - παραγωγής βιντεοπαιχνιδιών. Μια μηχανή παιχνιδιού είναι στην ουσία ένα σύστημα λογισμικού σχεδιασμένο να δουλεύει είτε σε κονσόλες, είτε σε λειτουργικά συστήματα ηλεκτρονικών υπολογιστών, όπως Microsoft Windows, Linux και Mac OS X.

Κατά τα τελευταία χρόνια κυκλοφόρησαν πολλές ισχυρές μηχανές παιχνιδιού, δίνοντας την ευκαιρία στους επίδοξους ανεξάρτητους προγραμματιστές να δημιουργήσουν το παιχνίδι είχαν πάντα στο μυαλό τους. Οι πιο δημοφιλείς μηχανές παιχνιδιού είναι οι **Unity**, **Unreal Engine 4** και **CryEngine**. Και οι τρεις είναι εξαιρετικά ισχυρές μηχανές παιχνιδιού και κάθε μια έχει τα θετικά της. Για να προσδιοριστεί ποια λειτουργεί καλύτερα για το έργο κάποιου, θα πρέπει να αναρωτηθεί τι είδους παιχνίδι θα φτιάξει. Είναι ένα First Person Shooter (FPS); Ένα **Mobile Game**; Πρόκειται να είναι **2D** ή **3D**;

ΚΕΦΑΛΑΙΟ 1

ΕΙΔΟΣ ΠΑΙΧΝΙΔΙΟΥ

ΕΙΣΑΓΩΓΗ

Στην πτυχιακή αυτή έχει επιλεχθεί να αναπτυχθεί ένα FPS Indie Horror Game χρησιμοποιώντας το Unity σαν εργαλείο ανάπτυξης. Το **First Person Shooter (FPS)** είναι ένα είδος τρισδιάστατου βιντεοπαιχνιδιού που επικεντρώνεται στην προοπτική πρώτου προσώπου. Ο παίκτης βιώνει τη δράση μέσα από τα μάτια του πρωταγωνιστή, και σε ορισμένες περιπτώσεις, του ανταγωνιστή. Δεν έχουν καμιά σχέση με τα παιχνίδια τρίτου προσώπου, στα οποία ο παίκτης μπορεί να δει (συνήθως από πίσω) το χαρακτήρα που ελέγχει. Το κύριο στοιχείο του σχεδιασμού είναι η μάχη, που αφορούν κυρίως τα πυροβόλα όπλα. Συχνά εστιάζουν τη δράση του παιχνιδιού, με γρήγορο ρυθμό και αιματηρή ανταλλαγή πυρών. Ωστόσο κάποια δίνουν μεγαλύτερη έμφαση στην αφήγηση, την επίλυση προβλημάτων και τη λογική πάζλ, όπως το παιχνίδι αυτής της εργασίας.

Εικόνα 3 "Doom, ένα από τα πρωτοπορειακά παιχνίδια του είδους"

Επειδή λαμβάνουν χώρα σε ένα 3D περιβάλλον, αυτά τα παιχνίδια έχουν την τάση να είναι κάπως πιο ρεαλιστικά από τα υπόλοιπα είδη παιχνιδιών και να έχουν πιο ακριβή αναπαραστάσεις της βαρύτητας, του φωτισμού, του ήχου και των συγκρούσεων (**collisions**). Τα βιντεοπαιχνίδια που παίζονται σε προσωπικούς υπολογιστές ελέγχονται συνδυάζοντας το πληκτρολόγιο και το ποντίκι. Ο συγκεκριμένος τρόπος ελέγχου του FPS παιχνιδιού είναι

δημοφιλέστερος από εκείνον που χρησιμοποιούν οι κονσόλες, δηλαδή αναλογικούς μοχλούς, λόγω ευκολίας χειρισμού. Είναι συνηθισμένο να εμφανίζονται τα χέρια του πρωταγωνιστή και τα όπλα στην κύρια προβολή, η υγεία, τα πυρομαχικά, η τοποθεσία και διάφορες λεπτομέρειες ψηλά στην οθόνη. Επίσης, είναι δυνατόν να φαίνεται και ένας χάρτης της γύρω περιοχής.

Τα First-person Shooter μπορεί να αποτελούνται από διάφορα δομικά επίπεδα ή να χρησιμοποιούν την τεχνική της συνεχούς αφήγησης στην οποία το παιχνίδι δεν εγκαταλείπει ποτέ την προοπτική πρώτου προσώπου. Άλλα πάλι διαθέτουν μεγάλα περιβάλλοντα **sandbox**, τα οποία δεν είναι χωρισμένα σε επίπεδα αλλά μπορούν να εξερευνηθούν ελεύθερα.

Τέλος, οι πρωταγωνιστές μπορούν να αλληλεπιδρούν με το περιβάλλον σε διάφορους βαθμούς. Από τα βασικά, όπως το άνοιγμα και το κλείσιμο μιας πόρτας ή η συλλογή αντικειμένων, που περιλαμβάνονται και στην παρούσα εργασία και θα αναλυθούν αργότερα, μέχρι και την επίλυση προβλημάτων πάζλ που βασίζονται σε μια ποικιλία από διαδραστικά αντικείμενα.

ΚΕΦΑΛΑΙΟ 2

ΜΗΧΑΝΗ ΠΑΙΧΝΙΔΙΟΥ

ΕΙΣΑΓΩΓΗ

Μια μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών. Υπάρχουν πολλές μηχανές παιχνιδιών οι οποίες είναι σχεδιασμένες να δουλεύουν σε κονσόλες βιντεοπαιχνιδιών και λειτουργικά συστήματα επιτραπέζιων υπολογιστών όπως τα Microsoft Windows, το Linux και το Mac OS X.

Η κεντρική λειτουργικότητα που παρέχεται τυπικά από μια μηχανή παιχνιδιού περιλαμβάνει μια μηχανή φωτοαπόδοσης (renderer) για 2D ή 3D γραφικά, μια μηχανή φυσικής ή εντοπισμού συγκρούσεων (collision detection, καθώς και collision response), ήχο, scripting, animation, τεχνητή νοημοσύνη, δικτύωση, streaming, διαχείριση μνήμης, νήματα (threading), υποστήριξη τοπικοποίησης και ένα γράφο σκηνής (scene gragh). Η διαδικασία της ανάπτυξης παιχνιδιού συχνά οικονομικοποιείται με το ότι σε μεγάλο μέρος η ίδια μηχανή παιχνιδιού επαναχρησιμοποιείται για να δημιουργηθούν διαφορετικά παιχνίδια.

Πριν από τις μηχανές παιχνιδιών, τα παιχνίδια τυπικά γραφόντουσαν ως μοναδικές οντότητες. Ένα παιχνίδι για το Atari 2600 (Εικόνα 4), για παράδειγμα, έπρεπε να σχεδιαστεί από το μηδέν για να κάνει βέλτιστη χρήση του υλικού εμφάνισης.

Εικόνα 4 "Atari 2600"

Ο όρος μηχανή παιχνιδιού ήρθε στην επιφάνεια στα μέσα της δεκαετίας του 1990, ειδικά με 3D παιχνίδια όπως τα παιχνίδια βολών πρώτου προσώπου (FPS). Η δημοτικότητα των παιχνιδιών Doom & Quake ανάγκασε καινούριους Developers αντί να δουλέψουν από μηδενική βάση να πάρουν την άδεια από τον λειτουργικό

κώδικα του λογισμικού και σχεδίασαν τα δικά τους γραφικά, χαρακτήρες, όπλα και επίπεδα – το περιεχόμενο του παιχνιδιού ή τα στοιχεία του παιχνιδιού, βασικές έννοιες όμως όπως ο έλεγχος σύγκρουσης και οι οντότητες του παιχνιδιού δεν σήμαινε ότι οι δεν μπορούσαν να μεγαλώσουν και να ειδικευτούν.

Οι μοντέρνες μηχανές παιχνιδιών είναι μερικές από τις πιο πολύπλοκες εφαρμογές για γράψιμο και συχνά χαρακτηρίζονται ως τελειοποιημένα συστήματα που αλληλεπιδρούν για να εγγυώνται μια ακριβώς ελεγχόμενη εμπειρία χρήστη. Η συνεχής εξέλιξη των μηχανών παιχνιδιών έχει δημιουργήσει ένα ισχυρό διαχωρισμό ανάμεσα στο rendering, το scripting, την τέχνη και το σχεδιασμό επιπέδων. Τυχαίνει όμως, μια τυπική ομάδα ανάπτυξης παιχνιδιού να έχει πολλές φορές περισσότερους καλλιτέχνες απ’ ότι προγραμματιστές.

ΕΠΙΛΟΓΟΣ

Τα παιχνίδια FPS παραμένουν οι κυρίαρχοι χρήστες μηχανών παιχνιδιών τρίτων μερών, αλλά τώρα επίσης χρησιμοποιούνται και σε άλλα είδη. Για παράδειγμα το RPG The Elder Scrolls IIIL Morrowind και το MMORPG Dark Age of Camelot βασίζονται στην μηχανή Gamebryo, και το MMORPG Lineage II (Εικόνα 5) βασίζεται στην Unreal Engine.

Εικόνα 5 "Lineage II"

ΚΕΦΑΛΑΙΟ 3

UNITY

ΕΙΣΑΓΩΓΗ

Η Unity (Εικόνα 4) είναι μια από τις πιο γνωστές μηχανές κατασκευής βιντεοπαιχνιδιών σε πολλές και διαφορετικές πλατφόρμες (cross-platform) και αποτελεί δημιουργία της εταιρείας Unity Technologies.

Εικόνα 6 "Το λογότυπο της Unity"

UNITY (ΜΗΧΑΝΗ ΠΑΙΧΝΙΔΙΟΥ)

Η Unity περιλαμβάνει μια μηχανή κατασκευής βιντεοπαιχνιδιών και ένα ενοποιημένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE). Χρησιμοποιείται στο σύνολό της για τη δημιουργία και κυκλοφορία βιντεοπαιχνιδιών σε ιστότοπους, φορητούς και μη - ηλεκτρονικούς υπολογιστές, κονσόλες όπως Playstation, Xbox, καθώς και για Smartphones.

Αρχικά η Unity κυκλοφόρησε για λειτουργικά συστήματα Mac παράλληλα με την εμφάνισή της στο Worldwide Developers Conference, με την Apple ως διοργανώτρια και μετέπειτα στόχευσε την επέκτασή της σε παραπάνω από δεκαπέντε πλατφόρμες. Φυσικά είναι αξιοσημείωτη η δυνατότητα που προσφέρει η συγκεκριμένη μηχανή στον εκάστοτε Developer να δημοσιεύσει το παιχνίδι του σε οποιουδήποτε είδους πλατφόρμα επιθυμεί ο ίδιος.

ΤΟ ΣΥΣΤΗΜΑ ΤΗΣ UNITY

Η Unity περιλαμβάνει ένα διακομιστή (server) για την αποθήκευση των διαφόρων αρχείων, καθώς και τη μηχανή προσομοίωσης φυσικής PhysX της Nvidia (Εικόνα 5). Για την καλύτερη χρήση των γραφικών στα παιχνίδια της, απαιτεί την ύπαρξη Direct3D για Windows και Xbox, OpenGL για Mac OS, Linux, Android και iOS.

Εικόνα 7 "Το λογότυπο της PhysX (Nvidia)"

Η Unity παρέχει τη δυνατότητα συμπίεσης και αλλαγής της ανάλυσης αρχείων, εικόνων και γραφικών γενικότερα, ανάλογα με τις απαιτήσεις κάθε πλατφόρμας. Η μηχανή της είναι φτιαγμένη με Mono που αποτελεί την εφαρμογή ανοιχτού κώδικα της Net Framework. Ο προγραμματιστής μπορεί να δημιουργήσει το παιχνίδι του σε τρεις διαφορετικές γλώσσες προγραμματισμού, C#, Javascript και Boo.

CROSS – PLATFORM ΜΗΧΑΝΗ ΠΑΙΧΝΙΔΙΟΥ

Ο όρος **Cross - platform** δηλώνει πως η Unity μπορεί και κυκλοφορεί τα παιχνίδια που παράγονται με αυτή, σε πολλές και διαφορετικού είδους μεταξύ τους κονσόλες. Πιο συγκεκριμένα, οι πλατφόρμες που υποστηρίζονται από την Unity είναι οι εξής:

- BlackBerry 10
- Windows, Windows Phone 8
- iOS, iOS X
- Android
- Linux
- Adobe Flash
- Playstation 3, 4, Vita
- Xbox 360, One

- Wii, Wii U
- Ιστότοπους και <<Facebook>> (με απαραίτητη εγκατάσταση του Unity Web Player).

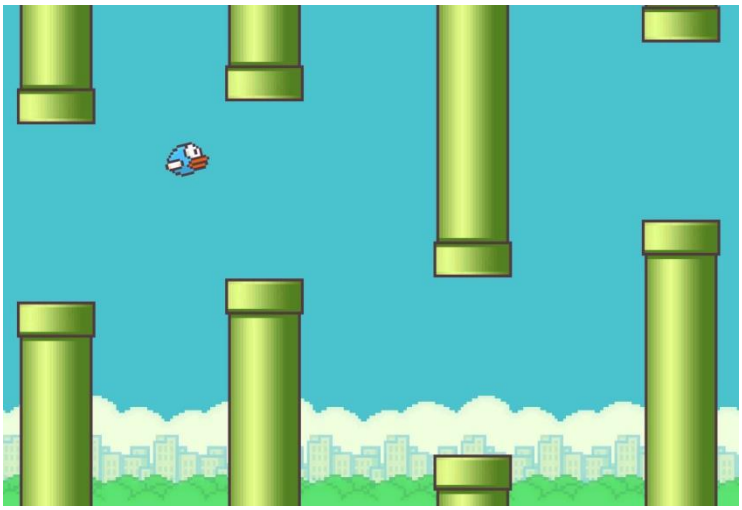
Η Unity τέλος, αποτελεί το προκαθορισμένο πλέον λογισμικό πακέτο (Software Development Kit - SDK) για την κονσόλα Wii U της Nintendo. Μαζί παρέχεται μια ελεύθερη άδεια για κάθε προγραμματιστή παιχνιδιών της κονσόλας Wii U από την ίδια την Nintendo.

UNITY TECHNOLOGIES

Η εταιρεία Unity Technologies είναι ο δημιουργός της Unity. Ιδρύθηκε το 2004 από τους David Helgason, Nicholas Francis και Joachim Ante στην Κοπεγχάγη, Δανία. Οι τρεις τους αναγνώρισαν την αξία των μηχανών κατασκευής παιχνιδιών καθώς και των εργαλείων ανάπτυξής τους και αποφάσισαν έτσι, να δημιουργήσουν μια μηχανή η οποία να είναι και φθηνή και προσιτή γενικά στο κοινό των προγραμματιστών.

Η επιτυχία της μηχανής τους οφείλεται στο γεγονός ότι στηρίζει τους ανεξάρτητους Developers σε όλο τον κόσμο, οι οποίοι δεν είναι σε θέση να δημιουργήσουν τη δικιά τους μηχανή παιχνιδιών, είτε να αποκτήσουν άδειες από άλλες μηχανές για να εκμεταλλευτούν πλήρως διάφορες δυνατότητες. Ο σκοπός της Unity Technologies είναι μια Δημοκρατική Ανάπτυξη Βιντεοπαιχνιδιών και η κατασκευή δισδιάστατων (2D) και τρισδιάστατων (3D) γραφικών να είναι προσιτή σε όσο δυνατόν περισσότερους ανθρώπους ανά τον κόσμο γενικότερα.

Η Unity Technologies βγήκε στην επιφάνεια κυρίως με την άφιξη των iPhone της Apple στην αγορά, το 2009, καθώς η μηχανή τους ήταν από τις πρώτες που παρείχαν πλήρη υποστήριξη στη συγκεκριμένη πλατφόρμα της Apple. Σύμφωνα με μια έρευνα προγραμματιστών, η Unity αποτελεί το εργαλείο ανάπτυξης παιχνιδιών για πάνω από το πενήντα τοις εκατό των προγραμματιστών που ασχολούνται με τη δημιουργία παιχνιδιών για κινητά τηλέφωνα, με χιλιάδες παιχνίδια να κυκλοφορούν σήμερα για συσκευές Android και iOS όπως τα Flappy Bird (Εικόνα 8) και Temple Run (Εικόνα 9) που είναι από τα πιο γνωστά.



Εικόνα 8 "Flappy Bird"

Εικόνα 9 "Temple Run"

Το 2009 η εταιρεία παραχώρησε στο προγραμματιστικό κοινό μια δωρεάν έκδοση της Unity, με αποτέλεσμα ο αριθμός των developers να αυξάνεται ραγδαία, ώσπου να γνωρίσει μεγάλη επιτυχία τέσσερα χρόνια αργότερα με τον αριθμό των προγραμματισμών της να φτάνει τα δύο εκατομμύρια, από τους οποίους οι τριακόσιοι χιλιάδες χρησιμοποιούν τη μηχανή σε μηνιαία βάση.

ΠΑΙΧΝΙΔΙΑ ΑΝΕΠΤΥΓΜΕΝΑ ΜΕ ΤΗ UNITY

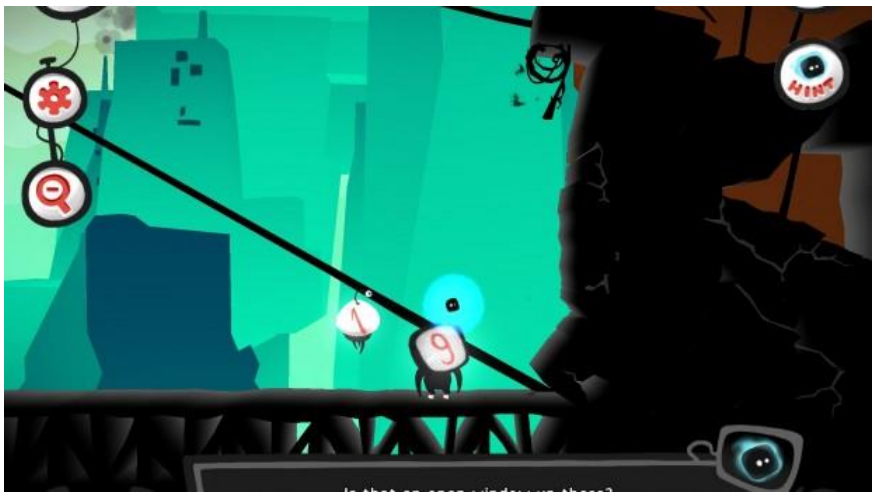
Το πρώτο παιχνίδι των δημιουργών της Unity ήταν το GooBall (Εικόνα 10) με το οποίο δε γνώρισαν την επιτυχία που επιθυμούσαν. Αμέσως μετά την επίσημη κυκλοφορία της μηχανής, εκατοντάδες τίτλοι έκαναν την εμφάνισή τους με αποτέλεσμα να εκτινάξουν την Unity στα ύψη.

Εικόνα 10 "GooBall"

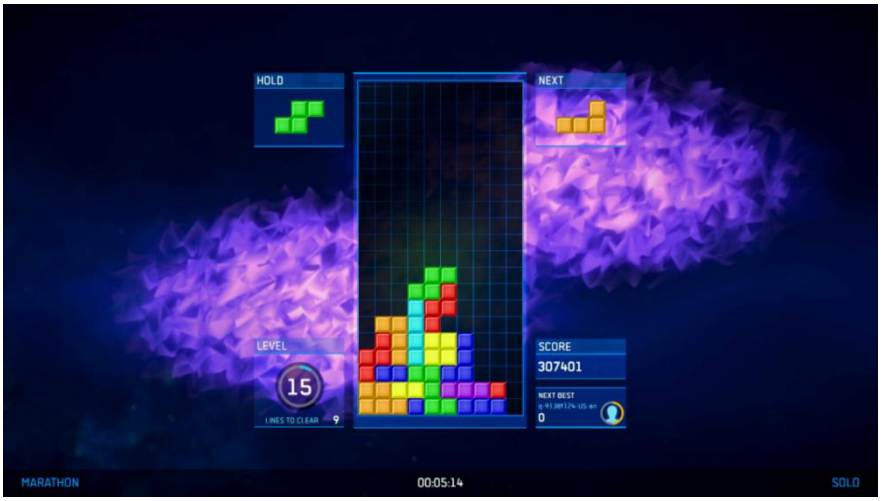
2D - Δισδιάστατα παιχνίδια

Μερικοί γνωστοί τίτλοι δισδιάστατου περιβάλλοντος ανεπτυγμένοι είτε από developers της εταιρείας, είτε ανεξάρτητους:

- Twelve a Dozen. Είδος: Platformer, Puzzle, της Bossa Studios, για iOS συσκευές. (Εικόνα 11)
- Cuphead. Είδος: Platformer, Shooter, της StudioMHR, για Linux, Mac, Windows, Xbox One.
- Tetris Ultimate. Είδος: Puzzle, της SoMa Play, για PSVITA, PS4, Windows, Xbox One. (Εικόνα 12)
- Pavilion. Είδος: Adventure, Puzzle, της Visiontrick, για PSVITA, PS4.
- Paradise Lost. Είδος: Adventure, Platformer, της Ashtree Works, για Linux, Mac, Wii U, Windows.



Εικόνα 11 "Twelve a Dozen"



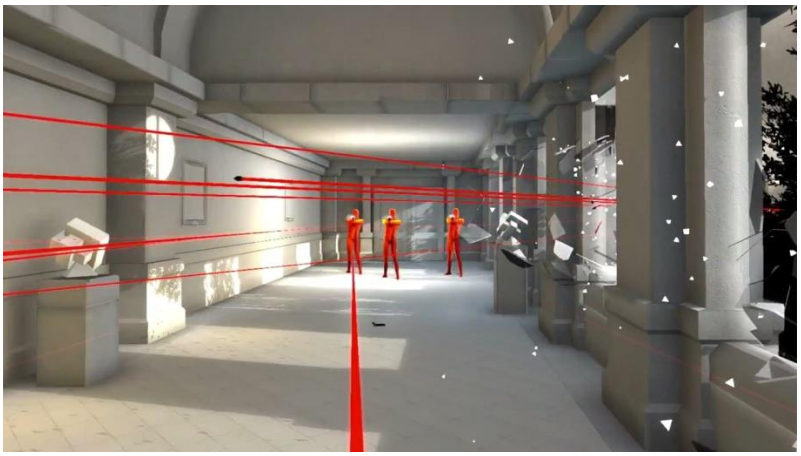
Εικόνα 12 "Tetris Ultimate"

3D - Τρισδιάστατα Παιχνίδια

Παρακάτω παρατίθενται μερικοί γνωστοί τίτλοι τρισδιάστατου περιβάλλοντος ανεπτυγμένοι με Unity:

- Wasteland 2. Είδος: RPG, της inXile Entertainment, για Linux, Mac, Windows. (Εικόνα 13)
- Superhot. Είδος: Action, Puzzle, Shooter, της Superhot Team, για Linux, Mac, Oculus Rift, Windows. (Εικόνα 14)
- Lucky's Tale. Είδος: MOBA, των Hammer & Chisel, για iOS.
- Shadow Blade. Είδος: Action, Platformer, της DeadMage για iOS.

Εικόνα 13 "Wasteland 2"



Εικόνα 14 "Superhot"

ΚΕΦΑΛΑΙΟ 4

MONODEVELOP

ΕΙΣΑΓΩΓΗ

Η Unity προσφέρει γενικά πολλές δυνατότητες. Για να βγει όμως ένα καλό αποτέλεσμα όσον αφορά το σύνολο ενός παιχνιδιού θα πρέπει να υπάρχει και κώδικας που να εξυπηρετεί τις διάφορες απαιτήσεις του παιχνιδιού αυτού. Το

MonoDevelop (Εικόνα 15) είναι το πρόγραμμα το οποίο εξυπηρετεί αυτές τις απαιτήσεις και προσφέρει στον προγραμματιστή τη δυνατότητα να γράψει τον κώδικα για το παιχνίδι του.

Εικόνα 15 "MonoDevelop"

Η Unity σε συνδιασμό με το MonoDevelop υποστηρίζει τρεις γλώσσες προγραμματισμού για την κατασκευή παιχνιδιών. Αυτές είναι:

- C# (C Sharp): Αντικειμενοστραφής γλώσσα προγραμματισμού, ανεπτυγμένη από την Microsoft.
- JavaScript: Διερμηνευμένη γλώσσα προγραμματισμού επηρεασμένη από τη C και ανεπτυγμένη από τη Netscape.
- Boo: Στατική γενικού σκοπού γλώσσα προγραμματισμού εμπνευσμένη από τη σύνταξη της Python και ανεπτυγμένη από τον Rodrigo B. De Oliveira.

Ο λόγος που οι τρεις αυτές γλώσσες προγραμματισμού έχουν υιοθετηθεί από τη Unity και αποτελεί το βασικό τους κοινό χαρακτηριστικό, είναι πως και οι τρεις είναι αντικειμενοστραφείς.

Το MonoDevelop είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment – IDE) ανεξάρτητο από τη Unity, ωστόσο συνεργάζεται μαζί με τη μηχανή για τον προγραμματισμό παιχνιδιών. Υπάρχουν πολλά εργαλεία – προγράμματα που βοηθούν στην ανάπτυξη ενός παιχνιδιού.

ΚΕΦΑΛΑΙΟ 5

ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΠΑΙΧΝΙΔΙΟΥ

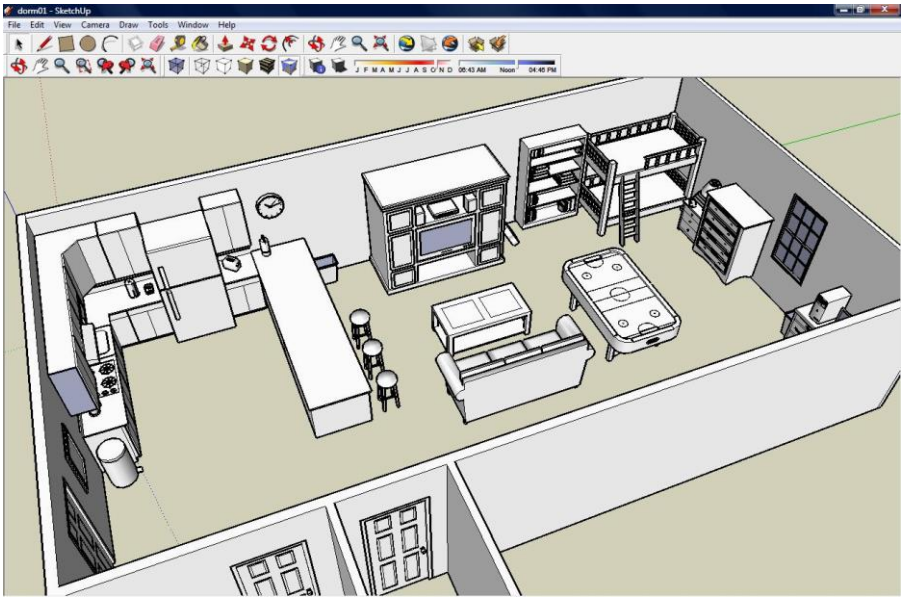
ΕΙΣΑΓΩΓΗ

Ένα εργαλείο ανάπτυξης παιχνιδιών είναι μια εξειδικευμένη εφαρμογή λογισμικού που βοηθά ή διευκολύνει την κατασκευή ενός βιντεοπαιχνιδιού. Μερικές διεργασίες που χειρίζονται αυτά τα εργαλεία περιλαμβάνουν τη μετατροπή των assets (όπως 3D μοντέλα, textures, κ.λπ.) σε μορφές που απαιτούνται από το παιχνίδι.

Σχεδόν όλα τα εργαλεία ανάπτυξης παιχνιδιών αναπτύχθηκαν από τους δημιουργούς για ένα παιχνίδι. Ωστόσο μπορούν να χρησιμοποιηθούν εκ νέου για επόμενα παιχνίδια. Προγράμματα επεξεργασίας 3D όπως Maya, 3D Studio Max και 2D όπως το Photoshop, και IDE όπως το Microsoft Visual Studio ενώ βοηθούν στην ανάπτυξη ενός παιχνιδιού, δεν θεωρούνται εργαλεία ανάπτυξης παιχνιδιών δεδομένου ότι έχουν χρήσεις πέρα από την ανάπτυξη παιχνιδιών.

ΧΡΗΣΗ

Πολλά εργαλεία μπορούν να χρησιμοποιηθούν για να βοηθήσουν στην ανάπτυξη του παιχνιδιού. Συχνά οι προγραμματιστές χρησιμοποιούν εργαλεία για να μετατρέψουν μορφές 3D μοντέλων και εικόνες γραφικών μορφές που μπορούν να εισαχθούν στο παιχνίδι. Άλλα εργαλεία, όπως το Google SketchUp (Εικόνα 16) που χρησιμοποιήθηκε στην πτυχιακή αυτή, χρησιμοποιούνται για να δημιουργήσουν τα περιβάλλοντα που θα χρησιμοποιηθούν στο παιχνίδι. Για ένα σύγχρονο εμπορικό παιχνίδι μπορεί να χρησιμοποιηθούν καμιά φορά και περισσότερα από δέκα τέτοια εργαλεία για να βοηθήσουν στη διαδικασία ανάπτυξης παιχνιδιού.



Εικόνα 16 "Google SketchUp"

ΚΕΦΑΛΑΙΟ 6

ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ

ΕΙΣΑΓΩΓΗ

Το κεφάλαιο αυτό αποτελεί το πρακτικό μέρος αυτής της πτυχιακής εργασίας. Εδώ περιγράφεται η διαδικασία ανάπτυξης ενός FPS Indie Horror Game για υπολογιστή με τη βοήθεια αρκετών εργαλείων ανάπτυξης παιχνιδιών και της Unity. Ο σκοπός δημιουργίας του ακόλουθου παιχνιδιού είναι να δείξει πως μπορεί κάποιος να κατασκευάσει ένα παιχνίδι από την αρχή απλά και εύκολα, εκμεταλλευόμενος διάφορες δυνατότητες της Unity.

ΠΕΡΙΛΗΨΗ ΠΑΙΧΝΙΔΙΟΥ

Στο συγκεκριμένο παιχνίδι ο χρήστης ελέγχει έναν χαρακτήρα που έχει χάσει ένα στοίχημα και πρέπει τώρα να επισκεφτεί ένα εγκαταλελειμμένο κτήριο με σκοπό να μαζέψει κάποιες σελίδες τις οποίες άφησε νωρίτερα ο φίλος του. Όσο μαζεύει όμως τις σελίδες στο σκοτεινό αυτό μέρος, θα συνειδητοποιήσει ότι δεν είναι μόνος και ότι το μέρος είναι στοιχειωμένο.

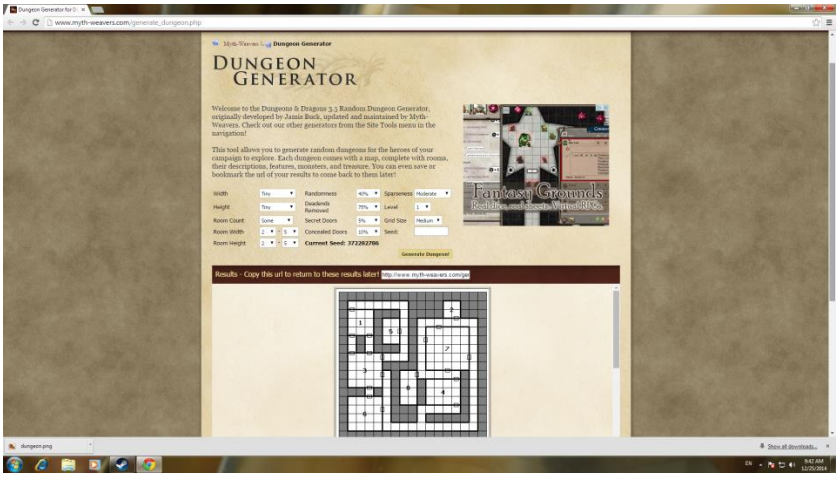
Ο χειρισμός του παιχνιδιού είναι απλός. Ο χαρακτήρας κινείται στον χώρο του παιχνιδιού με τα βελάκια ενώ μπορεί να αλληλεπιδράσει με μερικά αντικείμενα, π.χ. άνοιγμα πόρτας, συλλογή σελίδας, με το πάτημα ενός πλήκτρου. Ο αντίπαλος του χαρακτήρα, ένας εχθρός (π.χ. φάντασμα) πλησιάζει τον χαρακτήρα του παίκτη

όσο αυτός έχει στην κατοχή του σελίδες και η απόστασή τους ελαττώνεται όσο περισσότερες σελίδες έχει συλλέξει.

ΣΧΕΔΙΑΣΜΟΣ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

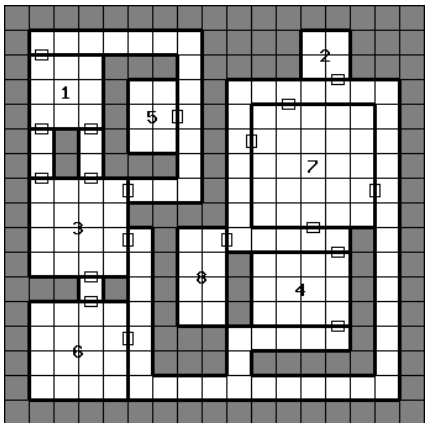
Όπως αναφέρθηκε στην προηγούμενη ενότητα, τις περισσότερες φορές χρησιμοποιούνται ένα ή περισσότερα βοηθητικά εργαλεία στην ανάπτυξη ενός παιχνιδιού πέρα από την μηχανή παιχνιδιού. Σ'αυτή τη πτυχιακή το SketchUp της Google σε συνδυασμό με το Photoshop της Adobe προτιμήθηκε για τον σχεδιασμό του περιβάλλοντος.

Ο χώρος του παιχνιδιού επιλέχθηκε να είναι ένα κτήριο, συγκεκριμένα ένας όροφος με πολλά δωμάτια. Στο Διαδίκτυο (Internet) υπάρχουν ιστοσελίδες (Εικόνα 17) που δημιουργούν τυχαίους χάρτες, με διάφορα κριτήρια πολυπλοκότητας, και κυρίως προορίζονται για επιτραπέζια παιχνίδια ωστόσο εξυπηρετεί και τα βιντεοπαιχνίδια. Οι σελίδες αυτές δίνουν την δυνατότητα να κατεβάσεις τον χάρτη σαν εικόνα. Ό,τι χρειαζόταν για μια αρχή.

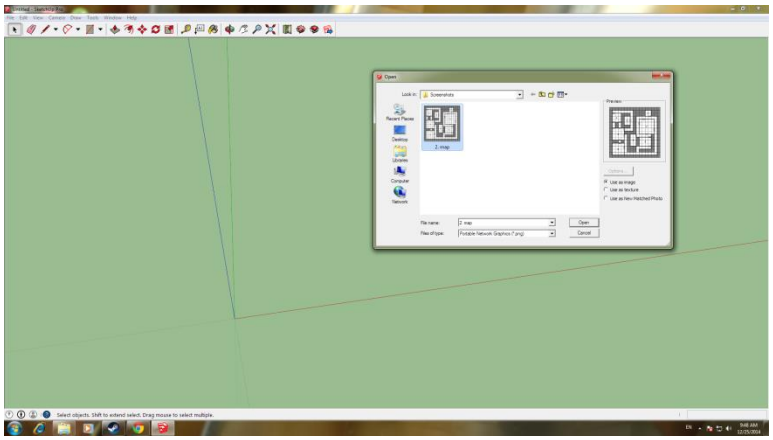


Εικόνα 17 "Μια ιστοσελίδα που δημιουργεί τυχαίους χάρτες"

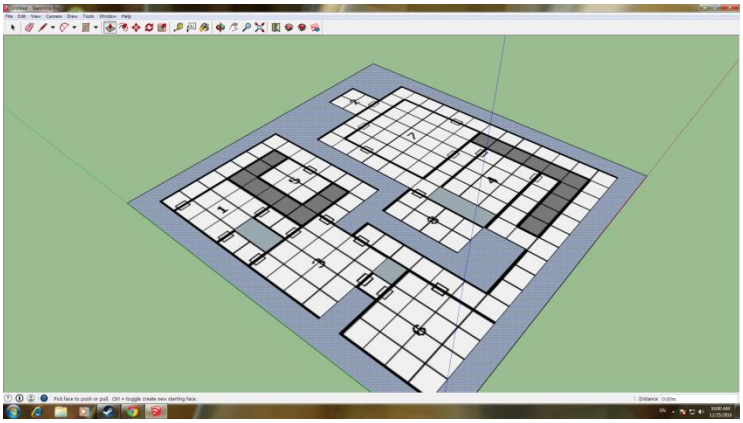
Για τον σχεδιασμό του περιβάλλοντος του παιχνιδιού της πτυχιακής χρησιμοποιήθηκε το SketchUp. Πρώτο βήμα είναι η δημιουργία ενός καινούριου project στο SketchUp και η εισαγωγή της εικόνας, χάρτη του παιχνιδιού για να γίνει η πατιτούρα των τοιχωμάτων. (Εικόνες 18 – 20) Η «αντιγραφή» του χάρτη γίνεται με τη βοήθεια των εργαλείων Line και Rectangle του SketchUp.



Εικόνα 18 "Ο χάρτης του παιχνιδιού"

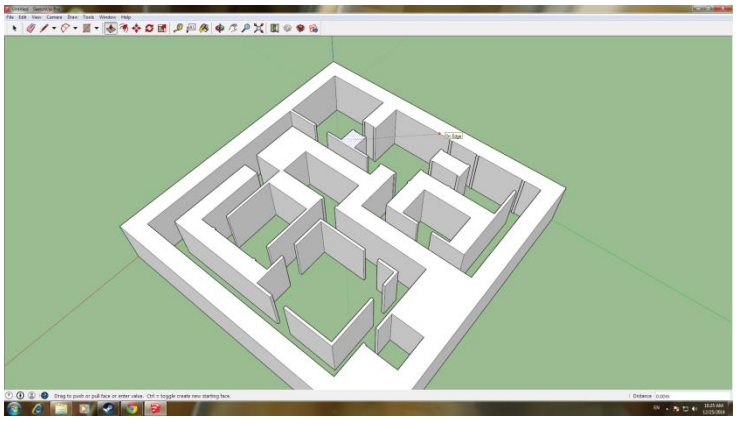


Εικόνα 19 "Εισαγωγή της εικόνας του χάρτη στο SketchUp"

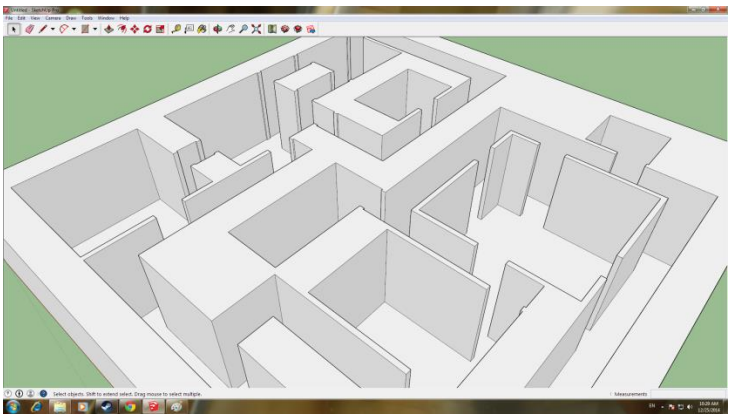


Εικόνα 20 "Πατιτούρα με τα εργαλεία Line και Rectangle"

Στην συνέχεια με την βοήθεια του εργαλείου Push/Pull του προγράμματος «σηκώνονται» οι τοίχοι (Εικόνες 21 - 22) και τόσο γρήγορα είναι έτοιμος ο όροφος του κτηρίου του βιντεοπαιχνιδιού.

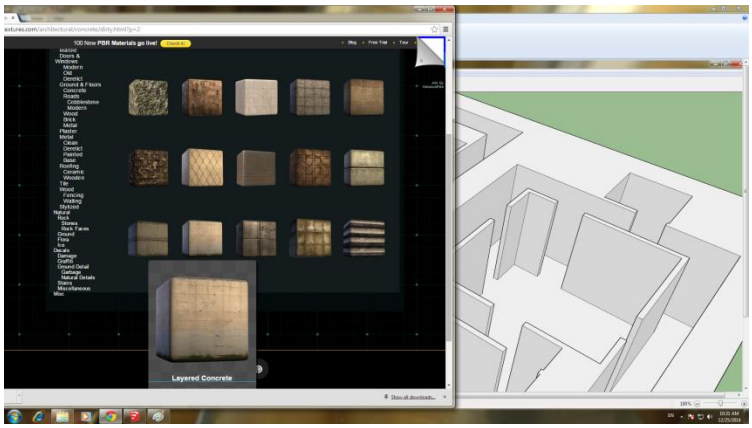


Εικόνα 21 "Χρήση εργαλείου Push/Pull"

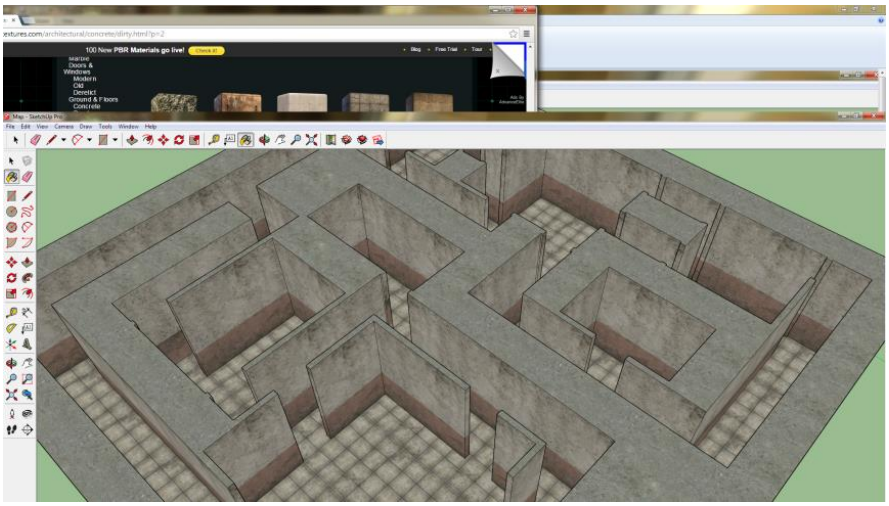


Εικόνα 22 "Χρήση Εργαλείου Push/Pull"

Σε αυτό το σημείο έχουμε έτοιμο τον κύριο χάρτη του παιχνιδιού σε ένα βασικό 3D μοντέλο, ωστόσο είναι άχρωμος και μπορούμε είτε να τον δώσουμε απλώς διάφορα χρώματα είτε να του περάσουμε και υφή (textures). Το SketchUp επιτρέπει να χρησιμοποιούμε διάφορα υλικά, με χρώμα και υφή, ώστε να προσθέσουμε λεπτομέρεια και ρεαλισμό στα μοντέλα. Με την βοήθεια του διαδικτύου μπορούμε να βρούμε textures, είτε δωρεάν είτε με πληρωμή, να τα εισάγουμε στο SketchUp και να αρχίσουμε να «βάφουμε» τους τοίχους. Για την πτυχιακή αυτή αγοράσθηκαν textures από το GameTextures.com (Εικόνες 23 - 24).

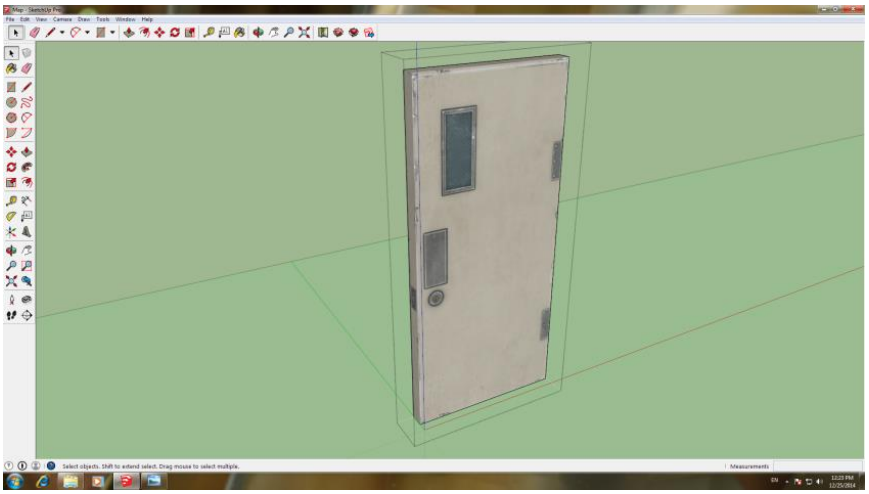


Εικόνα 23 "Gametextures.com"

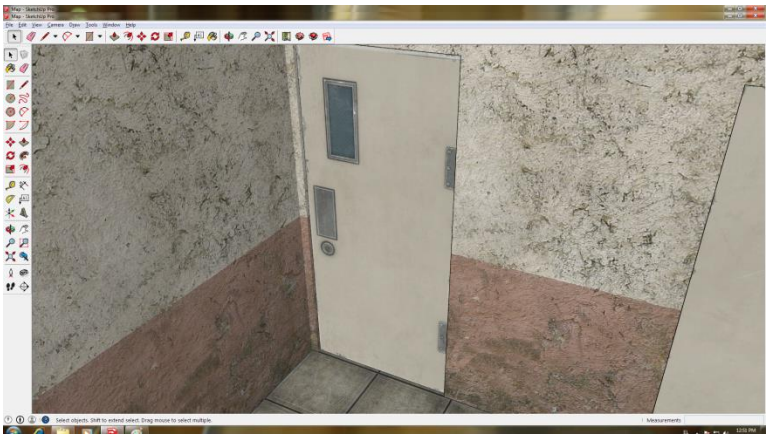


Εικόνα 24 "Προσθήκη υλικών στους τοίχους του παιχνιδιού"

Στη συνέχεια με τη βοήθεια του Photoshop, αφού επεξεργάστηκαν κάποιες εικόνες από πόρτες, σχεδιάστηκαν στο SketchUp τα μοντέλα πορτών που θα χρησιμοποιηθούν στο παιχνίδι και προστέθηκαν παρομοίως τα textures ώστε να μοιάζουν ρεαλιστικές. Έπειτα τοποθετήθηκαν σε διάφορα σημεία στον χάρτη για να προσθέσουν πολυπλοκότητα στο παιχνίδι (Εικόνες 25-26).

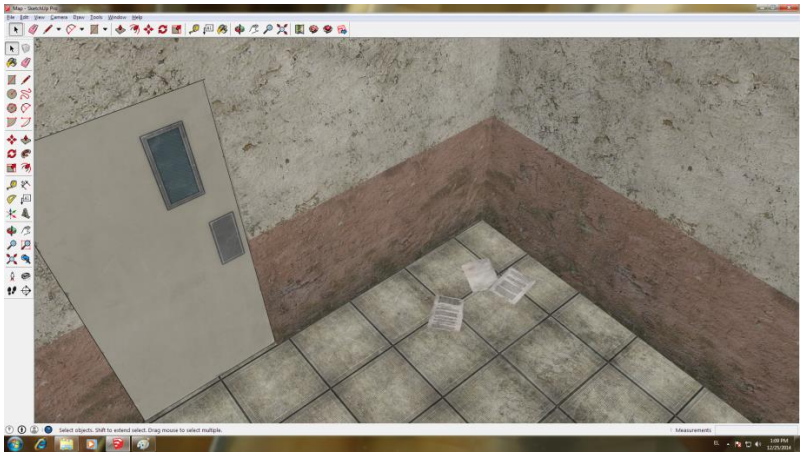


Εικόνα 25 "Δημιουργία πόρτας στο SketchUp"

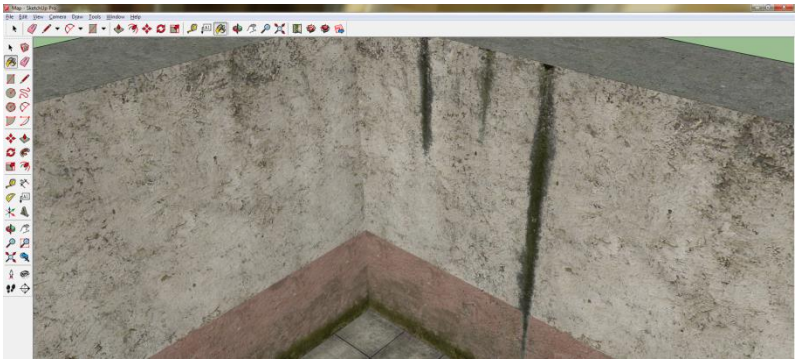


Εικόνα 26 "Τοποθέτηση του μοντέλου της πόρτας σε διάφορα σημεία"

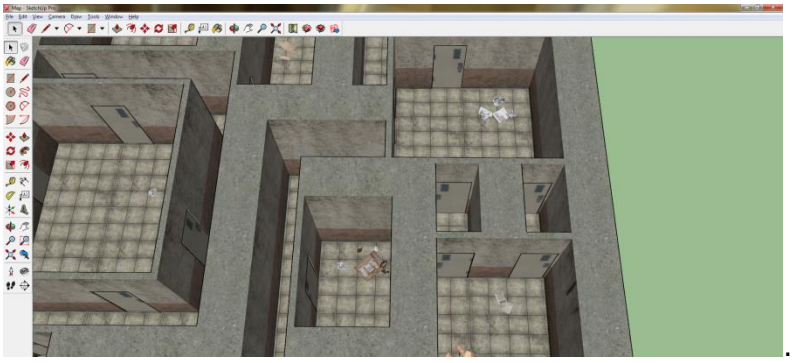
Σε αυτό το σημείο της πτυχιακής έχουμε ένα μοντέλο του χάρτη του παιχνιδιού με τοίχους και πόρτες τα οποία έχουν κάποιο χρώμα και υφή. Ωστόσο, για να προχωρήσει ένα βήμα παραπέρα και να μην έχει όλο το περιβάλλον του παιχνιδιού τον ίδιο τοίχο και το ίδιο πάτωμα κ.λπ. προστέθηκαν διάφορα decal textures ώστε να υπάρχει κάποια λεπτομέρεια στο περιβάλλον. Decal texture είναι μια εικόνα - υφή που επικαλύπτει άλλες υφές σε γραφικά υπολογιστών. Τέτοια μπορεί να απεικονίζουν μια ρωγμή σε έναν τοίχο, ένα σκουπίδι σε ένα πάτωμα κ.λπ. Συνήθως προτιμάται να τοποθετούνται decals αντί να σχεδιάζεται στο μοντέλο η ρωγμή, ή το σκουπίδι σε συνέχεια του προηγούμενου παραδείγματος, για να μην προστίθενται πολλά πολύγωνα στο τελικό μοντέλο. Στο περιβάλλον του παιχνιδιού της πτυχιακής αυτής τοποθετήθηκαν διάφορα decals όπως, υγρασία σε τοίχους, σκουπίδια στο πάτωμα, graffiti σε τοίχους κ.λπ. (Εικόνες 27-29).



Εικόνα 27 "Τοποθέτηση Decals σε διάφορα σημεία του χάρτη"



Εικόνα 28 "Τοποθέτηση Decals σε διάφορα σημεία του χάρτη"



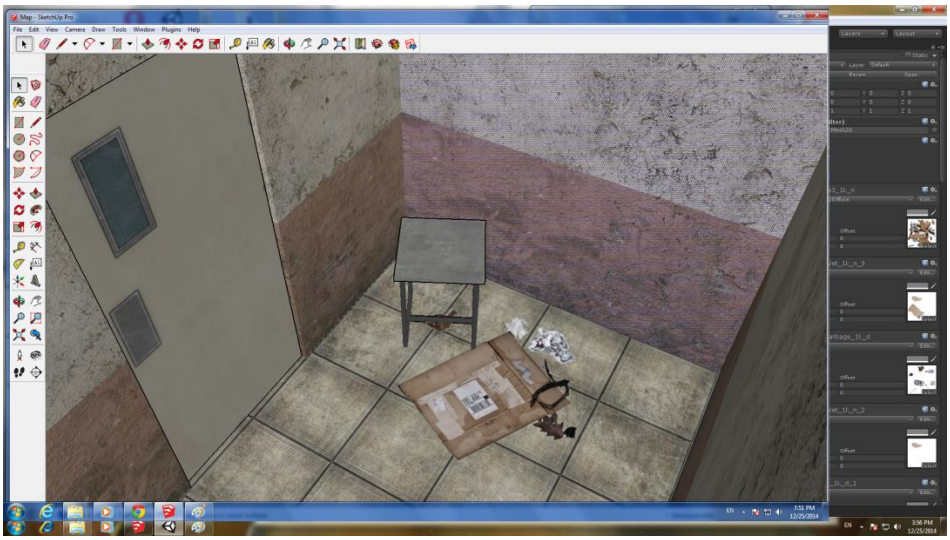
Εικόνα 29 "Τοποθέτηση Decals σε διάφορα σημεία του χάρτη"

Έπειτα, αφού έχουν τοποθετηθεί και μερικά decal textures σε διάφορα σημεία του χάρτη του παιχνιδιού, ήρθε η ώρα να τοποθετηθούν και πραγματικά αντικείμενα στο περιβάλλον ώστε να μην μοιάζει άδειο. Υπάρχουν δύο περιπτώσεις. Μία να κατασκευαστούν από την αρχή κάποια 3D μοντέλα, όπως βαρέλια, κουτιά κ.λπ., που δεν συνηθίζεται σε ανάπτυξη Indie Games, και η

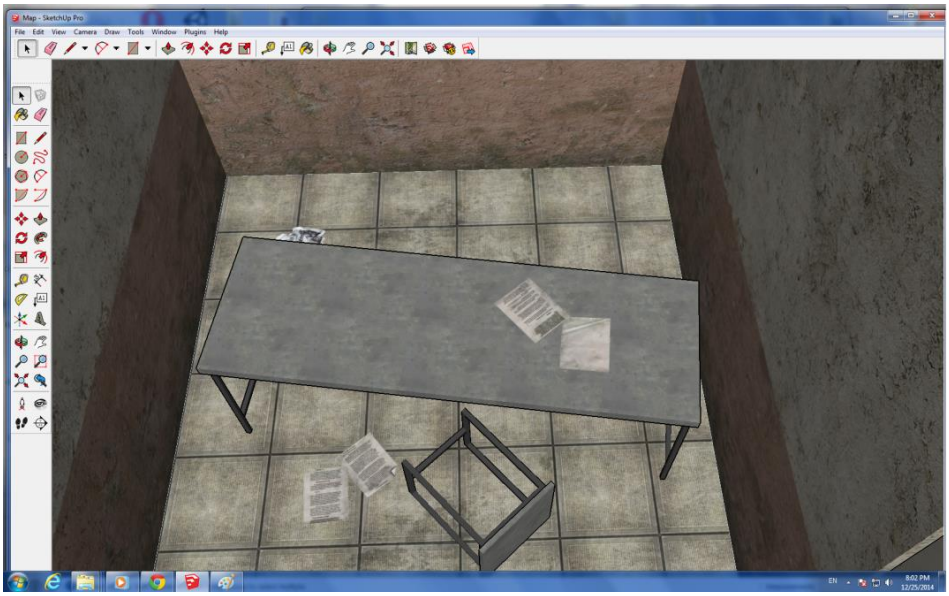
περίπτωση όπου ο developer απλώς αναζητεί στο διαδίκτυο 3D μοντέλα είτε δωρεάν είτε με πληρωμή. Συνήθως τα καλύτερα μοντέλα κοστίζουν. Για την συγκεκριμένη πτυχιακή έχουν βρεθεί δωρεάν μοντέλα από το 3D Warehouse της Google.

Αφού έχουν επιλεχθεί κάποια μοντέλα που ταιριάζουν με το περιβάλλον του παιχνιδιού, τοποθετήθηκαν σε διάφορα σημεία ώστε να δώσουν λεπτομέρεια σε όλο το χώρο (Εικόνες 30-33). Σε αυτό το σημείο είναι καλό να αναφερθεί πως το 3D Warehouse είναι συμβατό με το SketchUp, αφού δίνει τη δυνατότητα download ενός μοντέλου σε αρχείο *.skp που είναι αρχείο της εφαρμογής. Εξάλλου και τα δύο είναι της Google.

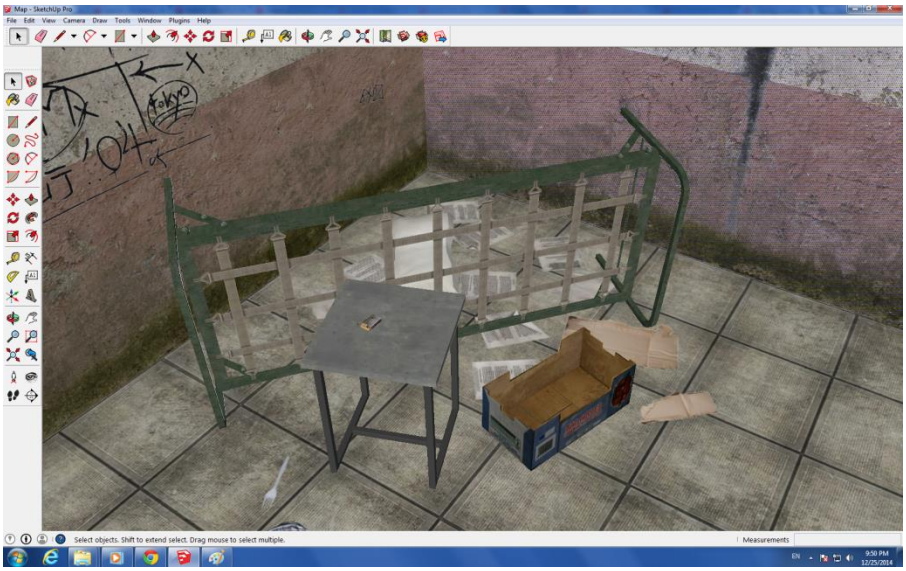
Η διαδικασία εισαγωγής και τοποθέτησης των 3D μοντέλων είναι απλή. Αυτό γίνεται από την επιλογή Import... στο μενού File.



Εικόνα 30 "Τοποθέτηση 3D μοντέλων σε διάφορα σημεία του παιχνιδιού"



Εικόνα 31 "Τοποθέτηση 3D μοντέλων σε διάφορα σημεία του παιχνιδιού"



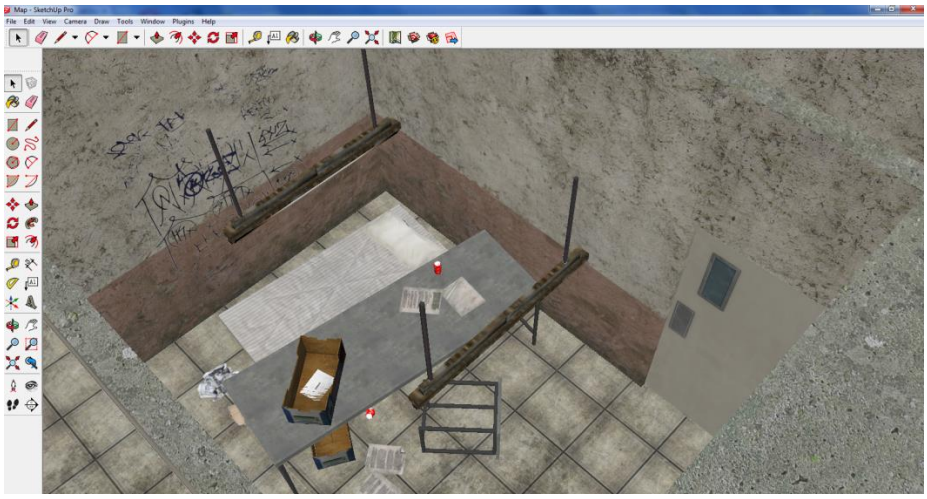
Εικόνα 32 "Τοποθέτηση 3D μοντέλων σε διάφορα σημεία του παιχνιδιού"

Εικόνα 33 "Τοποθέτηση 3D μοντέλων σε διάφορα σημεία του παιχνιδιού"

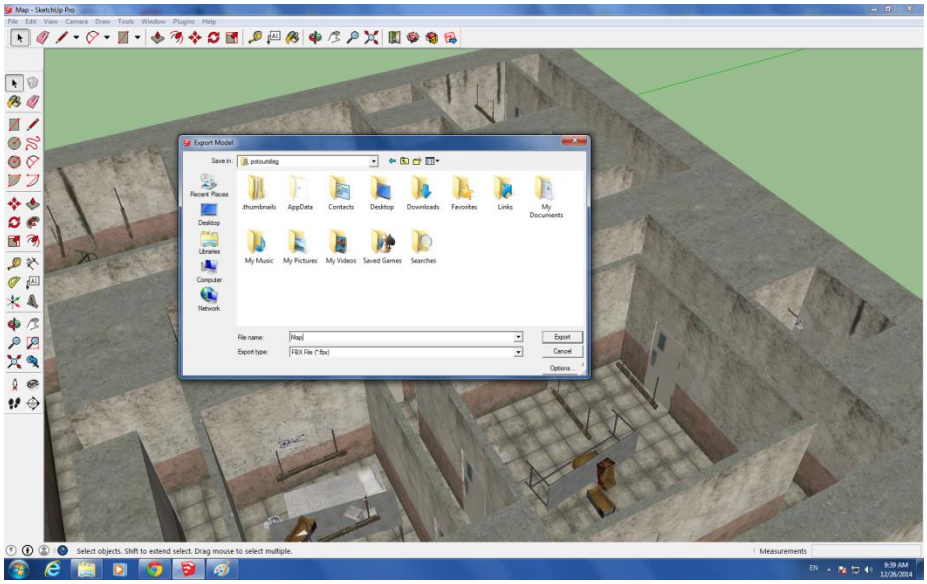
Στην συνέχεια με την βοήθεια του Photoshop ξανά, σχεδιάστηκαν οι σελίδες που θα συλλέγει ο παίκτης και έπειτα «κρύφτηκαν» και αυτές σε διάφορα σημεία. (Εικόνες 34-35)

Τέλος, αφού έχει ολοκληρωθεί το περιβάλλον σε ένα ικανοποιητικό βαθμό για το μάτι του παίκτη, έγινε εξαγωγή του project σε μορφή *.fbx ώστε να γίνει η εισαγωγή του στην Unity και να αρχίσει ο προγραμματισμός του παιχνιδιού. (Εικόνα 36)

Εικόνα 34 "Οι σελίδες που θα συλλέγει ο παίκτης σχεδιάστηκαν με τη βοήθεια του Photoshop"



Εικόνα 35 "Τοποθέτηση σελίδων σε διάφορα κρυφά σημεία"

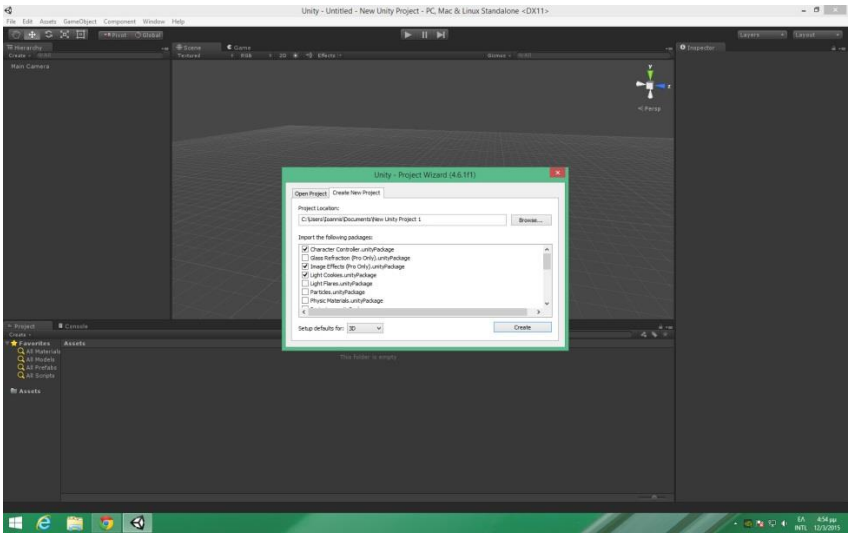


Εικόνα 36 "Εξαγωγή του μοντέλου σε μορφή .fbx"

ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ PROJECT ΣΤΗΝ UNITY

Ανοίγοντας την Unity πρώτη φορά θα εμφανισθεί ένα παράθυρο και θα ζητήσει να δημιουργήσουμε ένα καινούριο project. Σε αυτό το παράθυρο (Εικόνα 37) έχουμε να αποφασίσουμε δύο πράγματα, την διαδρομή που θα αποθηκεύσουμε το project και τα πακέτα που θα εισάγουμε σ'αυτό. Τα πακέτα αυτά μπορούν να εισαχθούν και αργότερα, αφού δημιουργηθεί το project. Πρόκειται για διάφορα έτοιμα στοιχεία που χρησιμοποιούνται στη δημιουργία του

παιχνιδιού π.χ. το Character Controller που επιτρέπει να κάνεις εύκολα κίνησή του παίκτη στον χώρο του παιχνιδιού με collisions κ.λπ. ή Image Effects που είναι διάφορα effects που χρησιμοποιούνται στην κάμερα και προσθέτουν πολλά στην εμφάνιση και την αίσθηση του παιχνιδιού χωρίς να καταναλώνουμε πολύ χρόνο στην δημιουργία καινούριων.

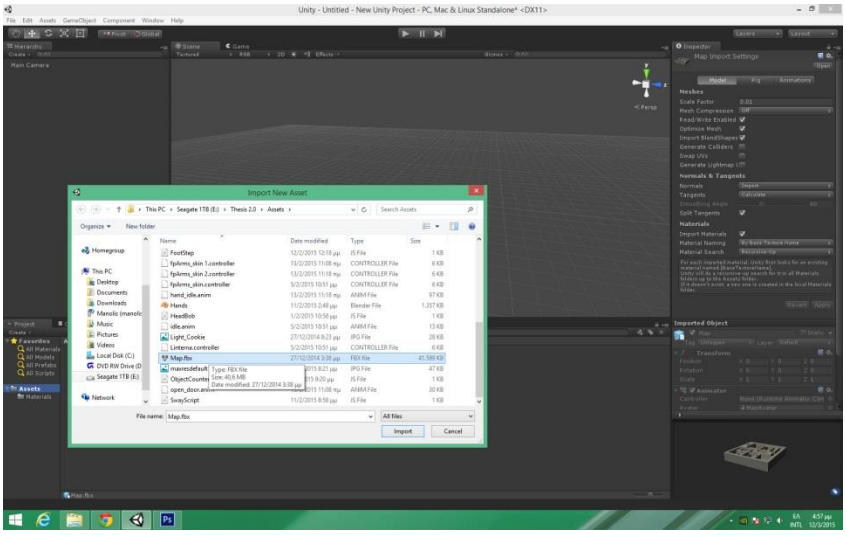


Εικόνα 37 "Δημιουργία νέου project στη Unity"

Υπάρχουν μερικοί βασικοί τύποι στοιχείων (assets) που μπορούν να εισαχθούν στο project. Τέτοιοι είναι τα τρισδιάστατα σχέδια, όπως το περιβάλλον του παιχνιδιού που σχεδιάστηκε, διάφορα ηχητικά effects ή textures κ.λπ. Αυτά τα στοιχεία μπορούν να εισαχθούν στο project είτε από τον explorer του λειτουργικού συστήματος του υπολογιστή είτε από την Unity όπου και είναι προτιμότερο.

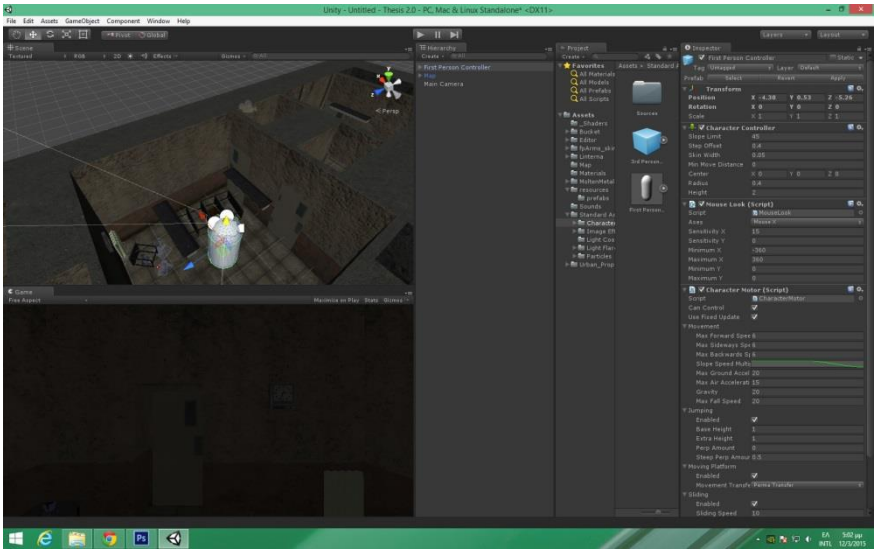
Στο GUI της Unity υπάρχουν διάφορα views με πληροφορίες, το Scene View που βλέπουμε τη σκηνή του παιχνιδιού που δημιουργούμε, το Inspector View που έχει διάφορες παραμέτρους σε κάποιο στοιχείο που έχουμε επιλεγμένο κ.λπ. Στο Project View φαίνεται η δομή των φακέλων του project και εκεί μπορούμε να διαχειριζόμαστε διάφορα στοιχεία που εισάγουμε ή μετακινούμε σ’αυτό.

Πηγαίνουμε στο Project View λοιπόν στον φάκελο Assets και με δεξί click διαλέγουμε εισαγωγή νέου Asset και βρίσκουμε το αρχείο (Εικόνα 38) που έγινε export από το SketchUp νωρίτερα μαζί με τον φάκελο που δημιουργήθηκε και έχει τα textures του περιβάλλοντος ώστε να περαστούν αυτόματα στο μοντέλο μέσα στο Unity. Αφού γίνει η εισαγωγή του ενεργοποιούμε το Generate Colliders και τροποποιούμε αν θέλουμε το μέγεθός του από το Inspector View και πατάμε Apply. Με την επιλογή Generate Colliders ο χαρακτήρας θα «τρακάρει» στους τοίχους και τα αντικείμενα και δεν θα περνάει από μέσα τους.



Εικόνα 38 "Εισαγωγή του μοντέλου στη Unity"

Στην συνέχεια «σέρνουμε» το μοντέλο του χάρτη του παιχνιδιού στο Scene View. Με τον ίδιο τρόπο ψάχνουμε το Character Controller στα Assets και σέρνουμε το First Person Controller (Εικόνα 39) σε ένα μέρος του χάρτη που θέλουμε να ξεκινάει ο παίκτης. Σε αυτό το σημείο αν πατήσουμε το Play θα μπορούμε να περιηγηθούμε στο παιχνίδι με προοπτική πρώτου προσώπου, ωστόσο δεν θα μπορούμε να αλληλεπιδράσουμε με το περιβάλλον ακόμα.

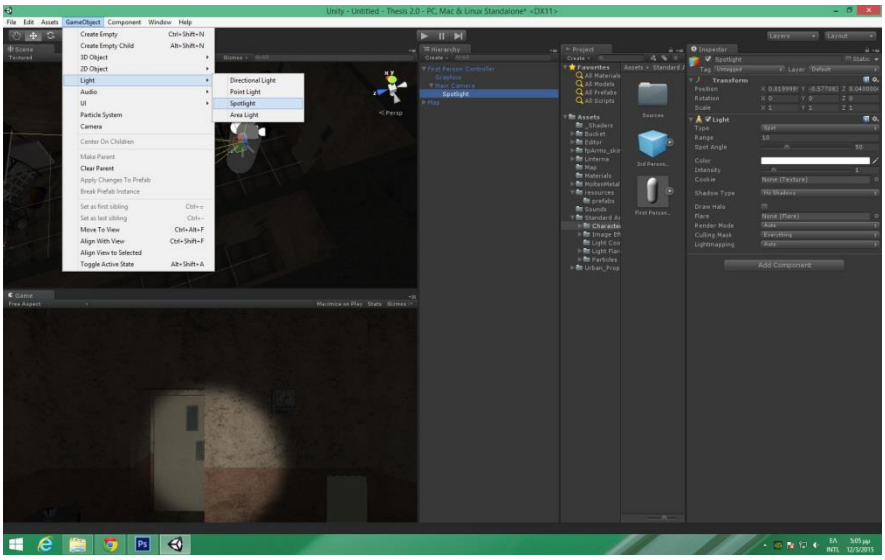


Εικόνα 39 "Εισαγωγή του First Person Controller στο Scene"

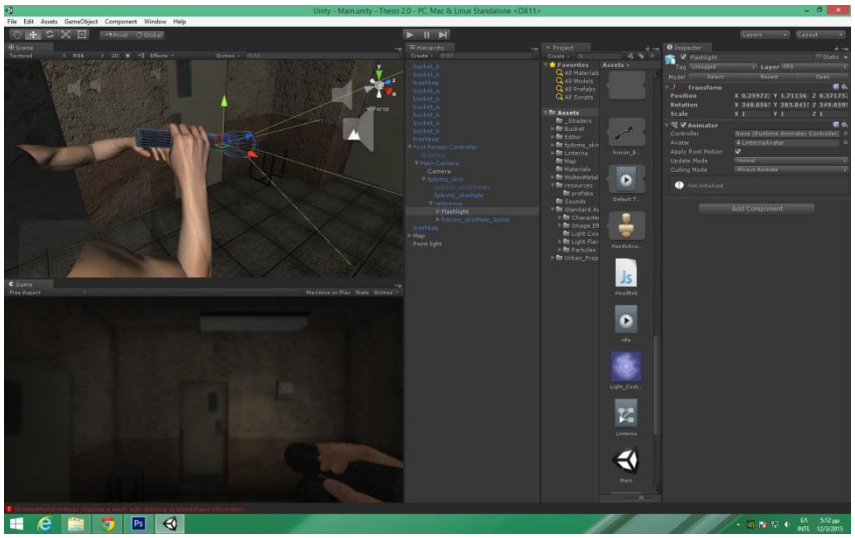
ΔΗΜΙΟΥΡΓΙΑ FLASHLIGHT ΤΟΥ ΧΑΡΑΚΤΗΡΑ

Για να δώσουμε μια πιο ρεαλιστική έκδοση πρώτου προσώπου όπου ο χαρακτήρας θα κρατάει φακό και θα ψάχνει μέσα στο κτήριο θα κάνουμε εισαγωγή μιας πηγής φωτός από το μενού GameObject -> Light -> Spotlight (Εικόνα 40). Αφού το επιλέξουμε και επεξεργαστούμε από το Inspector View την κατεύθυνση που θα φωτίζει (προς τη μεριά που βλέπει η κάμερα αν χρειαστεί), την φωτεινότητα κ.λπ. θα κάνουμε το φως μας (Spotlight) παιδί της κάμερας ώστε να ακολουθεί την κάμερα και να έχουμε την αίσθηση ότι κρατάμε φακό. Αυτό γίνεται μετακινόντας απλώς το Spotlight από το Hierarchy View μέσα στο Main Camera ώστε να πάει από κάτω του.

Με τον ίδιο τρόπο έγινε εισαγωγή 3D μοντέλων χεριών και φακού στα Assets, μπήκαν στο Scene και τοποθετήθηκαν μπροστά από την κάμερα κατάλληλα (Εικόνα 41) ώστε να δίνει την αίσθηση στο παίκτη ότι βλέπει τα χέρια του να κρατάνε φακό. Παρομοίως, πρέπει να γίνουν παιδιά της κάμερας ώστε να την ακολουθούν.



Εικόνα 40 "Εισαγωγή ενός Light GameObject στο Scene"

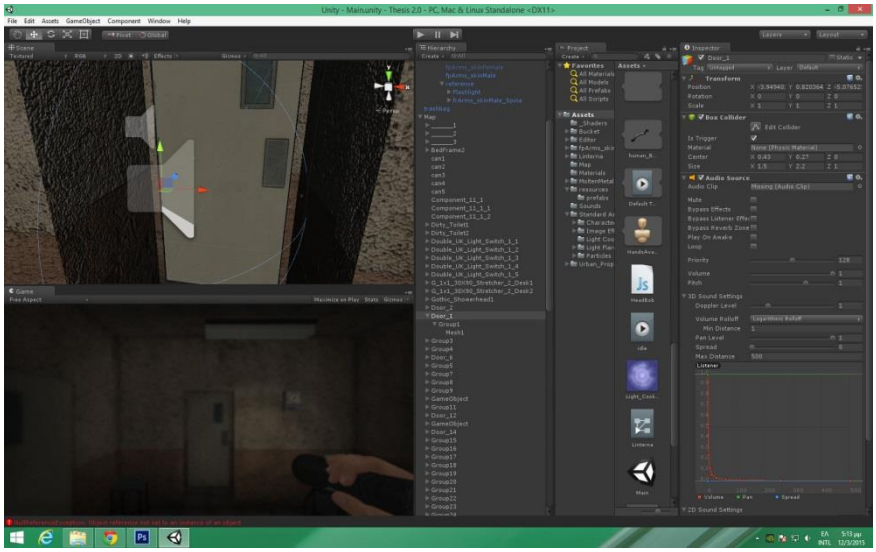


Εικόνα 41 "Η εισαγωγή χεριών και φακού στο παιχνίδι"

Στη συνέχεια θα προσθέσουμε διάφορους κώδικες (scripts), για να μπορεί να αλληλεπιδράσει ο χαρακτήρας με το περιβάλλον, δηλ. να ανοίγει πόρτες και να μπορεί να συλλέξει τις σελίδες.

ΑΛΛΗΛΕΠΙΔΡΑΣΗ ΜΕ ΤΙΣ ΠΟΡΤΕΣ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Αρχικά, θα γίνει για μια πόρτα και στη συνέχεια θα γίνει στις υπόλοιπες με την ίδια διαδικασία. Κάνουμε εισαγωγή ενός κενού GameObject και το κάνουμε Parent στην πόρτα. Ο λόγος που γίνεται αυτό είναι για να θέσουμε ένα σημείο από όπου που θα περιστρέφεται η πόρτα (την άκρη της) για να μην περιστρέφεται γύρω από τον εαυτό της (Εικόνα 42). Έχοντας διαλεγμένο το GameObject μετακινούμε από το Position στο Inspector View τον άξονα x, y, z στην άκρη του μοντέλου της πόρτας. Ονομάζουμε το GameObject κατάλληλα για να αναφερόμαστε σε αυτό εύκολα π.χ. Door. Στη συνέχεια στο Inspector View θα κάνουμε εισαγωγή καινούριου Component, ενός Box Collider. Αυτό θα μας εμφανίσει κάποια όρια που θα «τρακάρει» ο παίκτης και δεν θα μπορεί να περάσει από μέσα. Ωστόσο επιλέγουμε το Is Trigger ώστε να μπορεί να περάσει από το Collider. Τα τοποθετούμε κατάλληλα έτσι ώστε σε ποιο σημείο πιστεύουμε θα φτάσει ο χαρακτήρας για να μπορεί να ανοίξει την πόρτα. Αφού τελειώσουμε αυτή τη διαδικασία είμαστε έτοιμοι να προσθέσουμε τον κώδικα Javascript όπου κατά τον οποίο, όποτε εισέρχεται ο παίκτης στο Trigger Collider θα μπορεί με το πάτημα ενός πλήκτρου να ανοίγει την πόρτα, ουσιαστικά να την κάνει rotate 90 μοίρες στο σημείο που προσθέσαμε νωρίτερα.



Εικόνα 42 "Επεξεργασία παραμέτρων του GameObject της πόρτας"

Δημιουργούμε ένα αρχείο Javascript μέσα στα Assets, κάνουμε επικόλληση τον παρακάτω κώδικα, και εφαρμόζουμε το αρχείο αυτό σε ένα Script Component στο GameObject της πόρτας (Door) και είμαστε έτοιμοι να ανοίξουμε μερικές πόρτες, εφαρμόζοντας την ίδια διαδικασία και στις υπόλοιπες.

```
var IsOpen : boolean = false;

var CanOpen : boolean = false;

var Volume : float = 0.5;


function Start () {

    GetComponent.<AudioSource>().volume = Volume;

}

function Update () {

    if(Input.GetKeyUp(KeyCode.E) && !IsOpen && CanOpen) {

        GetComponent.<AudioSource>().PlayDelayed(0.140);

        Opening();

        IsOpen = true;

    }
```

```
    else if(Input.GetKeyUp(KeyCode.E) && IsOpen && CanOpen) {
        GetComponent.<AudioSource>().PlayDelayed(0.140);
        Closing();
        IsOpen = false;
    }
}

function Opening() {
    for (var i = 0; i < 100; i++) {
        transform.Rotate(0,-0.9,0);
        yield WaitForSeconds(0.0140);
    }
}

function Closing() {
    for (var i = 0; i < 100; i++) {
        transform.Rotate(0,0.9,0);
        yield WaitForSeconds(0.0140);
    }
}

function OnTriggerEnter (other : Collider) {
    if(other.gameObject.tag == "Player") {
        CanOpen = true;
    }
}
```

```
function OnTriggerExit (other : Collider) {
    if(other.gameObject.tag == "Player") {
        CanOpen = false;
    }
}
@script RequireComponent(AudioSource)
```

Αυτό που ελέγχει ουσιαστικά ο παραπάνω κώδικας Javascript είναι αν ο χαρακτήρας βρίσκεται κοντά στην πόρτα να μπορεί να την ανοίξει, δηλαδή να περιστραφεί αυτή 90 μοίρες ώστε να ανοίξει.

Ορίζονται αρχικά 2 μεταβλητές, IsOpen και CanOpen, οι οποίες θα ελέγχουν αν μια πόρτα είναι ανοιχτή και αν μπορεί να ανοίξει. Αν θα μπορεί μια πόρτα να ανοίξει εξαρτάται από το αν ο παίκτης είναι κοντά της, δηλαδή βρίσκεται μέσα στο Trigger Box Collider που τοποθετήθηκε προηγμένως στην πόρτα. Ο παραπάνω έλεγχος γίνονται με τις δύο μεθόδους OnTriggerEnter() και OnTriggerExit().

```
function OnTriggerEnter (other : Collider) {
    if(other.gameObject.tag == "Player") {
        CanOpen = true;
    }
}
```

Η μέθοδος OnTriggerEnter ελέγχει αν έχει πλησιάσει ένα αντικείμενο με ετικέτα «Player», την οποία έχει ο χαρακτήρας του παιχνιδιού. Αν ισχύει, η μεταβλητή CanOpen παίρνει την τιμή true. Δηλαδή αυτή η πόρτα μπορεί να ανοίξει.

```
function OnTriggerExit (other : Collider) {
    if(other.gameObject.tag == "Player") {
        CanOpen = false;
    }
}
```

Η μέθοδος OnTriggerExit ελέγχει αν έχει απομακρυνθεί ένα αντικείμενο με ετικέτα «Player», Αν ισχύει, η μεταβλητή CanOpen παίρνει την τιμή false. Δηλαδή αυτή η πόρτα μπορεί να ΔΕΝ μπορεί να ανοίξει.

Δύο άλλες μέθοδοι, Opening() και Closing(), είναι υπεύθυνες για την περιστροφή της πόρτας, το άνοιγμα και το κλείσιμο, οι οποίες είναι βοηθητικές σε μια μεγαλύτερη μέθοδο που κάνει όλη τη δουλειά.

```
function Update () {
    if(Input.GetKeyUp(KeyCode.E) && !IsOpen && CanOpen) {
        GetComponent.<AudioSource>().PlayDelayed(0.140);
        Opening();
        IsOpen = true;
    }
    else if(Input.GetKeyUp(KeyCode.E) && IsOpen && CanOpen) {
        GetComponent.<AudioSource>().PlayDelayed(0.140);
        Closing();
        IsOpen = false;
    }
}
```

Η Update() ελέγχει αν έχει πατηθεί κάποιο πλήκτρο “E” (το “E” σ’αυτή τη περίπτωση θέλουμε να ανοίγει τις πόρτες) ΚΑΙ αν μια πόρτα δεν είναι ανοιχτή ΚΑΙ αν μπορεί να ανοίξει. Τότε ανοίγει με την Opening() και η μεταβλητή IsOpen γίνεται true. Διαφορετικά αν έχει πατηθεί το “E” ΚΑΙ είναι ανοιχτή ΚΑΙ μπορεί να ανοίξει (ίσως η μεταβλητή CanOpen να είχε διαφορετικό όνομα, σ’αυτή τη περίπτωση σημαίνει να μπορεί να κλείσει), τότε η πόρτα κλείνει με την μέθοδο Closing(); και η IsOpen παίρνει την τιμή false. Τέλος, με την γραμμή κώδικα που ακολουθεί που βρίσκεται στους δυο παραπάνω ελέγχους παίζει ένας ήχος π.χ. τρίξιμο πόρτας, αν έχει τοποθετηθεί ένα αντικείμενο τύπου AudioSource με τον συγκεκριμένο ήχο σε κάθε πόρτα.

```
GetComponent.<AudioSource>().PlayDelayed(0.140);
```

ΣΥΛΛΟΓΗ ΣΕΛΙΔΩΝ ΑΠΟ ΤΟ ΠΕΡΙΒΑΛΛΟΝ

Οι επόμενοι κώδικες που ακολουθούν έχουν την ίδια λογική όσο αφορά τα αντικείμενα στο Scene μας. Ακολουθούν οι σελίδες, τις οποίες θα συλλέγει ο παίκτης. Πάλι τοποθετείται ένα Collider στις σελίδες (Εικόνα 43) όπου θα ελέγχει αν ο παίκτης είναι κοντά και τότε θα τις συλλέγει, δηλαδή θα καταστρέφει το αντικείμενο της σελίδας από τον κόσμο. Αυτή τη φορά ο κώδικας Javascript για τη συλλογή σελίδων τοποθετείται σε Javascript Component στον χαρακτήρα.

Εικόνα 43 "Τοποθέτηση Sphere Collider σε κάθε συλλέξιμη σελίδα"

```
var paper : int = 0;
var paperToWin : int =5;

function OnTriggerEnter( other : Collider ) {
	if (other.gameObject.tag == "Paper") {
		GetComponent.<AudioSource>().PlayDelayed(0.040);
		paper += 1;
		Debug.Log("A page was picked up. Total pages = " + paper);
		Destroy(other.gameObject);
	}
```

```
}

function OnGUI() {

	if (paper < paperToWin)  {

		GUI.Box(Rect((Screen.width/2)-100, 10, 200, 35), "" + paper + " 
	pages");

	}

	else {

		GUI.Box(Rect((Screen.width/2)-100, 10, 200, 35), "All papers 
	collected!");

	}

}

@script RequireComponent(AudioSource)
```

Ο παραπάνω κώδικας είναι απλός. Έχει δύο μεταβλητές, τις paper και paperToWin οι οποίες δείχνουν πόσες σελίδες έχει στην κατοχή του ο παίκτης, και πόσες θέλει συνολικά για να κερδίσει. Έχει δύο μεθόδους, την OnTriggerEnter() και την OnGUI(). Η δεύτερη απλώς απεικονίζει με ένα απλό Interface, πόσες σελίδες έχει συλλέξει ο παίκτης.

```
function OnTriggerEnter( other : Collider ) {

	if (other.gameObject.tag == "Paper") {

		GetComponent.<AudioSource>().PlayDelayed(0.040);

		paper += 1;

		Debug.Log("A page was picked up. Total pages = " + paper);

		Destroy(other.gameObject);

	}

}
```

Η μέθοδος OnTriggerEnter() ελέγχει αν ο χαρακτήρας έχει πλησιάσει ένα αντικείμενο με ετικέτα “Paper”, όπου είναι οι σελίδες μας, τότε παίζει πάλι έναν ήχο (σαν να σηκώνεις ένα χαρτί) που βρίσκεται σε ένα Component Audio Source,

αυξάνει τη μεταβλητή paper κατά 1, και καταστρέφει το αντικείμενο της σελίδας με την Destroy(other.gameObject).

ΣΤΑΤΙΣΤΙΚΑ ΤΟΥ ΠΑΙΚΤΗ

Για να υπάρχει ενδιαφέρον και κάποια θετική ή αρνητική εξέλιξη στο παιχνίδι χρειάζεται να κρατάμε κάποια στατιστικά στους παίκτες, όπως είναι η ζωή, η αντοχή, η πείνα και διάφοροι άλλοι παράγοντες που επηρεάζουν τον χαρακτήρα. Τα στατιστικά αποθηκεύονται σε public μεταβλητές στον χαρακτήρα και μπορούν να τροποποιηθούν όταν αυτός αλληλεπιδράσει με άλλα αντικείμενα, όπως για παράδειγμα αν τον επιτεθεί ο εχθρός η ζωή του θα μειωθεί. Για αυτή τη περίπτωση στη παρούσα πτυχιακή έχει δημιουργηθεί ο παρακάτω κώδικας Javascript και τοποθετήθηκε σαν Component στο First Person Controller.

```
#pragma strict

var MaxHealth = 100;
var Health : int;

function Start ()
{
	Health = MaxHealth;
}

function ApplyDamage (TheDamage : int)
{
	Health -= TheDammage;

	if(Health <= 0)
	{
		Dead();
	}
}

function Dead()
{
	RespawnMenuV2.playerIsDead = true;
	Debug.Log("Player Died");
}

function RespawnStats ()
```

```
{
        Health = MaxHealth;
}
```

Ο παραπάνω κώδικας είναι απλός. Αποθηκεύει την maximum τιμή που μπορεί να πάρει η ζωή του χαρακτήρα στη μεταβλητή MaxHealth, και την τρέχουσα τιμή της ζωής στη μεταβλητή Health. Κατά την εκκίνηση του παιχνιδιού η τρέχουσα ζωή είναι maximum, δηλαδή ο παίκτης ξεκινά με full life (Health = MaxHealth). Στη συνέχεια υλοποιούνται 3 απλές μέθοδοι. Η ApplyDamage(TheDamage: int) όπου αφαιρεί «damage» από τη ζωή του χαρακτήρα. Αν αυτή φτάσει το 0 (ή λιγότερο!) τότε ο χαρακτήρας πεθαίνει και καλείται η μέθοδος Dead(). Η Dead() απλώς δίνει σε μια μεταβλητή playerIsDead την τιμή true ώστε να ενεργοποιηθεί ένα άλλο scene όπου βρίσκεται το μενού του παιχνιδιού. Έτσι, όταν ο παίκτης χάσει θα επιστρέψει στο κεντρικό μενού. Τέλος, υλοποιείται η RespawnStats όπου απλώς αρχικοποιεί τα στατιστικά, στην προκειμένη περίπτωση η τρέχουσα ζωή μεγιστοποιείται πάλι παίρνοντας την τιμή της MaxHealth.

ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΠΑΛΟΥ

Για το φάντασμα - αντίπαλο που θα στοιχειώνει το εγκαταλελειμμένο χώρο του παιχνιδιού επιλέχθηκε ένα έτοιμο 3D model από το Asset Store της Unity, συγκεκριμένα το Total Horror (Εικόνα 44) από τον δημιουργό Arteria3D. Το συγκεκριμένο asset παρέχει ένα έτοιμο 3D μοντέλο, κάτι σαν μεταλλαγμένος άνθρωπος ή εξωγήινο ον, και θεωρήθηκε κατάλληλο για την χρήση του σε τέτοιου είδους παιχνίδι. Μαζί με το 3D μοντέλο συμπεριλαμβάνονται και διάφορα animations, όπως η idle κατάσταση του, το περπάτημα, η επίθεση κ.λπ.

Εικόνα 44 "Total Horror asset από την Arteria3D"

Αρχικά έγινε εισαγωγή του asset στη Unity κάνοντας δεξί κλικ στον φάκελο Assets και επιλέγοντας Import New Asset…. Έπειτα, δημιουργήθηκε ένα κενό GameObject, έγινε μετονομασία του σε Enemy και τοποθετήθηκαν σε αυτό σαν Components ένα Character Controller (αντί κάποιου άλλου Collider, απλώς για καλύτερο χειρισμό), ένα Rigidbody για να υπάρχει Collision με τους τοίχους και να υπάρχει και βαρύτητα. Στη συνέχεια έγινε εισαγωγή του Prefab του Total Horror (το μοντέλο μαζί με τα animation, materials κ.λπ) και μπήκε κάτω από το Enemy GameObject ώστε να είναι «παιδί» του. Έπειτα δημιουργήθηκε ο παρακάτω κώδικας Javascript για να «δώσει ζωή» στον κακό του παιχνιδιού, και τοποθετήθηκε σαν Component στο Total Horror.

Στις public μεταβλητές που εμφανίζονται στο Inspector Window σαν Target «σέρνουμε» τον χαρακτήρα του παιχνιδιού (First Person Controller) και στο controller τον αντίπαλο, δηλαδή το Enemy GameObject. Οι υπόλοιπες μεταβλητές ορίζουν τις διάφορες αποστάσεις του αντιπάλου από τον παίκτη, π.χ. σε πόση απόσταση θα κοιτάζει τον παίκτη, σε πόση θα αρχίσει να τον κυνηγάει και σε πόση θα αρχίσει να τον χτυπάει, και κάθε περίπτωση θα πραγματοποιείτε με ελέγχους if. Τέλος υπάρχουν και μεταβλητές που ορίζουν την ταχύτητα του αντιπάλου όσο αφορά το περπάτημα και την επίθεσή του αλλά και την περιστροφή του όταν αλλάζει κατεύθυνση. Ακολουθεί ο κώδικας Javascript.

```
var Distance;
var Target : Transform;
var lookAtDistance = 25.0;
var chaseRange = 15.0;
var attackRange = 1.5;
var moveSpeed = 5.0;
var Damping = 6.0;
var attackRepeatTime = 1;

var TheDamage = 40;

private var attackTime : float;

var controller : CharacterController;
var gravity : float = 20.0;
private var MoveDirection : Vector3 = Vector3.zero;

function Start ()
{
	attackTime = Time.time;
}

function Update ()
{
	if(RespawnMenuV2.playerIsDead == false)
	{
		Distance = Vector3.Distance(Target.position, transform.position);

		if (Distance < lookAtDistance)
		{
			lookAt();
		}

		if (Distance > lookAtDistance)
		{
			renderer.material.color = Color.green;
		}

		if (Distance < attackRange)
		{
			attack();
		}
		else if (Distance < chaseRange)
		{
```

```
                    chase ();
                }
        }
}

function lookAt ()
{
        renderer.material.color = Color.yellow;
        var rotation = Quaternion.LookRotation(Target.position -
transform.position);
        transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
Time.deltaTime * Damping);
}

function chase ()
{
        renderer.material.color = Color.red;

        moveDirection = transform.forward;
        moveDirection *= moveSpeed;

        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
}

function attack ()
{
        if (Time.time > attackTime)
        {
                Target.SendMessage("ApplyDamage", TheDamage);
                Debug.Log("The Enemy Has Attacked");
                attackTime = Time.time + attackRepeatTime;
        }
}

function ApplyDamage ()
{
        chaseRange += 30;
        moveSpeed += 2;
        lookAtDistance += 40;
}
```

Οι έλεγχοι για το πώς θα δρα ο αντίπαλος ανάλογα με την απόστασή του από τον παίκτη γίνεται μέσα στην Update() με μερικές if που ενσωματώνονται μέσα σε μια άλλη που ελέγχει αν ο παίκτης είναι ζωντανός. Αρχικά ορίζεται η απόσταση μεταξύ του παίκτη και του αντιπάλου στην μεταβλητή Distance με την εκτέλεση του Vector3.Distance(Target.position, transform.position) που επιστρέφει την απόσταση μεταξύ των δύο παραμέτρων σε float, και έπειτα γίνονται οι έλεγχοι. Έτσι, ελέγχεται στη συνέχεια αν η απόσταση είναι μικρότερη από την απόσταση που θέλουμε να κοιτάει ο αντίπαλος, η οποία έχει οριστεί αρχικά στην μεταβλητή lookAtDistance, τότε εκτελείται η μέθοδος lookAt(). Παρομοίως αν η Distance μειωθεί κάτω από την attackRange τότε ο αντίπαλος επιτίθεται με την εκτέλεση της attack(), και αν πάει κάτω από την chaseRange ο εχθρός κυνηγάει τον παίκτη με την εκτέλεση της chase(). Κατά την αλλαγή καταστάσεων μπορούν να εκτελούνται τα διάφορα animation του εχθρού με την κλήση της animation.Play(“”) και σαν παράμετρο σε String το όνομα του animation.

Στη συνέχεια του κώδικα υλοποιούνται οι παραπάνω τρεις μέθοδοι. Συγκεκριμένα στη lookAt() αποθηκεύεται στην μεταβλητή rotation με την εκτέλεση της Quaternion.LookRotation(Target.position - transform.position) η κατεύθυνση προς την οποία θα κοιτάξει ο αντίπαλος, και αυτή είναι αυτή του παίκτη (Εικόνα 45), και με την επόμενη σειρά ουσιαστικά γίνεται σφαιρικό rotation του αντιπάλου προς αυτή τη κατεύθυνση με την Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * Damping).

Εικόνα 45 "Η απόσταση έχει μειωθεί και ο αντίπαλος μας κοιτάει"

Παρομοίως στην chase() υπολογίζεται η γωνία προς την οποία πρέπει να πάει ο εχθρός και μετακινείται προς αυτόν τον άξονα με μια συγκεκριμένη ταχύτητα, και με την attack() γίνεται επίθεση στον παίκτη και μειώνεται η ζωή του κατά όσο έχουμε ορίσει στην μεταβλητή TheDammage με την ApplyDamage().

Στη συνέχεια προστέθηκαν κώδικες Javascript οι οποίοι προσθέτουν στην ακουστική και του παιχνιδιού όσο αφορά τον ρεαλισμό.

ΠΡΟΣΘΗΚΗ ΗΧΟΥ - ΠΑΤΗΜΑΣΙΕΣ

Ακουθεί ο κώδικας Javascript ο οποίος τοποθετήθηκε στο First Person Controller και ουσιαστικά ορίζει μια μεταβλητή ήχου, η οποία θα περιέχει διάφορα κλιπ, σ'αυτή τη περίπτωση διαφορετικούς ήχους footstep (πατημασιάς) και ελέγχει αν ο παίκτης κινείται τότε παίζει τυχαίο κλιπ από τις "πατημασιές".

```
var Steps : AudioClip[];
private var isWalking : boolean = false;
private var controller : CharacterController;

function Awake()
{
	controller = GetComponent(CharacterController);
}

function Update()
{
	if(controller.velocity.sqrMagnitude > 0.15 )
	{
		isWalking = true;
	}
	else
	{
		isWalking = false;
	}
}

InvokeRepeating("Walking", 0, 0.7);

function Walking()
{
	if(isWalking)
	{
		AudioSource.PlayClipAtPoint(Steps[Random.Range(0,Steps.length)]
	, gameObject.transform.position);
	}
}
```

Αρχικά ορίζονται τρεις μεταβλητές, η Steps περιέχει τα κλιπ των πατημασιών, η isWalking που θα παίρνει την τιμή true ή false ανάλογα με το αν περπατάει ο παίκτης ή όχι και η controller που θα είναι ο παίκτης, δηλαδή το First Person Controller. Στη συνέχεια με έναν έλεγχο if ελέγχεται αν μετακινείται ο παίκτης με τη γραμμή κώδικα controller.velocity.sqrMagnitude > 0.15 και εισχωρεί την κατάλληλη τιμή στην μεταβλητή isWalking. Η μέθοδος Walking() απλώς ελέγχει με μια if αν η isWalking είναι true, δηλαδή αν ο παίκτης κινείται, τότε παίζει παίζει έναν τυχαίο ήχο από τον πίνακα ήχων Steps (Steps[Random.Range(0,Steps.length)]) που δηλώσαμε στο σημείο που είναι ο παίκτης gameObject.transform.position. Τέλος με την InvokeRepeating("Walking", 0, 0.7); εκτελεί την μέθοδο που βρίσκεται στη πρώτη παράμετρο, σ'αυτή τη περίπτωση την Walking(), ξεκινόντας από 0 δεπτερόλεπτα, κάθε 0.7 δεπτερόλεπτα. Ουσιαστικά επαναλαμβάνει έναν τυχαίο ήχο πατημασιάς κάθε 0.7 δεπτερόλεπτα.

ΔΗΜΙΟΥΡΓΙΑ ΚΕΝΤΡΙΚΟΥ ΜΕΝΟΥ

Για τη δημιουργία ενός κεντρικού μενού δημιουργήθηκε ένα καινούριο Scene, αντιγραφή του Game που είναι το παιχνίδι, απλώς αφαιρέθηκαν τα GameObjects που δεν είναι απαραίτητα. Η κάμερα δείχνει σε ένα μέρος του παιχνιδιού και είναι fixed, δεν μπορεί να κινηθεί. Προστέθηκαν δύο GameObject τύπου Text που γράφουν Play και Quit. Έπειτα προτέθηκε ο παρακάτω κώδικας Javascript σε κάθε Text GameObject.

```
var isQuit=false;




function OnMouseEnter()

{
        //change text color
        renderer.material.color=Color.red;

}

function OnMouseExit()

{
```

```
        //change text color
        renderer.material.color=Color.white;

}


function OnMouseUp()

{
        //is this quit
        if (isQuit==true)

        {
                //quit the game
                Application.Quit();
        }
        else

        {
                //load level
                Application.LoadLevel(1);
        }

}

function Update()

{
        //quit game if escape key is pressed
        if (Input.GetKey(KeyCode.Escape))

        {

                Application.Quit();

        }

}
```

Ο παραπάνω κώδικας αλλάζει το χρώμα του κάθε Text Object με τις μεθόδους OnMouseEnter() και OnMouseExit() κάθε φορά που ο δείκτης του ποντικιού βρίσκεται πάνω από το κείμενο ή όχι αντίστοιχα. Στη συνέχεια ελέγχει με μια μεταβλητή isQuit και πράττει ανάλογα, είτε βγαίνει από το παιχνίδι με τη χρήση της εντολής Application.Quit() είτε φορτώνει το παιχνίδι με την εντολή Application.LoadLevel(1). Σ'αυτό το σημείο πρέπει να αναφερθεί ότι οι πίστες δηλώνονται από το μενού File -> Build Settings... και ανάλογα με τη σειρά που θα προστεθούν τα Scene στην λίστα οθονών, παίρνουν τον αριθμό ξεκινώντας από

τον αριθμό μηδέν. Τέλος, αν πατηθεί το πλήκτρο Escape τότε πάλι γίνεται έξοδος από το παιχνίδι.

Εικόνα 46 ¨Κεντρικό Μενού"

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Κίνηση εμπρός – Πλήκτρο W

Κίνηση πίσω – Πλήκτρο S

Κίνηση αριστερά – Πλήκτρο A

Κίνηση δεξιά – Πλήκτρο D

Αλλιλεπίδραση με πόρτες – Πλήκτρο E

Συλλογή σελίδας – Πλήκτρο E

Έξοδος – Πλήκτρο Escape

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η Unity προσφέρει στον ενδιαφερόμενο “developer” την ευκαιρία να μπει στη θέση των δημιουργών διάφορων αγαπημένων του παιχνιδιών (Εικόνα 47) και να δημιουργήσει ο ίδιος τον αγαπημένο του χαρακτήρα, να του δώσει ζωή και να πραγματοποιήσει το δικό του φανταστικό περιβάλλον. Σκοπός της παρούσας πτυχιακής είναι να δείξει στο κοινό πως μπορεί να εκμεταλλευτεί το περιβάλλον της Unity και με λίγες γνώσεις προγραμματισμού να φτιάξει ένα παιχνίδι από το μηδέν. Η Unity παρέχει οτιδήποτε χρειάζεται κάποιος για την ανάπτυξη του παιχνιδιού του, από οδηγούς στον ιστοχώρο της, μέχρι και απίστευτα χαρακτηριστικά και επιλογές στο πρόγραμμά της. Τέλος, η πολλαπλή επιλογή πλατφορμών για την κυκλοφορία των παιχνιδιών την κάνει πιο προσιτή αφού οι κονσόλες, οι υπολογιστές και γενικά μία συσκευή αναπαραγωγής παιχνιδιών δε λείπει από τις ζωές των ανθρώπων, με τα βιντεοπαιχνίδια να αποτελούν μία από τις πιο «δυνατές» μορφές ψυχαγωγίες στις μέρες μας, όπου η τεχνολογία κιόλας αυξάνεται συνεχώς.

Εικόνα 47 "Διάφοροι ήρωες αγαπημένων παιχνιδιών"