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

Πτυχιακή εργασία

Δημιουργία παιγνιδιού με Unity 3D και JavaScript

Του φοιτητή
Αργυρούδη Δημήτριου
Ράπτης
Αρ. Μητρώου: 09/3457

Επιβλέπων
καθηγητής
δρ. Πασχάλης

Θεσσαλονίκη **2015**

Θεσσαλονίκη 2015

Πνευματικά δικαιώματα

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Περίληψη

Η παρούσα πτυχιακή εργασία είχε στόχο την ανάπτυξη ενός πλήρους λειτουργικού ηλεκτρονικού παιχνιδιού, με συνοχή σεναρίου και ομαλή λειτουργία. Μέσα από την πορεία της ανάπτυξης της εφαρμογής του συγκεκριμένου ηλεκτρονικού παιχνιδιού διαφαίνονται οι σύγχρονες τάσεις και τεχνικές της βιομηχανίας των ηλεκτρονικών παιχνιδιών καθώς αναφέρονται και οι κυριότερες αντιλήψεις στο τεχνολογικό πλαίσιο της εν λόγω εφαρμογής.

Θεματική Περιοχή: Γραφικά Υπολογιστών

Λέξεις Κλειδιά: Unity3D, Ανάπτυξη Παιχνιδιού, Javascript, Γραφικά Υπολογιστών

Abstract

This project was aimed at developing a fully functional computer game, concrete scenario, and smooth gameplay. Through the course of development of the current application, game industry's modern trends and techniques emerge and also the main concepts of the technological context of this application.

Subject Area: Computer Graphics

Keywords: Unity3D, Game Development, Javascript, Computer Graphics

Κατάλογος περιεχομένων

Κατάλογος εικόνων

Εισαγωγή

Ένα πιθανό μονοπάτι που δύναται να ακολουθήσει ένας απόφοιτος τμήματος πληροφορικής είναι αυτό της βιομηχανίας του θεάματος. Οπτικά εφέ, επεξεργασία ήχου, μορφοποίηση εικόνων είναι λίγα μόνο θέματα ενδιαφέροντος από τα διαθέσιμα με τα οποία μπορεί κάποιος να ασχοληθεί. Ένα εκ των ποιο ενδιαφερόντων, κατά την γνώμη μου, είναι το κομμάτι της ανάπτυξης ηλεκτρονικών παιχνιδιών ή της ψηφιακής πραγματικότητας. Ο κλάδος της ανάπτυξης παιχνιδιών είναι ένας τομέας εργασίας με πολύ μεγάλους τζίρους. Στόχος αυτής της πτυχιακής εργασίας είναι, να παρουσιαστούν και να αναλυθούν, κατά το δυνατό, οι σύγχρονες τεχνικές ανάπτυξης που ακολουθούνται από τους μεγάλους οργανισμούς ανάπτυξης παιχνιδιών αλλά και οι τεχνικές και τα βασικά αντικείμενα που χρησιμοποιούν οι προγραμματιστές κατά την διάρκεια της ανάπτυξης. Στόχος είναι να αναπτυχθεί ένα ολοκληρωμένο ηλεκτρονικό παιχνίδι δημιουργημένο σύμφωνα με τα πρότυπα της αγοράς αλλά και της σύγχρονης γνωστικής τεχνοτροπίας. Συνοπτικά, θα ακολουθήσει μια παρουσίαση της της παράδοσης και της ιστορίας του ηλεκτρονικού παιχνιδιού καθώς και μια πιο συγκεκριμενοποιημένη ανάλυση του είδους το οποίο θα μας απασχολήσει κατά την διάρκεια της συγκεκριμένης πτυχιακής εργασίας. Θα παρουσιαστούν επίσης βασικές τεχνικές ανάλυσης, σχεδίασης και κωδικοποίησης καθώς και παράθεση του απαραίτητου/προτεινόμενου λογισμικού που χρησιμοποιήθηκε κατά την πορεία της εξέλιξης της εργασίας. Τέλος, θα σχολιαστούν τα κυριότερα σημεία της εφαρμογής που αναπτύχθηκε, συντάσσοντας ταυτόχρονα και έναν οδηγό χρήσης για αυτήν.

1 Εισαγωγή στην Ανάπτυξη Ηλεκτρονικών Παιχνιδιών

Ανάπτυξη ηλεκτρονικού παιχνιδιού ονομάζεται η διαδικασία της δημιουργίας ενός βιντεοπαιχνιδιού. Ο όρος βιντεοπαιχνίδι αναφέρεται σε ένα παιχνίδι υλοποιημένο προς και αποδιδόμενο από ψηφιακά μέσα εικόνας και ήχου όπως η τηλεόραση ή ο υπολογιστής. Η διαδικασία ανάπτυξης ενός ηλεκτρονικού παιχνιδιού μπορεί να είναι δουλειά ενός και μόνον προγραμματιστή ή να απασχολήσει μια, ίσως και παραπάνω σε πολλές περιπτώσεις, εταιρείες. Στην εποχή μας, μια φυσιολογική διαδικασία ανάπτυξης ενός ηλεκτρονικού παιχνιδιού, συνήθως, χρηματοδοτείται από έναν εκδότη ή εκδοτικό οργανισμό, υλοποιείται σαν έργο από μια εταιρεία ανάλογου αντικειμένου και μπορεί να διαρκέσει μερικά χρόνια. Ο αντίλογος είναι τα Indie Games (Ανεξάρτητα Παιχνίδια). Τα Indie Games είναι τίτλοι που αναπτύσσονται και δημοσιεύονται από έναν ή μια μικρή ομάδα προγραμματιστών και έχουν από μηδαμινό έως πολύ μικρό κοστολόγιο. Τέτοια παιχνίδια χαρακτηρίζονται από τον απλοϊκό σχεδιασμό τους και την έλλειψη εντυπωσιακών εφέ, με τα χρόνια όμως έχουν δημιουργήσει την δικιά τους κουλτούρα και το ανάλογο “καλτ” κοινό. Η εμπορική ανάπτυξη βιντεοπαιχνιδιών ξεκίνησε περί το 1970. Τότε η χαμηλές δυνατότητες των υπολογιστικών συστημάτων καθιστούσαν περισσότερο λογική την επιλογή ανάπτυξης Indie τίτλων παρά την σύσταση μεγάλων εταιρειών για μια δουλειά τόσο χαμηλού κοστολογίου. Με την πάροδο τον χρόνων όμως η βιομηχανία ανάπτυξης βιντεοπαιχνιδιών άρχισε να αναπτύσσεται ραγδαία, σε σημείο εν έτει 2015 να υπάρχουν εταιρείες – οικονομικοί κολοσσοί με μέσο όρο κόστος ανάπτυξης ενός παιχνιδιού τα 25 εκατομμύρια δολάρια ενώ υπάρχουν περιπτώσεις που το κόστος παραγωγής άγγιξε τα 265 εκατομμύρια δολάρια!

1.1 Φάσεις

Η διαδικασία ανάπτυξης ενός βιντεοπαιχνιδιού περνάει από συγκεκριμένες φάσεις ανάπτυξης. Αρχικά προετοιμάζονται κάποια πρωτότυπα και έγγραφα σχεδίασης ώστε να τεθεί η ιδέα προς συζήτηση. Εάν η ιδέα γίνει αποδεκτή τότε η ομάδα ανάπτυξης λαμβάνει επιχορήγηση και ξεκινάει η πλήρους μεγέθους ανάπτυξη. Συνήθως η ομάδα ανάπτυξης αποτελείται από 20 με 100 άτομα ποικίλων δραστηριοτήτων. Τα άτομα

μπορούν να περιλαμβάνουν σχεδιαστές γραφικών και μοντέλων, προγραμματιστές testers και managers που έχουν τον ρόλο να συντονίζουν τα διάφορα μέλη.

1.1.1 Pre-production

Η φάση προ-παραγωγής ή αλλιώς σχεδίασης είναι η περίοδος σχεδιασμού του παιχνιδιού όπου περιλαμβάνει την επεξεργασία της ιδέας και του πλαισίου και την σύνταξη των πρώτων εγγράφων σχεδιασμού. Ο στόχος είναι η καλή σύνταξη των στόχων και των απαιτήσεων του λογισμικού και τα χρονικά πλαίσια στα οποία αυτά θα υλοποιηθούν. Το ένα από τα δύο σημαντικότερα αντικείμενα σε αυτό το σημείο είναι να περιγραφεί καλά η πρόταση του παιχνιδιού (High Concept) και να αναπτυχθεί όσο το δυνατόν αναλυτικότερο πλάνο εργασιών, στόχων, πόρων και χρονοδιαγράμματος όσων αφορά την εφαρμογή. Το δεύτερο σημαντικότερο αντικείμενο της προ-παραγωγικής φάσης είναι να δημιουργηθούν πρωτότυπα. Τα πρωτότυπα είναι κομμάτια που παρουσιάζουν σημεία της λειτουργικότητας του τελικού παιχνιδιού χωρίς να δίνεται σημασία στα γραφικά και γενικότερα στο front-end επίπεδο. Μεγάλη βάση δίνεται στην ανάπτυξη και εφαρμογή αλγορίθμων καθώς και λειτουργικών στιγμιότυπων της εφαρμογής τα οποία μπορούν να χρησιμοποιηθούν αρχικά ως παρουσίαση της ιδέας και σε δεύτερο χρόνο ως δομικό στοιχείο της ανάπτυξης του προϊόντος.

1.1.2 Παραγωγή

Παραγωγή ονομάζεται η κυρίως φάση της ανάπτυξης ενός βιντεοπαιχνιδιού. Σε αυτό το σημείο παράγονται τα αγαθά (assets) και ο πηγαίος κώδικας για το παιχνίδι. Ως αγαθά (assets) χαρακτηρίζονται όλα τα αντικείμενα τα οποία συμβάλλουν στην δημιουργία του τελικού αποτελέσματος και αποτελούνται από γραφικά, ηχητικά clip, τρισδιάστατα μοντέλα, σκηνές, animations και ο,τιδήποτε άλλο μπορεί να αποτελέσει κομμάτι του παιχνιδιού. Σε αυτή τη φάση οι εργασίες διακρίνονται σε ομάδες. Παραδοσιακά υπάρχει η ομάδα των προγραμματιστών που γράφουν τον κώδικα για το παιχνίδι, η ομάδα των γραφιστών που σχεδιάζουν γραφικά και δισδιάστατες υφές (textures) και η ομάδα των σχεδιαστών που σχεδιάζουν και εμψυχώνουν (animate) τα τρισδιάστατα αντικείμενα, η ομάδα των παραγωγών ήχου, συνθετών, ηχοληπτών και μουσικών που συνθέτουν, ηχογραφούν και βελτιστοποιούν την μουσική για το

παιχνίδι, η ομάδα των σχεδιαστών επιπέδων (level designers) όπου, στα παιχνίδια που αυτό είναι επιθυμητό, σχεδιάζουν και αναπτύσσουν διαφορετικές πίστες και επίπεδα δυσκολίας και η ομάδα των σεναριογράφων που συλλαμβάνουν και συγγράφουν το σενάριο, τους διαλόγους και τους χαρακτήρες των NPC. NPC είναι τα αρχικά του Non Player Character και συμβολίζουν οποιονδήποτε άλλο χαρακτήρα στο παιχνίδι μπορεί να αλληλεπιδράσει με τον παίκτη χωρίς να τίθεται ποτέ υπό τον έλεγχό του. Στην παρούσα εργασία δεν θα επεκταθώ στην μεθοδολογία και την διαδικασία παραγωγής όλων των ομάδων ανάπτυξης παρά μόνο στον προγραμματισμό, που είναι το σχετικό κομμάτι και επηρέασε άμεσα την προκείμενη εφαρμογή.

1.1.3 Προγραμματισμός

Ακόμα και εάν επιλεχθεί μια έτοιμη μηχανή ανάπτυξης παιχνιδιών, όπως και γίνεται στο πλείστο των περιπτώσεων, ένα πολύ μεγάλο κομμάτι προγραμματισμού είναι απαραίτητο. Η ομάδα ή οι ομάδες των προγραμματιστών επιλέγουν και υλοποιούν πρωτότυπα ιδεών και διορθώνουν τα διάφορα προβλήματα (bugs) που προκύπτουν κατά την διάρκεια της ανάπτυξης. Συνήθως οι ομάδες δουλεύουν με ευέλικτες (agile) μεθοδολογίες καθότι το παραδοσιακό μοντέλο του καταρράκτη αυξάνει την πολυπλοκότητα και το κόστος, δυσχεραίνει την διαδικασία ανάπτυξης και αποδίδει ένα κατώτερο ποιοτικά προϊόν στην βιομηχανία των βιντεοπαιχνιδιών. Συνήθως χρησιμοποιείται η επαναληπτική διαδικασία επί των αρχικών προτύπων. Έτσι σε σύντομες περιόδους παραδίδεται από την ομάδα λειτουργικό προϊόν που μπορεί να δοκιμαστεί, να παρουσιαστεί και το κυριότερο, να επεκταθεί κατά βούληση. Η λογική είναι ένας project manager να καθοδηγεί την ομάδα σε κάθε επανάληψη, βοηθώντας στην διευκρίνηση των στόχων, την επιλογή διαδικασιών προς υλοποίηση και το όλο πλαίσιο της ανάπτυξης (framework, τεχνολογία, τεχνοτροπία). Μετά το πέρας συγκεκριμένης και ορισμένης από την ομάδα ημερομηνίας, συναντιούνται ώστε να συζητήσουν τους μέχρι τώρα πραγματοποιημένους στόχους και τις μελλοντικές ενέργειες, ξεκινώντας έτσι έναν νέο κύκλο επανάληψης. Ανά συγκεκριμένο αριθμό ολοκληρωμένων κύκλων δίνεται πιθανώς και μια έκδοση στον πελάτη και στις υπόλοιπες ενδιαφερόμενες ομάδες.

1.1.4 Testing

Εξίσου σημαντικό κομμάτι στην διαδικασία παραγωγής είναι οι δοκιμές. Δοκιμές γίνονται καθ' όλη την διάρκεια της ανάπτυξης από την στιγμή που υπάρχουν playable levels (επίπεδα που μπορούν να παιχθούν). Όσο η ανάπτυξη του παιχνιδιού πλησιάζει στο τέλος, προσλαμβάνεται μεγάλος αριθμός για full-time απασχόληση, σκοπό να δοκιμαστούν όλα τα καινούρια χαρακτηριστικά καθώς και γενικότερα οι μηχανισμοί του παιχνιδιού σε βάθος και η αλληλεπίδραση του παίκτη με το σύνολο της εφαρμογής. Το testing είναι από τα σημαντικότερα και πολύ κρίσιμα σημεία της ανάπτυξης διότι θα παραχθεί το πρώτο feedback για τους προγραμματιστές από μεγάλο αριθμό ανθρώπων, θα κριθούν τα λάθη και οι αστοχίες του παιχνιδιού και θα προταθούν ίσως κάποιες τελευταίες βελτιώσεις. Για αυτό το λόγο οι εταιρείες δίνουν αρκετά λεφτά στην πρόσληψη ανάλογων ανθρώπων μόνο για να δοκιμάσει την εφαρμογή. Επιτυχές δοκιμές σημαίνουν αυτόματα και επιτυχής σύνδεση όλων των χαρακτηριστικών του παιχνιδιού αναμεταξύ τους. Επίσης συνεπάγεται την ευχαρίστηση του πελάτη και του end user. Αντίθετα ελλιπές testing μπορεί να οδηγήσει σε ασυμφωνίες χαρακτηριστικών λίγο πριν την τελική παράδοση (όπου το κόστος διόρθωσής τους αυξάνεται) και δυσαρέσκεια του πελάτη και των χρηστών.

1.1.5 Alpha

Η φάση Alpha ενεργοποιείται όταν το προϊόν θεωρείται playable, τα χαρακτηριστικά του είναι ως επί το πλείστον υλοποιημένα και τα assets σχεδόν ολοκληρωμένα. Στην φάση alpha το προϊόν διανέμεται προς testing σε επιλεγμένη μερίδα κοινού, όπου είναι πλέον end users και όχι προσληφθέντες της εταιρείας. Συνήθως ένα παιχνίδι είναι στην Alpha φάση του 8 με 10 μήνες πριν την παράδοση. Κύριο μέλημα των προγραμματιστών σε αυτή τη φάση είναι να μαζέψουν όσο το δυνατόν περισσότερο feedback, να διορθώσουν τυχόν λάθη, να ολοκληρώσουν τις βασικές λειτουργίες και το βασικό codebase (εάν αυτό δεν έχει γίνει ήδη) και να ετοιμάσουν το λογισμικό για την επόμενη φάση.

1.1.6 Beta

Στην Beta φάση το παιχνίδι θεωρείται ολοκληρωμένο από θέμα κώδικα, assets και χαρακτηριστικών και περιέχει μόνο bugs που δεν εμποδίζουν όμως την

λειτουργικότητά του. Για άλλη μια φορά το παιχνίδι διανέμεται σε end users, κατά βάση πολυπληθέστερους από την Alpha φάση για να εντοπίσουν αυτά τα bugs και να παραδώσουν feedback. Συνηθίζεται πολλές φορές να γίνεται Open Beta Phase σε ένα παιχνίδι όπου η διαφορά είναι ότι οι χρήστες δεν επιλέγονται με πρόσκληση και δεν είναι περιορισμένοι σε αριθμό αλλά το παιχνίδι μπορεί να διατεθεί σε όποιον το επιθυμεί, δωρεάν με σκοπό πρώτον την διαφήμιση και μια πρόγευση του παιχνιδιού και δεύτερον το feedback από πολύ μεγαλύτερο και διαφορετικό σε γνωστικό επίπεδο κοινό. Συχνά το Open Beta γίνεται σε διαδικτυακά παιχνίδια με σκοπό μια εταιρεία να δοκιμάσει την αντοχή των server της.

1.1.7 Gold Master

Gold Master είναι η φάση ακριβώς πριν την μαζική παραγωγή. Το παιχνίδι θεωρείται ολοκληρωμένο και στο μέτρο του δυνατού bug-free και γίνεται το build της βασικής έκδοσης που θα τεθεί υπό διανομή.

1.1.8 Post-Production

Μετά την παραγωγή του, ένα βιντεοπαιχνίδι μπαίνει στην φάση της συντήρησης. Για τα βιντεοπαιχνίδια που παράγονταν για κονσόλες μέχρι πρόσφατα, δεν υπήρχε καθόλου η φάση της συντήρησης καθώς το παιχνίδι παρέμενε ως είχε και δεν γινόταν καμία επιπλέον παραγωγή. Πλέον, με την εισαγωγή του internet και στις κονσόλες ακόμα και τα βιντεοπαιχνίδια για κονσόλες έχουν φάση συντήρησης όπου καταλήγει στον καταναλωτή συνήθως με την μορφή DLC (Downloadable Content). Για τα βιντεοπαιχνίδια που παράγονται για ηλεκτρονικούς υπολογιστές η φάση της συντήρησης ήταν και είναι πάντα παρούσα. Η διαδικασία έχεις ως εξής, όταν κριθεί ότι έχει συλλεχθεί αρκετή πληροφορία που αφορά bugs ή μικροαλλαγές στο gameplay (τρόπο παιχνιδιού) ξεκινά η παραγωγή ενός patch από τους προγραμματιστές. Ένα patch μπορεί να φέρει από διορθώσεις μέχρι και καινούρια, επιπρόσθετα χαρακτηριστικά σε ένα παιχνίδι και η παραγωγή του μπορεί να διαρκέσει από βδομάδες έως μήνες, ανάλογα με την ποσότητα των προσθηκών ή των διορθώσεων. Η φάση της συντήρησης κρατάει επ' αορίστου χρόνου, έως ότου η αρμόδια εταιρεία ανακοινώσει την λήξη της.

1.1.9 Marketing

Η βιομηχανία των βιντεοπαιχνιδιών έχει ολόκληρο κλάδο για την διαφήμιση καθώς πλέον ένα τέτοιο έργο κοστίζει εκατομμύρια και αναμένεται να έχει κέρδος. Οι βασικές αρχές που ακολουθούνται στην διαφήμιση και στην διανομή είναι ανάλογες με τις μεθόδους που χρησιμοποιούνται στην βιομηχανία των κινηματογράφου και της μουσικής. Αρχικά γίνεται μια ανακοίνωση από αρμόδιους εκπροσώπους της εταιρείας για την ιδέα και τα σχέδια υλοποίησης, όπου συνήθως είναι μια εκτιμώμενη ημερομηνία όπου η ιδέα θα μπορεί να είναι διαθέσιμο προϊόν. Όσο η ανάπτυξη προχωράει καταγράφεται ένα trailer, ανάλογο με αυτό των ταινιών, το οποίο μπορεί να περιέχει τόσο animation και cinematics όσο και σκηνές του ίδιου του gameplay. Συνηθίζεται η ανακοίνωση και η πρώτη προβολή ενός trailer να γίνεται σε κάποιο μεγάλο γεγονός ή συνέδριο. Κλασικό παράδειγμα αποτελεί η ετήσια έκθεση Electronic Entertainment Expo γνωστή κατά κόρον ως E3. Στην E3 κάθε χρόνο ανακοινώνονται, παρουσιάζονται και σχολιάζονται, μεταξύ άλλων, όλοι οι μεγάλοι τίτλοι από όλες τις μεγάλες εταιρείες της βιομηχανίας βιντεοπαιχνιδιών που βρίσκονται ήδη υπό ανάπτυξη και αναμένονται μέσα στον επόμενο χρόνο, καθώς και οι νέες κυκλοφορίες. Αυτό γίνεται για αρκετούς λόγους. Σε μια τόσο μεγάλη διοργάνωση θα δημιουργηθεί ανταγωνισμός ανάμεσα σε εταιρείες, ακόμα και ανάμεσα σε διαφορετικούς τίτλους μιας ίδιας εταιρείας, για μερίδιο της αγοράς. Εάν έχεις εξέχον προϊόν μπορείς να “χτυπήσεις” άμεσα και νόμιμα τον ανταγωνισμό. Επίσης, ένα τέτοιο γεγονός έχει μεγάλη συρροή κόσμου καθώς και πολλές εκατομμύρια προβολές, streamings και μεταφορτώσεις, γεγονός που σημαίνει μαζική γνωστοποίηση και διαφήμιση στην κοινότητα. Οπότε είναι μια μοναδική ευκαιρία διαφήμισης καθώς όλα τα φώτα της δημοσιότητας πέφτουν πάνω στο προϊόν σου μαζικά. Παρόλα αυτά trailer βιντεοπαιχνιδιών δημοσιεύονται ανά την διάρκεια του έτους και ανεξάρτητα εκδηλώσεων, απευθείας από την εταιρεία ή τον αρμόδιο φορέα. Πολλές φορές αυτό γίνεται είτε γιατί ο τίτλος δεν είναι έτοιμος ή δεν έχει μπει καν στην διαδικασία παραγωγής. Επίσης μπορεί να αποτελεί ένα διαφημιστικό τέχνασμα ούτως ώστε μια εταιρεία να μην διχάσει το αγοραστικό κοινό ανάμεσα σε δύο διαφορετικούς τίτλους της. Σε κάποιες περιπτώσεις τα trailer έρχονται να συμπληρώσουν ή να επαυξήσουν ένα προηγούμενο trailer του ίδιου τίτλου. Εκτός από τα trailer όμως μπορεί να διατεθεί

και ένα demo του βιντεοπαιχνιδιού στα πλαίσια κάποιας εκδήλωσης ούτως ώστε οι χρήστες να το γνωρίσουν από πρώτο χέρι. Σε μερικές περιπτώσεις μάλιστα, οργανώνεται και ένας διαγωνισμός με κάποιο χρηματικό έπαθλο γύρω από το demo ούτως ώστε να προσελκύσει μεγαλύτερο κοινό και ανταπόκριση των streamers, χρηστών - gamers όπου παίζουν παιχνίδια επαγγελματικά και διαθέτουν την όλη διαδικασία στο internet. Τέλος, μια από τις ισχυρότερες μορφές διαφήμισης είναι η επιλογή ενός τίτλου για benchmark testing. Συγκεκριμένα, πολλές εταιρείες hardware υπολογιστών για να διαφημίσουν τα τεχνικά χαρακτηριστικά του προϊόντος τους επιλέγουν συγκεκριμένα βιντεοπαιχνίδια τα οποία χαρακτηρίζονται από υψηλές απαιτήσεις υλικού και υποβάλλουν το προϊόν σε διάφορα test αντοχής και απόδοσης στο πλαίσιο του βιντεοπαιχνιδιού.

Μπορεί όλα τα ηλεκτρονικά παιχνίδια να έχουν παρόμοια διαδικασία ανάπτυξης, παρόλα αυτά, καθώς υπάρχουν πολλά διαφορετικά ήδη παιχνιδιών, υπάρχουν και κάποιες διαφορές στην υλοποίηση. Στην συνέχεια θα δούμε τις ειδικές λεπτομέρειες που αφορούν το παιχνίδι περιπέτειας ή adventure game, είδος στο οποίο κατατάσσεται η εν λόγω εφαρμογή.

2 Παιχνίδι Περιπέτειας

Παιχνίδι Περιπέτειας είναι το είδος εκείνο του ηλεκτρονικού παιχνιδιού όπου ο παίκτης αναλαμβάνει τον ρόλου του πρωταγωνιστή σε μια διαδραστική ιστορία οδηγούμενη από γρίφους. Βασικό χαρακτηριστικό ενός παιχνιδιού περιπέτειας είναι η αλληλεπίδραση του παίκτη με διάφορα αντικείμενα ενός χώρου ή/και ο διάλογος με NPC (μη-χειριζόμενους χαρακτήρες). Κατά βάση, η αρχή και το τέλος του παιχνιδιού ορίζονται από ένα συγκεκριμένο, συνήθως στατικό, σενάριο και δεν υπάρχει η ελευθερία που υπάρχει σε παιχνίδια τύπου RPG (παιχνίδι ρόλων). Ο παίκτης καλείται να επιλύσει διαφόρων ειδών γρίφους για την εξέλιξη της πορείας του σεναρίου. Οι θεματικές κατηγορίες ενός παιχνιδιού περιπέτειας καλύπτουν ένα πολύ μεγάλο εύρος όπως δράσης, μυστηρίου, τρόμου, κωμωδίας, σάτιρας και φαντασίας. Οι γρίφοι συνήθως βασίζονται σε ικανότητες του παίκτη όπως επίλυση σπαζοκεφαλιών, puzzle και παρατηρητικότητα καθώς και σε γνώσεις, μαθηματικών, ιστορίας, μουσικής, φυσικής και άλλων. Τα παιχνίδια περιπέτειας άκμασαν από τα τέλη της δεκαετίας του

'80 έως και τα μέσα της δεκαετίας του '90, όπου θεωρούνταν το ανώτερο τεχνολογικά είδος παιχνιδιών. Ενώ συνεχίζουν να βγαίνουν τίτλοι στον δυτικό κόσμο έχουν πλέον πολύ μικρότερο μερίδιο της αγοράς. Αντίθετα στην Ιαπωνία συνεχίζουν να κατέχουν μεγάλο κομμάτι της αγοράς (περίπου 70% των παιχνιδιών για ηλεκτρονικό υπολογιστή) υπό την μορφή του οπτικού μυθιστορήματος (visual novel). Τα οπτικά μυθιστορήματα παρόλο που ανήκουν στην κατηγορία των παιχνιδιών δράσης έχουν αρκετά διαφορετικά χαρακτηριστικά καθώς η αλληλεπίδραση του χρήστη περιορίζεται συνήθως σε ανάγνωση διαλόγων και επιλογή διαλόγων, κάτι που εξαφανίζει σχεδόν ολοκληρωτικά την ύπαρξη γρίφων καθώς επίσης και το γραφικό τους περιβάλλον είναι σχεδιασμένο σε τεχνοτροπία manga.

Ο όρος Παιχνίδι Περιπέτειας έχει τις ρίζες του στην δεκαετία του 70 όπου ένα βιντεοπαιχνίδι εν ονόματι Adventure όρισε ένα δικό του στυλ παιχνιδιού το οποίο μεταγενέστερα αντιγράφτηκε σε πολλές περιπτώσεις και εξαπλώθηκε, μέχρι που έγινε κατηγορία από μόνο του. Ο σχεδιασμός τέτοιου είδους παιχνιδιών μπορεί να αναλυθεί σε πέντε κύρια χαρακτηριστικά, την επίλυση γρίφων, την εύρεση και χρήση διαφόρων αντικειμένων, την ιστορία, το σκηνικό και θεματολογία, τους διαλόγους και τα διαλογικά δέντρα που δημιουργούνται, τους στόχους και την επιτυχία ή την αποτυχία.

2.1 Επίλυση Γρίφων

Η διαδικασία επίλυσης γρίφων είναι από τα σημαντικότερα στοιχεία ενός παιχνιδιού περιπέτειας. Αποτελούν εμπόδια για τον παίκτη στην εξέλιξη του σεναρίου και είναι σχεδιασμένα έτσι ώστε να δοκιμάζουν της επαγωγικές ικανότητές του. Οι γρίφοι έχουν να κάνουν με την αποκωδικοποίηση μηνυμάτων, την εύρεση ή και τον συνδυασμό αντικειμένων, το ξεκλείδωμα κλειδαριών ή την διενέργεια συγκεκριμένων λειτουργιών. Οι γρίφοι μπορεί να απαιτούν ικανότητα συμπερασματικής σκέψεις από τον παίκτη, γνώσεις κάποιων επιστημών, ή την γνώση μουσικών ή καλλιτεχνικών αντικειμένων. Όλα αυτά μπορούν να μεταφερθούν στο περιβάλλον του παιχνιδιού με την επίλυση κάποιου puzzle, την χρήση μουσικού οργάνου ή την επίλυση ενός μαθηματικού προβλήματος. Ενώ ο βασικός σκοπός ενός γρίφου είναι να κάνει το παιχνίδι ενδιαφέρον και εκλυστικό προς τον χρήστη υπάρχουν περιπτώσεις όπου η υπερβολική δυσκολία μπορεί να θεωρηθεί αποτρεπτική και να τον κάνει να χάσει το

ενδιαφέρον του. Υπάρχουν αρκετές κριτικές από την κοινότητα των παικτών διαφόρων παιχνιδιών για γρίφους που έχουν θεωρηθεί απαράδεκτα δύσκολοι ή για γρίφους που δεν απαιτήσουν καμία ικανότητα από πλευράς χρήστη αλλά να μαντέψει μια λύση χωρίς καθόλου στοιχεία. Τρία πολύ συχνά παραδείγματα γρίφων ακολουθούν στις επόμενες φωτογραφίες.

Η εικόνα 1 δείχνει ένα στιγμιότυπο από το παιχνίδι δράσης Vampire Legends: The True Story of Kisilova. Σε αυτό το στιγμιότυπο βλέπουμε έναν από τους χαρακτηριστικότερους γρίφους που καλείται να επιλύσει ένα παίκτης παιχνιδιών περιπέτειας, τον συνδυασμό αντικειμένων. Στην προκειμένη, ο συνδυασμός γινόταν σε διάφορα κομμάτια χαρτί ούτως ώστε να προκύψει ένα ολοκληρωμένο γράμμα. Ο συνδυασμός αντικειμένων είναι μια πολύ συνηθισμένη τεχνική καθώς απαιτεί από τον χρήστη όχι μόνο να εντοπίσει τα αντικείμενα που χρειάζεται αλλά να σκεφτεί και ότι

Εικόνα 1: Vampire Legends: The True Story of Kisilova

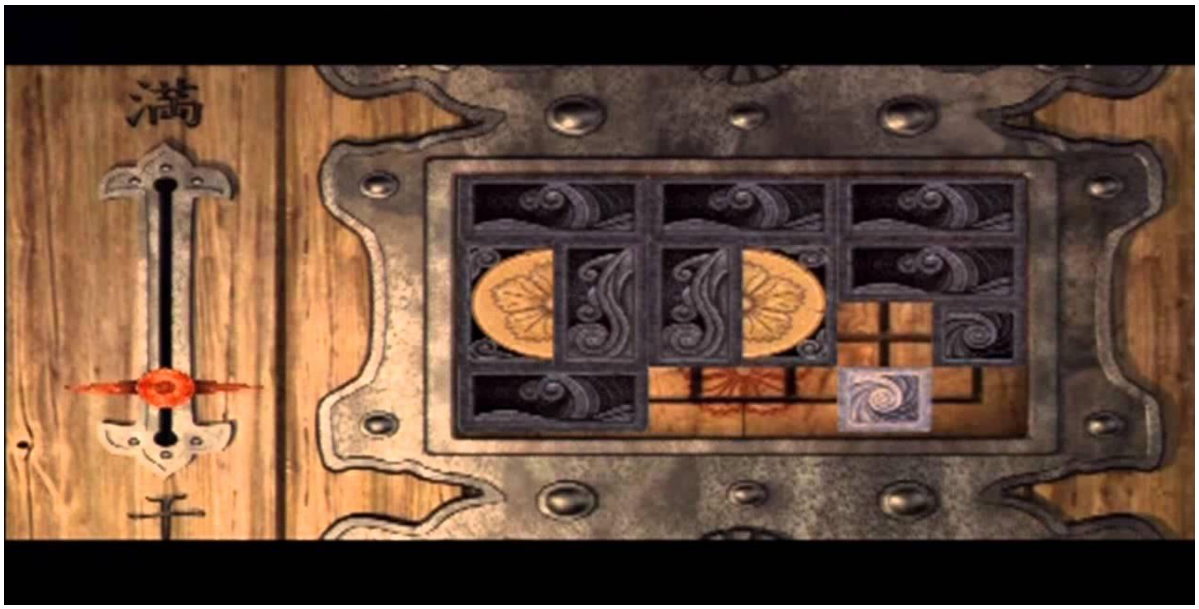
αυτά μπορεί να αλληλεπιδρούν αναμεταξύ τους.

Στην εικόνα 2 φαίνεται ένα screenshot από το indie παιχνίδι Darkfall, το πρώτο της σειράς και δείχνει ένα από τους συχνότερα χρησιμοποιήσιμους γρίφους, την αποκωδικοποίηση μηνυμάτων. Συγκεκριμένα ανάμεσα από τους δύο καθρέφτες βλέπουμε ένα σύμβολο το οποίο ο παίκτης πρέπει να θυμάται και να ερμηνεύσει την κατάλληλη στιγμή στην πορεία του παιχνιδιού. Η δυσκολία με τους συγκεκριμένους τύπους γρίφων έγκειται στο ότι δεν γνωρίζει ο παίκτης πως να τους ερμηνεύσει μέχρι να βρει κάποιο επιπλέον στοιχείο, ενώ πολλές φορές μπορεί να μην κρύβουν και καμία σημασία και να βρίσκονται στο παιχνίδι σαν μηνύματα με σκοπό να τον μπερδέψουν περισσότερο. Ένα πολύ γνωστό παράδειγμα τέτοιου γρίφου που δεν προσέδιδε κάτι στην εξέλιξη του σεναρίου ούτε κάποιο στοιχείο στον παίκτη ήτανε στην σειρά παιχνιδιών Pokemon για την κονσόλα GameBoy, όπου ο παίκτης έβρισκε μια σειρά από γραμματόμορφα πλάσματα τα οποία μπορούσαν να μεταφραστούν σε λέξεις τις αγγλικής γλώσσας και να σχηματίσουν προτάσεις, χωρίς όμως αυτό να έχει κάποιο νόημα.

Εικόνα 2: Darkfall: The Journey

Η εικόνα 3 παρουσιάζει έναν τρίτο τύπο γρίφων που χρησιμοποιήθηκαν κατά κόρον σε παιχνίδια κονσολών όπου η έλλειψη πληκτρολογίου έκανε δύσκολη την χρήση γρίφων που απαιτούσαν εισαγωγή κειμένου ή αριθμών. Το συγκεκριμένο στιγμιότυπο είναι παρμένο από το παιχνίδι Onimusha: Warlords, το πρώτο της σειράς Onimusha. Στον συγκεκριμένο γρίφο έπρεπε να κουνήσεις τα κομμάτια συμπληρώνοντας

ουσιαστικά έναν τύπο puzzle (η ακριβής ορολογία είναι sliding puzzle δηλαδή κυλιόμενη σπαζοκεφαλιά) στον διαθέσιμο χρόνο που οριζόταν από την κάθετη μπάρα στα αριστερά της οθόνης. Αυτή η μέθοδος χρησιμοποιήθηκε τόσο πολύ την περασμένη δεκαετία λόγο της ευκολίας υλοποίησης και των θετικών κριτικών που



Εικόνα 3: Onimusha: Warlords

πλέον σπάνια την συναντάει κανείς ακόμα και εάν παίζει indie τίτλους.

2.2 Συλλογή και Χρήση Αντικειμένων

Άλλο ένα στοιχείο που χαρακτηρίζει τα παιχνίδια περιπέτειας είναι η συλλογή και η χρήση διαφόρων αντικειμένων. Συνήθως πριν την επίλυση κάποιου γρίφου ή μετά την ολοκλήρωσή του αποκτάς κάποιο αντικείμενο. Αυτό το αντικείμενο μπορεί να βρίσκεται κάπου στην σκηνή και οι προκλήσεις που μπορεί να δίνονται στον παίκτη είναι είτε να το βρει, είτε να πρέπει να συνδυάσει δύο αντικείμενα για την συνέχεια της πλοκής. Αντίθετα με τους γρίφους υπάρχει κατά βάση απεριόριστος χρόνος για την εύρεση κάποιου αντικειμένου οπότε βασίζεται στην λογική του παίκτη παρά στην γρήγορη σκέψη. Ένα πρόβλημα που είχε αυτή η τεχνική στα παιχνίδια τύπου “point and click” ήταν ότι οι παίκτες εκτελούσαν μια διεργασία γνωστή ως pixel hunting, όπου έκαναν κλικ με το ποντίκι σε όλα τα σημεία κάθε οθόνης. Για να αποτραπεί η χρήση του pixel hunting ένα αντικείμενο που είναι βασικό για τον χρήστη θα γίνεται διακριτό όταν ο παίκτης το πλησιάσει με το ποντίκι είτε εκπέμποντας κάποια λάμψη είτε αλλάζοντας το σχήμα του κέρσορα είτε εμφανίζοντας κάποιο μήνυμα

Η εικόνα 4 δείχνει ένα παράδειγμα διάκρισης αντικειμένων σε μια σκηνή ενός εκ των γνωστότερων παιχνιδιών δράσης, του Post Mortem. Στο συγκεκριμένο παιχνίδι ο χρήστης όταν περνάει το κέρσορα πάνω από ένα αντικείμενο το οποίο μπορεί να το συλλέξει, αλλάζει την μορφή του κέρσορα σε ένα χέρι, κάτι που ειδοποιεί τον χρήστη

Εικόνα 4: Post Mortem

ότι το συγκεκριμένο αντικείμενο μπορεί να συλλεχθεί.

2.3 Ιστορία Θεματολογία Σκηνογραφία

Ένα τρίτο χαρακτηριστικό των παιχνιδιών περιπέτειας είναι το σενάριο τους το οποίο είναι παίζει πολύ σημαντικότερο ρόλο από οποιοδήποτε άλλο είδος παιχνιδιού. Η αφήγηση ή εξέλιξη της πλοκής είναι από τα πιο σημαντικά σημεία διότι δίνουν στον παίκτη το επόμενο στοιχείο που χρειάζεται κάθε φορά έτσι η σκηνογραφία και η ανάπτυξη της ιστορίας κατά την διάρκεια του παιχνιδιού είναι οι κύριες κινητήριες δυνάμεις προς το επόμενο επίπεδο. Μέσα από την αφήγηση ο παίκτης μαθαίνει τις δυνατότητές του, τους στόχους του και τον λόγω για τον οποίο συμβαίνει ότι συμβαίνει. Μάλιστα, ένα παιχνίδι περιπέτειας θεωρείται ότι έχει αποτύχει εάν το σενάριο του δεν είναι πιστικό και η πλοκή του δεν κρατάει το ενδιαφέρον, ενώ στα υπόλοιπα παιχνίδια το σενάριο και η πλοκή είναι ένα ακόμα χαρακτηριστικό στην τελική βαθμολογία.

2.4 Διάλογος και Διαλογικά Δέντρα

Άλλο ένα βασικό συστατικό πολλών παιχνιδιών περιπέτειας είναι οι διάλογοι. Ο παίκτης μπορεί, πολλές φορές, να διαδράσει με κάποιον NPC (Non Playable Character) πραγματοποιώντας έναν διάλογο. Η πεπατημένη είναι να εμφανίζεται στον παίκτη ένα διάλογος συνοδευόμενος από κάποιες επιλογές που αντιπροσωπεύουν την απάντηση του παίκτη. Αυτές οι επιλογές μπορεί να περιλαμβάνουν στοιχεία για

Εικόνα 5: Fallout 3

κάποιον γρίφο ή κάποιο αντικείμενο ή απλά να είναι κομμάτια της σεναριακής πλοκής. Σε πολλές περιπτώσεις η επιλογή μιας απάντησης απενεργοποιεί της άλλες επιλογές και οδηγεί τον παίκτη σε ένα συγκεκριμένο διαλογικό δέντρο. Ανάλογα με το διαλογικό δέντρο που θα ακολουθήσει κάποιος παίκτης είναι δυνατόν να μεταβάλει το σενάριο ή ακόμα και το τελείωμα ενός παιχνιδιού.

Χαρακτηριστικό παράδειγμα της μεγάλης επίδρασης που μπορεί να έχουν τα διαλογικά δέντρα είναι τα παιχνίδια Fallout 3 και Fallout: New Vegas της Bethesda. Αριστερά φαίνεται ένα παράδειγμα διαλόγου με NPC όπου ο παίκτης ανάλογα με την επιλογή που θα δώσει αλλάζει τόσο την άμεση τροπή που θα πάρει το παιχνίδι όσο και το γενικότερο σενάριο και στις περισσότερες περιπτώσεις, το τελείωμα που θα έχει το παιχνίδι.

2.5 Στόχοι και Επιτυχία-Αποτυχία

Τελευταίο στοιχείο ενός παιχνιδιού περιπέτειας είναι οι στόχοι του και το αποτέλεσμα,

είτε αυτό είναι μια επιτυχία είτε μια αποτυχία. Ένα παιχνίδι περιπέτειας τοποθετεί τον παίκτη σε μια αναζήτηση, είτε αυτή είναι για την λύση ενός μυστηρίου είτε για την αναζήτηση ενός αντικειμένου είτε η εύρεση κάποιου χαρακτήρα. Κάθε σενάριο αποτελείται από σαφείς στόχους για τον παίκτη. Η πραγματοποίηση ή όχι αυτών τον στόχων καθορίζει και την επιτυχία ή την αποτυχία του παιχνιδιού. Σε κάποια παιχνίδια ο παίκτης μαθαίνει στο τέλος εάν έχει αποτύχει ή επιτύχει, ανάλογα με τις ενέργειές του ενώ σε άλλα η ολοκλήρωση της αποστολής δεν γίνεται μέχρι την επιτυχία του παίκτη. Στην πρώτη περίπτωση υπάρχουν δύο λογικές υλοποιήσεις για την αποτυχία, είτε το παιχνίδι τερματίζει με ένα τελείωμα αποτυχίας είτε ο παίκτης μπαίνει σε μια κατάσταση που είναι αδύνατον να κάνει κάτι άλλο. Αυτή η κατάσταση μπορεί να είναι να “πεθάνει” ο χαρακτήρας οπότε να σε παραπέμπει να ξεκινήσεις το παιχνίδι απ την αρχή ή να βρεθεί σε σημείο όπου δεν θα μπορεί να γίνει περαιτέρω αλληλεπίδραση με το παιχνίδι. Η τελευταία λύση έχει λάβει αρκετή αποδοκιμασία από τους χρήστες, κυρίως γιατί δεν μπορείς να μάθεις, στις περισσότερες περιπτώσεις, πια επιλογή κατά την διάρκεια του παιχνιδιού σε οδήγησε σε αυτό το αδιέξοδο.

Καθώς οι παραπάνω λεπτομέρειες των παιχνιδιών περιπέτειας που αναφέρθηκαν δεν αποτελούν κανόνα και απαραίτητο στοιχείο, είναι τόσο ευρέως διαδεδομένες και απαντώνται στο σύνολο τον παιχνιδιών αυτού του είδους, κάτι που τις καθιστά οδηγούς σχεδίασης. Μπορεί, ενδεχομένως να σχεδιαστεί κάποιο παιχνίδι που υπολείπεται μερικών, αλλά η παράδοση προδικάζει ότι αυτό θα υστερεί έναντι των υπολοίπων.

3 Προγράμματα που Χρησιμοποιήθηκαν

3.1 Audacity

Το audacity είναι ένα βραβευμένο, ανοιχτού κώδικα, πρόγραμμα επεξεργασίας ήχου και ηχογράφησης. Το επέλεξα διότι θεωρώ ότι είναι από τις καλύτερες open source επιλογές για επεξεργασία ήχου. Συγκεκριμένα τα χαρακτηριστικά που με οδήγησαν στην επιλογή του είναι:

- Η δωρεάν διάθεση του για κατέβασμα και χρήση όντας open source.
- Η ευκολία χρήσης και εξοικείωσης με το περιβάλλον και τα εργαλεία του.

- Η αφθονία διαθέσιμων μορφών εξαγωγής αρχείων, κάποιες από τις οποίες είναι οι WAV, AIFF, AU, FLAC, Ogg Vorbis, AC3, MP3, MP2, ACC, WMA, M3A (AAC).

Η χρήση του έγινε κυρίως για επεξεργασία ήχου. Αναλυτικά, εκτεταμένα χρησιμοποίησα τα εργαλεία της παραγωγής ησυχίας, την αλλαγή του tempo, το κόψιμο και την επέκταση ενός κομματιού και την εξαγωγή σε μορφές WAV και AIFF ανάλογα το μέγεθος του κομματιού και την χρήση του.

3.2 GIMP

Το GIMP είναι ανοιχτού κώδικα λογισμικό για την επεξεργασία γραφικών τύπου raster. Το όνομά του προέρχεται από τα αρχικά των GNU Image Manipulation Program. Μπορεί να επεκταθεί με μια πληθώρα επεκτάσεων (plug-in) για την προσθήκη εξαιρετικά μεγάλου αριθμού πρόσθετων λειτουργιών. Στην διαδικτυακή κοινότητα θεωρείται ως μια από τις καλύτερες δωρεάν εναλλακτικές αντί του Photoshop. Η επιλογή του έγινε γιατί είναι ανοιχτού κώδικα, άρα δωρεάν και για τον μεγάλο αριθμό εργαλείων του. Ένα από τα σπουδαία χαρακτηριστικά του πέραν της επεξεργασίας και καλλιέργειας εικόνων και του φωτομοντάζ είναι η δημιουργία κινούμενων εικόνων σε μορφή gif ή mpeg.

Κυρίως χρησιμοποιήθηκε για την επεξεργασία σε πολλαπλά επίπεδα (layers) και για την συνένωση (merging) εικόνων. Η δημιουργία animation δεν χρησιμοποιήθηκε καθώς θα ήταν εκτός πλαισίου της συγκεκριμένης πτυχιακής εργασίας.

3.3 SketchUp (Google SketchUp)

Το SketchUp, γνωστό παλαιότερα και ως Google SketchUp είναι ένα πρόγραμμα τρισδιάστατης μοντελοποίησης που χρησιμοποιείται κυρίως από πολιτικούς μηχανικούς, μηχανολόγους μηχανικούς, αρχιτέκτονες και σχεδιαστές βιντεοπαιχνιδιών. Είναι εύκολο στην χρήση του και η εκμάθησή του για βασική λειτουργία γίνεται πολύ γρήγορα. Παρόλο που οι περισσότεροι στον χώρο της ανάπτυξης τρισδιάστατων μοντέλων για παιχνίδια χρησιμοποιούν κυρίως το Blender ή το 3D Studio Max για την πτυχιακή αυτή επιλέχθηκε το SketchUp για τους εξής λόγους:

- Διανέμεται δωρεάν για μια βασική λειτουργικότητα, παρόλο που μπορεί να

αγοραστεί η πλήρης έκδοση.

- Είναι πολύ πιο εύκολο από τους ανταγωνιστές του στην χρήση και την εκμάθηση
- Είχα προηγούμενη εμπειρία λόγο πρακτικών ασκήσεων σε μάθημα της σχολής.
- Περιέχει το 3D Warehouse ένα εκτενέστατο repository με δωρεάν προς χρήση και μετατροπή μοντέλα

Χρησιμοποιήθηκε για την μοντελοποίηση όλων των χώρων της εφαρμογής. Επίσης χρησιμοποιήθηκε ως επί το πλείστον για την εύρεση μοντέλων στο 3D Warehouse και την εξαγωγή τους.

3.4 Autodesk FBX Convert

Το FBX Converter της Autodesk είναι δωρεάν λογισμικό για την μετατροπή αρχείων τύπου OBJ, DXF, DAE, και 3DS σε αρχεία τύπου fbx. Όλοι αυτοί οι τύποι αρχείων αναφέρονται σε αρχεία τρισδιάστατων μοντέλων.

Χρησιμοποιήθηκε για την μετατροπή των αρχείων τύπου DAE που εξάγει παραδοσιακά το SketchUp σε fbx στις περιπτώσεις όπου κάποια μοντέλα ήταν πολύ μεγάλα καθώς τα fbx έχουν μικρότερο μέγεθος. Ο λόγος ήταν η καλύτερη και γρηγορότερη συμπίεση και εισαγωγή στην Unity, την μηχανή στην οποία εισαγόταν.

3.5 MS Paint

Το Paint της Microsoft είναι το βασικό εργαλείο ζωγραφικής που έρχεται εγκατεστημένο μαζί με οποιοδήποτε λειτουργικό σύστημα Windows. Δίνει την δυνατότητα στον χρήση εύκολα και γρήγορα να σχεδιάσει απλές εικόνες ή να επεξεργαστεί σε πολύ βασικό επίπεδο αρχεία εικόνας. Η μοναδική χρήση που είχε ήταν για γρήγορο resize σε όσα textures χρειαζόταν προσαρμογή.

3.6 LunaPic

LunaPic είναι μια διαδικτυακή εφαρμογή επεξεργασίας εικόνων με δυνατότητες παρόμοιες του GIMP. Έχει επιπλέον λειτουργικότητες όπως την δημιουργία κάποιων βασικών animation ή την εφαρμογή διαφόρων εφέ. Η χρήση της είναι δωρεάν.

Χρησιμοποιήθηκε συγκεκριμένα για την μετατροπή του background διαφόρων

εικόνων και texture σε διαφανές. Η ίδια λειτουργία θα μπορούσε να γίνει μέσω του GIMP αλλά η LunaPic είναι πολύ γρήγορη και πιο ευκολοδούλευτη για επεξεργασία σε ένα επίπεδο.

3.7 Fontspace και Pookatoo

Το Fontspace και το Pookatoo είναι δύο επίσης διαδικτυακές εφαρμογές για την δημιουργία προσαρμοσμένων γραμματοσειρών δωρεάν. Τόσο το Fontspace σε 2D όσο και το Pookatoo σε 3D σου δίνουν την δυνατότητα επιλογής από μια πληθώρα γραμματοσειρών και εισαγωγής κειμένου της επιλογής σου. Το τελικό αποτέλεσμα παράγεται σε εικόνα που μπορεί να κατέβει και να αποθηκευτεί τοπικά.
Χρησιμοποιήθηκαν για την δημιουργία του τίτλου του παιχνιδιού στην αρχική οθόνη και του μηνύματος στον τοίχο στο τελικό δωμάτιο.

3.8 Unity3D

Η Unity είναι μια cross-platform μηχανή ανάπτυξης παιχνιδιών με το δικό της ενσωματωμένο προγραμματιστικό περιβάλλον (IDE). Θεωρείται μια από τις πιο πετυχημένες μηχανές ανάπτυξης παιχνιδιών λόγο της δωρεάν έκδοσής της και της ευκολίας χρήσης της. Χρησιμοποιείται από indie developers μέχρι και σε μεγάλα studio. Μερικοί από τους γνωστότερους τίτλους της είναι: “Warhammer 40,000: Space Wolf, OddworldL New 'n' Tasty, Rust, Might & Magic X Legacy, Deus Ex: The Fall” και πολλά άλλα. Ένα από τα σημαντικά της χαρακτηριστικά της μηχανής είναι η δυνατότητα παράδοσης του αποτελέσματος σε πολλές διαφορετικές πλατφόρμες με μικρή έως ελάχιστη παραμετροποίηση. Κάποιες από τις πλατφόρμες είναι οι ακόλουθες: Windows, Linux, Adobe Flash, PlayStation 2/3/4/Vita/ Xbox 360/One Wii και άλλες. Ενώ η Unity είναι μια αρκετά πλήρης μηχανή στην δωρεάν της έκδοση, μπορεί να αγοραστεί και η έκδοση Unity Pro η οποία έχει αρκετές επιπρόσθετες λειτουργίες κυρίως σε θέμα γραφικών, deployment και animation. Η τιμή της Unity Pro ανέρχεται στα 1500$ ή μπορεί να ενοικιασθεί για 75$ τον μήνα. Για το add-on του Android Pro ή του iOS Pro χρειάζονται άλλα 1500$ καθώς και το team license κοστίζει επιπλέον 500$. Το κόστος έπειτα μεταβάλλεται ανάλογα των αγορών απ το Asset Store. Επιλέχθηκε έναντι των UDK, CryENGINE και άλλες λιγότερο γνωστές για τους παρακάτω λόγους:

- Υπήρχε εξοικείωση σε μεγάλο βαθμό λόγω προηγούμενων project, παρακολούθηση tutorial και σεμιναρίων και ανάλογης θεματολογίας εργασία και παρουσίαση στο μάθημα των γραφικών.
- Η δωρεάν της έκδοση δίνει πολλές παραπάνω δυνατότητες από τις υπόλοιπες μηχανές.
- Προσωπικά θεωρώ τα αποτελέσματά σε indie projects εντυπωσιακότερα σε αντιπαράθεση με τις υπόλοιπες διαθέσιμες επιλογές.
- Asset Store. Ένα υπερμέγεθες repository με project και assets διαθέσιμα είτε δωρεάν είτε έναντι πολύ λογικής τιμής. Παρόλο που δεν αγοράστηκε κάτι σε αυτό το project καθώς ο στόχος ήταν η ανέξοδη ανάπτυξη λογισμικού, με βρίσκω σύμφωνο με αυτή τη λογική, τη τεχνοτροπία της επαναχρησιμοποιησιμότητας και την άμεση προώθηση τον προϊόντων του γραφίστα στον προγραμματιστή και vise versa.
- Community. Η διαδικτυακή κοινότητα της Unity τόσο σε θέματα forum όσο και σε Wiki είναι πληρέστατη, ίσως κατά πολύ μεγαλύτερη από ότι οι ανάλογες κοινότητες των υπολοίπων μηχανών ανάπτυξης παιχνιδιών. Πολύ εύκολα βρίσκει κανείς documentation, έτοιμο κώδικα σε μορφή script και συμβουλές σε θέματα τεχνικής, αντιμετώπισης προβλημάτων και αντιμετώπισης νέων ιδεών. Ένα ακόμα θετικό στοιχείο της συγκεκριμένης κοινότητας είναι ότι έχει πολλούς ενεργούς χρήστες και συνεπώς δίνει ταχύτατες απαντήσεις σε ερωτήματα στα forum και ταυτόχρονα διατηρεί φιλικό χαρακτήρα και προσέγγιση απέναντι σε αρχάριους χρήστες.
- Μεγάλο πλήθος επιτρεπτών τύπων αρχείων προς εισαγωγή. Η δυνατότητα της Unity να επεξεργάζεται τόσο μεγάλο αριθμό διαφορετικών τύπων αρχείων μου έδωσε την δυνατότητα να χρησιμοποιήσω κατά βούληση όλα τα προαναφερθέντα λογισμικά χωρίς να ανησυχώ για θέματα συμβατότητας.
- Προγραμματισμός με Javascript. Η Unity σου δίνει την δυνατότητα να γράψεις κώδικα τύπου scripting σε 3 γλώσσες. C#, Javascript και Boo. Στην παρούσα πτυχιακή χρησιμοποιήθηκε C# και Javascript, με την

δεύτερη να είναι η βασική επιλογή καθώς με αυτήν είχα μεγαλύτερη εξοικείωση. C# χρησιμοποιήθηκε για την επεξεργασία έτοιμων script που ήταν ήδη γραμμένα σε αυτήν την γλώσσα.

- Στην έκδοση 4.6 προστέθηκαν εξαιρετικά χαρακτηριστικά ως προς την σχεδίαση του UI. Συγκεκριμένα έκαναν τόσο απλή την διαδικασία δημιουργίας, σχεδίασης και διαχείρισης UI τόσο από προγραμματιστικό όσο και από σχεδιαστικό κομμάτι που θα ήτανε χάσιμο χρόνου να χρησιμοποιηθεί κάτι άλλο για τον ίδιο σκοπό.
- Animator. Το βασικό σύστημα καταγραφής και επεξεργασίας animation της Unity ήταν υπέρ αρκετό για τα animation που χρησιμοποιήθηκαν στην παρούσα εργασία οπότε θα ήταν έξτρα χρόνος η εκμάθηση animation σε έναν τρίτο λογισμικό για την εισαγωγή του σε κάποια άλλη μηχανή.

Χρησιμοποιήθηκε ως βασικό εργαλείο ανάπτυξης του λογισμικού της συγκεκριμένης πτυχιακής εργασίας. Όλα τα αντικείμενα που χρησιμοποιήθηκαν δημιουργήθηκαν στο SketchUp είτε από εμένα είτε κατεβάστηκαν από το 3D Warehouse και επεξεργάστηκαν από λίγο έως ολοκληρωτικά. Τα textures που χρησιμοποιήθηκαν είναι όλα δωρεάν διαθέσιμα δείγματα από διάφορες ιστοσελίδες και βρέθηκαν μέσω της μηχανής αναζήτησης Google. Όλα τα particle System που χρησιμοποιήθηκαν είναι είτε δικής μου δημιουργίας είτε αναπροσαρμογή των βασικών particle assets που προσφέρει η Unity δωρεάν με την εγκατάστασή της. Η κάμερα και το το First Person Shooter System που χρησιμοποιήθηκαν περιλαμβάνονται επίσης στα βασικά assets που διανέμει η Unity με την εγκατάστασή της. Οι ήχοι και τα ηχητικά εφέ που χρησιμοποιήθηκαν βρίσκονται στην σελίδα freesound.org που προσφέρει δωρεάν ηχητικά samples και εφέ. 2D icons και GUI Textures είτε κατασκευάστηκαν από εμένα για τους σκοπούς της συγκεκριμένης εργασίας είτε βρέθηκαν δωρεάν διαθέσιμα στο διαδίκτυο.

3.9 Monodevelop

Monodevelop είναι ένα IDE που επικεντρώνεται σε Mono και .net frameworks. Είναι δωρεάν ως προϊόν ανοιχτού λογισμικού και είναι ο πιο γρήγορος τρόπος να ξεκινήσεις την συγγραφή script για την εφαρμογή σου στη Unity καθώς μπορείς να

εγκαταστήσεις μια προσαρμοσμένη έκδοση μαζί με την εγκατάσταση της μηχανής. Υποστηρίζει στοιχεία όπως η αυτόματη συμπλήρωση κώδικα, GUI και Web Designer. Το Monodevelop υποστηρίζει την συγγραφή στις γλώσσες Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Python, Vala, and Visual Basic.NET. Φυσικά θα μπορούσε να γίνει η χρήση οποιουδήποτε IDE για την συγγραφή script αλλά η Unity διασυνδέεται άμεσα με το Monodevelop όπως και με το Visual Studio για την συγγραφή σε C#. Η χρήση του Monodevelop έναντι του visual studio ή κάποιου άλλου IDE έγινε για τους εξής λόγους:

- Διανέμεται δωρεάν ως λογισμικό ανοιχτού κώδικα.
- Επιταχύνει τις διαδικασίες καθώς μπορεί να εγκατασταθεί στον ίδιο χρόνο με την Unity.
- Είναι πιο ελαφρύ λογισμικό από το visual studio, το Eclipse και το Netbeans που ήταν οι υπόλοιπες λογικές επιλογές
- Το περιβάλλον του μου ήταν πιο οικείο και σύνηθες για την συγγραφή Javascript.

Χρησιμοποιήθηκε κυρίως για την συγγραφή και ανάγνωση κώδικα σε Javascript (εκτός ελάχιστων περιπτώσεων που χρησιμοποιήθηκε C#) σε μορφή script προορισμένα για την μηχανή Unity.

Σε επόμενο κεφάλαιο θα αναλυθεί και θα αναπτυχθεί η μεθοδολογία ανάλυσης και ανάπτυξης, η διαδικασία σύλληψης του σεναρίου, η συλλογή του απαραίτητου υλικού και η αναζήτηση των διαφόρων προγραμμάτων που χρησιμοποιήθηκαν κατά την διάρκεια της εκπόνησης της συγκεκριμένης εργασίας.

3.10 Markup Highlighter

Η υπηρεσία Highlighter που κάνει διαθέσιμη η ιστοσελίδα markup.su είναι μια υπηρεσία διαμόρφωσης κειμένου ανάλογα με κάποια γλώσσα προγραμματισμού και κάποιο στυλ μορφοποίησης. Εντοπίζει αυτόματα την γλώσσα προγραμματισμού της οποίας εισάγεις κείμενο και βγάζει το αποτέλεσμα επεξεργασμένο σύμφωνα με κάποιο πρότυπο. Τα προτερήματα αυτού του εργαλείου είναι τα εξής:

- Εύκολα και γρήγορα μπορείς να μορφοποιήσεις τον κώδικα σου σύμφωνα με κάποιο πρότυπο, σαν να έγραφες στο ανάλογο editor ώστε να

χρησιμοποιήσεις το αποτέλεσμα σε κάποιον κειμενογράφο τύπου Microsoft Word.

- Προσφέρεται δωρεάν το api της εφαρμογής ούτως ώστε ο οποιοσδήποτε να μπορεί να υλοποιήσει την συγκεκριμένη υπηρεσία στην ιστοσελίδα του.

Χρησιμοποιήθηκε κυρίως για τον πρώτο λόγο, για την εύκολη μορφοποίηση κώδικα προκειμένου να καταγραφεί στο OpenOffice Writer.

Όλα τα προγράμματα που χρησιμοποιήθηκαν, έγινε προσπάθεια να είναι open source ή να διανέμονται προς δωρεάν χρήση ώστε να διατηρηθεί ένας ανέξοδος χαρακτήρας στην πτυχιακή εργασία και να υλοποιηθεί ένα παράδειγμα ανάπτυξης ηλεκτρονικού παιχνιδιού με μηδαμινό προϋπολογισμό.

4 Σενάριο

Το χρονικογεωγραφικό πλαίσιο του παιχνιδιού είναι τοποθετημένο σε πραγματικό μέρος και χρόνο. Το παιχνίδι λαμβάνει χώρα στα προάστια της Belize City, μιας πόλης στην κεντρική Αμερική που ιδρύθηκε από Βρετανούς αποικιοκράτες, ξυλοκόπους στα θεμέλια μιας μικρής πόλης των Maya, την Holzuz το 1968. Η ημερομηνία είναι 31 Οκτωβρίου 1961. Η συγκεκριμένη πόλη και ημερομηνία επιλέχθηκαν για αρκετούς λόγους. Η πόλη Belize την μέρα εκείνη χτυπήθηκε από τον τυφώνα Hattie που κατέστρεψε την μισή πόλη και σκότωσε περίπου 400 ανθρώπους, οι καιρικές συνθήκες ήταν απολύτως ταιριαστές με το περιβάλλον που ήθελα να δημιουργήσω. Το γεγονός ότι ο τυφώνας έλαβε χώρα στις 31 Οκτωβρίου, μέρα που γιορτάζεται το Halloween στην Αμερική, βοηθάει στο να γίνουν πειστικά τα υπερφυσικά στοιχεία που περιέχονται στο σενάριο. Η ημερομηνία είναι ιδανική για τον τύπο σπιτιού που ήθελα να δημιουργήσω και γιατί είναι ίδια με την ημερομηνία που παίχτηκε η σειρά από την οποία εμπνεύστηκε το συγκεκριμένο σενάριο. Το γεγονός ότι είναι μια πόλη που δημιουργήθηκε από αποικιοκράτες βοηθάει στο να είναι ρεαλιστικά επιπλέον στοιχεία του σεναρίου.

Η ιστορία έχει ως εξής. Είναι νύχτα, 31 Οκτωβρίου 1961. Ο πρωταγωνιστής, ένας δικηγόρος που μένει στην διπλανή πόλη αλλά εργάζεται στην Belize επιστρέφει σπίτι μετά από μια επιτυχημένη συμβολαιογραφική δουλειά. Καθώς βρίσκεται στον περιφερειακό οι κακές καιρικές συνθήκες τον αναγκάζουν να βγει εκτός δρόμου και να ζητήσει άσυλο σε ένα προαστιακό σπίτι. Φτάνοντας στο σπίτι βρίσκει την εξώπορτα του σπιτιού ανοιχτή και μπαίνει μέσα για να στεγνώσει και να περιμένει την μπόρα να κοπάσει.

Ανοίγοντας τα φώτα βρίσκει τον εαυτό του σε ένα χολ με μαρμάρινο πάτωμα και γύψινο τοίχο και ένα παράθυρο με τζάμι βιτρό στον δεξιό τοίχο. Το μεγάλο ρολόι τοίχου δείχνει δώδεκα παρά τέταρτο. Αφού δεν απαντάει κανείς στο κάλεσμα του

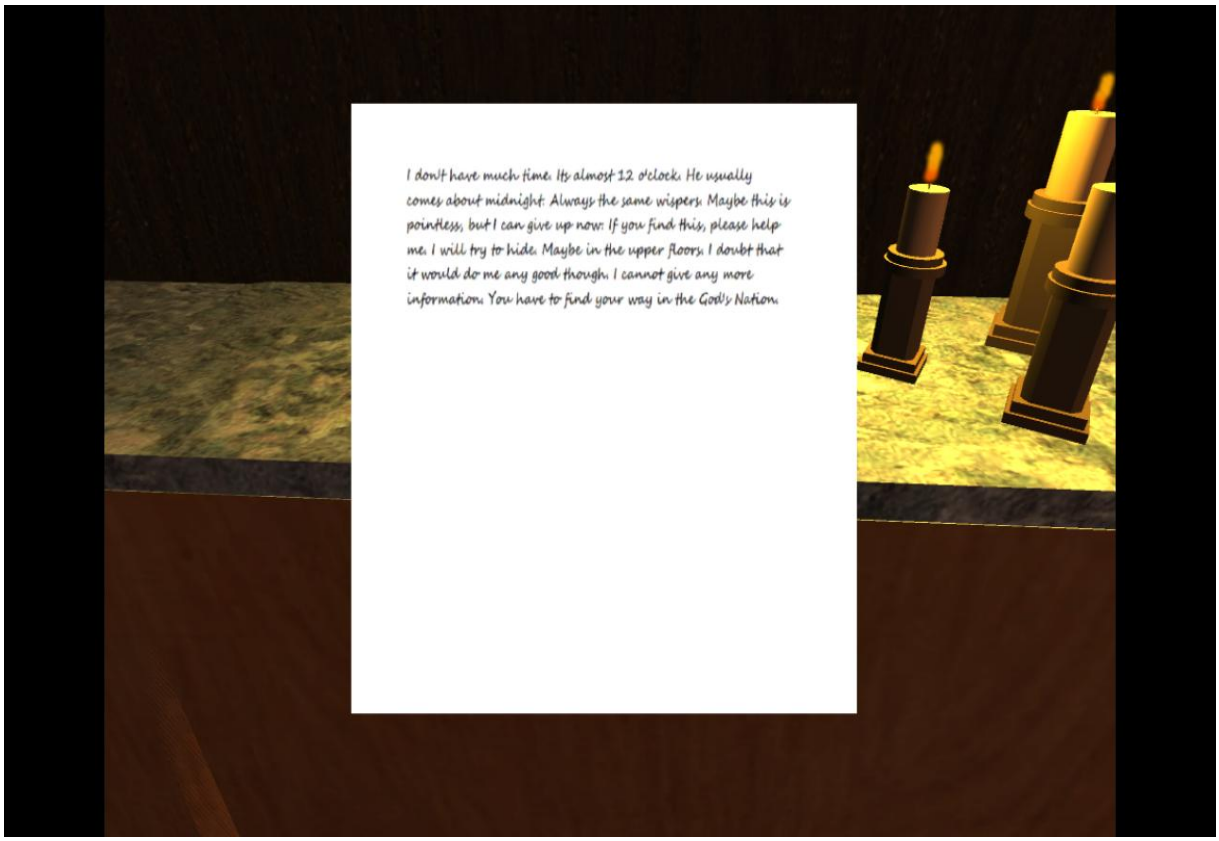
Εικόνα 6: Last Mansion - Hall

ανοίγει τις ξύλινες διπλές πόρτες για να περιμένει τον ένοικο του σπιτιού στο σαλόνι καθώς θα ήταν αγένεια να κάνει οτιδήποτε άλλο και το να φύγει ή να επιστρέψει στο αμάξι του δεν ήταν πλέον επιλογή με αυτόν τον καιρό.

Περνώντας την διπλή ξύλινη πόρτα οδηγείται στο σαλόνι του παλιού αρχοντικού όπου στο τζάκι καίει ακόμα φωτιά. Το πάτωμα είναι παρκέ, ενώ οι τοίχοι που περιβάλλουν το σαλόνι φτιαγμένοι από ξύλο. Ξεχωρίζει ο παλιός πολυέλαιος, το μεγάλο χαλί, το τζάκι στο βάθος καθώς και τα έπιπλα βικτοριανής εποχής που κοσμούν το δωμάτιο. Ο διακόπτης του φωτισμού βρίσκεται στα αριστερά της εισόδου και φωτίζει το σκοτεινό δωμάτιο αποκαλύπτοντας στον παίκτη λεπτομέρειες όπως δύο μεγάλα κάδρα και μια μεγάλη βιβλιοθήκη στα δεξιά του δωματίου.

Εικόνα 7: Last Mansion - Living room

Με λίγη αναζήτηση ο παίκτης βρίσκει ένα γράμμα, γραμμένο από τον ένοικο του σπιτιού.



Εικόνα 8: Last Mansion - First letter

Το γράμμα είναι μια έκλυση βοήθειας. Εξηγεί ότι ο ένοικος του σπιτιού χρειάζεται βοήθεια, κάποιος τον κυνηγάει και ζητάει από όποιον το βρει βοήθεια. Μη μπορώντας να δώσει περισσότερα στοιχεία λόγο του διώκτη του αναφέρει ότι περισσότερες πληροφορίες μπορεί ο παίκτης να βρει στο "Έθνος του Θεού".

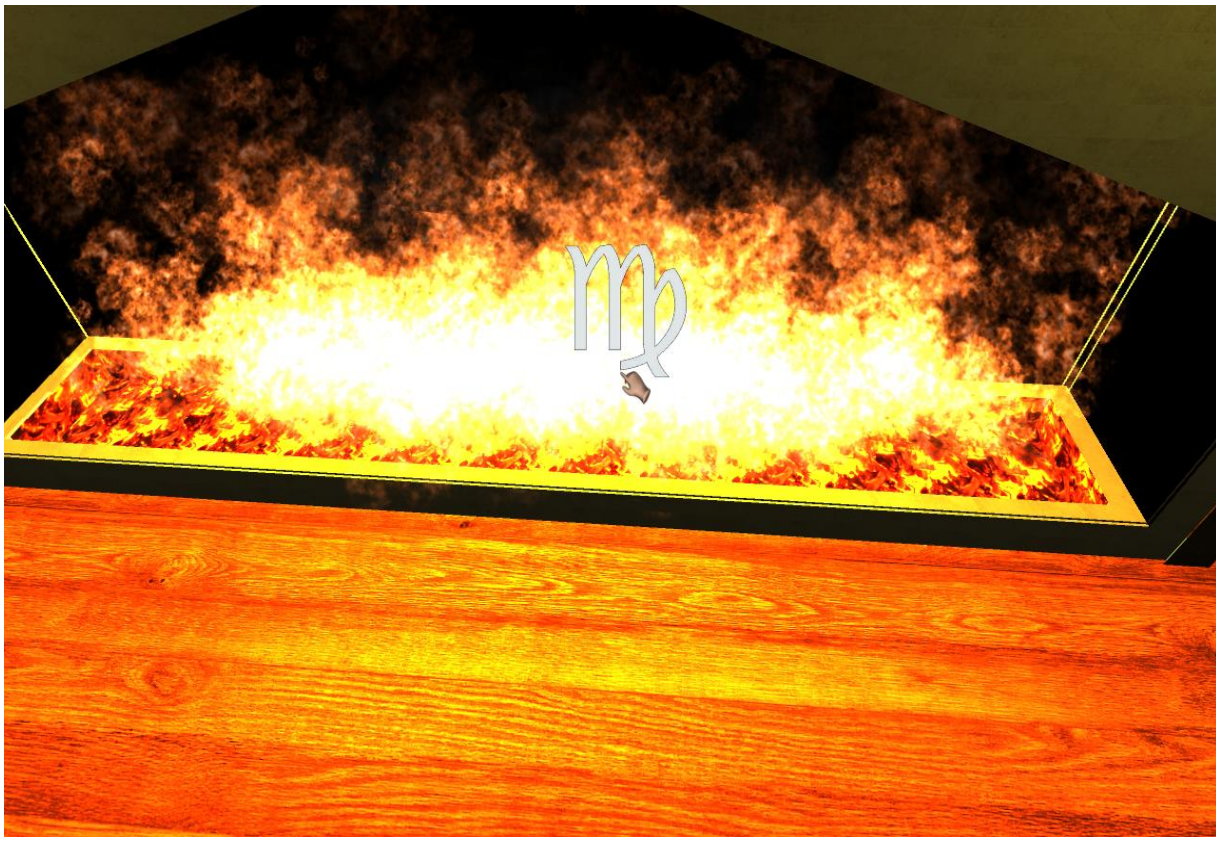
Μετά από προσεκτική παρατήρηση του δωματίου, ο παίκτης μπορεί να δει τον ακόλουθο πίνακα.

Εικόνα 9: Last Mansion - One Nation Under God

Ο πίνακας αυτός είναι έργο του Jon McNaughton και λέγεται One Nation Under God. Η παραπομπή “το έθνος του θεού” αναφέρεται σε αυτόν τον πίνακα. Εάν ο παίκτης εστιάσει μπορεί να εξετάσει τον πίνακα και να αποκτήσει έτσι ένα κλειδί. Αυτό το κλειδί ξεκλειδώνει την κλειδαριά του ντουλαπιού που βρίσκεται στο σαλόνι. Σε αυτό το ντουλάπι περιέχονται τρία νέα αντικείμενα, ένας φακός, ένα κλειδί για το υπόγειο και ένα γράμμα. Στο γράμμα περιγράφεται ο επόμενος γρίφος. Ο παίκτης πρέπει να αποκτήσει ένα εργαλείο απ το υπόγειο, διότι το επόμενο στοιχείο βρίσκεται κριμένο στο ίδιο αντικείμενο από το όπου ο μυθικός φοίνικας αναγεννιέται. Ο γρίφος αναφέρεται στις στάχτες του τζακιού, από τις οποίες μπορεί ο παίκτης να πάρει το επόμενο στοιχείο αφού πρώτα ανακτήσει από το υπόγειο το σκαλιστήρι που φαίνεται στην επόμενη εικόνα.

Εικόνα 10: Last Mansion - Basement

Στο υπόγειο του σπιτιού δεν υπάρχει καθόλου φωτισμός παρά μόνο ένα αμυδρό φως από έναν παλιό καυστήρα και το φως του φακού που ο παίκτης έχει αποκτήσει από το ντουλάπι του προηγούμενου ορόφου. Το ζητούμενο εργαλείο βρίσκεται σε μια θήκη με εργαλείο τζακιού. Καθ' όλη τον διάρκεια που ο παίκτης βρίσκεται στο υπόγειο ακούγονται ήχοι από αλυσίδες που κροταλίζουν. Ο χρήστης εάν αποκωδικοποιήσει τον γρίφο του γράμματος μπορεί πλέον να αποκτήσει το επόμενο στοιχείο απ το τζάκι, το οποίο είναι το εξής

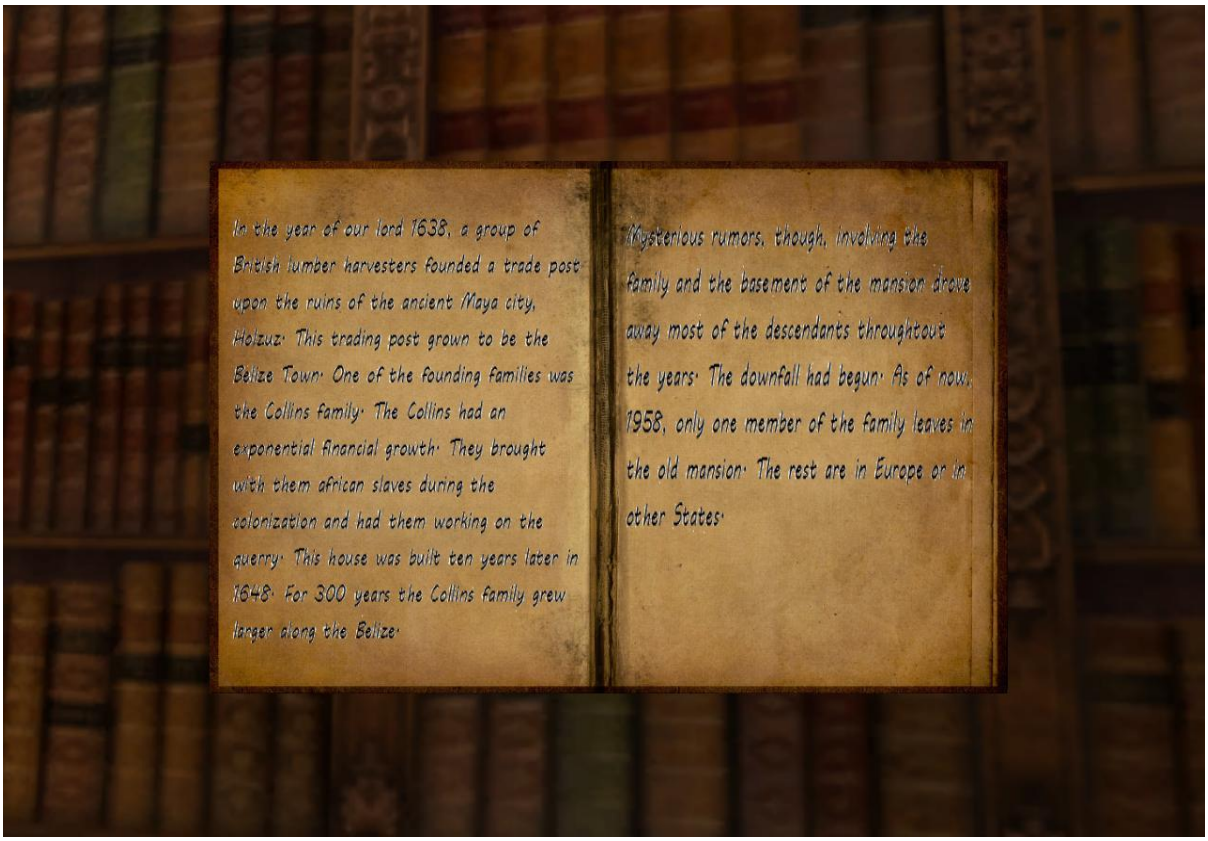


Εικόνα 11: Last Mansion - Fireplace item

Αυτό που φαίνεται στην παραπάνω εικόνα είναι ένα σύμβολο του ζωδιακού και συγκεκριμένα το σύμβολο της παρθένου. Ο επόμενος γρίφος αφορά αυτό το σύμβολο που ο παίκτης μόλις απέκτησε. Το ρολόι τοίχου που βρίσκεται στο χολ της εισόδου στο καντράν του αντί για τους αριθμούς της ώρας έχει τα σύμβολα του ζωδιακού, πλην ενός, της παρθένου. Τοποθετώντας ο παίκτης το σύμβολο στο ρολόι αποκτά το κλειδί που ξεκλειδώνει την πόρτα του δευτέρου ορόφου.

Εικόνα 12: Last Mansion - Second Floor Hall

Μπαίνοντας στον δεύτερο όροφο ο παίκτης ακούει μια γυναίκα να φωνάζει και να κλαίει, καταλαβαίνοντας έτσι ότι δεν έχει πολύ χρόνο. Η πόρτα από την οποία ακούγεται η κραυγή είναι κλειδωμένη εκ των έσω και δεν μπορεί να ξεκλειδωθεί. Ο παίκτης πρέπει πρώτα να ξεκλειδώσει την πόρτα που είναι κλειδωμένη με μια αναλογική κλειδαριά συνδυασμού. Τον κωδικό της κλειδαριάς ο παίκτης μπορεί να τον βρει σε ένα βιβλίο στην βιβλιοθήκη του πρώτου ορόφου.



Εικόνα 13: Last Mansion - Library Book

Το συγκεκριμένο βιβλίο καταγράφει κάποιες λεπτομέρειες για την πόλη και την δημιουργία αυτού του σπιτιού καθώς και την ιστορία της οικογένειας στην οποία ανήκει. Ο τετραψήφιος κωδικός που θα χρειαστεί ο παίκτης είναι η χρονιά που χτίστηκε το συγκεκριμένο σπίτι. Τοποθετώντας τον κωδικό στην κλειδαριά, το λουκέτο αφαιρείται και ο παίκτης μπορεί να προχωρήσει στο πρώτο υπνοδωμάτιο.

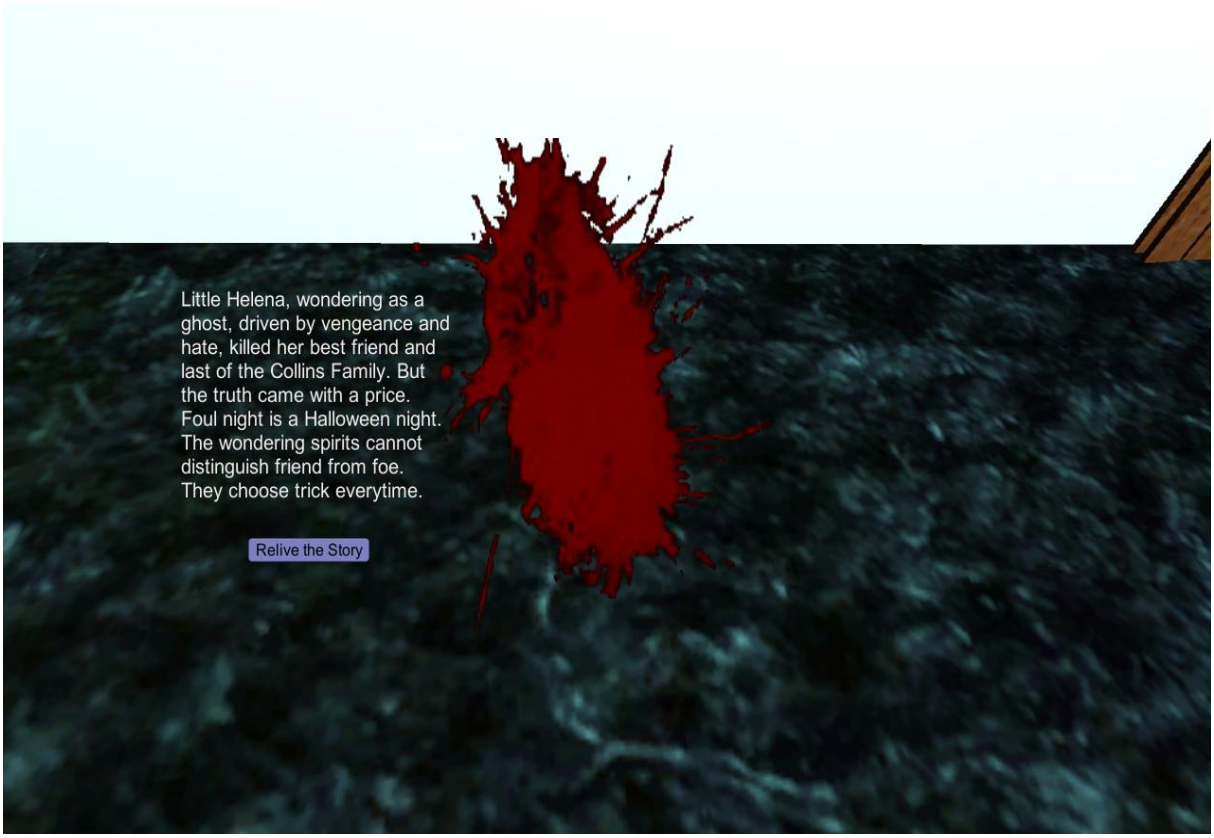
Εικόνα 14: Last Mansion - First Bedroom

Σε αυτό το υπνοδωμάτιο ο παίκτης μπορεί να αλληλεπιδράσει με το επιτραπέζιο Ouija, ένα είδος παιχνιδιού που χρησιμοποιείται ανά τον κόσμο για επικοινωνία με πνεύματα. Η λειτουργία του είναι η εξής, κάποιος κουνάει το τρίγωνο ξύλο στα γράμματα που θέλει ούτως ώστε να σχηματίσει μια πρόταση και έπειτα το πνεύμα αποκρίνεται σχηματίζοντας μια απάντηση με τον ίδιο τρόπο. Στο συγκεκριμένο παιχνίδι το πνεύμα σχηματίζει την λέξη “revenge” που σημαίνει εκδίκηση. Αφού ο παίκτης παρακολουθήσει το μήνυμα του Ouija μπορεί πλέον να πάει στο τελευταίο δωμάτιο, το οποίο έχει ξεκλειδωθεί.

Εικόνα 15: Last Mansion - Last Bedroom

Μπαίνοντας στο τελευταίο υπνοδωμάτιο ο παίκτης δεν μπορεί να δει ή να ανοίξει κάποιο φως διότι η λάμπα έχει σπάσει και βρίσκεται στο πάτωμα και ο μόνος φωτισμός είναι αυτός από τους κεραυνούς της καταιγίδας. Τα τρία διακριτά σημεία είναι τα εξής, το πτώμα του ενοίκου του σπιτιού, ο οποίος είναι μια γυναίκα, κόρη της οικογένειας και τελευταία απόγονος που μένει στο σπίτι, ένα μήνυμα γραμμένο με αίμα και ένα γράμμα γραμμένο από την αποθανούσα. Στο γράμμα καταγράφονται τα τελευταία λόγια της ενοίκου και ολοκληρώνεται στον παίκτη η ιστορία. Ο παίκτης μαθαίνει ότι η συγκεκριμένη οικογένεια συνήθιζε να βασανίζει τους υπηρέτες της στο υπόγειο προς τιμωρία για τα λάθη που έκαναν. Η κοπέλα που μόλις πέθανε είχε γίνει φίλη στο παρελθόν με την κόρη μιας υπηρέτριας, την οποία όμως όταν οι γονείς της πήραν για να την βασανίσουν η ίδια δεν έκανε τίποτα για να το αποτρέψει. Το κοριτσάκι πέθανε και έχει επιστρέψει τώρα ως πνεύμα για να πάρει την εκδίκησή του από την παλιά του φίλη που το πρόδωσε. Το γράμμα καταλήγει λέγοντας ότι πλέον η κυρία Collins είναι έτοιμη για τον θάνατο που την περιμένει, διότι έπραξε λάθος και πρέπει να πληρώσει ώστε το πνεύμα της φίλης της να ξεκουραστεί και να προχωρήσει. Καθώς ο παίκτης έχει μάθει την αλήθεια και πηγαίνει στην πόρτα για να

φύγει, πέφτει νεκρός από ένα χτύπημα στο κεφάλι και του εμφανίζεται η ακόλουθη εικόνα



Εικόνα 16: Last Mansion - Ending Scene

Εξηγείται στον παίκτη ότι το χτύπημα και ο θάνατός του προκλήθηκαν από τον πνεύμα της Έλενας, της μικρής που πέθανε από τα βασανιστήρια. Το πνεύμα καθοδηγούμενο από εκδίκηση και θυμό σκότωσε την παλιά του φίλη αλλά λόγω ημέρας αυτό δεν κατέστη αρκετό για να το κάνει να ηρεμήσει. Συγκεκριμένα η τελευταία πρόταση αναφέρει ότι “την ημέρα του Halloween τα πνεύματα δεν ξεχωρίζουν φίλους από εχθρούς, διαλέγουν το τρόμαγμα κάθε φορά”, σαν λογοπαίγνιο στην φράση που λέγεται την συγκεκριμένη μέρα “trick or treat”, που σημαίνει τρόμαγμα ή κέρασμα. Ο παίκτης μπορεί επιλέγοντας να “ξαναζήσει την ιστορία” να βρεθεί στο αρχικό μενού και εάν το επιθυμεί να ξαναρχίσει το παιχνίδι από την αρχή.

4.1 Χειρισμός

Ο χειρισμός του παίκτη είναι παρόμοιος με τα περισσότερα παιχνίδια fps (First Person Sooter). Ο χαρακτήρας μετακινείτε με τα πλήκτρα “α” για αριστερά, “δ” για δεξιά, “σ”

για πίσω και “ς” για μπροστά. Με το πλήκτρο “space” μπορεί να κάνει ένα μικρό άλμα. Με το πλήκτρο “ε” εκτελεί ενέργειες προς τα αντικείμενα της σκηνής και εστιάζει στα αντικείμενα που του δίνουν αυτή την δυνατότητα, όπως για παράδειγμα τα γράμματα. Με το πλήκτρο “escape” φεύγει από την κατάσταση εστίασης και τέλος με το πλήκτρο “tab” πηγαίνει το δρομέα στα πλαίσια ώστε να εισάγει τους απαραίτητους κωδικούς. Η κίνηση με το ποντίκι είναι ανάλογη και αντιστοιχίζει στην κίνηση της κάμερας και συνεπώς στην αίσθηση της κίνησης του κεφαλιού του παίκτη. Η επιλογή των αντικειμένων γίνεται βάση ενός στατικού δισδιάστατου γραφικού που απεικονίζει ένα χέρι, ή σε συγκεκριμένες περιπτώσεις ένα χέρι με έναν μεγεθυντικό φακό.

5 Υλοποίηση

Μετά την σύλληψη του σεναρίου το παιχνίδι υλοποιήθηκε σε τρία διακριτά βήματα, τον σχεδιασμό των μοντέλων, την εισαγωγή και την προσαρμογή των μοντέλων στην μηχανή Unity και τον προγραμματισμό της λειτουργικότητας των διαφόρων αντικειμένων μέσω scripting.

Για το πρώτο στάδιο, σχεδιάστηκαν αρχικά οι έξι βασικοί χώροι του παιχνιδιού οι οποίοι ήταν:

- Το χολ του πρώτου ορόφου
- Το σαλόνι του πρώτου ορόφου
- Το υπόγειο
- Το χολ του δευτέρου ορόφου
- Το πρώτο υπνοδωμάτιο του δευτέρου ορόφου
- Το δεύτερο υπνοδωμάτιο του δευτέρου ορόφου

Κάποιοι από τους χώρους εμπνεύστηκαν από την αμερικάνικη τηλεοπτική σαπουνόπερα, Dark Shadows. Οι χώροι δημιουργήθηκαν κατεξοχήν στο SketchUp από εμένα και εμπλουτίστηκαν με δωρεάν textures που βρέθηκαν στο google. Μετά την ολοκλήρωση των χώρων ξεκίνησε η διακόσμησή τους. Όλα τα μοντέλα που περιέχονται στους έξι προαναφερθέντες χώρους πάρθηκαν δωρεάν από το 3D Warehouse. Τα πιο πολλά μοντέλα υπέστησαν κάποια μορφοποίηση, είτε ήταν rescale είτε ήταν αλλαγή χρώματος ή αποκοπή μερικών πολυγώνων έτσι ώστε να ταιριάζουν καλύτερα με τον χώρο και με την ζητούμενη αισθητική. Η όλη διαδικασία

της δημιουργίας και της εύρεσης των τρισδιάστατων μοντέλων διήρκεσε περίπου ένα μήνα, το ένα τρίτο της διάρκειας της συγκεκριμένης εργασίας.

Στο δεύτερο στάδιο, τα μοντέλα που παράχθηκαν στο SketchUp, αφού υπέστησαν την ανάλογη επεξεργασία (συμπίεση ή μετατροπή) εισήχθησαν στην μηχανή Unity. Με την εισαγωγή έγινε το κατάλληλο rescale και η συμπίεση ώστε να δημιουργηθεί ομοιογένεια ανάμεσα στα διαφορετικά μοντέλα. Στην Unity δημιουργήθηκαν εφτά διακριτές σκηνές, δύο με το αρχικό μενού και την εισαγωγή και πέντε για κάθε βασικό χώρο. Το χολ του πρώτου ορόφου καθώς και το σαλόνι του πρώτου ορόφου έχουν συγχωνευθεί σε μία σκηνή. Οι μεγάλες προκλήσεις σε αυτό το σημείο ήτανε τρεις. Αρχικά έπρεπε όλα τα μοντέλα να βρεθούν στο κατάλληλο scale για να υπάρχει μια συνοχή μεταξύ τους αλλά και με τον παίκτη. Αυτό ήταν εύκολο στους χώρους που σχεδίασα ο ίδιος καθώς με την χρήση μέτρου και μεζούρας μπόρεσα να διατηρήσω τους χώρους σε παρόμοιες διαστάσεις μεταξύ τους. Τα μοντέλα που πάρθηκαν έτοιμα απ το 3D Warehouse και προστέθηκαν όμως, είχαν μεγάλες ανομοιογένειες. Αφού όλα τα αντικείμενα μεταφέρθηκαν στο κατάλληλο μέγεθος η δεύτερη πρόκληση ήταν να προετοιμαστεί το έδαφος για το scripting. Αυτό σημαίνει ότι κάποιες ομάδες διακριτών αντικειμένων έπρεπε να διαχωριστούν στα επιμέρους στοιχεία τους και να ονομασθούν κατάλληλα καθώς και κάποια διάχυτα αντικείμενα να ομαδοποιηθούν κάτω από ένα όνομα. Η διαδικασία αυτή επέτρεψε αργότερα την ευκολότερη εύρεση των αντικειμένων που θα έπρεπε ο παίκτης να αλληλεπιδράσει, διευκόλυνε τον προγραμματισμό τους και απλούστευσε τα τυχόν animation που δημιουργήθηκαν επί αυτών. Τέλος, αρκετά επίπονη αποδείχθηκε η διαδικασία της προσθήκης των colliders στα μοντέλα. Colliders ονομάζονται οι αόρατοι “τοίχοι” που εμποδίζουν τον παίκτη ή οποιοδήποτε αντικείμενο που φέρει colliders να παρέλθει αυτών. Η unity δίνει την επιλογή του generate colliders σε κάθε μοντέλο. Με αυτήν την επιλογή κάθε διακριτή επιφάνεια αποκτά έναν ξεχωριστό collider. Όσο απλή και βολική επιλογή ακούγεται αυτή, τόσα προβλήματα περιέχει. Ένα μοντέλο σε μια σκηνή μπορεί να περιέχει κατά μέσο όρο 2000 πολύγονα, άρα αρκετές εκατοντάδες επιφάνειες. Εάν σε μια σκηνή περιέχονται 20 με 30 αντικείμενα αυτόματα ο αριθμός επιφανειών αυξάνεται κατά πολύ. Αντί λοιπόν να παράξουμε colliders για κάθε μια επιφάνεια και να βάλουμε πολύ μεγάλο φόρτο στο παιχνίδι, παράγουμε χειροκίνητα colliders μόνο για τις επιφάνειες

που εμείς επιθυμούμε. Έτσι επιτυγχάνεται ένα πολύ λειτουργικό και ελαφρύ από πλευράς απαιτήσεων λογισμικό αλλά αυξάνεται ο χρόνος επεξεργασίας του.

Το τρίτο στάδιο ήταν ο προγραμματισμός των μοντέλων. Οτιδήποτε περιέχει μια λειτουργία και μια ενέργεια, φτιάχτηκε με το χέρι. Ο προγραμματισμός δεν σημαίνει κατ'ανάγκη και κώδικα. Οι φλόγες των κεριών και του τζακιού ή οι σπίθες τις σπασμένης λάμπας δημιουργήθηκαν με το λεγόμενο particle system. Το particle system στην αρχική του μορφή απέχει πάρα πολύ από το αποτέλεσμα το οποίο κάποιος επιθυμεί και ο τεράστιος αριθμός μεταβλητών που περιέχει χρειάζονται από αρκετά λεπτά έως και ώρες ρυθμίσεων. Άλλη μορφή προγραμματισμού είναι η δημιουργία animation. Παρόλο που δεν χρησιμοποιήθηκε third party software για την εξειδικευμένη δημιουργία animation αλλά ο animator της Unity η διαδικασία καθεαυτή συνεχίζει να είναι χρονοβόρα. Ανάλογα με το animation που θέλει να επιτύχει, μπορεί κανείς να ρυθμίσει μια πληθώρα παραμέτρων που αφορούν ένα αντικείμενο καθώς και μεταβλητές του περιβάλλοντός του. Κάθε αλλαγή παράγει μια καμπύλη στον άξονα του χρόνου που μπορεί επίσης να μεταβληθεί και να παραμετροποιηθεί αναλόγως. Σαν να μην φτάνουν όλα αυτά στα frames ενός animation μπορούν να εισαχθούν event που θα καλούν κάποιο κομμάτι κώδικα. Το τρίτο στάδιο, λοιπόν, που χρειάστηκε άλλον ένα μήνα από το χρονοδιάγραμμα της συγκεκριμένης εργασίας ήταν αυτών των ειδών οι προγραμματισμοί. Κυριότερο κομμάτι όμως του προγραμματισμού ήταν ο κώδικας όπου υλοποιήθηκε σε scripts χρησιμοποιώντας την γλώσσα javascript. Η βασική λειτουργία των script ήταν να προσαρτώνται σε κάθε αντικείμενο που μπορεί ο παίκτης να αλληλεπιδράσει και να του προσδίδουν λειτουργικότητα. Η λειτουργικότητα αυτή μπορεί να ήταν από το παίξιμο ενός animation ή ενός clip ήχου, μέχρι την δημιουργία κάποιου γραφικού ή την εισαγωγή κειμένου. Επιπλέον μέσα στα scripts υλοποιήθηκε σταδιακά ένα σύνολο μεταβλητών αληθείας που σε κάθε στιγμή το σύνολό τους ορίζει το ποσοστό ολοκλήρωσης του παιχνιδιού. Περισσότερες πληροφορίες για τον προγραμματισμό των script θα παρατεθούν στο ανάλογο κεφάλαιο.

Με αυτά τα τρία στάδια ολοκληρώθηκε η διαδικασία της υλοποίησης του παιχνιδιού. Το επόμενο κομμάτι, όπου φάνηκε αρκετά σημαντικό παρά το μικρό μέγεθος της εφαρμογής, ήταν το testing του παιχνιδιού.

6 Testing

Μετά την ολοκλήρωση του παιχνιδιού περάσαμε στην φάση του testing. Κατά κανόνα ένα παιχνίδι δοκιμάζεται σε alpha φάση από ανθρώπους που πληρώνονται για αυτήν την δουλειά και σε beta φάση από επιλεγμένους τελικούς χρήστες. Η μικρή κλίμακα του συγκεκριμένου παιχνιδιού επέτρεψε να δοκιμαστεί κατευθείαν από άλλους χρήστες. Έγινε επιλογή 6 ατόμων, με διαφορετικό επίπεδο γνώσεων των ηλεκτρονικών παιχνιδιών καθώς και διαφορετικό αισθητικό και γνωστικό υπόβαθρο. Από την διαδικασία των δοκιμών προέκυψαν αρκετά προβλήματα (bugs) που δεν είχαν εντοπιστεί προηγουμένως και οδήγησαν στην περαιτέρω βελτίωση του παιχνιδιού καθώς και της ποιότητάς του.

Αποδείχθηκε ότι η φάση του testing είναι πολύ χρήσιμη και κρίσιμη ακόμα και σε μικρής κλίμακας εφαρμογές, όπως ήταν το συγκεκριμένο παιχνίδι. Αποκαλύπτει σφάλματα όπου ο σχεδιαστής μπορεί να παραλείψει καθώς προσφέρει και μια ανατροφοδότηση χρηστών καθιστώντας το προϊόν πιο ελκυστικό και εμπορικό.

7 Scripting

Το κυριότερο μέρος της ανάπτυξης αυτής της εφαρμογής και βασικό στόχος της πτυχιακής εργασίας ήταν η ανάπτυξη του κώδικα του παιχνιδιού. Η ανάπτυξη κώδικα σε μια έτοιμη μηχανή δημιουργίας βιντεοπαιχνιδιών καλείται scripting. Κατά την διαδικασία του scripting ο προγραμματιστής δημιουργεί έγραφα κώδικα καλούμενα ως scripts τα οποία προσδίδουν λειτουργικότητα στα διάφορα συστατικά ή επιμέρους συστήματα της μηχανής. Τα scripts ορίζουν όλη τη λειτουργικότητα ενός παιχνιδιού και μπορεί να αναφέρονται από την διάδραση του παίκτη με τα διάφορα αντικείμενα έως την χρήση και την παραμετροποίηση διάφορων στοιχείων της ίδιας της μηχανής. Συγκεκριμένα, στην Unity τα scripts αναφέρονται και ως Behaviour Components (συστατικά συμπεριφοράς) και προσαρτώνται στα διάφορα αντικείμενα μιας σκηνής. Στην συνέχεια θα αναφερθούμε στα βασικά χαρακτηριστηκά ενός script, θα δούμε παραδείγματα χρήσης και θα αναφερθούμε σε βασικές λειτουργίες που υλοποιήθηκαν μέσω script στην συγκεκριμένη εργασία.

7.1 Συστατικά

Τα scripts χαρακτηρίζονται από 3 κυρίως συστατικά, τις συναρτήσεις, τις μεταβλητές

και τα statements. Οι διάφοροι συνδυασμοί των 3 είναι που μας δίνουν την επιθυμητή λειτουργικότητα.

7.1.1 Συναρτήσεις:

Οι συναρτήσεις χαρακτηρίζονται από την χρήση της δεσμευμένης λέξης function, ένα όνομα και μια λίστα από μηδέν ή περισσότερες παραμέτρους. Η δεσμευμένη λέξη function φροντίζει ώστε ο compiler να "αντιληφθεί" ότι η επερχόμενη λέξη είναι το όνομα μιας συνάρτησης. Το όνομα της συνάρτησης είναι η λέξη με την οποία από εδώ και πέρα θα μπορεί η συνάρτηση να κληθεί. Η λίστα των παραμέτρων έχει τον ρόλο της εισόδου δεδομένων στην συνάρτηση κατά την κλήση της. Μια πάραμετρος μπορεί να είναι μια τιμή ή μια έκφραση με την μορφή μιας μεταβλητής. Από την δημιουργία, του ένα script στην Unity περιέχει δύο βασικές συναρτήσεις, την function Start() και την function Update(). Παρόλο που οι δύο αυτές συναρτήσεις δημιουργούνται εκ προεπιλογής μπορεί ο προγραμματιστής να αφήσει το σώμα τους κενό ή ακόμα και να τις διαγράψει από το script. Η λειτουργία τους είναι αρκετά απλή αλλά η σημασία της πολύ μεγάλη. Η συνάρτηση Start() καλείται με την δημιουργία του script, με το instantiate (δημιουργία υπόστασης) δηλαδή, του αντικειμένου που φέρει το script ως component του. Ουσιαστικά, η συνάρτηση Start() προσφέρει στον προγραμματιστή μια δυνατότητα παραμετροποίησης και αρχικοποίησης. Συχνή χρήση της Start() γίνεται για να εντοπιστούν αντικείμενα ή να αρχικοποιηθούν μεταβλητές ως εξής:

```
var x;
function Start(){
        x=0;
}
```

Ο παραπάνω κώδικας δίνει αρχική τιμή το 0 στην μεταβλητή x κατά την δημιουργία του αντικειμένου στο οποίο έχει τοποθετηθεί το script.

Η συνάρτηση Update() αντίθετα καλείται έπειτα από το instantiate του εν λόγω αντικειμένου και σε κάθε frame έπειτα μέχρι την καταστροφή του αντικειμένου και είναι από τις πιο συχνά χρησιμοποιήσιμες συναρτήσεις. Κατά κύριο λόγο στην συνάρτηση Update() ο προγραμματιστής βάζει ελέγχους και κλήσεις άλλων συναρτήσεων. Η ιδιαιτερότητα της Update() είναι ότι απαγορεύει την χρήση statements που ζητάνε καθυστέρηση επεξεργασίας κώδικα. Χαρακτηριστικό παράδειγμα είναι το εξής:

```
function Update(){
    yield WaitForSeconds(3);
}
```

Αυτός ο κώδικας ζητάει αναμονή τριών δευτερολέπτων πριν ο κώδικας μπορεί να συνεχίσει να εκτελείται. Αυτό όμως πάει κόντρα στην ίδια την λειτουργία της Update() έτσι ο compiler θα εμφανίσει λάθος.

Εκτός από την Start() και την Update() ένα script συνήθως περιέχει και άλλες συναρτήσεις. Κάποιες είναι δημιουργία του προγραμματιστή ενώ κάποιες άλλες ανήκουν στο σύστημα όπως οι Start() και η Update() και ο προγραμματιστής απλά συμπληρώνει το σώμα τους.

Παράδειγμα μιας συνάρτησης δημιουργημένης από τον προγραμματιστή είναι η παρακάτω:

```
public function pressExit(){
        Application.Quit();
}
```

Η συγκεκριμένη συνάρτηση που ονομάζεται pressExit() όταν κληθεί τερματίζει την εφαρμογή, εκτός και αν η εφαρμογή τρέχει στον editor, ο οποίος στην προκείμενη είναι η Unity. Η δεσμευμένη λέξη public στην αρχή της δήλωσης της συνάρτησης δηλώνει ότι η συγκεκριμένη συνάρτηση θα μπορεί να είναι διαθέσιμη και πέραν του πλαισίου του script που ανήκει.

Παράδειγμα μιας συνάρτησης του συστήματος δηλωμένη από τον προγραμματιστή είναι η παρακάτω:

```
function OnGUI () {
        if (drawGUI) {
                GUI.Box (Rect(Screen.width*0.5-51,Screen.height*0.35, 120, 22),"Leave Room");
        }
}
```

Η συνάρτηση OnGUI καλείται σε κάθε frame εάν υπάρχει και επεξεργάζεται rendering και handling GUI events. Στο συγκεκριμένο παράδειγμα, όταν η μεταβλητή drawGUI

γίνει αληθής θα δημιουργηθεί το event GUI.Box το οποίο θα σχεδιάσει στην οθόνη ένα κουτί με τις διαστάσεις του πολύγονου που αναφέρονται και μήνυμα το “Leave Room”.

7.1.2 Μεταβλητές:

Οι μεταβλητές χαρακτηρίζονται από την δεσμευμένη λέξη var ακολουθούμενη από το όνομα της μεταβλητής. Προαιρετικά μπορεί να δωθεί και τιμή στην μεταβλητή με την δήλωση της. Επίσης προαιρετικά μπορεί να δωθεί και τύπος δεδομένων που θα περιέχει η μεταβλητή αλλά κάνοντάς το αυτό, αφαιρείται ένα εξαιρετικό χαρακτηριστικό της Javascript, να ορίζει τύπο δεδομένων για μια μεταβλητή την στιγμή που αυτή αρχικοποιείται. Συνηθίζεται να ορίζεται το scope (έκταση) της μεταβλητής με την δεσμευμένη λέξη public ή private όπου δηλώνει το αν η μεταβλητή είναι θα είναι προσβάσιμη από άλλα scripts ή μόνο από το script στο οποίο δημιουργείται. Τέλος μια μεταβλητή πολλές φορές ορίζεται ως static εάν κριθεί απαραίτητο η τιμή της να διατηρείται κατά την αλλαγή των σκηνών. Πολλές φορές αυτό γίνεται για να διατηρούνται πληροφορίες όπως κάποιο score ή το username κ.α στο σύνολο της εφαρμογής. Ακολουθεί ένα παράδειγμα:

```
public static var isOpen = true;
```

Στο συγκεκριμένο παράδειγμα δημιουργείται και ταυτόχρονα αρχικοποιείται μια μεταβλητή με το όνομα isOpen, παίρνει τύπο δεδομένων boolean, τιμή ίση με true, παραμένει στατική ακόμα και εάν αλλάξει η σκηνή και γίνεται διαθέσιμη σε όλα τα scripts μέσω του public.

7.1.3 Statements

Statements είναι αυτό που στην φυσική γλώσσα θα λέγαμε πρόταση και αποτελεί ένα ξεχωριστό κομμάτι εκτέλεσης. Ακολουθεί παράδειγμα ενός statement:

```
if (Input.GetKeyDown(KeyCode.Escape)) {
        debug.log("Button Escape was pressed");
}
```

Το παραπάνω κομμάτι κώδικα δείχνει ένα if statement στο οποίο ελέγχεται εάν έχει πατηθεί το πλήκτρο “Escape” στο πληκτρολόγιο και εμφανίζει στην κονσόλα ένα μήνυμα. Εννοείται για τέτοιου είδους ελέγχους η ορθότητα ή όχι του statement θα

πρέπει να ελέγχεται σε κάθε frame και άρα τοποθετείται στην συνάρτηση Update().

7.2 Σημαντικά Σημεία

Προηγουμένως αναπτύχθηκε η θεωρία που αφορά το scripting και δόθηκε η βασική ορολογία για να μπορέσουμε να εξετάσουμε σε βάθος στιγμιότυπα κώδικα από την συγκεκριμένη εργασία που παρουσιάζουν κάποιο ενδιαφέρον και να αναφερθούμε σε κάποιες ιδιάζουσες περιπτώσεις.

7.2.1 First Person Controller

Το αντικείμενο First Person Controller υπάρχει στην Unity στο πακέτο των βασικών assets που μπορεί να ενσωματωθεί μαζί με την εγκατάσταση της μηχανής και αντιπροσωπεύει τον χαρακτήρα που θα κινεί ο παίκτης σε είδος παιχνιδιού FPS (First Person Shooter). Η κίνηση του χαρακτήρα (player) υλοποιείται από προϋπάρχον script που είναι ενσωματωμένο στον παίκτη από την δημιουργία. Η λογική είναι ότι ένα script με όνομα FPSInputController εντοπίζει σε κάθε frame την κατεύθυνση στην οποία “κοιτάμε” και ενημερώνει το script CharacterMotor το οποίο είναι υπεύθυνο για την κίνηση. Το CharacterMotor κάνει όλους τους απαραίτητους ελέγχους και μετακινεί το αντικείμενο player ανάλογα με την κατεύθυνση και την ταχύτητα. Ακολουθεί το κομμάτι του script που είναι υπεύθυνο για την κίνηση. Έχουν αφαιρεθεί όλοι οι έλεγχοι για ευκολότερη ανάγνωση καθώς το script είναι αρκετά εκτενές.

```
// Save lastPosition for velocity calculation.
var lastPosition : Vector3 = tr.position;

// We always want the movement to be framerate independent.  Multiplying by Time.deltaTime does this.
var currentMovementOffset : Vector3 = velocity * Time.deltaTime;

// Reset variables that will be set by collision function
movingPlatform.hitPlatform = null;
groundNormal = Vector3.zero;

// Move our character!
```

```
    movement.collisionFlags = controller.Move (currentMovementOffset);

    movement.lastHitPoint = movement.hitPoint;
    lastGroundNormal = groundNormal;
```

Πέραν αυτών, γίνονται έλεγχοι για την ταχύτητα, την βαρύτητα, την κλίση του εδάφους, το collision (σύγκρουση), την απόσταση από το έδαφος και άλλα. Το script δίνει την δυνατότητα παραμετροποίησης της μέγιστης ταχύτητας προς τα μπροστά, τα πλάγια και πίσω, του ύψους που μπορεί να πηδήξει ο παίκτης την βαρύτητα, την ταχύτητα πτώσης και την μέγιστη επιτάχυνση. Φυσικά μπορεί να αλλαχθεί και το script καθεαυτό αλλά δεν προτείνεται καθώς μπορεί να προκύψουν προβλήματα.

Η πλοήγηση του χαρακτήρα στον χώρο χαρακτηρίζεται από δύο πράγματα, τον τρόπο που μετακινείται και τον τρόπο που κινείται η κάμερα σε διάδραση με τον χρήστη. Τον ρόλο της κίνησης της κάμερας λαμβάνει το δεύτερο script που είναι ενσωματωμένο στον First Person Controller, το MouseLook, γραμμένο σε C#. Το MouseLook “αντιλαμβάνεται” την κίνηση του ποντικιού του χρήστη και περιστρέφει το αντικείμενο First Person Controller ανάλογα. Ας δούμε πως γίνεται αυτό:

```
void Update ()
    {
        if (axes == RotationAxes.MouseXAndY)
        {
            float rotationX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") * sensitivityX;

            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            rotationY = Mathf.Clamp (rotationY, minimumY, maximumY);

            transform.localEulerAngles = new Vector3(-rotationY, rotationX, 0);
        }
        else if (axes == RotationAxes.MouseX)
        {
            transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityX, 0);
```

```
        }
        else
        {
            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            rotationY = Mathf.Clamp (rotationY, minimumY, maximumY);

            transform.localEulerAngles = new Vector3(-rotationY,
transform.localEulerAngles.y, 0);

        }
    }
```

Αυτό που κάνει το script είναι ουσιαστικά να περιστρέφει το First Person Controller με το transform ανάλογα με την περιστροφή του ανάλογου άξονα του ποντικιού όπως αυτός εντοπίζεται από το Input.GetAxis.Οι μεταβλητές που λαμβάνουν χώρα στην συνάρτηση Update() αρχικοποιούνται πιο πριν σύμφωνα με τις επιλογές του προγραμματιστή ή του χρήστη. Στην συγκεκριμένη εφαρμογή, ήταν επιθυμητό να μην εμφανίζεται κάποιος κέρσορας στον χρήστη παρά μόνο ένα στάσιμο δισδιάστατο texture οπότε στην συνάρτηση Start() του MouseLook προστέθηκε η ακόλουθη γραμμή:

```
Screen.showCursor = false;
```

Όπου δίνει την εντολή στο αντικείμενο οθόνη να μην εμφανίζει τον κέρσορα.

7.2.2 Διάδραση με Αντικείμενα

Ο παίκτης με κάποιον τρόπο θα πρέπει να μπορεί να διαδρά με τα διάφορα αντικείμενα της σκηνής. Αυτό για να μπορεί να γίνει έχει τρεις προϋποθέσεις: πρώτον, να είναι ο παίκτης σε μια απόσταση κοντινή με το αντικείμενο, δεύτερον, να το κοιτάει, όπου στο παιχνίδι καθορίζεται από το εάν είναι ο κέρσορας ή στον προκειμένη το δισδιάστατο γραφικό, πάνω από το αντικείμενο και τρίτον να δηλώνει την ενέργεια, κατά βάση πατώντας κάποιο πλήκτρο ή κάνοντας κλικ.

```
function Update () {
    var ray : Ray = Camera.main.ViewportPointToRay(guiTex.transform.position);
    var hit : RaycastHit;
```

```
    if(canopen){
        if(Physics.Raycast(ray, hit) && (hit.transform.name == "group_1" ||
hit.transform.name == "group_0")) {
            if (drawGUI == true && Input.GetKeyDown(KeyCode.E)) {
                toggleDoorState();
            }
        }
    }
}

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

Το παραπάνω script χωρίζεται σε τρία σημεία, στην συνάρτηση Update() στον εμφωλευμένο έλεγχό της και στην συνάρτηση OnTriggerEnter(). Ο κώδικας στην παραπάνω συνάρτηση Update() λύνει το πρόβλημα του να “κοιτάς” ένα αντικείμενο. Αυτό γίνεται ως εξής: σε κάθε frame εκπέμπεται από την κάμερα μια ακτίνα, με αρχικό σημείο το δισδιάστατο texture. Έπειτα διαβάζεται το όνομα του αντικειμένου στο οποίο έχει προσκρούσει η εκπεμπόμενη ακτίνα και ελέγχεται. Εάν το όνομα του αντικειμένου ταιριάζει με το ζητούμενο αντικείμενο τότε ο παίκτης “κοιτάει” το αντικείμενο. Έπειτα ελέγχεται εάν την στιγμή που κοιτάει το αντικείμενο είναι σε κοντινή απόσταση και εάν έχει δώσει εντολή διάδρασης. Η εντολή φαίνεται από το εάν έχει πατήσει το πλήκτρο “E”. Η συνάρτηση OnTriggerEnter() έρχεται να λύσει το πρόβλημα της απόστασης από το αντικείμενο. Στο αντικείμενο που μας ενδιαφέρει προσθέτουμε ένα Collider με επιλογή Is Trigger επιλεγμένη. Αυτό σημαίνει ότι δεν θα εμποδίζει το collide αλλά αντ'αυτού θα ενεργοποιεί την OnTriggerEnter() και την OnTriggerExit() σε αντίθετη περίπτωση. Η OnTriggerEnter() και Exit() όπως γίνεται φανερό, είναι και αυτές συναρτήσεις του συστήματος που έρχεται να υλοποιήσει απλά ο προγραμματιστής. Ελέγχουμε λοιπόν στην OnTriggerEnter εάν το αντικείμενο που εισήλθε ονομάζεται

Player (όπως θα πρέπει να έχουμε δώσει για tag στο αντικείμενο First Person Controller) και εάν είναι, τότε σηματοδοτούμε, ως επί το πλείστο με μια flag μεταβλητή, μεταβλητή τύπου boolean δηλαδή, το γεγονός. Αυτό το κομμάτι κώδικα βρίσκεται σε κάθε script που περιλαμβάνει την διάδραση του παίκτη με αντικείμενα της σκηνής με μικρές ίσως διαφοροποιήσεις.

7.2.3 Πόρτες

Οι πόρτες δίνουν πρόσβαση στον παίκτη σε μια επιπλέον περιοχή. Αυτό γίνεται με δύο τρόπους. Είτε ανοίγουν για να επιτρέψουν στον παίκτη να περάσει στην επιπλέον περιοχή που οριοθετούν είτε φορτώνουν και μεταφέρουν τον παίκτη σε μια άλλη σκηνή. Από την στιγμή που ο παίκτης θα δώσει την εντολή, η ανάλογη διαδικασία υλοποιείται με τους εξής δύο τρόπους:

```
1{
        doubleDoors.animation.CrossFade("Open LivingRoom Doors");
        doubleDoors.audio.PlayOneShot();
        doorIsClosed = false;
}
2{
        audio.Play();
        WaitForSeconds(1);
        Application.LoadLevel("livingroom and hall");
}
```

Στο πρώτο script εφόσον έχουμε μια πόρτα που είναι ένα φυσικό αντικείμενο που πρέπει να ανοίξει απλά παίζουμε ένα animation που φροντίζει να περιστρέφει την πόρτα κατά το δοκούν του προγραμματιστή οπότε ο χρήστης αποκτάει πρόσβαση στον επόμενο χώρο. Κατά την διάρκεια του animation πιθανόν είναι να παίζει και κάποιος ήχος όπως και να κρατηθεί η αλλαγή κατάστασης σε μια flag μεταβλητή.

Στο δεύτερο script αρχικά παίζει ένας ανάλογος ήχος και αφού τελειώσει, η μηχανή δίνει πρόσβαση στον χρήστη σε έναν καινούριο χώρο φορτώνοντας μια διαφορετική σκηνή. Για να λειτουργήσει η μέθοδος LoadLevel θα πρέπει όλες οι σκηνές που είναι να φορτωθούν καθ' αυτόν τον τρόπο να δηλωθούν στα build setting του παιχνιδιού ως Scenes In Build. Προσοχή, με την αλλαγή της σκηνής όλες οι μεταβλητές και τα

αντικείμενα της προηγούμενης σκηνής θα επανέλθουν στην αρχική τους κατάσταση κατά την επιστροφή εκτός από μεταβλητές που έχουν οριστεί static και από αντικείμενα για τα οποία έχει δοθεί οδηγία να μην καταστραφούν με την εντολή DontDestroyOnLoad.

7.2.4 Φωτισμός

Για τον φωτισμό γράφτηκαν τρία διαφορετικά script για της ανάγκες της συγκεκριμένης πτυχιακής εργασίας. Ένα για το εφέ του flicker (τρεμόσβυμα) στο φως που παράγεται από φωτιά, ένα για να ανοίγουν ή να κλείνουν τα φώτα σε κάποιο δωμάτιο και ένα για την προσομοίωση φωτισμού από αστραπές. Ακολουθεί η υλοποίηση των τριών διαφορετικών περιπτώσεων.

```
function toggleLightState() {
    audio.Play();
    if (lightOn) {
        roomLight.intensity = 0;
        lightOn = false;
    }
    else {
        roomLight.intensity = intensityOn;
        lightOn = true;
    }
}
```

Το παραπάνω κομμάτι κώδικα είναι υπεύθυνο για το άνοιγμα και το σβήσιμο των φώτων σε έναν δωμάτιο. Όταν καλείται η συνάρτηση toggleLightState() ακούγεται ο ήχος του διακόπτη και ελέγχεται μια flag μεταβλητή που προσδιορίζει εάν το φως είναι ανοιχτό ή κλειστό. Εάν είναι ανοιχτό τότε θέτει την ένταση του φωτός, μέσω της ανάλογης ιδιότητας του ανάλογου αντικειμένου Light, ίση με μηδέν και θέτει την τιμή της flag μεταβλητής ως ψευδή. Στην περίπτωση που το φως είναι κλειστό θέτει την ένταση του φωτός ίση με μια προκαθορισμένη απ τον προγραμματιστή τιμή.

```
function Start()
 {
    random = Random.Range(0.0f, 65535.0f);
```

```
}

 function Update()
{
    var noise = Mathf.PerlinNoise(random, Time.time*2);
    light.intensity = Mathf.Lerp(minIntensity, maxIntensity, noise);
}
```

Το παραπάνω κομμάτι κώδικα ευθύνεται για το flicker εφέ. Παράγει τιμή ενός θορύβου ανάλογη του χρόνου και μιας τυχαίας τιμής και αλλάζει την ένταση του φωτός σε κάθε frame με μια τυχαία τιμή που παράγεται από την παρεμβολή μια τυχαίας τιμής ανάμεσα στην μέγιστη και στην ελάχιστη ένταση που μπορεί να έχει η πηγή φωτός προσθέτοντας και τον θόρυβο.

Και τέλος:

```
var minTime = .5; var thresh = .5;
var counter=0;

private var lastTime = 0;

function Start()
{
   light.enabled=false;
}

 function Update()
{
   counter++;
   if(counter==10){
      thunder();
      counter=0;
   }
```

```
}

function thunder(){
    if ((Time.time - lastTime) > minTime) {
     if (Random.value > thresh)
         light.enabled = true;
     else
         light.enabled = false;
     }
    lastTime = Time.time;
}
```

Το παραπάνω κομμάτι είναι το πλήρες script που χρησιμοποιείται για την προσομοίωση του εφέ του κεραυνού. Ξεκινώντας το script απενεργοποιεί τον φωτισμό. Στην συνέχεια ένας μετρητής μετρά την πάροδο δέκα frame μέχρι πριν καλέσει την συνάρτηση thunder(). Η συνάρτηση thunder() ελέγχει εάν έχει περάσει αρκετός χρόνος από την τελευταία αστραπή και κατόπιν ελέγχει εάν μια τυχαία τιμή είναι μεγαλύτερη από ένα κατώφλι. Εάν οι παραπάνω έλεγχοι είναι αληθείς τότε ενεργοποιείται η πηγή φωτός που προσομοιώνει την αστραπή, εάν η τυχαία τιμή είναι κατώτερη του κατωφλίου τότε η πηγή φωτός απενεργοποιείται ξανά. Οι διπλοί έλεγχοι και η αναμονή των δέκα frames βρίσκονται για να προσομοιώσουν όσο το δυνατόν ρεαλιστικότερα την διάρκεια του φωτισμού μιας αστραπής καθώς και την συχνότητα των αστραπών.

7.2.5 Ανάγνωση Σελίδων

Είδαμε προηγουμένως πως ο παίκτης μπορεί να διαδρά με τα διάφορα αντικείμενα του χώρου εκδίδοντας μια εντολή διάδρασης. Στην αλληλεπίδραση με της σελίδες της σκηνής εμφανίζεται στον παίκτη ένα δισδιάστατο texture κειμένου το οποίο ο παίκτης καλείται να διαβάσει πριν κάνει το οτιδήποτε άλλο. Στην συνέχεια παραθέτεται η υλοποίηση του συγκεκριμένου μηχανισμού.

```
var lookArround01 : MouseLook;
var lookArround02 : MouseLook;
var charMotor : CharacterMotor;
```

```
function Start() {
    lookArround01 = GameObject.Find("Main Camera").GetComponent(MouseLook);
    lookArround02 = GameObject.Find("First Person Controller").GetComponent(MouseLook);
    charMotor = GameObject.Find("First Person Controller").GetComponent(CharacterMotor);
}

function Hide() {
    guiTexture.enabled = false;
    lookArround01.enabled = true;
    lookArround02.enabled = true;
    charMotor.enabled = true;
}

function Show(paper : Paper) {

    if (!guiTexture.enabled) {
        var t = paper.popupTexture;
        var rect = Rect(-t.width/2+250, -t.height/2+300, t.width*0.5, t.height*0.55);
        guiTexture.pixelInset = rect;
        guiTexture.texture = t;
        guiTexture.enabled = true;
        lookArround01.enabled = false;
        lookArround02.enabled = false;
        charMotor.enabled = false;
    }
}

function Update () {
```

```
    if (Input.GetKeyDown(KeyCode.Escape)) {
        Hide();
    }

}
```

Ας αναλύσουμε το παραπάνω script σε κομμάτια λειτουργικότητας αποτελούμενα από τις συναρτήσεις του.

Αρχικά, στην συνάρτηση Start() εντοπίζονται τα τρία script που είναι υπεύθυνα για την περιήγηση του παίκτη και την κίνηση της κάμερας, τα οποία συζητήθηκαν σε προηγούμενο σημείο του κεφαλαίου και τα καταχωρεί σε ανάλογες μεταβλητές. Το συγκεκριμένο script καλείται μέσω ενός script που είναι προσαρτημένο στο αντικείμενο της σελίδας με την εξής εντολή:

```
paperPopup.Show(this);
```

Άρα ενεργοποιείται η συνάρτηση Show() με παράμετρο το script το οποίο την κάλεσε. Η Συνάρτηση Show() ελέγχει εάν έχει εμφανιστεί ήδη το ζητούμενο texture στην οθόνη και εάν όχι, το προετοιμάζει χρησιμοποιώντας το δισδιάστατο texture που είναι προσαρτημένο στο αντικείμενο που είναι και το script που την κάλεσε και το εμφανίζει στην οθόνη σε προεπιλεγμένο σημείο με προεπιλεγμένες διαστάσεις ενώ παράλληλα θέτει τα τρία script της περιήγησης που εντοπίστηκαν σε ανενεργά. Αυτό βεβαιώνει ότι ο χρήστης δεν θα μπορέσει να μετακινηθεί όσο διαβάζει την σελίδα που του εμφανίστηκε. Τέλος, ελέγχεται σε κάθε frame μέσω της συνάρτησης Update() εάν έχει πατηθεί το πλήκτρο “Escape”. Όταν η έκφραση βρεθεί αληθής τότε καλείται η μέθοδος Hide() η οποία εξαφανίζει το texture της σελίδας από την οθόνη του χρήστη και επανενεργοποιεί τα script της περιήγησης.

7.2.6 **Επανεκκίνηση**

Κατά την επανεκκίνηση του παιχνιδιού, μέσω της ανάλογης επιλογής θέλουμε να επανέρχονται όλα στην αρχική τους κατάσταση και να ξεκινάει ο παίκτης απ το σημείο που ξεκίνησε και την πρώτη φορά. Για να γίνει αυτό πρέπει να κρατάμε μια λίστα με τις static μεταβλητές που έχουμε χρησιμοποιήσει και να τις επαναφέρουμε όλες στην αρχική τους κατάσταση. Στην συγκεκριμένη πτυχιακή εργασία αυτή η διαδικασία

υλοποιήθηκε κατά τον ακόλουθο τρόπο:

```
public function pressRestart(){
    Application.LoadLevel("Start Menu");

    //restart all statics
    basementDoorTrigger.hasBasementKey = false;
    BedroomDoorTrigger.bedroomlocked = true;
    CupboardTrigger.drawFlashlight = true;
    CupboardTrigger.drawKeyFlag = false;
    CupboardTrigger.drawPageFlag = false;
    CupboardTrigger.locked = true;
    CupboardTrigger.hasCupboardKey = false;
    FireplaceItemTrigger.gotVirgo = false;
    FireplacePokerTrigger.destroyed = false;
    OneNationUnderGodTrigger.triggered = false;
    ouijaTrigger.ouijaPlayed = false;
    Paper.readIt = false;
    SecondFloorTrigger.hasSecondFloorKey = false;
    zodiacTrigger.hassecondfloorkey=false;
    scream.screamed = false;
}
```

Η μέθοδος pressRestart() καλείται με την χρήση ενώ UI κουμπιού (διαθέσιμα από την Unity 4.6 και έπειτα) και κάνει υλοποιεί δύο λειτουργίες. Φορτώνει το πρώτο αρχικό μενού που συναντά ο παίκτης καθώς ξεκινάει το παιχνίδι και επαναφέρει όλες τις static μεταβλητές που χρησιμοποιήθηκαν σε όλα τα script στην αρχική τους κατάσταση.

7.3 *Παρατηρήσεις*

Καθ' όλη την διαδικασία του προγραμματισμού της εν λόγω πτυχιακής εργασίας προέκυψαν κάποιες δυσκολίες οι οποίες καταπολεμήθηκαν με τα ανάλογα προγραμματιστικά τεχνάσματα. Στο συγκεκριμένο υποκεφάλαιο θα αναφερθούν παρατηρήσεις όσων αφορά το scripting, τόσο σε επίπεδο κώδικα όσο και σε επίπεδο

τεχνικών προγραμματισμού.

7.3.1 Διαχείριση στατικών μεταβλητών

Μπορούμε από ένα script να προσπελάσουμε public μεταβλητές ή μεθόδους ενός άλλου script. Για αυτόν τον λόγο πολλές φορές χρησιμοποιούμε public static μεταβλητές για διάφορους ελέγχους ή αναφορές καταστάσεων. Οι μεταβλητές ενός script όμως παίρνουν τιμή με την αρχικοποίηση του αντικειμένου στο οποίο βρίσκονται στην εκάστοτε σκηνή. Οπότε, εάν θέλουμε στην ίδια σκηνή να προσπελάσουμε μεταβλητές ενός script από ένα άλλο script τότε δεν πρέπει να κάνουμε την εκχώρηση στην δήλωση της μεταβλητής αλλά στο σημείο όπου θα την χρειαστούμε, καθώς μια αρχική δήλωση θα δώσει τιμή στην μεταβλητή που δεν θα μπορεί να αλλάξει ακόμα και εάν η αντίστοιχή της στο αναφερόμενο script έχει αλλάξει. Για να γίνει καλύτερα κατανοητή αυτή η τεχνική ας δούμε ένα παράδειγμα από τα scripts της εν λόγω πτυχιακής εργασίας.

Script A:

```
public static var destroyed;

function triggerFireplacePoker(){
    destroyed = true;
    Destroy (gameObject);
}
```

Script B:

```
private var hasPoker = A.destroyed;
```

Με αυτόν τον τρόπο η μεταβλητή hasPoker του script B θα ενημερωθεί όταν η μεταβλητή destroyed του script A λάβει τιμή και όχι με την αρχικοποίηση (που πιθανώς θα λάμβανε μια λανθασμένη ή προσωρινή τιμή).

7.3.2 Διαφοροποίηση της λειτουργικότητας

Μια καλή τεχνική προγραμματισμού είναι η αποσύνδεση (decoupling) της λειτουργικότητας. Αυτό συνεπάγεται καθαρότερο κώδικα και αυξημένη επαναχρησιμοποιησιμότητα. Στην javascript αυτό μπορεί να επιτευχθεί ως εξής. Αντί

να επιφορτίζεται η συνάρτηση Update() με μεγάλο κομμάτι της λειτουργικότητας του συστήματος είναι καλύτερα η Update() να καλεί μια άλλη συνάρτηση για κάθε διαφορετική λειτουργία. Πέραν των προαναφερθέντων θετικών, η συγκεκριμένη τεχνική αυξάνει και την ταχύτητα του συστήματος καθώς κάθε κλήση μιας συνάρτησης θα “τρέχει” σε διαφορετικό thread οπότε θα επεξεργάζεται παράλληλα. Ακολουθεί παράδειγμα από τον πραγματικό κώδικα που χρησιμοποιήθηκε στην εφαρμογή.

```
function Update(){
  if (drawGUI && !ouijaPlayed && Input.GetKeyDown(KeyCode.E)) {
    triggerOuija();
  }
}

function triggerOuija(){
   ouijaPlayed = true;
   audio.Play();
   board.animation.CrossFade("Ouija");
}
```

Σε αυτό το παράδειγμα φαίνεται πως ακόμα και εάν μια λειτουργία είναι αρκετά απλή (το παίξιμο ενός ήχου και ενός animation) είναι καλύτερο να το τοποθετούμε σε ξεχωριστή συνάρτηση την οποία καλούμε από την Update(). Αν θεωρήσουμε ότι ο έλεγχος μπορεί να προσδώσει κάποια καθυστέρηση μπορούμε να τον προσθέσουμε και αυτόν μέσα στην συνάρτηση, αν και αυτό δεν είναι καλή τεχνική γιατί σημαίνει ότι η συνάρτηση θα καλείται σε κάθε frame. Συγκεκριμένα στην Unity, εφαρμόζοντας αυτήν την τεχνική της μετάθεση της λειτουργικότητας από την συνάρτηση Update() σε μια άλλη συνάρτηση δίνει το προνόμιο στον προγραμματιστή να την χαρακτηρίσει ως public και να έχει πρόσβαση σε αυτήν ακόμα και εάν το αντικείμενο που φέρει το συγκεκριμένο script έχει καταστραφεί ή βρίσκεται σε διαφορετική σκηνή.

Σε αυτό το κεφάλαιο συζητήθηκαν οι βασικότερες λεπτομέρειες του scripting στα πλαίσια του της συγκεκριμένης εφαρμογής. Επίσης, καταγράφηκαν σημαντικά σημεία ή τεχνικές οι οποίες συνηθίζονται στην αγορά εργασίας και υλοποιήθηκαν για καλύτερα αποτελέσματα. Τέλος, παρουσιάστηκαν τα σημεία που παρουσίαζαν το

μεγαλύτερο ενδιαφέρον όσον αφορά το προγραμματιστικό κομμάτι της εφαρμογής αυτής.

8 Συμπεράσματα

Η βιομηχανία των ηλεκτρονικών παιχνιδιών προσφέρει πολύ μεγάλες αμοιβές και μια θέση ως προγραμματιστής σε κάποιον μεγάλο τίτλο είναι σίγουρα αξιοζήλευτη. Υπάρχουν όμως και αρκετά παραδείγματα προγραμματιστών οι οποίοι μόνοι ή με μια μικρή ομάδα έφτιαξαν indie τίτλους οι οποίοι όχι μόνο είχαν μεγάλα κέρδη αλλά κατάφεραν να αποκτήσουν φήμη και κοινό εφάμιλλα αυτών των διασημότερων εταιρειών. Η συγκεκριμένη πτυχιακή εργασία έδειξε πως, με γνώμονα τις σύγχρονες τεχνικές και τεχνολογίες, μπορεί κάποιος να είναι ανταγωνιστικός και να παράξει ένα αξιοπρεπές αποτέλεσμα ακόμα και με μηδενικό προϋπολογισμό. Συγκεκριμένα, φάνηκε σύμφωνα με τις φάσεις ανάπτυξης που παρουσιάστηκαν και αναλύθηκαν, τα βήματα τα οποία μπορούν να ακολουθηθούν για την δημιουργία μιας πλήρης εφαρμογής ηλεκτρονικού παιχνιδιού. Σημαντικά σημεία στα οποία πρέπει να σταθεί ένας προγραμματιστής που διαβάζει την συγκεκριμένη πτυχιακή είναι:

- Ακόμα και ένα άτομο είναι αρκετό
- Υπολόγισε το ως ελάχιστο χρονικό περιθώριο τους 2 μήνες
- Προσπάθησε να έχεις όσο το δυνατόν καινοτόμες ιδέες
- Ακολούθησε την πεπατημένη μεθοδολογία

Ακολουθώντας τα παραπάνω απλά συμπεράσματα είναι δυνατόν κάποιος με μέτρια γνώση και αρκετή όρεξη να δημιουργήσει κάτι που μπορεί να έχει ακόμα και παγκόσμια απήχηση

9 Βιβλιογραφία

Adams, E. και Rollings, A. (2003). Andrew Rollings and Ernerst Adams on game design. New Riders Publishing.

Bates B. (2004). Game Design (2nd ed.). Thomson Course Technology.

Bethke, E. (2003). Game development and production. Texas: Wordware Publishing, Inc.

Brathwaite, B. και Schreiber, I. (2009). Challenges for Game Designers. Charles River Media.

Chandler, H. M. (2009). The Game Production Handbook (2nd ed.). Hingham, Massachusetts: Infinity Science Press.

McGuire, M. και Jenkins, O. C. (2009). Creating Games: Mechanics, Content, and Technology. Wellesley, Massachusetts: A K Peters.

McShaffry, M. (2009). Game Coding Complete. Hingham, Massachusetts: Charles River Media.

Moore, M. E. και Novak, J. (2010). Game Industry Career Guide. Delmar: Cengage Learning.

Oxland, K. (2004). Gameplay and design. Addison Wesley.

Thirslung A. (2012). How to create a Survival Game – Unity Course. Αναρτημένο στο κανάλι https://www.youtube.com/user/Brackeys/playlists

Thirslung A. (2012). UI in Unity 4.6. Αναρτημένο στο κανάλι https://www.youtube.com/user/Brackeys/playlists

Οτιδήποτε επιπλέον δεν ανήκει σε αυτήν την βιβλιογραφία έχει παραχθεί από προσωπική γνώση, δοκιμή και λογικά συμπεράσματα.