ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

«ΑΝΑΠΤΥΞΗ ΔΙΑΔΡΑΣΤΙΚΟΥ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΠΑΙΧΝΙΔΙΟΥ ΒΑΣΙΣΜΕΝΟ ΣΤΗ ΦΥΣΙΚΗ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΜΕΛΕΤΙΟΣ ΣΤΑΘΗΣ

Εξεταστική επιτροπή

Επ.Καθ. Αικατερίνη Μανιά (επιβλέπουσα)
Επ.Καθ. Μιχαήλ Γ. Λαγουδάκης
Επ.Καθ. Αντώνιος Δεληγιαννάκης

Χανιά 2009

Στην οικογένειά μου

Περίληψη

Η εργασία αυτή παρουσιάζει ένα διαδραστικό τρισδιάστατο παιχνίδι για υπολογιστές βασισμένο σε τεχνολογίες υπολογιστικών γραφικών οι οποίες υποστηρίζουν εξομοίωση φυσικής. Το παιχνίδι υλοποιήθηκε κάνοντας χρήση της μηχανής παιχνιδιών Unity3D και είναι τύπου τρισδιάστατης δράσης. Αποτελείται απο ένα περιβάλλον στο οποίο ο χρήστης καλείται να περιπλανηθεί και να περάσει δοκιμασίες πριν να φτάσει στο τερματισμό. Υλοποιήθηκαν η γραφική διεπαφή για τον χρήστη, τα μενού διεπαφής, τα χαρακτηριστικά που βασίζονται στην φυσική, κινήσεις περπατήματος, άλματος, γροθιάς, διάφορα εφφέ και ήχοι, η καταγραφή υψηλότερων βαθμολογιών, αποθήκευσης σε εξυπηρετητή δικτύου και η συμπεριφορά αντιπάλων του χρήστη βασισμένη σε αρχές της τεχνητής νοημοσύνης. Η εφαρμογή, αξιολογήθηκε με χρήση των ερωτηματολογίων QUIS (Questionnaire for User Interaction Satisfaction) από χρήστες και των δυο φύλων, διαφορετικής ηλικίας και επαγγελματικής προέλευσης.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στην επιβλέπουσα καθηγήτριά μου κ. Κατερίνα Μανιά για την επίβλεψη και καθοδήγησή της στην εκπόνηση αυτής της διπλωματικής εργασίας. Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές κ. Λαγουδάκη Μιχάλη και κ. Δεληγιαννάκη Αντώνιο για το χρόνο που θα αφιερώσουν στην ανάγνωση του κειμένου και για τις εποικοδομητικές παρατηρήσεις τους. Ακόμα, θα ήθελα να ευχαριστήσω όσους αφιέρωσαν χρόνο και με βοήθησαν συμπληρώνοντας τα ερωτηματολόγια.

Τέλος, ένα μεγάλο ευχαριστώ στους ανθρώπους που μου έδωσαν κουράγιο, έμπνευση και έδειξαν κατανόηση, όλα αυτά τα χρόνια που δεν ήταν πάντα εύκολα, αλλά μαζί τους έγιναν παραπάνω από ευχάριστα και εποικοδομητικά.

Οκτώβριος 2009
Μελέτιος Στάθης

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΔΙΑΓΡΑΜΜΑΤΑ ΚΑΙ ΕΙΚΟΝΕΣ

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

Στόχος της παρούσας διπλωματικής εργασίας ήταν η υλοποίηση ενός διαδραστικού τρισδιάστατου παιχνιδιού για υπολογιστές το οποίο βασίζεται σε τεχνολογίες υπολογιστικών γραφικών οι οποίες υποστηρίζουν εξομοίωση φυσικής. Το εν λόγω παιχνίδι μπορεί να κατηγοριοποιηθεί στα παιχνίδια τρισδιάστατης δράσης (3D action) [1]. Τα παιχνίδια τρισδιάστατης δράσης, στην πλειονότητά τους, μπορούν να οριστούν ως τα παιχνίδια που περιλαμβάνουν ποικιλία προκλήσεων για τον παίκτη, όπως αγώνες ταχύτητας, γρίφους, συγκομιδή αντικειμένων κ.ά. Το κυριότερο και πιο σημαντικό γνώρισμα αυτής της κατηγορίας παιχνιδιών είναι ότι ο χρήστης πρέπει να είναι γρήγορος στις αντιδράσεις του και με αποδοτικό συντονισμό καλής όρασης και χειρισμού του χαρακτήρα. Ο παίκτης είναι συνήθως κάτω από χρονική πίεση και έτσι δεν μπορεί να καταστρώσει πολύπλοκες στρατηγικές. Γενικά, τα παιχνίδια που εξελίσσονται πιο γρήγορα είναι και πιο διασκεδαστικά. Οι αντίπαλοι πρέπει να επιδεικνύουν «έξυπνη» συμπεριφορά ώστε να μπορούν να αντιδράσουν μετά από κινήσεις του παίκτη.

1.2 Περιγραφή παιχνιδιού

1.2.1 Σύντομη περιγραφή

Το παιχνίδι που παρουσιάζεται σε αυτή τη διπλωματική είναι ένα παιχνίδι τρίτου προσώπου και δράσης. Ο κύριος χαρακτήρας θα πρέπει να περάσει απο κάποιες δοκιμασίες για να φτάσει στον τερματισμό. Οι δοκιμασίες έχουν να κάνουν με την εύρεση αντικείμενων, καταστροφή των εχθρών και περάτωση αποστολών σε συγκεκριμένα χρονικά όρια.

1.2.2 Πίστα

Εν γένει, σε κάθε παιχνίδι δράσης ο παίκτης προχωρεί όπως τον καθοδηγεί η πλοκή του παιχνιδιού τερματίζοντας «πίστες». Κάθε «πίστα» έχει αντιπάλους, με συγκεκριμένη εμφάνιση και συμπεριφορά και διαδραματίζεται στο ίδιο γραφικό περιβάλλον. Κάθε «πίστα» περιλαμβάνει προκλήσεις τις οποίες ένας παίκτης πρέπει να αντιμετωπίσει με επιτυχία, ώστε να καταφέρει να προχωρήσει στην επόμενη. Στην δικιά μας περίπτωση, έχει δημιουργηθεί ένα γραφικό περιβάλλον το οποίο απεικονίζει ένα νησί. Το νησί περιλαμβάνει σημεία από τα οποία ο παίκτης μπορεί να «αναστηθεί» κάθε φορά που χάνει μια ζωή. Όπως σε πλήθος παιχνιδιών, στο περιβάλλον υπάρχουν χώροι που ο παίκτης μπορεί να εισχωρήσει, έχοντας συγκεκριμένα κλειδιά.

Η χρησιμοποίηση περιορισμών κινήσεων είναι αναγκαία ώστε να αυξήσουμε το επίπεδο πρόκλησης και δυσκολίας. Για παράδειγμα, ο παίκτης μπορεί να καεί όταν εισέρχεται στο ηφαίστειο ή η να έχει δυσκολία ορατότητας του περιβάλλοντος του όταν εισέρχεται στο βυθό της θάλασσας.

1.2.3 Δυνατότητες Χαρακτήρα

Ο παίκτης ελέγχει ένας χαρακτήρα, ο οποίος είναι ο πρωταγωνιστής του παιχνιδιού. Ο χαρακτήρας έχει την ικανότητα να πλοηγείται, να κινείται και συχνά να συλλέγει αντικείμενα. Έχει ακόμα την δυνατότητα να γρονθοκοπεί, να χτυπάει δέντρα και να σπάει κανάτες. Ο παίκτης μπορεί να βρει διάφορα αντικείμενα που του δίνουν ενέργεια ή μπορούν να βελτιώσουν την φυσική του κατάσταση όπως να γίνει δυνατότερος, γρηγορότερος και πιο αλτικός .

1.2.4 Εμπόδια και Αντίπαλοι

Ο παίκτης καθώς πλοηγείται στο περιβάλλον θα αντιμετωπίσει εμπόδια, παγίδες και αντιπάλους. Οι αντίπαλοι εμπεριέχουν αρχές τεχνητής νοημοσύνης ώστε να αντιλαμβάνονται την παρουσία του παίκτη, να τον κυνηγούν και να του επιτίθονται. Όσο προχωράμε προς τον τερματισμό του παιχνιδιού οι αντίπαλοι γίνονται πιο ικανοί. Μπορούν να τρέχουν πιο γρήγορα και να πετάν πιο πολλές καρύδες οι οποίες αποτελούν το όπλο που χρησιμοποιούν εναντίον του κυρίως παίκτη.

1.2.5 Υγεία και ζωές

Ο παίκτης έχει ενέργεια, η οποία μειώνεται καθώς διαδραματίζονται επιτυχημένες επιθέσεις αντιπάλων. Ακόμα, η ενέργεια μειώνεται μετά από χτυπήματα από βράχους, που του πέφτουν «ουρανοκατέβατα». Η ενέργεια μπορεί να αναπληρωθεί συλλέγοντας αντικείμενα που βρίσκονται στην πίστα. Όταν η ενέργεια του παίκτη τελειώσει, τότε μειώνονται οι ζωές. Όταν ο χαρακτήρας «χάσει» μια ζωή, θα ξεκινήσει από το τελευταίο σημείο ελέγχου (checkpoint) που έχει ενεργοποιήσει. Ο παίκτης μπορεί να κερδίσει και μερικές έξτρα ζωές βρίσκοντας το συγκεκριμένο αντικείμενο της «ζωής» στην πίστα.

1.2.6 Γραφικό περιβάλλον

Το παιχνίδι διαδραματίζεται σε ένα τρισδιάστατο χώρο και είναι ορατό στον χρήστη από συγκεκριμένη προοπτική, αυτή της κάμερας. Η κάμερα που δείχνει τι συμβαίνει στον συγκεκριμένο χώρο μετακινείται καθώς ο παίκτης εξερευνά το περιβάλλον και είναι τοποθετημένη, στην πλειονότητα των περιπτώσεων, σε μια συγκεκριμένη απόσταση και ύψος από τον παίκτη. Είναι η λεγόμενη κάμερα τρίτου προσώπου (third person camera). Ότι χρειάζεται να γνωρίζει ο παίκτης για την κατάσταση του παιχνιδιού βρίσκεται στην οθόνη (head-up display,HUD). Αυτή η οθόνη περιέχει σημαντικές πληροφορίες για τον παίκτη όπως το επίπεδο της διαθέσιμης ενέργεια, τις ζωές, τα νομίσματα, τον αριθμό των μπουκαλιών ρούμι έχει πιει, την βαθμολογία που

έχει μαζέψει και ένα μικρό χάρτη της πίστας ώστε ο παίκτης να μπορεί να κάνει ευκολότερη την πλοήγηση του.

1.2.7 Βαθμολογία και τερματισμός παιχνιδιού

Το παιχνίδι, όπως τα περισσότερα παιχνίδια δράσης, έχει απλούς στόχους και είναι αρκετά προφανές για τον παίκτη πως θα φτάσει στην διεκπεραίωσή τους. Οι στόχοι παρουσιάζονται με την μορφή μίας δομημένης ιστορίας, δηλαδή ενός σεναρίου (κεφάλαιο 4.2), με το ανάλογο "happy end", όταν τερματιστεί το παιχνίδι.

Στο παιχνίδι καταγράφεται η βαθμολογία του κάθε παίκτη. Ο παίκτης παίρνει βαθμούς όταν συλλέγει νομίσματα, σκοτώνει αντιπάλους, σπάει κανάτες, όταν τερματίσει το παιχνίδι και ανάλογα με τα δευτερόλεπτα που του έχουν απομείνει όταν τελειώνει μια αποστολή. Παρότι δεν είναι ο κύριος στόχος του παιχνιδιού να μαζέψει όσο το δυνατόν μεγαλύτερη βαθμολογία, είναι ένα αρκετά σημαντικό στοιχείο ώστε να θεωρηθεί ένας παίκτης ικανός. Η ικανότητά του στο παιχνίδι εξαρτάται από το εάν θα καταφέρει να πετύχει μεγαλύτερη βαθμολογία από τους παίκτες που έπαιξαν πριν από αυτόν.

1.3 Πλατφόρμα

Το παιχνίδι έχει φτιαχτεί ώστε να λειτουργεί σε PC και σε Mac. Η Unity3D δημιουργεί το εκτελέσιμο για Windows standalone player, MacOS X standalone player, πρόγραμμα πλοήγησης στο διαδίκτυο. Για την λειτουργία του σε πρόγραμμα πλοήγησης υπό την μορφή εκτελέσιμου προϋποθέτει την ύπαρξη μιας πρόσθετης εφαρμογής (plug-in) της Unity3D.

1.4 Βασικά τεχνικά χαρακτηριστικά

Τα βασικά τεχνικά χαρακτηριστικά του παιχνιδιού είναι τα εξής: το σύστημα γραφικών, το σύστημα φυσικής, το σύστημα του ήχου, το σύστημα για τον φωτισμό, το σύστημα που διαχειρίζεται την τεχνητή νοημοσύνη, το σύστημα που εισέρχονται και καταγράφονται οι βαθμολογίες των παικτών και τέλος το σύστημα για την δημιουργία του γραφικού περιβάλλοντος χρήστη.

1.4.1 Σύστημα γραφικών

Στο εν λόγω σύστημα περιλαμβάνονται όλες οι διεργασίες που απαιτούνται ώστε να γίνεται η εμφάνιση των γραφικών στο χρήστη. Περιλαμβάνονται εργαλεία για την εισαγωγή τρισδιάστατων μοντέλων, εισαγωγή εικόνων, γραμματοσειρών, χειρισμό των υφών(texture) που χρησιμοποιούμε και εφαρμογής τους στα μοντέλα που έχουν εισαχθεί, μοντελοποίηση επιφανειών ανάλογα με την διαβάθμιση του γκρίζου σε μια ασπρόμαυρη εικόνα (heightmaps) και την τοποθέτηση όλων αυτών κατά βούληση του σχεδιαστή.

1.4.2 Σύστημα φυσικής

Η Unity3D χρησιμοποιεί την NVIDIA PhysX μηχανή φυσικής, με έναν συντάκτη και με ενσωμάτωση API. Μπορούμε να καθορίσουμε τους όγκους σύγκρουσης και τους συνδέσμους βάζοντας συστατικά σε GameObjects τον συντάκτη και χρησιμοποιώντας και υλοποιώντας scripts φυσικής.

1.4.3 Σύστημα ήχου

Το σύστημα ήχου μας επιτρέπει να κάνουμε εισαγωγή στερεοφωνικών και μονοφωνικών αρχείων ήχου, με ή χωρίς διακοπή, τοποθετημένα σε συγκεκριμένο σημείο στον τρισδιάστατο χώρο, αρχεία μπορούν να παίζονται με καθυστέρηση, και με έλεγχο τόνου, έντασης, καθώς εξασθένισή του ήχου.

1.4.4 Σύστημα φωτισμού

Για να είναι ορατά και ρεαλιστικά τα γραφικά είναι απαραίτητος ο φωτισμός των μοντέλων και των υφών τους (textures). Το σύστημα φωτισμού μας επιτρέπει να τοποθετούμε τρία είδη πηγών φωτισμού: Το φωτισμό που προέρχεται από ένα σημείο και κατευθύνεται προς όλες τις κατευθύνσεις (σαν λάμπα), το φωτισμό από ένα σημείο προς μία συγκεκριμένη κατεύθυνση και φωτισμό από μια συγκεκριμένη περιοχή προς μια κατεύθυνση (όπως ο ήλιος). Μετά την χρήση των πηγών φωτισμού μπορεί να δημιουργηθεί ο λεγόμενος χάρτης φωτισμού (lightmap) -όπου είναι δυνατό-, ώστε να μειώσουμε όσο μπορούμε τους υπολογισμούς που χρειάζεται να κάνει ο κεντρικός επεξεργαστής.

1.4.5 Σύστημα τεχνητής νοημοσύνης

Το σύστημα τεχνητής νοημοσύνης χρησιμοποιείται από τους αντιπάλους του παίκτη ώστε να μπορούν να πλοηγούνται στον χώρο, να αποφεύγουν τους όμοιούς τους και να έχουν μία συγκεκριμένη συμπεριφορά που ορίζεται από τον σχεδιαστή.

1.4.6 Σύστημα αποθήκευσης και ανάκτησης δεδομένων

Είναι σημαντικό να αποθηκεύεται η βαθμολογία των παικτών ώστε να ενισχύεται ο ανταγωνισμός μεταξύ τους. Η αποθήκευση των βαθμολογιών γίνεται σε μια βάση δεδομένων που αποθηκεύει Η ανάκτησή τους γίνεται μέσω της διασύνδεσης του παιχνιδιού με αυτήν. Οι υψηλότερες βαθμολογίες εμφανίζονται στην οθόνη.

1.4.7 Σύστημα δημιουργίας του γραφικού περιβάλλοντος χρήστη

Ο χρήστης μπορεί να βλέπει σημαντικές πληροφορίες για την κατάσταση του παιχνιδιού και του χαρακτήρα του. Αυτό επιτυγχάνεται με την δημιουργία κατάλληλου γραφικού περιβάλλοντος για τον χρήστη. Το σύστημα μας παρέχει την δυνατότητα εισαγωγής κουμπιών, παράθυρων, πεδίων για κείμενο, κειμένου, μετακινητών κάθετης και οριζόντιας μπάρας παραθύρου και δημιουργία δικών μας παραμετροποιημένων εκδοχών όλων αυτών.

1.5 Δομή της εργασίας

Μετά το τρέχον -πρώτο κεφάλαιο- το οποίο αποτελεί και την εισαγωγή της διπλωματική εργασίας, ακολουθούν εφτά ακόμα κεφάλαια.

Στο δεύτερο κεφάλαιο παρουσιάζονται η σχετικές με την διπλωματική εργασία, τεχνολογίες και γνώσεις πάνω στην ανάπτυξη παιχνιδιών για υπολογιστές. Παρουσιάζονται έξι μηχανές γραφικών, και αναλύεται ο λόγος για τον οποίο επιλέξαμε την Unity3D. Η Unity3D συγκρίνεται με τη γενική βιβλιοθήκη για δημιουργία τρισδιάστατων γραφικών (OpenGL).

Στο τρίτο κεφάλαιο περιγράφονται η αρχιτεκτονική και τα συστατικά της μηχανής παιχνιδιών που χρησιμοποιήσαμε για την δημιουργία του παιχνιδιού.

Στο τέταρτο κεφάλαιο παρουσιάζεται η αρχιτεκτονική του παιχνιδιού. Περιγράφεται ο τρόπος με τον οποίο σχεδιάστηκε το παιχνίδι και γιατί. Παρουσιάζονται οι μονάδες που το αποτελούν, οι οποίες αναλύονται με τέτοιο τρόπο έτσι ώστε να γίνει κατανοητή η χρησιμότητα καθεμιάς και ο τρόπος με τον οποίο αλληλεπιδρούν μεταξύ τους.

Στο πέμπτο κεφάλαιο αναλύεται λεπτομερώς η υλοποίηση της συγκεκριμένης αρχιτεκτονικής και επίσης παραθέτονται παραδείγματα που κάνουν πιο κατανοητή τη συγκεκριμένη υλοποίηση. Ακόμα, γίνεται μια τεχνική αποτίμηση των διαφορετικών μονάδων που χρησιμοποιούνται για την υλοποίηση του παιχνιδιού.

Στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα αξιολόγησης της χρήσης του παιχνιδιού που μας βοηθάνε να αξιολογήσουμε τη σταθερότητα και την ευχρηστία του. Παρουσιάζονται οι διορθώσεις που έγιναν με βάση τις παρατηρήσεις των χρηστών.

Στο έβδομο και τελευταίο κεφάλαιο γίνεται ανακεφαλαίωση της διπλωματικής διατριβής, αναφέρονται τα συμπεράσματα της εργασίας και επισημαίνονται μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ 2
ΕΠΙΣΚΟΠΗΣΗ ΣΧΕΤΙΚΗ ΕΡΕΥΝΑΣ

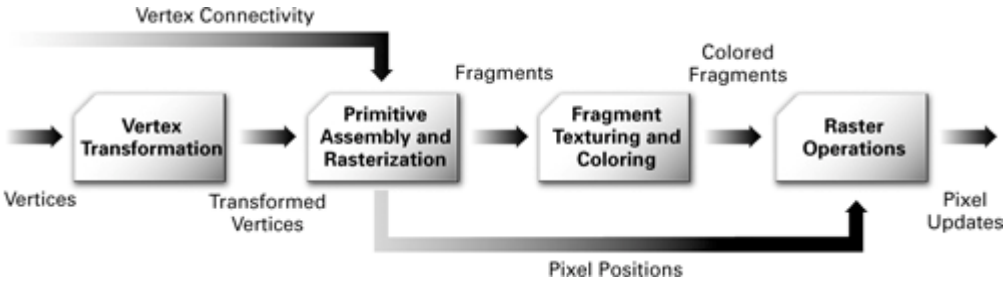
2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν γνώσεις σχετικά με τα τρισδιάστατα παιχνίδια και τις τεχνολογίες που χρησιμοποιούνται. Θα γίνει ανάλυση επιλεγμένων μηχανών γραφικών που χρησιμοποιούνται για την δημιουργία τους και θα διασαφηνιστεί ο λόγος που επιλέξαμε την συγκεκριμένη στην παρούσα εργασία.

2.2 Κορυφές, κομμάτια και διοχέτευση υλικού γραφικών

2.2.1 Διοχέτευση υλικού γραφικών (Graphics Hardware Pipeline)

Παρακάτω εμφανίζεται η διοχέτευση σε επίπεδο υλικού των γραφικών που χρησιμοποιείται από τις σύγχρονες κάρτες γραφικών (βλ. Εικόνα 1). Μια τρισδιάστατη εφαρμογή στέλνει στην μονάδα επεξεργασίας γραφικών (GPU-Graphics Processing Unit) μια σειρά απο κορυφές οι οποίες αποτελούν τα γεωμετρικά σχήματα: τυπικά πολύγωνα, γραμμές, και σημεία.



Εικόνα 1: Η διοχέτευση υλικού γραφικών

Κάθε κορυφή έχει μια θέση αλλά και διάφορες άλλες ιδιότητες όπως χρώμα, δευτεροβάθμιο (ή κατοπτρικό (specular)) χρώμα, μια ή περισσότερες ομάδες συντεταγμένων των υφών του, και ένα κανονικό (normal) διάνυσμα. Το κανονικό διάνυσμα προσδιορίζει ποια είναι η κατεύθυνση της επιφάνειας των κορυφών, και τυπικά χρησιμοποιείται στους υπολογισμούς για τον φωτισμό.

Μετασχηματισμός κορυφών

Ο μετασχηματισμός κορυφών είναι το πρώτο στάδιο επεξεργασίας στην διοχέτευση υλικού γραφικών. Εκτελείται μια ακολουθία μαθηματικών διαδικασιών σε κάθε κορυφή. Αυτές οι διαδικασίες περιλαμβάνουν το μετασχηματισμό της θέσης της κορυφής σε μια θέση της οθόνης, για χρήση από την μονάδα που κάνει απεικόνιση διανυσματικών γραφικών. Αυτό γίνεται με χρήση εικονοστοιχείων, που παράγουν τις συντεταγμένες για εφαρμογή των υφών στην επιφάνεια και έπειτα φωτίζεται η κορυφή για να καθοριστεί το χρώμα της.

Πρωταρχική συναρμολόγηση και απεικόνιση

Οι μετασχηματισμένες κορυφές προχωρούν σειριακά στο επόμενο στάδιο, το οποίο ονομάζεται πρωταρχική συναρμολόγηση και απεικόνιση (primitive assembly and rasterization). Πρώτα, συναθροίζονται οι κορυφές και σχηματίζουν γεωμετρικά σχήματα βασισμένα στις πληροφορίες που έχουν οι κορυφές. Αυτό έχει ως αποτέλεσμα μια ακολουθία απο τρίγωνα, γραμμές, ή σημεία. Αυτά τα γεωμετρικά σχήματα ενδέχεται να απαιτήσουν περικοπή στην ορατή περιοχή του τρισδιάστατου χώρου (view frustum). Ο απεικονιστής (rasterizer) μπορεί επίσης να απορρίψει τα πολύγωνα ελέγχοντας εάν οι επιφάνειές του βλέπουν προς τα εμπρός ή προς τα πίσω.

Αυτή η διαδικασία είναι γνωστή ως ξεδιάλεγμα (culling). Τα πολύγωνα που επιζούν αυτών των βημάτων περικοπής και ξεδιαλέγματος πρέπει να **γίνει απεικόνιση διανυσματικών γραφικών με χρήση εικονοστοιχείων (rasterization)**. Η απεικόνιση (rasterization) είναι η διαδικασία που γίνεται για να καθοριστεί το σύνολο των εικονοστοιχείων που καλύπτονται απο ένα γεωμετρικό σχήμα. Τα πολύγωνα, οι γραμμές, και τα σημεία απεικονίζονται με χρήση εικονοστοιχείων, σύμφωνα με τους κανόνες που έχουν οριστεί για το κάθε ένα τύπο. Τα αποτελέσματα της απεικόνισης (rasterization) είναι ένα σύνολο θέσεων εικονοστοιχείου καθώς επίσης και ένα σύνολο που ορίζουν ένα **κομμάτι (fragment)**. Δεν υπάρχει καμία σχέση μεταξύ του αριθμού των κορυφών που ένα γεωμετρικό σχήμα έχει και με τον αριθμό κομματιών που παράγονται όταν γίνεται απεικόνιση (rasterization). Παραδείγματος χάριν, ένα τρίγωνο φτιαγμένο πάνω από ακριβώς τρεις κορυφές θα μπορούσε να καταλάβει ολόκληρη οθόνη, και επομένως να παραγάγει εκατομμύρια κομμάτια.

Η διάκριση μεταξύ ενός κομματιού και ενός εικονοστοιχείου είναι σημαντική. Ένα εικονοστοιχείο αντιπροσωπεύει τα περιεχόμενα μιας μνήμης αποθήκευσης καρέ σε μια συγκεκριμένη θέση, όπως το χρώμα, το βάθος, και οποιεσδήποτε άλλες τιμές που συνδέονται με εκείνη την θέση. Ένα κομμάτι είναι η κατάσταση που ενδεχομένως χρειάζεται για να ενημερώσει ένα συγκεκριμένο εικονοστοιχείο.

Ο όρος *κομμάτι (fragment)* χρησιμοποιείται επειδή η απεικόνιση (rasterization) χωρίζει κάθε γεωμετρικό σχήμα, για παράδειγμα ένα τρίγωνο, σε μικρά κομμάτια για κάθε εικονοστοιχείο που καλύπτει το σχήμα. Ένα κομμάτι έχει για κάθε εικονοστοιχείο μια σχετική θέση, μια τιμή βάθους, και ένα σύνολο παραμέτρων όπως χρώμα, κατοπτρικό (specular) χρώμα, και ένα ή περισσότερα σύνολο συντεταγμένων υφής. Αυτές οι παράμετροι προέρχονται από μετασχηματισμένες κορυφές που απαρτίζουν το ιδιαίτερο γεωμετρικό σχήμα που χρησιμοποιείτε για να παράγουν τα κομμάτια. Μπορούμε να σκεφτούμε κάθε κομμάτι ως ένα "πιθανό εικονοστοιχείο".

Εάν ένα κομμάτι περνά τις διάφορες δοκιμές απεικόνισης (rasterization) τότε ενημερώνει ένα εικονοστοιχείο στην μνήμη αποθήκευσης καρέ.

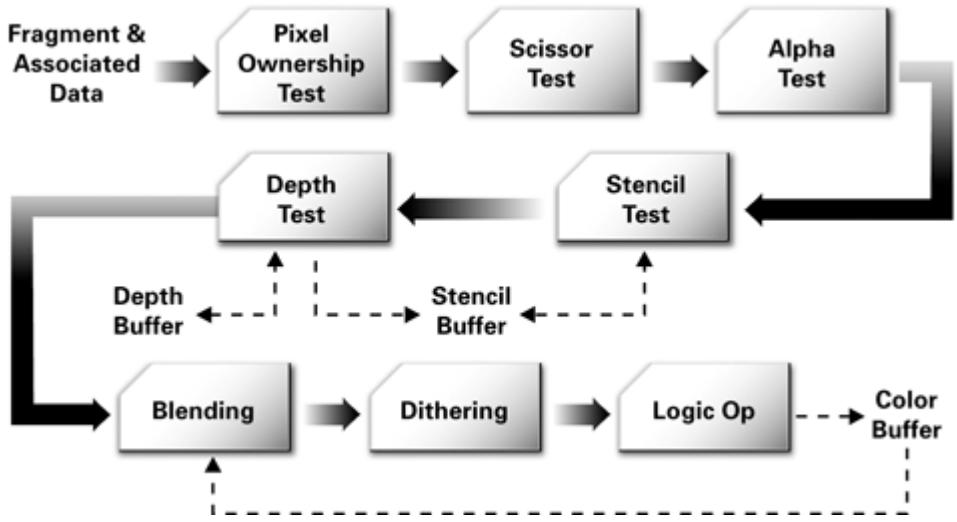
Παρεμβολή, υφές και χρωματισμός

Όταν ένα σχήμα έχει απεικονιστεί σε μια συλλογή απο μηδέν ή περισσότερων κομματιών, το στάδιο της παρεμβολής, των υφών, και του χρωματισμού, παρεμβάλλει τις παραμέτρους των κομματιών, όπου κρίνεται αναγκαίο, εκτελεί μια ακολουθία εφαρμογής υφών και μαθηματικών διαδικασιών, και καθορίζει το τελικό χρώμα για κάθε κομμάτι. Εκτός από τον καθορισμού του τελικού χρώματος του κομματιού, αυτό το στάδιο μπορεί επίσης να καθορίσει ένα νέο βάθος ή μπορεί ακόμη και να απορρίψει ένα κομμάτι για να αποφύγει την ανανέωση του αντίστοιχου εικονοστοιχείου στη μνήμη αποθήκευσης καρέ. Επιτρέποντας το στάδιο αυτό να μπορεί να απορρίψει ένα κομμάτι, εκπέμπει ένα ή κανένα χρωματισμένα κομμάτια για κάθε κομμάτι που λαμβάνει απο την είσοδο.

Διαδικασίες απεικόνισης (Rasterization)

Το στάδιο απεικόνισης διαδικασιών εκτελεί μια τελική ακολουθία διαδικασιών ανά-κομμάτι αμέσως πριν ενημερώνει την μνήμη αποθήκευσης καρέ. Αυτές οι διαδικασίες είναι ένα τυποποιημένο μέρος των βιβλιοθηκών OpenGL και Direct3D. Κατά τη διάρκεια αυτού του σταδίου, οι κρυμμένες επιφάνειες αποβάλλονται μέσω μιας διαδικασίας γνωστής ως *δοκιμή βάθους (depth testing)* . Άλλα αποτελέσματα, όπως η ανάμειξη (blending) και η σκίαση βασισμένη στο ***stencil***, εμφανίζονται επίσης κατά τη διάρκεια αυτού του σταδίου.

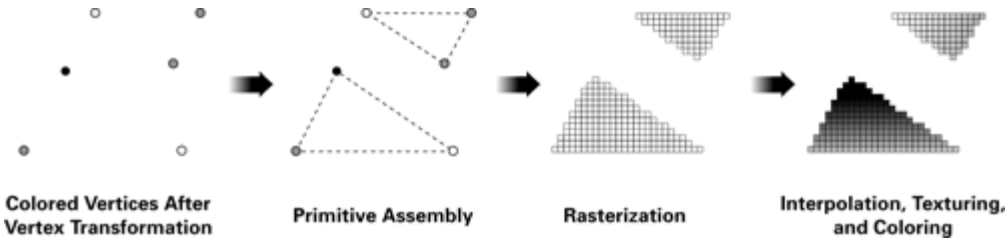
Το στάδιο απεικόνισης διαδικασιών ελέγχει κάθε κομμάτι βασισμένο σε διάφορες δοκιμές, του ψαλιδιού (scissor test), άλφα (alpha test), του stencil, και των δοκιμών βάθους (depth test). Αυτές οι δοκιμές περιλαμβάνουν το τελικό χρώμα ή το βάθος του κομματιού, τη θέση εικονοστοιχείου, και τις ανά-εικονοστοιχείο τιμές όπως του βάθους και του stencil. Εάν οποιαδήποτε δοκιμή αποτύχει, αυτό το στάδιο απορρίπτει να κάνει ενημέρωση στην τιμή χρώματος του εικονοστοιχείου. Περνώντας την δοκιμή βάθους μπορεί να αντικαταστήσει την τιμή βάθους του εικονοστοιχείου τιμή βάθους του κομματιού. Μετά από τις δοκιμές, μια λειτουργία ανάμειξης (blending) συνδυάζει το τελικό χρώμα του κομματιού με την αντιστοιχία της τιμής χρώματος του εικονοστοιχείου. Τέλος, μια μνήμη αποθήκευσης καρέ γράφει ότι η λειτουργία αντικαθιστά το χρώμα του εικονοστοιχείου με το αναμεμειγμένο χρώμα (βλ. Εικόνα 2) .



Εικόνα 2: Τυπικές διαδικασίες απεικόνισης OpenGL και Direct3D

Σχηματική απεικόνιση της διοχέτευσης γραφικών (Graphics Pipeline)

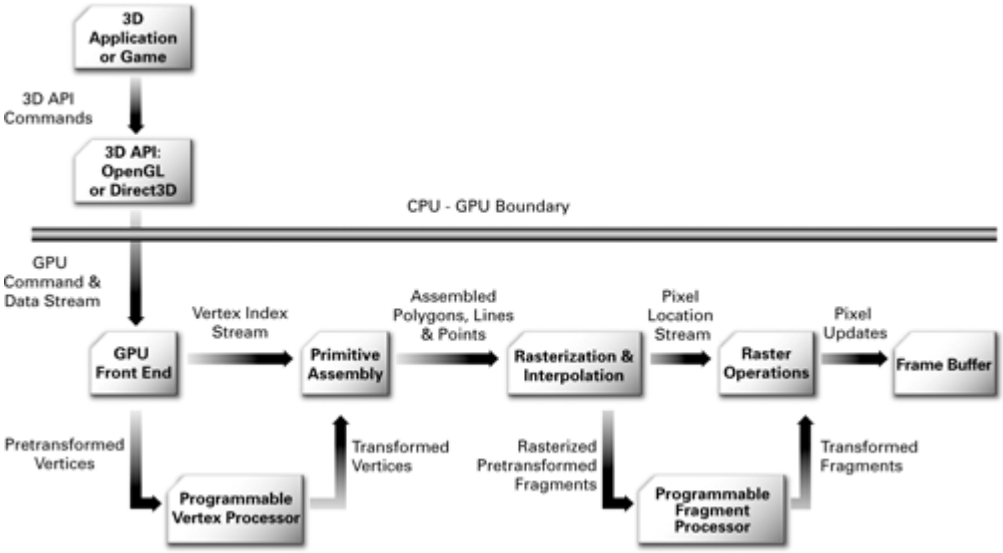
Παρακάτω φαίνονται τα στάδια της διοχέτευσης. Στην Εικόνα 3, δυο τρίγωνα απεικονίζονται. Η διαδικασία αρχίζει με το μετασχηματισμό και τον χρωματισμό των κορυφών. Έπειτα, το βήμα που γίνεται η συνάθροιση των σχημάτων δημιουργεί τα τρίγωνα απο τις κορυφές όπως υποδεικνύουν οι διακεκομμένες γραμμές. Μετά απο αυτό, ο απεικονιστής «γεμίζει» τα τρίγωνα με κομμάτια. Τέλος, οι καταχωρημένες τιμές απο τις κορυφές παρεμβάλλονται, χρησιμοποιούνται υφές και χρωματίζονται. Αξίζει να προσέξουμε πόσα πολλά κομμάτια παράγονται απο μερικές κορυφές.



Εικόνα 3: Σχηματική απεικόνιση της διοχέτευσης γραφικών

2.2.2 Προγραμματιζόμενη διοχέτευση γραφικών

Η κυρίαρχη τάση στο σχεδιασμού του υλικού των γραφικών σήμερα είναι να ενσωματωθεί μεγαλύτερη δυνατότητα προγραμματισμού των δυνατοτήτων την μονάδα επεξεργασίας γραφικών (GPU). Στην Εικόνα 4 φαίνονται τα στάδια επεξεργασίας των κορυφών και κομματιών στην διοχέτευση μιας προγραμματιζόμενης GPU.

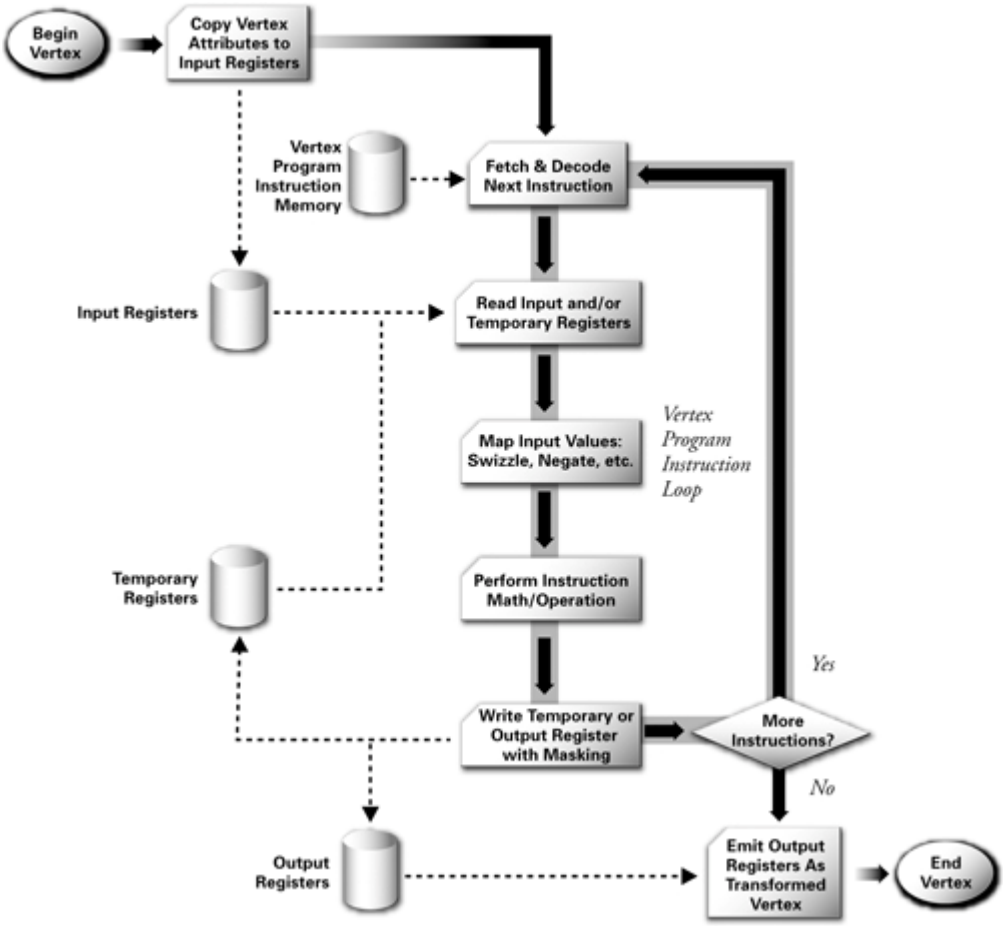


Εικόνα 4: Η προγραμματιζόμενη διοχέτευση γραφικών

Στην Εικόνα 4 φαίνεται ότι οι επεξεργασίες των κορυφών και κομματιών έχουν διασπαστεί σε προγραμματιζόμενες μονάδες. Ο προγραμματιζόμενος επεξεργαστής κορυφών είναι η μονάδα υλικού που τρέχει τα προγράμματα Cg (**C** for **g**raphics) κορυφών, ενώ ο προγραμματιζόμενος επεξεργαστής κομματιών είναι η μονάδα που τρέχει τα Cg προγράμματα κομματιών.

Προγραμματιζόμενος Επεξεργαστής Κορυφών

Στην Εικόνα 5 φαίνεται ένα διάγραμμα ροής για ένα τυπικό προγραμματιζόμενο επεξεργαστή κορυφών. Το μοντέλο ροής-δεδομένων για την επεξεργασία κορυφών ξεκινά με την εισαγωγή των ιδιοτήτων των κορυφών (θέση, χρώμα, συντεταγμένες υφών κ.ά.) στον επεξεργαστή κορυφών. Έπειτα, ο επεξεργαστής κορυφών επανειλημμένα φέρνει μια νέα εντολή και την εκτελεί μέχρι το πρόγραμμα κορυφών να τερματιστεί. Οι εντολές έχουν πρόσβαση σε διάφορες διακριτές ομάδες από τράπεζες καταχωρητών που περιλαμβάνουν τις τιμές των διανυσμάτων, όπως θέσης, κανονικής θέσης (normal), ή χρώματος. Οι ιδιότητες των καταχωρητών κορυφών είναι μόνο για ανάγνωση και περιλαμβάνουν ορισμένες ομάδες ιδιοτήτων για τις κορυφές, ειδικώς για την εφαρμογή. Οι προσωρινοί καταχωρητές μπορούν να διαβαστούν, να εγγραφούν και χρησιμοποιούνται για να υπολογίσουν ενδιάμεσα αποτελέσματα. Τα αποτελέσματα των καταχωρητών είναι μόνο για εγγραφή. Το πρόγραμμα είναι υπεύθυνο για το γράψιμο των αποτελεσμάτων σε αυτούς τους καταχωρητές. Όταν το πρόγραμμα των κορυφών τερματιστεί, το αποτέλεσμα των καταχωρητών εξόδου περιλαμβάνει τις καινούριες μετασχηματισμένες κορυφές. Μετά την εγκατάσταση και την απεικόνιση του τριγώνου, οι παρεμβαλλόμενες τιμές για κάθε καταχωρητή περνάνε στον επεξεργαστή κομματιών.



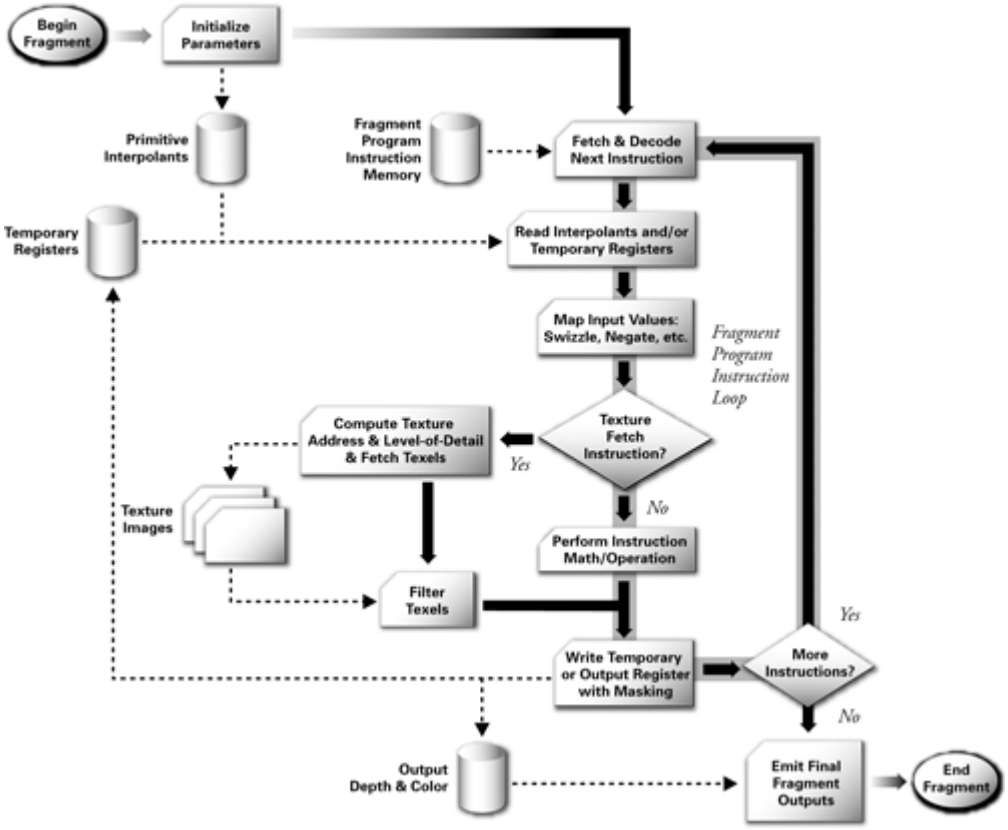
Εικόνα 5:Διάγραμμα ροής για τον προγραμματιζόμενο επεξεργαστή κορυφών

Προγραμματιζόμενος Επεξεργαστής Κομματιών

Ο προγραμματιζόμενος επεξεργαστής κομματιών απαιτεί πολλές απο τις ίδιες μαθηματικές διαδικασίες που θέλουν και οι επεξεργαστές κορυφών. Η διαφορά τους είναι ότι υποστηρίζει διαδικασίες για την εφαρμογή υφών. Οι διαδικασίες για την εφαρμογή υφών δίνουν την δυνατότητα στον επεξεργαστή να έχει πρόσβαση στην εικόνα υφής χρησιμοποιώντας τις συντεταγμένες, και έπειτα, να επιστρέφει ένα φιλτραρισμένο δείγμα της εικόνας υφής.

Στην παρακάτω εικόνα φαίνεται ένα διάγραμμα ροής για ένα σύγχρονο προγραμματιζόμενο επεξεργαστή κομματιών. Όπως ο επεξεργαστής κορυφών, η ροή των δεδομένων περιλαμβάνει την εκτέλεση μια σειράς απο εντολές μέχρι να τερματιστεί το πρόγραμμα. Και εδώ, υπάρχει μια ομάδα απο καταχωρητές εισόδου. Ωστόσο, σε αντίθεση με τις ιδιότητες των κορυφών, οι καταχωρητές εισόδου του επεξεργαστή κομματιών περιλαμβάνουν παρεμβαλλόμενες παραμέτρους για κάθε κομμάτι που έχει οριστεί απο πριν και προήλθε απο τις παραμέτρους των κορυφών που αποτελούν ένα σχήμα. Υπάρχουν προσωρινοί καταχωρητές για ανάγνωση και εγγραφή των ενδιάμεσων τιμών. Οι διαδικασίες της εγγραφής για τους καταχωρητές εξόδου περιλαμβάνουν το χρώμα και μερικές φορές την νέα τιμή βάθους του κάθε

κομματιού. Οι εντολές για τα προγράμματα των κομματιών περιλαμβάνουν και ενώσεις με την υφή.



Εικόνα 6: Διάγραμμα ροής για τον προγραμματιζόμενο επεξεργαστή κομματιού

2.3 Όροι και έννοιες

2.3.1 Φυσική (Physics)

Οι νόμοι φυσικής για ηλεκτρονικά παιχνίδια (game physics) περιλαμβάνουν την εισαγωγή των αρχών των νομών της φυσικής σε μια προσομοίωση ή σε μια μηχανή παιχνιδιών, ειδικά στα τρισδιάστατα γραφικά υπολογιστών, με σκοπό να κάνουν τις τεχνικές εντυπωσιασμού (effects) να φαίνονται ρεαλιστικότερα. Τυπικά, η προσομοίωση φυσικής είναι η μόνη κοντινή διαδικασία προσέγγισης της επιστημονικής φυσικής και ο υπολογισμός της γίνεται χρησιμοποιώντας διακριτές τιμές.

2.3.2 Φωτισμός

Όταν λέμε φωτισμό εννοούμε την προσομοίωση του φωτός με γραφικά υπολογιστών. Αυτή η προσομοίωση μπορεί να είναι εκπληκτικά ακριβής όπως σε εφαρμογές που αναζητάνε την ενέργεια της ροής του φωτός καθώς αυτό αλληλεπιδρά με διάφορα υλικά στο τρισδιάστατο χώρο. Εναλλακτικά, η προσομοίωση μπορεί απλά να επηρεαστεί από την φυσική του φωτός σαν την περίπτωση όπου γίνεται μη

φωτορεαλιστική σχεδίαση (render). Και στις δυο περιπτώσεις ένα φωτοσκιασμένο μοντέλο (shading model) χρησιμοποιείται για να αποδώσει το τρόπο που οι επιφάνειες ανταποκρίνονται στο φως. Μεταξύ αυτών των δυο άκρων, υπάρχουν πολλές διαφορετικές προσεγγίσεις σχεδίασης οι οποίες μπορούν να αποδοθούν ώστε να πετύχουμε σχεδόν όποιο οπτικό αποτέλεσμα επιθυμούμε.

Η χαρτογράφιση φωτός (lightmapping) είναι μια τεχνική που χρησιμοποιείται στα ηλεκτρονικά παιχνίδια για την μείωση των υπολογισμών του φωτός σε πραγματικό χρόνο. Χάρτης φωτός, ονομάζεται μια δομή δεδομένων που περιλαμβάνει την φωτεινότητα των επιφανειών. Υπολογίζονται απο πριν και χρησιμοποιούνται σε στατικά αντικείμενα. Οι πιο συνηθισμένες μέθοδοι χαρτογράφησης φωτός είναι είτε να υπολογίζεται απο πριν ο φωτισμός των γωνιών χρησιμοποιώντας την απόσταση της κάθε γωνίας απο την πηγή φωτισμού, είτε χρησιμοποιώντας πολυχαρτογράφηση και εφαρμόζοντας δεύτερη υφή που θα περιέχει τα φωτιστικά δεδομένα.

2.3.3 Υφές (Texturing)

Η χαρτογράφηση υφών (texture mapping) είναι μια μέθοδος για να αποδώσουμε λεπτομέρεια η να χρωματίσουμε ένα γραφικό φτιαγμένο από υπολογιστή η ένα τρισδιάστατο μοντέλο.

Υπάρχουν πολλά είδη χαρτογράφησης υφών. Παρακάτω αναφέρονται κάποια από τα βασικότερα και ευρέως χρησιμοποιούμενα.

Πολυχαρτογράφηση (multitexturing) ονομάζεται η χρησιμοποίηση περισσότερων από μια υφές την φορά πάνω σε ένα πολύγωνο. Για παράδειγμα, ένας χάρτης φωτός (light map) χρησιμοποιημένος σαν υφή μπορεί να χρησιμοποιηθεί για να φωτίσει μια επιφάνεια σαν μια εναλλακτική για να επαναϋπολογίζει τον φωτισμό κάθε φορά που θα σχεδιάζεται αυτή η επιφάνεια.

Άλλη μια τεχνική πολυχαρτογράφησης είναι η χαρτογράφιση ανώμαλης επιφάνειας (bump mapping ή normal mapping) , το οποίο επιτρεπεί μια υφή κατευθείαν να ελέγχει την επίστρωση μιας επιφάνειας με σκοπό των υπολογισμό του φωτισμού. Αυτό μπορεί να δώσει πολύ καλή εμφάνιση σε πολύπλοκες επιφάνειες όπως είναι ένας κορμός δέντρου η το τραχύ τσιμέντο, και έτσι να δώσει φωτιστικές λεπτομέρειες σε συνδυασμό με το συνηθισμένο λεπτομερή χρωματισμό. Η χαρτογράφηση ανώμαλης επιφάνειας έγινε διάσημη σε ηλεκτρονικά παιχνίδια που σχεδιάστηκαν πρόσφατα καθώς οι κάρτες γραφικών έγιναν αρκετά δυνατές ώστε να μπορούν να το υποστηρίζουν.

2.3.4 Σκιαστές (Shaders)

Ένας σκιαστής (shader) στο τομέα των γραφικών με υπολογιστή είναι ένα σετ από εντολές λογισμικού, οι οποίες χρησιμοποιούνται από τους γραφικούς πόρους κυρίως για να εκτελέσουν τα σχεδιαστικά εφφέ. Οι σκιαστές (shaders) χρησιμοποιούνται για να βοηθήσουν τον σχεδιαστή μιας τρισδιάστατης εφαρμογής να προγραμματίσει την μονάδα επεξεργασίας γραφικών (graphics processing unit (GPU)), η οποία έχει σχεδόν αντικαταστήσει το παλαιότερο "fixed-function pipeline", αφήνοντας

μεγαλύτερη ευελιξία στο να γίνεται χρησιμοποίηση των προχωρημένων προγραμματιζόμενων χαρακτηριστικών στις μονάδες επεξεργασίας γραφικών.

2.3.5 Κίνηση (Animation)

Η κίνηση αναφέρεται στην προσωρινή θέση-περιγραφή ενός αντικειμένου. Για παράδειγμα, πως αυτό μετακινείται και πως παραμορφώνεται στο πέρασμα του χρόνου. Δημοφιλείς μέθοδοι-τεχνικές είναι το key framing, η αντίστροφη κινηματική (inverse kinematics, IK) και η σύλληψη κίνησης (motion capture, MoCap). Ωστόσο, πολλές από αυτές τις τεχνικές χρησιμοποιούνται πολλές φορές και σε συνδυασμό μεταξύ τους.

2.3.6 Δίκτυα (Meshes)

Τα δίκτυα (meshes) είναι μια συλλογή από κορυφές, ακμές και επιφάνειες που ορίζουν μια μορφή ενός πολυεδρικού αντικειμένου στα τρισδιάστατα γραφικά και στα μοντέλα στερεών αντικειμένων. Οι επιφάνειες τους συνήθως απαρτίζονται από τρίγωνα, τετράπλευρα και άλλα απλά κυρτά πολύγωνα.

2.3.7 Σύγκριση και ανάλυση μηχανών γραφικών

Με μία πιο προσεκτική ματιά στις υπάρχουσες μηχανές παιχνιδιών που κυκλοφορούν μπορούμε να δούμε ότι υπάρχει μια πληθώρα από αυτές. Είτε είναι ελεύθερες στην μεταφόρτωση και χρησιμοποίησή τους, είτε θα πρέπει οι σχεδιαστές παιχνιδιών να αγοράσουν την απαιτούμενη άδεια για την χρησιμοποίησή τους. Ακολουθεί μια παρουσίαση μερικών από αυτών:

Source Engine

Από τις πιο γνωστές μηχανές γραφικών που υπάρχουν στις μέρες μας. Βρίσκεται σε συνεχή βελτίωση από την Valve Software και λέγεται ότι είναι μια από τις καλύτερες μηχανές γραφικών που κυκλοφορούν για PC. Έχει χρησιμοποιηθεί για το σχεδιασμό πολύ γνωστών παιχνιδιών όπως το Half-life 2, το Portal και το Team Fortress 2. Είναι βελτιστοποιημένη για να υποστηρίζει υψηλού επιπέδου πρώτου προσώπου (FPP) παιχνίδια. Στην παρούσα έκδοσή της υποστηρίζει Windows, Playstation 3, και Xbox πλατφόρμα και απαιτεί πολλά χρήματα για την απόκτηση της άδειας χρησιμοποίησή της. Οπότε ουσιαστικά είναι διαθέσιμη μόνο για μεγάλα studios παιχνιδιών [2] .

FIFE engine

Σε αντίθεση με την Source, ο στόχος της FIFE είναι να αναπτύξει μια βέλτιστη μηχανή παιχνιδιών για δυσδιάστατα παιχνίδια με ισομετρική προοπτική, όπως αυτή η προοπτική που χρησιμοποιείται κατά κόρον στα παιχνίδια στρατηγικής (Warcraft, Settlers κ.ά.). Ο κώδικας της μηχανής είναι ακόμα σε πρώιμο beta στάδιο και είναι

διαθέσιμη κάτω από την GPL άδεια που σημαίνει ότι μπορεί ο οποιοσδήποτε να μεταφορτώσει τον κώδικα δωρεάν και να συνεισφέρει στην βελτίωσή της. Την στιγμή που το API είναι σε συνεχή τροποποίηση είναι δύσκολο να καθορίσουμε πόσο εύκολο είναι να χρησιμοποιηθεί το ολοκληρωμένο προϊόν από τους σχεδιαστές παιχνιδιών. Η FIFE έχει σχεδιαστεί για Windows και Linux πλατφόρμες [3] .

Εικόνα 7: Παραδείγματα από παιχνίδια που κατασκευάστηκαν από την Source engine (Half-life 2, αριστερά) και FIFE project Group (δεξιά).

Spring

Η Spring είναι διαφορετική απο τις άλλες μηχανές γραφικών. Στην Spring χρειάζεται η χρήση της LUA (scripting γλώσσα). Χρησιμοποιείται για να προγραμματιστεί όλο το gameplay του παιχνιδιού με την δυνατότητα να εισάγουμε τα δικά μας μοντέλα χαρακτήρων και υφές. Ενώ είναι εύκολη στην χρήση της το κύριο πρόβλημα είναι ότι είναι σχεδιασμένη για παιχνίδια στρατηγικής πραγματικού χρόνου και αυτό περιορίζει την εφαρμογή σε ένα είδος gameplay που δεν συνάδει με το δικό μας [4] .

Εικόνα 8: Παράδειγμα παιχνιδιού φτιαγμένο στην Spring μηχανή γραφικών

Panda3D

Η μηχανή γραφικών Panda3D, από τα αρχικά του Platform Agnostic Networked Display Architecture, είναι μια ισχυρή μηχανή για σχεδίαση σε SGI, Linux, Sun, και Windows [6]. Αρχικά, αναπτύχθηκε από τα Studios εικονικής πραγματικότητας της Walt Disney και χρησιμοποιήθηκε για να υλοποιήσουν το MMOG παιχνίδι Toontown Online. Από τότε έχει δοθεί στην κοινότητα και είναι ανοικτού κώδικα. Ο πυρήνας

της μηχανής γραφικών είναι γραμμένος σε C++. Η Panda3D/ DIRECT παρέχει διεπαφή για Python scripting.

Η Panda3D είναι μια μηχανή γραφικών σκηνής. Αυτό σημαίνει ότι ο εικονικός κόσμος είναι αρχικά ένας άδειος καρτεσιανός χώρος που μέσα σε αυτόν οι σχεδιαστές βάζουν τρισδιάστατα μοντέλα.

Όταν σχεδιάστηκε στην αρχή δεν υπήρχαν οι σκιαστές εικονοστοιχείο και κορυφής. Με προγραμματιζόμενους από χρήστες σκιαστές κατάφερε να τους υποστηρίζει απο το 2005. Οι προγραμματιστές που αναπτύσσουν την Panda3D πιστεύουν ότι επειδή ο προγραμματισμός των σκιαστών είναι αρκετά δύσκολος, πολλοί σχεδιαστές παιχνιδιών θέλουν να τους χειρίζεται η μηχανή γραφικών αυτόματα.

Έτσι, οι προγραμματιστές της Panda3D έχουν δώσει πρόσφατα την δυνατότητα στους χρήστες της να συνθέτουν κάποιους σκιαστές αυτόματα. Αυτή η σύνθεση συμβαίνει όταν ο σχεδιαστής δημιουργήσει σε ένα πρόγραμμα τρισδιάστατης σχεδίασης το μοντέλο και εφαρμόσει ένα υλικό για την δημιουργία χαρτών ανώμαλης επιφάνειας, ένα γυαλιστερό υλικό (glossy), ένα αυτοφωτιζόμενο υλικό (self-illumination) ή μια άλλη δυνατότητα που υπερβαίνει το fixed-function pipeline. Κατά την εισαγωγή στην σκηνή το μοντέλο θα φαίνεται όπως ο σχεδιαστής το έχει σχεδιάσει στο πρόγραμμα τρισδιάστατης σχεδίασης, χωρίς καθόλου μεσολάβηση του προγραμματιστή.

Όντας μηχανή παιχνιδιών και όχι μηχανή γραφικών παρέχει και άλλες δυνατότητες εκτός από την τρισδιάστατη απεικόνιση [6] . Μερικές από τις κυριότερες είναι οι εξής:

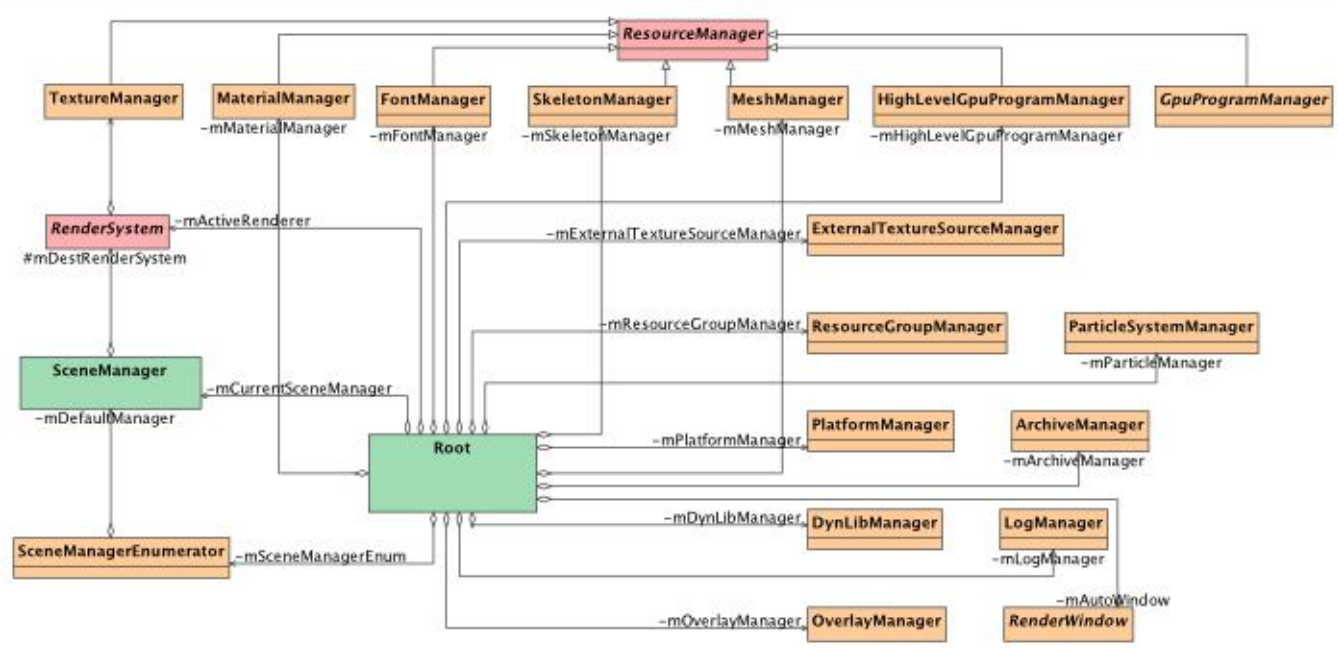
- Εργαλεία ανάλυσης της απόδοσης
- Εργαλεία εξερεύνησης του γράφου σκηνής
- Εργαλεία για αποσφαλμάτωση
- Μια πλήρης διοχέτευση για εισαγωγή και εξαγωγή αρχείων.
- Τρισδιάστατο ήχο, χρησιμοποιώντας είτε το FMOD, OpenAL ή Miles Sound System
- Εντοπισμός συγκρούσεων
- Σύστημα φυσικής και πλήρη ανάμειξη με την Open Dynamics Engine
- Υποστήριξη πληκτρολογίου και ποντικιού
- Υποστήριξη για ασυνήθιστες συσκευές εισόδου και εξόδου
- Μηχανές πεπερασμένων καταστάσεων

Εικόνα 9: Παραδείγματα από παιχνίδια που έχουν δημιουργηθεί από την Disney με την Panda3D (Disney's Pirates of the Caribbean αριστερά και Toontown Online δεξιά) .

Ogre3D

Η OGRE (*Object-Oriented Graphics Rendering Engine*) είναι μια ευέλικτη τρισδιάστατη σχεδιαστική μηχανή γραμμένη σε C++, προσανατολισμένη σύμφωνα με την σκηνή και σχεδιασμένη, για να γίνεται εύκολα κατανοητή και να χρησιμοποιηθεί από σχεδιαστές προγραμμάτων ώστε να παράγουν εφαρμογές εκμεταλλευόμενοι το υλικό των τρισδιάστατατων γραφικών. Η βιβλιοθήκη των κλάσεων συνοψίζει τις λεπτομέρειες χρησιμοποιώντας τις θεμελιώδεις βιβλιοθήκες Direct3D και OpenGL και παρέχει μια διασύνδεση βασισμένη στα αντικείμενα του κόσμου και άλλες κλάσεις υψηλού επιπέδου. Η OGRE έχει μια πολύ ενεργή κοινότητα.

Όπως λεει και το όνομα της, η OGRE είναι «απλά» μια σχεδιαστική μηχανή. Στην πραγματικότητα, η κύρια επιδίωξη των σχεδιαστών της είναι να παρέχει μια γενική λύση για το σχεδιασμό των γραφικών του υπολογιστή. Παρότι έρχεται και με άλλες δυνατότητες (διανυσματικές κλάσεις πινάκων, χειρισμό μνήμης), αυτές θεωρούνται συμπληρωματικές. Δεν είναι μια ολοκληρωμένη λύση για ανάπτυξη παιχνιδιών ή προσομοίωση, καθώς δεν έχει, για παράδειγμα, υποστήριξη ήχου και φυσικής. Γενικά, αυτό θεωρείται ένα από τα κύρια μειονεκτήματα της OGRE και ο λόγος ο οποίος δεν επιλέχθηκε από εμάς.



Εικόνα 10: Αρχιτεκτονική της μηχανής Ogre3D

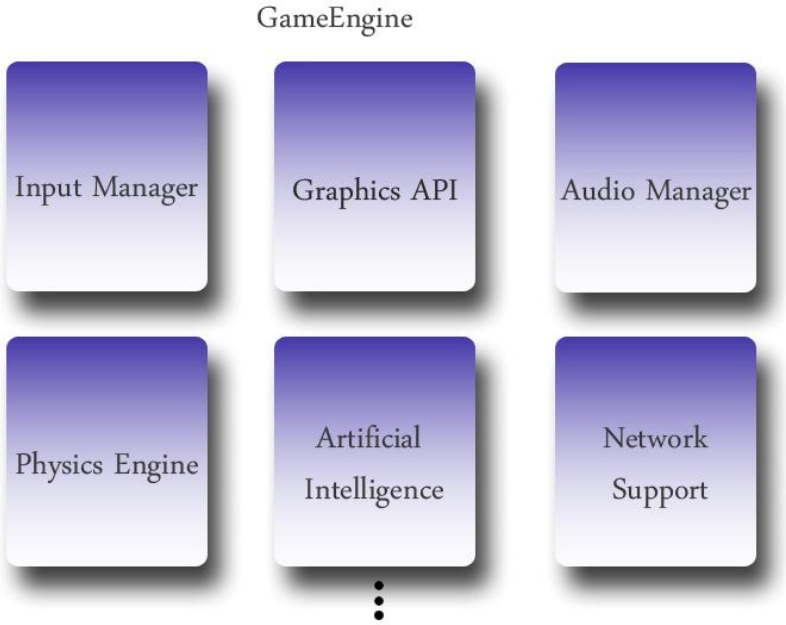
Το διάγραμμα κλάσεων δείχνει την «κύρια» κλάση (Root) της OGRE και όλες τις κλάσεις που γίνεται η διαχείριση και δίνουν πρόσβαση σε διαφορετικά υποσυστήματα. Η «κύρια» (Root) κρατάει αναφορά για κάθε «Διαχειριστή Σκηνής» (SceneManager) και ένα μετρητή που αφήνει τους άλλους διαχειριστές των γραφημάτων σκηνής να φορτώσουν. Το «Σύστημα Σχεδίασης» (RenderSystem) είναι μια αφηρημένη κλάση που διαχωρίζει την «κύρια» κλάση (Root) από την συγκεκριμένη υλοποίηση του, OpenGL ή Direct3D. Ο «Διαχειριστής Πόρων» (ResourceManager) είναι μια αφηρημένη κλάση που είναι υποκλάση ενός αριθμού διαχειριστών που ελέγχουν τους πόρους όπως είναι οι υφές (textures), τα υλικά (materials) και οι γραμματοσειρές (fonts). Όλες οι κλάσεις στο διάγραμμα εκτός από το «Σύστημα Σχεδίασης», «Διαχειριστή Σκηνής» και «Σχεδιαστή Παραθύρου» χρησιμοποιούν μια ξεχωριστή φόρμα για να σιγουρευτεί ότι υπάρχει μόνο ένα στιγμιότυπο από κάθε μια κλάση.

Εικόνα 11: Παραδείγματα παιχνιδιών αναπτυγμένα με την βοήθεια της Ogre3D (The age of Racing δεξιά και Zero Gear αριστερά) .

2.4 Γιατί δεν χρησιμοποιήθηκε γενική βιβλιοθήκη για τρισδιάστατα γραφικά

Για να μπορέσουμε να απαντήσουμε σε αυτήν την βασική ερώτηση θα πρέπει πρώτα να ορίσουμε τι είναι το OpenGL (γενική βιβλιοθήκη για τρισδιάστατα γραφικά). Το OpenGL είναι ένα πρότυπο που ορίζει μια γλώσσα για να γράφουμε εφαρμογές που παράγουν δυσδιάστατα και τρισδιάστατα γραφικά. Η διεπαφή αποτελείται από περίπου 250 διαφορετικές συναρτήσεις μηχανής οι οποίες μας επιτρέπουν να σχεδιάσουμε πολύπλοκες τρισδιάστατες σκηνές από σημεία και απλά πρωτογενή σχήματα. Μας δίνει την δυνατότητα να έχουμε αρκετά χαμηλού επιπέδου πρόσβαση στις τεχνολογίες των καρτών γραφικών και να γράφουμε τα πάντα μόνοι μας δημιουργώντας σημεία στον χώρο και ορίζοντας επιφάνειες μεταξύ τους. Προσφέρει στον χρήστη μερικές βασικές συναρτήσεις μαθηματικών μετασχηματισμών για διανύσματα και πίνακες οι οποίες είναι απαραίτητες για να πετύχουμε μετακίνηση ή περιστροφή στην οθόνη. Το OpenGL είναι γνωστό για την δυνατότητα ανεξαρτησίας από την πλατφόρμα που εκτελείται. Μπορεί να κάνει σχεδιασμό σχεδόν σε όλες τις αρχιτεκτονικές συστημάτων, όπως Linux, Windows και MacOS. Φημολογείται ότι έχει μία από τις καλύτερες τεχνικές τεκμηριώσεις, λόγω των πολυάριθμων χρηστών που συνεισφέρουν σε αυτήν.

Για την δημιουργία ενός παιχνιδιού δεν χρειάζονται μόνο σχεδιασμός γραφικών. Χρειάζεται ήχος, μια μηχανή φυσικής, χειρισμός εισόδων και αποκρίσεων χρήστη, τεχνητή νοημοσύνη κ.ά. Οι μηχανές παιχνιδιών (game engines) έρχονται με ένα πρότυπο API υψηλού επιπέδου που επιτρέπει τον χειρισμό όλων αυτών. Μας επιτρέπουν να επικεντρωθούμε σε πιο σημαντικά πράγματα, όπως το gameplay του παιχνιδιού και την σύνδεση όλων αυτών πολύ πιο γρήγορα.



Εικόνα 12: Μια τυπική μηχανή γραφικών

2.5 Επιλογή της Unity3D

Στην πράξη τώρα, για τον σχεδιασμό και την υλοποίηση του παιχνιδιού αρχικά χρησιμοποιήθηκε η Panda3D. Καθώς προχωρούσε η υλοποίηση εμφανίστηκαν προβλήματα βελτιστοποίησης της απόδοσης του παιχνιδιού. Πιο συγκεκριμένα ο ρυθμός καρέ ανά δευτερόλεπτο (framerate per second) είχε πέσει σε τέτοια επίπεδα τα οποία καθιστούσαν αδύνατο να εξελιχθεί η ροή του παιχνιδιού και να ακολουθηθεί από τον χρήστη. Η εισαγωγή αντιπάλων (NPCs-non player characters), η τεχνητή νοημοσύνη που είχαν, σε μορφή απλών μηχανών πεπερασμένων καταστάσεων (FSMs) και η έλλειψη βελτιστοποίησης στον έλεγχο συγκρούσεων με το έδαφος ήταν οι σημαντικότερες αιτίες για αυτό το χαμηλό ρυθμό καρέ. Κάθε παίκτης που υπήρχε στην σκηνή μας έκανε πολυάριθμους ελέγχους συγκρούσεων με το έδαφος. Επιπλέον η προσομοίωση φυσικής δεν θα μπορούσε να γίνει εφικτή με αυτές τις συνθήκες.

Η επιλογή της Unity3D έγινε με βάση αυτά τα προβλήματα. Στο τομέα των γραφικών η Unity3D έχει ένα αρκετά βελτιστοποιημένη διοχέτευση γραφικών και για DirectX και για OpenGL. Κινούμενα δικτυώματα, συστήματα σωματιδίων, προχωρημένο σύστημα για τον φωτισμό των χώρων και των σκιών. Κάτι το οποίο η Panda3D δεν διαθέτει. Η ταχύτητα του σχεδιασμού της Unity3D ήταν άκρως ικανοποιητική. Ο σχεδιασμός γίνεται ελαχιστοποιώντας τις αλλαγές καταστάσεων. Σε ένα σύστημα των ημερών μας η Unity3D σχεδιάζει εκατομμύρια πολύγωνα ανά δευτερόλεπτο.

Η Unity3D ενσωματώνει Mono για να τροφοδοτήσει το scripting περιβάλλον της. Μπορούμε να γράψουμε σε C#, JavaScript, ή Boo (μια παραλλαγή Python). Η Mono η ίδια είναι μια έκδοση ανοικτού-λογισμικού του .NET περιβάλλοντος ανάπτυξης. Πρέπει να σημειωθεί ότι αυτό δεν σημαίνει ότι η Unity3D απαιτεί .NET. Η Mono είναι εντελώς ξεχωριστή από .NET της Microsoft, και η Unity3D ενσωματώνει Mono. Μπορούμε να πάρουμε επίσης πλήρες .NET namespace, το οποίο σημαίνει ότι μια τεράστια ποσότητα κλάσεων είναι διαθέσιμη σε μας: Ανάλυση XML, σύστημα κρυπτογραφίας, υποδοχές, και άλλα. Η JavaScript της Unity3D δεν είναι πολύ παρόμοια με τη γλώσσα JavaScript που βρίσκεται στους φυλλομετρητές του Ιστού. Τα ενσωματωμένα σε Mono scripts έχουν απόλυτη πρόσβαση στην μηχανή της Unity3D μέσω του unity scripting API. Αυτό σημαίνει ότι μπορούμε να κάνουμε σχεδόν τα πάντα. Το MonoBehaviour, όπου είναι η βασική κλάση από όπου προέρχεται το κάθε script, παρέχει ένα αριθμό από ευκολίες. Τα πράγματα είναι αρκετά σαφή με το MonoBehaviour.

Υπάρχει ένας αριθμός συναρτήσεων για σκοπούς που εξυπηρετούν την λογική ενός παιχνιδιού:

* Start()
* Update()
* OnCollisionEnter()
* OnMouseDown()

Το περιβάλλον scripting υποστηρίζει τις υπορουτίνες, οι οποίες είναι πολύ χρήσιμες για σχεδόν όλα τα στοιχεία που μπορούμε να χρησιμοποιήσουμε στο παιχνίδι μας.

Εκτός από την δυνατότητα να γράφουμε scripts την στιγμή που εκτελείται το παιχνίδι μας, η Unity3D μας παρέχει ένα ισχυρό API για τον συντάκτη (editor) της ώστε να

δημιουργούμε δικά μας εργαλεία, παράθυρα, και εικονίδια ώστε να επισπεύσουμε την ροή της δουλεία μας. Στην συγκεκριμένη έκδοση της Unity3D που χρησιμοποιήθηκε, όλος ο συντάκτης (editor) έχει γραφτεί με το συγκεκριμένο Unity API οπότε θα μπορούσαμε θεωρητικά να κάνουμε ότι έχουν υλοποιήσει και οι σχεδιαστές της Unity.

ΚΕΦΑΛΑΙΟ 3
ΤΕΧΝΟΛΟΓΙΚΗ ΒΑΣΗ

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα κάνουμε μια τεχνική περιγραφή της αρχιτεκτονικής και των μονάδων της Unity3D. Θα αναφέρουμε τα βασικά χαρακτηριστικά της που βοηθάνε στην υλοποίηση του παιχνιδιού.

3.2 Τι είναι η Unity3D

Το πεδίο που καλύπτει η Unity3D καθιστά το συνοπτικό ορισμό της δύσκολο. Η Unity3D είναι πολλά πράγματα, και έχει χρησιμοποιηθεί διαφορετικά από διαφορετικές αρχές. Σε μια πρώτη ανάλυση η Unity3D είναι ένας ενσωματωμένος συντάκτης ο οποίος παρέχει ένα περιβάλλον επεξεργασίας όπου μπορούμε να οργανώνουμε τα στοιχεία (assets) του project μας, να δημιουργούμε τα αντικείμενα του παιχνιδιού (GameObjects), να προσθέτουμε υλοποιήσεις σε αυτά, να οργανώνουμε τα αντικείμενα σε επίπεδα (levels). Επιπλέον, η Unity3D παρέχει μια όψη «παιχνιδιού» για το περιεχόμενό μας. Λέγοντας όψη παιχνιδιού εννοούμε ότι μπορούμε οποιαδήποτε στιγμή να πατήσουμε το κουμπί "play" και ενώ ελέγχουμε τις τιμές να αλληλεπιδράσουμε με το περιεχόμενό, τις ρυθμίσεις που αλλάζουν, ακόμα και να μεταγλωττίσουμε ξανά τις υλοποιήσεις.

Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) είναι κατά ένα μεγάλο μέρος χωρίς καταστάσεις, δεδομένου ότι υπάρχει μικρή διάφορά μεταξύ της δημιουργίας των επιπέδων και παίζοντάς σε αυτά. Παραδείγματος χάριν, ο συντάκτης (editor) παραμένει λειτουργικά ίδιος είτε το περιεχόμενό μας σταματά, είτε παίζοντας αυτήν την περίοδο. Αυτό είναι ένα εξαιρετικά χρήσιμο χαρακτηριστικό, επειδή ενώ το περιεχόμενό μας βρίσκεται σε κατάσταση λειτουργίας ("παίζει") μπορούμε να πατήσουμε το κουμπί της παύσης και έπειτα να μετακινήσουμε πράγματα, να δημιουργήσουμε νέα αντικείμενα, να προσθέσουν υλοποιήσεις, και οτιδήποτε άλλο μπορούμε να χρειαστούμε για να εξετάσουμε το gameplay ή να αποσφαλματώσουμε το κώδικα μας. Ο συντάκτης καλύπτει μεγάλο εύρος των απαιτήσεων που μπορεί να έχει η δημιουργία ενός παιχνιδιού. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για δοκιμές καπνού και σωματιδίων, να οργανώνουμε τα στοιχεία ανά επίπεδα, να βελτιώνουμε τις υφές (textures) και διάφορα άλλα που έχουν να κάνουν με τα γραφικά του παιχνιδιού. Ακόμα, μπορούμε να εστιάσουμε την προσοχή μας πάνω στις τιμές και στους αριθμούς που αλλάζουν κατά τη εκτέλεση του παιχνιδιού.

Μια ενοποιημένη διεπαφή μας βοηθά παρά πολύ διότι δεν χρειάζεται να χρησιμοποιούμε διαφορετικά εργαλεία με διαφορετικές διεπαφές και συμβάσεις στην ροής της δουλειάς. Πρακτικά, αυτό σημαίνει ότι μειώνουμε κατά πολύ τον χρόνο που θα χρειαζόταν για μάθουμε τα εργαλεία, να τα εκκινήσουμε, να δημιουργήσουμε

κατάλληλα αρχεία που θα είναι συμβατά μεταξύ τους και να τα εισάγουμε στην ροή της δουλεία μας.

3.3 Βασική Αρχιτεκτονική της Unity3D μηχανής

Για να δουλέψουμε σε βάθος με διάφορες πτυχές της μηχανής θα πρέπει να κατανοήσουμε και να μελετήσουμε την αρχιτεκτονική της. Η Unity3D χρησιμοποιεί μια αρχιτεκτονική βασισμένη στα συστατικά (βλ. Εικόνα 13). Εάν αγνοήσουμε αυτό στη δημιουργία της λογικής του παιχνιδιού μας, δεν θα μπορούμε να έχουμε καλή κατανόηση του σχεδιασμού της Unity. Στην Unity, κάθε αντικείμενο στη σκηνή είναι ένα GameObject. Ένας αυθαίρετος αριθμός συστατικών είναι συνδεμένος με GameObjects για να καθορίσει τη συμπεριφορά τους. Ακολουθεί η ανάλυση αυτών των υπομονάδων (components) καθώς και η χρησιμότητά τους [5]:

Εικόνα 13: Οι υπομονάδες από τις οποίες αποτελείται η Unity3D μηχανή

3.3.1 Υπομονάδα κίνησης (Animation Component)

Παρέχει λειτουργικότητα για την κίνηση (animation) του κάθε μοντέλου. Την διάρκεια που θα εκτελείται, πόσες φορές, εάν θα αλληλεπιδρά με την μονάδα φυσικής.

3.3.2 Υπομονάδα στοιχείων (Asset Component)

Αυτή η υπομονάδα είναι υπεύθυνη για τις εισαγωγές και τους τύπους που χρειάζονται για να φτιάξουμε το παιχνίδι, π.χ. για τα μοντέλα, τις υφές, τους ήχους κ.ά.

3.3.3 Υπομονάδα ήχου (Audio Component)

Ομάδα των μεθόδων που παρέχει στην εφαρμογή μας υποστήριξη ήχου και χειρισμό διαφορετικών αρχείων ήχου.

3.3.4 Υπομονάδα φυσικής (Physics Component)

Η Unity3D έχει ενσωματωμένη υποστήριξη της Ageia PhysX μηχανής φυσικής. Αυτό επιτρέπει την προσομοίωση της συμπεριφοράς των αντικείμενων ως πραγματικά στέρεα άκαμπτα σώματα. Μπορούμε να εισάγουμε δυνάμεις που θα ενεργούν σε ένα αντικείμενο, αντικείμενα συγκρούσεων (colliders), συνδέσμους (joints) και αντικείμενα συγκρούσεων για προσομοίωση τροχών.

3.3.5 Διαχειριστής ρυθμίσεων (Settings Manager)

Επιτρέπει να διαχειριζόμαστε τις ρυθμίσεις όλων των στοιχείων τους project μας. Για παράδειγμα, ελέγχοντας τον διαχειριστή του ήχου θα μπορούμε να ελέγξουμε όλους τους ήχους που παίζονται στην σκηνή μας. Με τον διαχειριστή των γραφικών μπορούμε να ελέγξουμε την ποιότητα των γραφικών της σκηνής μας, τις σκιές κ.ά. Υπάρχουν διαχειριστές ρυθμίσεων για γραφικά, ήχο, εισαγωγές από πληκτρολόγιο ή ποντίκι, παίκτη, σχεδιασμό (rendering), χρόνο, δίκτυα, φυσική και για τις αναγνωριστικές πινακίδες (tags). Tag είναι μια λέξη που μπορούμε να συνδέσουμε ένα ή περισσότερα gameObject.

3.3.6 Υπομονάδα δικτυώματος (Mesh Component)

Τα τρισδιάστατα πλέγματα είναι τα θεμελιακά στοιχεία για γραφικά στην Unity3D. Γίνεται σχεδιασμός είτε κανονικών πλεγμάτων, είτε πλεγμάτων τύπου skinned (χρησιμοποιούνται για να σχεδιαστούν οι χαρακτήρες διότι έχουν συνήθως κίνηση στα κόκαλά τους). Ακόμα μπορούμε να εισάγουμε και τρισδιάστατο κείμενο και να σχεδιαστούν ίχνη ενός πλέγματος.

3.3.7 Σύνολο δικτύου (Network Group)

Αυτό το σύνολο περιέχει όλα τα συστατικά που σχετίζονται με τα παιχνίδια δικτύου όπου μπορούν να παίξουν ταυτόχρονα πολλοί παίκτες.

3.3.8 Υπομονάδα σωματιδίων (Particle Component)

Τα συστήματα σωματιδίων στην Unity3D χρησιμοποιούνται για την δημιουργία καπνού, φωτιάς και άλλων ατμοσφαιρικών εφφέ. Τα συστήματα σωματιδίων δουλεύουν με μια ή δύο υφές και σχεδιάζονται πολλές φορές, δημιουργώντας ένα χαοτικό εφφέ. Ένα τυπικό σύστημα σωματιδίων στην Unity3D είναι ένα αντικείμενο που περιέχει ένα σωματιδιακό εκπομπό (Particle Emitter), ένα σωματιδιακό κινητή (Particle Animator) και ένα σχεδιαστή σωματιδίων (Particle Renderer). Ο σωματιδιακός εκπομπός είναι υπεύθυνος για την παραγωγή σωματιδίων, ο σωματιδιακός κινητής μετακινεί τα σωματίδια κατά την διάρκεια του χρόνου και ο σχεδιαστής σωματιδίων είναι αυτός που τα εμφανίζει στην οθόνη.

3.3.9 Υπομονάδα σχεδιασμού (Rendering Component)

Αυτή η υπομονάδα περιλαμβάνει όλα τα συστατικά που έχουν να κάνουν με το σχεδιασμό κατά την διάρκεια του παιχνιδιού και με τα στοιχεία της διεπαφής του χρήστη. Τα ειδικά εφφέ και τα εφφέ φωτισμού περιλαμβάνονται σε αυτήν την υπομονάδα.

3.3.10 Υπομονάδα μετακίνησης (Transform Component)

Σε αυτή την κατηγορία συστατικών περιλαμβάνονται όλα τα συστατικά που έχουν να κάνουν με την θέση ενός αντικειμένου που είναι εκτός από το σύστημα φυσικής

3.3.11 UnityGUI

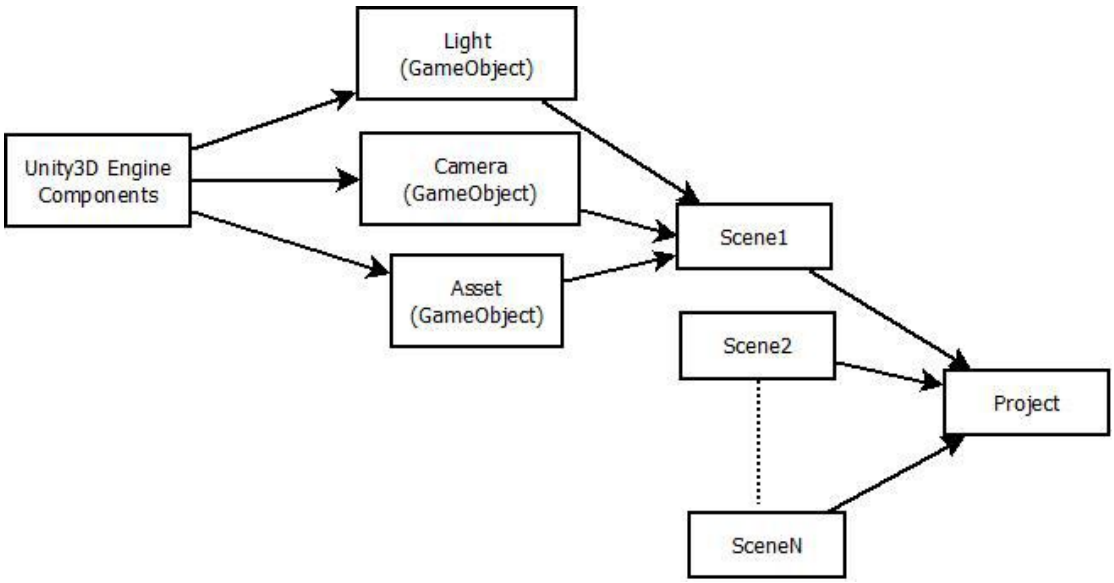
Το εν λόγω σύστημα είναι το σύστημα που μας παρέχει η Unity3D για την δημιουργία της γραφικής διεπαφής (GUI) του παιχνιδιού. Αποτελείται από διαφορετικούς τύπους ελέγχου, και ορισμό του περιεχομένου και της εμφάνισης αυτών.

3.4 Παρουσίαση της αρχιτεκτονικής

Όπως προαναφέρθηκε η Unity3D ορίζει ένα παιχνίδι ως project. Μέσα σε κάθε project μπορεί να υπάρχει μεγάλος αριθμός από σκηνές (scenes). Ο παίκτης βλέπει κάθε φορά μια σκηνή. Μέσα σε αυτήν την σκηνή μπορεί να εισαχθεί ένας αυθαίρετος αριθμός από GameObjects. Για να καθοριστεί συμπεριφορά σε αυτό θα πρέπει να έχουν συνδεδεμένα τα ανάλογα συστατικά (components). Για να ξεκινήσει μια σκηνή

θα πρέπει να έχει μια κάμερα. Η κάμερα θα πρέπει να έχει εισαχθεί ως GameObject. Έπειτα, θα χρειαστεί ένα φως για να βλέπουμε στον χώρο. Έτσι θα δημιουργήσουμε ένα άδειο GameObject το οποίο θα πρέπει να περιέχει και μια υπομονάδα τύπου σχεδιασμού (rendering) . Επόμενο βήμα για να εισάγουμε ένα στοιχείο παιχνιδιού είναι να φτιάξουμε ένα άδειο GameObject και μετά να προσθέσουμε κάποιο από τα στοιχεία που θα ανήκει στην υπομονάδα στοιχείων (asset component) και έτσι θα συνεχιστεί η διαδικασία.

Σχηματικά αυτή η αρχιτεκτονική μια τέτοιας σκηνής φαίνεται στην Εικόνα 14.



Εικόνα 14: Παρουσίαση απλής αρχιτεκτονικής ενός project στην Unity3D

3.5 Παρουσίαση του Unity Rendering Pipeline

Οι σκιαστές μπορούν να ορίσουν πως ένα αντικείμενο φαίνεται είτε χωρίς, είτε με την αλληλεπίδραση του φωτός. Στην Unity3D υπάρχουν Directional, Point ή Spot τύποι φωτισμού. Επιπροσθέτως, μέσω των ρυθμίσεων μπορεί να οριστεί και ατμοσφαιρικό (Ambient) φως.

Τα φώτα που φωτίζουν ένα αντικείμενο είναι καθορισμένα και κάθε φως έχει οριστεί απο πριν εάν θα σχεδιαστεί σε τύπου κορυφής ή εικονοστοιχείου (Vertex or Pixel) όταν αυτό σχεδιαστεί. Ο φωτισμός τύπου εικονοστοιχείου συνήθως είναι πολύ καλύτερος, αλλά είναι ακριβός για τον επεξεργαστή. Έτσι για κάθε αντικείμενο, μόνο ένα μικρό ποσοστό απο φώτα σχεδιάζονται σε ρύθμιση εικονοστοιχείου, ενώ τα υπόλοιπα σε φωτισμό τύπου κορυφής. Όταν δεν υπάρχουν φώτα να φωτίζουν ένα αντικείμενο τότε σχεδιάζεται σε ρύθμιση «τίποτα» (None).

Κάθε αντικείμενο σχεδιάζεται με τον ακόλουθο τρόπο:

- Εάν υπάρχουν φώτα κορυφής που το φωτίζουν, σχεδιάζονται κατευθείαν. Αυτό το στάδιο σχεδιάζει όλα τα περάσματα κορυφής μέσα στον σκιαστή,

και ο σκιαστής αναμένεται να λάβει υπ' όψιν του τον φωτισμό κορυφής και τον ατμοσφαιρικό φωτισμό.

- Εάν δεν υπάρχουν φώτα κορυφής να το φωτίζουν, ένας σκιαστής εκτελείται μία φορά, για να λάβει υπ' όψιν του το ατμοσφαιρικό φως. Αυτό το στάδιο σχεδιάζει όλα τα "none" περάσματα στον σκιαστή.
- Μετά απο αυτό, ο φωτισμός εικονοστοιχείου θα προστεθεί. Όλα τα περάσματα εικονοστοιχείου στον σκιαστή σχεδιάζονται για κάθε φως εικονοστοιχείου. Αυτός είναι ο κύριος λόγος που τα φώτα εικονοστοιχείου είναι περισσότερο ακριβά, επειδή χρειάζεται να γίνει σχεδιασμός του αντικείμενου για κάθε φως ξεχωριστά, αντί να γίνει για όλα τα φώτα κατευθείαν.

3.5.1 Φώτα κορυφής

Τα φώτα τύπου κορυφής σχεδιάζονται μέσω περασμάτων «κορυφής» (pass tags στην Unity). Όλα τα φώτα σχεδιάζονται κατευθείαν, χρησιμοποιώντας το fixed function OpenGL/Direct3D μοντέλο φωτισμού (Blinn-Phong). Ο φωτισμός κορυφής μπορεί να υπολογιστεί αυτόματα. Το μόνο που χρειάζεται να κάνουμε στον σκιαστή είναι να χρησιμοποιήσουμε τα υπολογισμένα χρώματα διάχυσης/κατοπτρικά (diffuse/specular), είτε σε ένα πρόγραμμα κομματιού (fragment program), είτε με χρήση συνδυασμού υφών.

3.5.2 Φώτα εικονοστοιχείου

Για την υλοποίηση ενός σκιαστή φωτισμού εικονοστοιχείου χρειάζονται περισσότερα πράγματα κυρίως λόγω των διαφορετικών ειδών φωτός αλλά και επειδή ο σκιαστής θα πρέπει να είναι ικανός να τα επεξεργαστεί. Για τα φώτα εικονοστοιχείου θα πρέπει να γραφτούν προσαρμοσμένα προγράμματα κορυφών τα οποία θα πρέπει να υπολογίζουν το φωτισμό.

ΚΕΦΑΛΑΙΟ 4
ΣΧΕΔΙΑΣΜΟΣ ΠΑΙΧΝΙΔΙΟΥ ΚΑΙ ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ

4.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εξετάσαμε την αρχιτεκτονική της Unity3D και είδαμε σε γενικές γραμμές τις υπομονάδες που την αποτελούν καθώς και την λειτουργικότητα τους. Στο παρών κεφάλαιο, θα δούμε αναλυτικότερα το τρόπο σχεδιασμού του παιχνιδιού με βάση αυτά και την χρηστικότητά τους.

4.2 Σενάριο

Ο ήρωας μας είναι ο Ζαχαρίας, ένας εύσωμος πορτοκαλί πειρατής που μπαίνει σε περιπέτειες στο νησί των Πράσινων Πειρατών. Ο ίδιος κατάγεται από το νησί των πορτοκαλί πειρατών το οποίο πρόσφατα υπέστη μια βίαιη και αναίτια επίθεση από τους πράσινους πειρατές. Τα εν λόγω ανεγκέφαλα και κτηνώδη πλάσματα (οι πράσινοι πειρατές) αποφάσισαν ότι βαριούνται την ζωή τους και ήθελαν δράση. Ήθελαν να δουν καινούρια μέρη και να καλοπεράσουν εκεί. Αποφάσισαν ότι το καλύτερο μέρος για αυτό ήταν το ειρηνικό νησί των Πορτοκαλί πειρατών. Εκεί που έμενε ο Ζαχαρίας, ο φίλος μας.

Έτσι λοιπόν πήγαν στο νησί του Ζαχαρία άρπαξαν ότι χρήματα είχε το νησί και ότι μπουκάλια από δροσερό και ζωογόνο -για οποιαδήποτε ράτσα πειρατών- ρούμι, εξαφάνισαν όσους αντιστέκονταν σε αυτούς και δεν τους έδιναν ότι ζητούσαν και έπειτα γύρισαν χορτασμένοι από εμπειρίες στο νησί τους. Έχοντας και μια παπαγαλίνα αιχμάλωτη. Ναι, καλά το μαντέψατε, την Μαλάμω, την παπαγαλίνα του Ζαχαρία. Ο Ζαχαρίας όταν τα πληροφορήθηκε όλα αυτά, γυρνώντας από δουλειές, αποφάσισε ότι δεν πήγαινε άλλο. "Ως εδώ και μην παρέκει..", σκέφτηκε. Θα χρησιμοποιούσε τις γνώσεις που είχε από τον στρατό και θα πήγαινε στο νησί των Πράσινων Πειρατών για να πάρει πίσω όλα τα νομίσματα και τα μπουκάλια ρούμι που πήραν από το νησί του οι πράσινοι. Ήταν αποφασισμένος ότι θα γυρίσει ήρωας πίσω με την Μαλάμω και έχοντας δώσει ένα μάθημα στους πράσινους. Έβαλε λοιπόν το ψεύτικο μαύρο πειρατικό κάλυμμα στο δεξί του μάτι και ξεκίνησε.

Ο Ζαχαρίας θα πρέπει να κινηθεί έξυπνα ώστε να μην αφανιστεί από τους πράσινους πειρατές. Γνωρίζει ότι υπάρχουν δύο αετομάτηδες φρουροί στην είσοδο του νησιού. Θα πρέπει κάπως να τους εξολοθρεύσει. Εξολοθρεύοντας τους 2 πειρατές θα έχει πρόσβαση στα επτασφράγιστα κιτάπια των αποθηκών του νησιού. Έτσι θα μπορέσει να βρει που βρίσκεται η Μαλάμω. Διότι στα κιτάπια βρίσκονται όλες οι πληροφορίες για περιεχόμενο των αποθηκών. Οι πράσινοι πειρατές έχουν ένα πολύ περίεργο συνήθειο. Να βάζουν τα κλειδιά τους σε κανάτες. Όταν ο Ζαχαρίας πάει στην αποθήκη που βρίσκεται η Μαλάμω, θα βρεθεί μπροστά σε μια δυσάρεστη έκπληξη. Η

Μαλάμω θα έχει γεράσει επικίνδυνα. Ο μάγος του νησιού έχει δώσει στην Μαλάμω ένα από τα φίλτρα του και εάν δεν πάρει το αντίδοτο θα πεθάνει. Μαζί με αυτήν θα πεθάνει από στενοχώρια και ο Ζαχαρίας μας. Όποτε θα πρέπει να βρει και να μαζέψει το αντίδοτο για την Μαλάμω μέσα σε συγκεκριμένο χρονικό διάστημα.

Σε αυτήν την αναζήτηση του όπως- είναι αναμενόμενο- θα τον δυσκολέψουν οι πράσινοι πειρατές που τριγυρνάνε στο νησί. Όταν αποκτήσει το αντίδοτο θα πρέπει να γυρίσει γρήγορα στην Μαλάμω και να της το δώσει. Όταν τελικά η Μαλάμω είναι ξανά νέα, ο Ζαχαρίας θα είναι ευτυχισμένος πάλι και δεν θα τον νοιάζει η εκδίκηση.

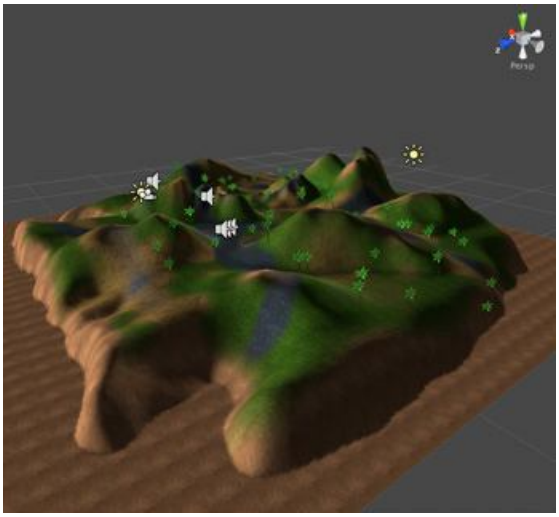
4.3 Σχεδιασμός παιχνιδιού

Αρχικά, για τον σχεδιασμό του παιχνιδιού κρίθηκε αναγκαίο να μελετήσουμε διάφορα άλλα παιχνίδια του ίδιου είδους. Βρίσκοντας κάποια κοινά χαρακτηριστικά, τα χαρακτηριστικά που αναφέραμε στο πρώτο κεφάλαιο, προσπαθήσαμε να τα ενσωματώσουμε σε ένα σενάριο ώστε να πετύχουμε μια συνοχή στο παιχνίδι μας.

Παρουσιάζουμε τα χαρακτηριστικά με ένα πιο τεχνικό τρόπο από ότι στο πρώτο κεφάλαιο.

4.3.1 Σχεδιασμός πίστας

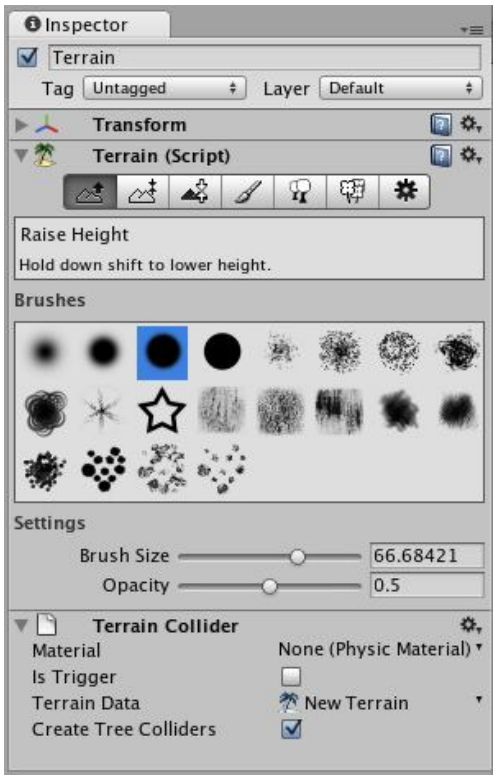
Κατασκευή ενός μοντέλου νησιού που θα έχει αρχική και τελική θέση, για την εκκίνηση και τον τερματισμό της πίστας. Η πλοήγηση του παίκτη θα πρέπει να είναι άνετη, με φανερά μονοπάτια και σημεία τα οποία δεν μπορεί να προσπελάσει. Σύμφωνα με το σενάριο θα υπάρχουν μέρη που θα χρησιμοποιούνται ως αποθήκες. Χρησιμοποιούμε τον συντάκτη εδάφους (Terrain Editor), που παρέχεται από την Unity3D, για τον σχεδιασμό του μοντέλου του νησιού και το 3D Studio Max για να σχεδιάσουμε τα σπίτια που θα χρησιμοποιηθούν σαν αποθήκες.



Εικόνα 15:Το έδαφος στην τελική του μορφή

Συντάκτης εδάφους (Terrain Editor)

Ο συντάκτης εδάφους (Terrain Editor) της Unity3D χρησιμοποιείται για την κατασκευή εδάφους. Μας παρέχει εργαλεία για ανύψωση εδάφους σε ένα σημείο ανάλογα με το μέγεθος της βούρτσας που έχουμε επιλέξει, για βύθιση του εδάφους, για ανύψωση εδάφους μέχρι ένα συγκεκριμένο ύψος, για εισαγωγή υφών στον έδαφος, δέντρων και διάφορων άλλων λεπτομερειών (βράχους, γρασιδιού, μπάλες κ.ά)



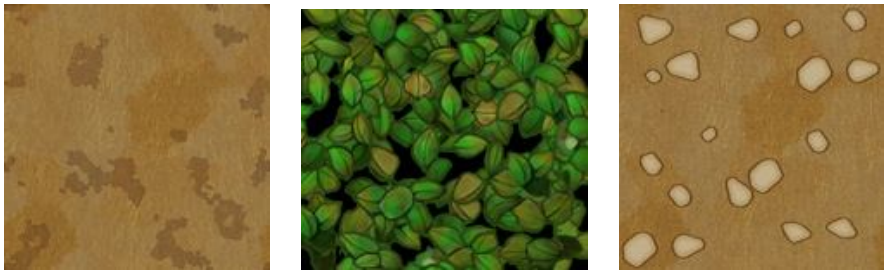
Εικόνα 16: Χρησιμοποίηση συντάκτη εδάφους (Terrain Editor)

Εικόνα 17: Το εργαλείο με το οποίο υψώνουμε σημείο του εδάφους

Χρησιμοποιώντας το εργαλείο για την διαχείριση των υφών του εδάφους (βλ. Εικόνα 18) και κάνοντας πολυχαρτογράφιση (blending ή multitexturing) βάλαμε πολλές υφές ώστε να φτιάξουμε την υφή που θέλουμε.

Εικόνα 18:Εργαλείο για διαχείριση υφών του εδάφους

Έτσι δημιουργήσαμε μια υφή που αποτελείται απο τρεις άλλες και ανάλογα με την μάσκα που εφαρμόζουμε σε αυτήν θα έχουμε και το βαθμό ορατότητάς της. Οι υφές είναι σκόπιμα μη φωτορεαλιστικές (βλ. Εικόνα 19) .



Εικόνα 19: Υφές για έδαφος (3D Total Toon Textures)

«Η χρησιμοποίηση περιορισμών είναι αναγκαία ώστε να αυξήσουμε το επίπεδο της πρόκλησης, όπως το να καίγεται ο παίκτης εάν εισέρχεται στο ηφαίστειο και να μην βλέπει καλά όταν εισέρχεται στο βυθό της θάλασσας.»
Στο ηφαίστειο έχει τοποθετηθεί ένας συγκρουστής (collider) ο οποίος ενεργοποιείται όταν ο παίκτης πέσει μέσα στο ηφαίστείο και προκαλεί απώλεια μιας «ζωής» στον χαρακτήρα του χρήστη.

Για το εφφέ της μειωμένης ορατότητας στον βυθό ελέγχεται η θέση του παίκτη ως προς την στάθμη της θάλασσας. Όταν εντοπιστεί ο παίκτης στον βυθό ενεργοποιείται το εφφέ θολώματος (blur effect) και το εφφέ για διόρθωση χρώματος (color correction effect) στην ενεργή κάμερα.

Εικόνα 20: Το ηφαίστειο τοποθετημένο στην πίστα (δεξιά) και στιγμιότυπο όταν ο παίκτης εισέρχεται στην θάλασσα (αριστερά) .

Για τον φωτισμό και την σκίαση του εδάφους χρησιμοποιείται η τεχνική χαρτογράφησης φωτός (lightmapping) και αυτό διότι το έδαφος είναι στατικό αντικείμενο. Φτιάχτηκε ένας χάρτης φωτός και χρησιμοποιήθηκε ώστε να μην επιβαρυνθεί ο επεξεργαστής απο τους επιπλέον υπολογισμούς φωτισμού πραγματικού χρόνου (real-time lighting).

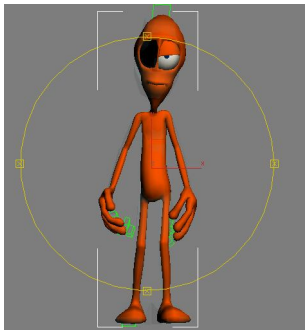
Δημιουργώντας ένα περιβάλλον νησιού, σημαντικό κομμάτι είναι η δημιουργία θάλασσας γύρω από αυτό. Έτσι φτιάχτηκε ένα μοντέλο το οποίο θα έπρεπε να του εφαρμοστεί ο κατάλληλος σκιαστής ώστε να φαίνεται η θάλασσα ρεαλιστική. Να κινείται, να έχει κύματα στην επιφάνεία της και να κάνει αντανάκλαση του περιβάλλοντος της.

4.3.2 Δυνατότητες Χαρακτήρα

Για τις δυνατότητες του χαρακτήρα που είναι η πλοήγηση, η απλή επίθεση, το σπάσιμο αντικειμένων, η συλλογή αντικειμένων θα χρειαστεί να σχεδιαστούν:

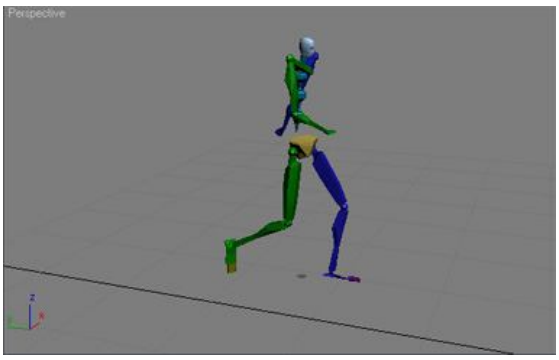
- μοντέλου χαρακτήρα
- ενέργεια για επιθέσεις
- αντικείμενα

Το **μοντέλο του χαρακτήρα** σχεδιάστηκε από τον Daniel Martinez Lara και έγινε εισαγωγή στη Unity3D με το κατάλληλο εισαγωγέα. Από την στιγμή που έγινε η εισαγωγή, κάθε φορά που γίνεται αλλαγή στο αρχείο μας θα το ανανεώνει και η Unity3D σε πραγματικό χρόνο.



Εικόνα 21: Μοντέλο κύριου χαρακτήρα

Για να σχεδιάσουμε **τις ενέργειες που γίνονται στις επιθέσεις**, προσθέσαμε motion capture animations και προσαρμόστηκαν στον χαρακτήρα μας. Φτιάχτηκε ένα μοντέλο δίποδου και συνδέθηκε με το μοντέλο. Αυτό το μοντέλο δίποδου είναι συμβατό με τα δεδομένα καταγραφής κίνησης (motion capture animations) που προσφέρει το πανεπιστήμιο Carnegie Mellon University (CMU) [9] .



Εικόνα 22: Σύστημα δίποδου (Biped system) για το προσθήκη κίνησης βασισμένη σε δεδομένα MoCap (κύκλος περπατήματος)

Σχεδιάστηκε **το μοντέλο των αντικειμένων** που θα συλλέγει ο χαρακτήρας.

Για το αντικείμενο του νομίσματος προστέθηκε ένα πλάνο που θα χρησιμοποιηθεί ως το εφφέ γυαλίσματος καθώς περιστρέφεται γύρω από τον άξονά του.

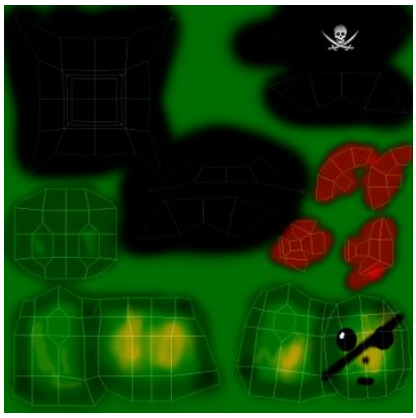
Εικόνα 23: Μοντέλο νομίσματος

4.3.3 Εμπόδια και Αντίπαλοι

Για τις δυνατότητες των αντιπάλων που είναι μετακίνηση, επίθεση με καρύδες, θα χρειαστεί να σχεδιαστούν:

- το μοντέλου του αντιπάλου
- η ενέργεια που θα γίνεται στις επιθέσεις
- τα αντικείμενα που θα εκτοξεύει προς τον παίκτη

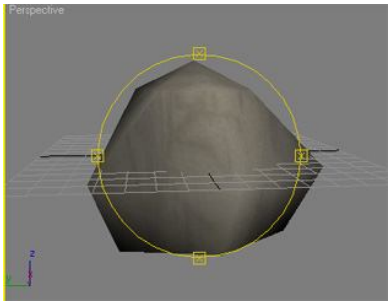
Το **μοντέλο του αντιπάλου** είναι όμοιο με το μοντέλο του χαρακτήρα μας με διαφορετικό χάρτη υφής.



Εικόνα 24:Χάρτης υφής (Texture map) του αντιπάλου

Η ενέργεια που θα γίνεται στις επιθέσεις είναι η εξής: ο αντίπαλος θα πετάει καρύδες όταν ο παίκτης φτάσει σε μια συγκεκριμένη απόσταση. Η καρύδα λοιπόν θα είναι μια απλή σφαίρα η οποία έχει την κατάλληλη υφή εφαρμοσμένη και θα χρησιμοποιηθεί ως αντικείμενο που θα εκτοξεύει ο παίκτης.

Τέλος, βράχοι θα χρησιμοποιηθούν ως εμπόδια και πέφτουν όταν ο παίκτης πατήσει συγκεκριμένα σημεία στην πίστα.



Εικόνα 25: Μοντέλο βράχου

Η τεχνητή νοημοσύνη του αντιπάλου θα αναλυθεί σε επόμενο κεφάλαιο.

4.3.4 Υγεία και ζωές

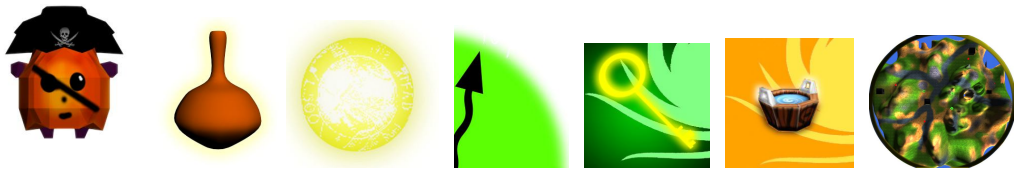
Τα σημεία ελέγχου (checkpoints) πρέπει να σχεδιαστούν και να εισαχθούν για να φαίνεται ο παίκτης από που θα ξεκινάει όταν χάνει μία ζωή .Επίσης μοντέλα για τα αντικείμενα (pick-ups) της ζωής και της υγείας και συστήματα που θα ενεργοποιούν την εκπομπή σωματιδίων όταν τα συλλέγει ο παίκτης.

4.3.5 Γραφικό περιβάλλον

Το γραφικό περιβάλλον χρήστη αποτελείται απο σελίδες, εικόνες, γραμματοσειρές και χρώματα που ταιριάζουν με το στυλ του παιχνιδιού. Οι οθόνες του μενού είχαν όλες την ίδια διαρρύθμιση.

Εικόνα 26: Δυσδιάστατη εικόνα για οθόνες εκτός παιχνιδιού

Απαραίτητες είναι και οι δυσδιάστατες εικόνες για να την παρουσίαση των βοηθητικών στοιχείων στην οθόνη (HUDs). Θα χρειαστούν εικόνες για μπάρα ενέργειας, την εμφάνιση των αντικείμενων που έχει μαζέψει ο παίκτης, την εικόνα που θα τοποθετηθεί ο χάρτης του νησιού, τα εικονίδια του καταλόγου (inventory) και τις διάφορες πληροφορίες που θα δίνουμε στον χρήστη με την μορφή κείμενου που εξασθενούν στο χρόνο.



Εικόνα 27: Δυσδιάστατες εικόνες για HUDs

4.3.6 Βαθμολογία και τερματισμός παιχνιδιού

Στο τερματισμό θα εισαχθούν ήχοι και κείμενο ώστε να πληροφορείται ο παίκτης ότι κέρδισε και θα εμφανίζεται μια οθόνη με τις υψηλότερες βαθμολογίες. Οπότε, χρειάζεται μια απλή βάση δεδομένων που θα συγκρατεί το όνομα και την βαθμολογία του κάθε παίκτη, καθώς και η διασύνδεση του παιχνιδιού με αυτήν.

4.4 Υλοποίηση σεναρίου

Ακολουθεί ο τρόπος που υλοποιήθηκε το σενάριο και η συσχέτισή του με το παιχνίδι. Σύμφωνα με το σενάριο, ο κύριος χαρακτήρας του παιχνιδιού θα πρέπει να είναι στο νησί των Πράσινων πειρατών. Έτσι κρίνεται απαραίτητο να φτιαχτεί ένα έδαφος που να μοιάζει με νησί. Έπειτα το σενάριο αναφέρει ότι υπάρχουν δύο φρουροί στην είσοδο του νησιού. Δύο αντίπαλοι χαρακτήρες θα τοποθετηθούν σε ένα σημείο που θα είναι κοντά στο σημείο που εμφανίζεται ο κύριος χαρακτήρας.

«Εξολοθρεύοντας τους 2 πειρατές θα έχει πρόσβαση στα επτασφράγιστα κιτάπια των αποθηκών του νησιού. Έτσι θα μπορέσει να βρει που βρίσκεται η Μαλάμω.»
Όταν νικήσει τους αντιπάλους ο κύριος χαρακτήρας και μόνο τότε, θα εμφανιστεί το κουμπί που θα δίνει πληροφορίες για την τοποθεσία των αντικειμένων (βλ. Εικόνες 36, 37).

Για να βρούμε τα κλειδιά που υπάρχουν στην πίστα πρέπει να σπάσουμε τις κανάτες. Για αυτό το κομμάτι σεναρίου φτιάχτηκε ένα μοντέλο κανάτας το οποίο έχει ενσωματωμένη την προσομοίωση φυσικής σπασίματός της σαν αρχείο κίνησης (animation) και το μοντέλο του κλειδιού. Το κλειδί τοποθετήθηκε έπειτα σε σημείο τέτοιο που να φαίνεται ότι είναι μέσα στην κανάτα.

«Όταν ο Ζαχαρίας πάει στην αποθήκη που βρίσκεται η Μαλάμω θα βρεθεί μπροστά σε μια δυσάρεστη έκπληξη. Η Μαλάμω θα έχει γεράσει επικίνδυνα.»
Για να μπει στην αποθήκη ο χαρακτήρας θα πρέπει να έχει στην κατοχή του το κλειδί. Όταν το έχει και βρει την αποθήκη που έχουν τον έξτρα χαρακτήρα θα εμφανιστεί ένα μήνυμα που θα του λεει τι έχει συμβεί και τι πρέπει να κάνει για συνεχίσει. Έχει σχεδιαστεί ένα μοντέλο για τον έξτρα χαρακτήρα και ένα για τις αποθήκες .

«Μαζί με αυτήν θα πεθάνει από στενοχώρια και ο Ζαχαρίας μας.»
Ο παίκτης θα χάσει το παιχνίδι εάν συμβεί αυτό το κομμάτι του σεναρίου.

«Όποτε θα πρέπει να βρει και να μαζέψει το αντίδοτο για την Μαλάμω μέσα σε συγκεκριμένο χρονικό διάστημα.»

Έχει τοποθετηθεί ένα αντικείμενο στην πίστα και ο παίκτης θα πρέπει να το βρει πριν τελειώσει ο χρόνος που έχει οριστεί για την συγκεκριμένη αποστολή.

«...θα τον δυσκολέψουν οι πράσινοι πειρατές που τριγυρνάνε στο νησί.»
Οι αντίπαλοι που βρίσκονται στο νησί θα κυνηγούν και θα προσπαθούν να σκοτώσουν τον χαρακτήρα που ελέγχει ο παίκτης.

«Όταν αποκτήσει το αντίδοτο θα πρέπει να γυρίσει γρήγορα στην Μαλάμω και να της το δώσει. Όταν τελικά η Μαλάμω είναι ξανά νέα, ο Ζαχαρίας θα είναι ευτυχισμένος πάλι και δεν θα τον νοιάζει η εκδίκηση.»
Ο χαρακτήρας όταν έχει στην κατοχή του το ζητούμενο αντικείμενο θα πρέπει να γυρίσει στην θέση που βρίσκεται ο έξτρα χαρακτήρας και να τον πλησιάσει. Όταν τον πλησιάσει αρκετά θα εμφανιστεί ένα μήνυμα που αναφέρει τι συνέβη και θα ακολουθήσει μια σειρά απο ενέργειες που δείχνουν ότι ο χαρακτήρας έχει καταφέρει τον στόχο του και είναι χαρούμενος για αυτό.

4.5 Παρουσίαση της γραφικής διεπαφής

4.5.1 Αρχικό μενού

Η αρχική σελίδα της εφαρμογής, φαίνεται στην παρακάτω εικόνα. Ο χρήστης έχει την δυνατότητα να πληροφορηθεί για τα κουμπιά χειρισμού του παιχνιδιού, για την ιστορία πάνω στην οποία βασίζεται το παιχνίδι, να εγκαταλείψει την εφαρμογή και, φυσικά, να παίξει.

Εικόνα 28: Αρχική οθόνη παιχνιδιού

Παρακάτω παρουσιάζουμε την λειτουργία του κάθε κουμπιού:

- **How to play:** Με αυτήν την επιλογή ο χρήστης μεταφέρεται σε μια οθόνη που του δίνει πληροφορίες σε σχέση με τα κουμπιά χειρισμού του παιχνιδιού (βλ. Εικόνα 29). Όταν μεταφερθεί εκεί έχει την επιλογή να κλείσει το παράθυρο, με το κουμπί «close», που ανοίχτηκε και να γυρίσει στην προηγούμενη οθόνη.

Εικόνα 29:Οθόνη που πληροφορείται ο χρήστης για τα κουμπιά χειρισμού

- **Play:** Με αυτήν την επιλογή ο χρήστης μεταφέρεται σε μια οθόνη που προτρέπεται απο το σύστημα να εισάγει το ψευδώνυμό του, ώστε να χρησιμοποιηθεί για την καταγραφή των υψηλότερων βαθμολογιών (βλ. Εικόνα 30). Έχει την δυνατότητα να το αφήσει κενό. Το σύστημα θα του δώσει ένα μήνυμα που θα τον πληροφορεί ότι στην καταγραφή των υψηλότερων βαθμολογιών θα φαίνεται ως φιλοξενούμενος («guest»). Στην οθόνη υπάρχουν τα κουμπιά «O.K.» για την είσοδο του παίκτη στο παιχνίδι και «Back» για την εμφάνιση της αρχικής οθόνης. Πατώντας «O.K.» εμφανίζεται μια οθόνη που μας πληροφορεί ότι το παιχνίδι «φορτώνει» (βλ. Εικόνα 31).

Εικόνα 30:Οθόνη που ο χρήστης εισάγει το ψευδώνυμο του

Εικόνα 31: Οθόνη φορτώματος

- **The Story:** Με αυτήν την επιλογή ο χρήστης πληροφορείται σε ξεχωριστό παράθυρο για την ιστορία στην οποία βασίζεται το παιχνίδι (βλ. Εικόνα 32). Στο παράθυρο που αναδύεται έχουμε κάθετη μπάρα για την εμφάνιση του κειμένου και ένα κουμπί «Close» για να κλείνει το παράθυρο και να μεταφερόμαστε στο προηγούμενο μενού.

Εικόνα 32: Το παράθυρο της ιστορίας του παιχνιδιού

- **Quit:** Με αυτήν την επιλογή εγκαταλείπουμε το παιχνίδι. Πριν κλείσει η εφαρμογή το σύστημα ρωτάει τον χρήστη εάν όντως θέλει να εγκαταλείψει το παιχνίδι και αν ο χρήστης απαντήσει καταφατικά τότε εγκαταλείπει το παιχνίδι αλλιώς επανέρχεται στην οθόνη του αρχικού μενού.

4.5.2 Κύριο γραφικό περιβάλλον παιχνιδιού

Το κύριο γραφικό περιβάλλον του παιχνιδιού αποτελείται απο τις λεγόμενες Heads-Up Displays (HUDs), απο τα κουμπιά που ελέγχουν τον κατάλογο του παίκτη (inventory) και τις βοηθητικές οθόνες του παιχνιδιού.

Οι HUDs έχουν τις εξής πληροφορίες για τις παίκτη:

- Ζωές
- Ενέργεια που του απομένει
- Βοηθητικός χάρτης
- Το σκορ
- Τα νομίσματα που έχει μαζέψει
- Τα μπουκάλια ρούμι που έχει πιει
- Πληροφορίες για την συμπεριφορά του αντιπάλου

Τα μόνα που χρειάζονται επεξήγηση απο τα παραπάνω είναι ο χάρτης του νησιού και οι πληροφορίες για την κατάσταση του αντιπάλου.

Στο **χάρτη του νησιού** υπάρχουν πολύχρωμες σφαίρες και ένας κύβος. Ο **κύβος** αντιπροσωπεύει τον χαρακτήρα που ελέγχει ο παίκτης. Οι **κίτρινες σφαίρες** αντιπροσωπεύουν τους αντίπαλους του χαρακτήρα, οι **κόκκινες σφαίρες**

αντιπροσωπεύουν τις τοποθεσίες των αποθηκών και τέλος η **πράσινη σφαίρα** αντιπροσωπεύει ένα έξτρα χαρακτήρα (τη Μαλάμω, σύμφωνα με το σενάριο) .

Παρουσίαση όλων αυτών στην παρακάτω εικόνα.

Εικόνα 33:Παρουσίαση HUDs

Οι συμπεριφορές του αντιπάλου μπορούν να είναι οι εξής:

- **Συμπεριφορά «καπνίσματος»:** Ο αντίπαλος έχει χαλαρώσει και καπνίζει το πούρο του. Μπορεί όμως οποιαδήποτε στιγμή να αλλάξει συμπεριφορά και να κυνηγήσει το χαρακτήρα (βλ. Εικόνα 34).

Εικόνα 34: Εικονίδιο για συμπεριφορά καπνίσματος

- **Συμπεριφορά «περίπολος»:** Ο αντίπαλος κάνει περιπολία. Θα πρέπει ο παίκτης να προσέχει να μην το δει, διότι θα τον κυνηγήσει (βλ. Εικόνα 35).

Εικόνα 35: Εικονίδιο για συμπεριφορά περιπολίας

- **Συμπεριφορά «ύπνος»:** Ο αντίπαλος κοιμάται αλλά δεν ξέρουμε πότε θα ξυπνήσει. Οπότε καλό είναι να προσέχει ο χαρακτήρας του παίκτη μην περάσει κοντά του (βλ. Εικόνα 36).

Εικόνα 36: Εικονίδιο για συμπεριφορά ύπνου

Όπως αναφέρθηκε παραπάνω, ο χρήστης μπορεί να ενημερωθεί και για την κατάσταση του καταλόγου των αντικειμένων που έχει στην κατοχή του (inventory).

Το σχετικό γραφικό περιβάλλον φαίνεται στην Εικόνα 37:

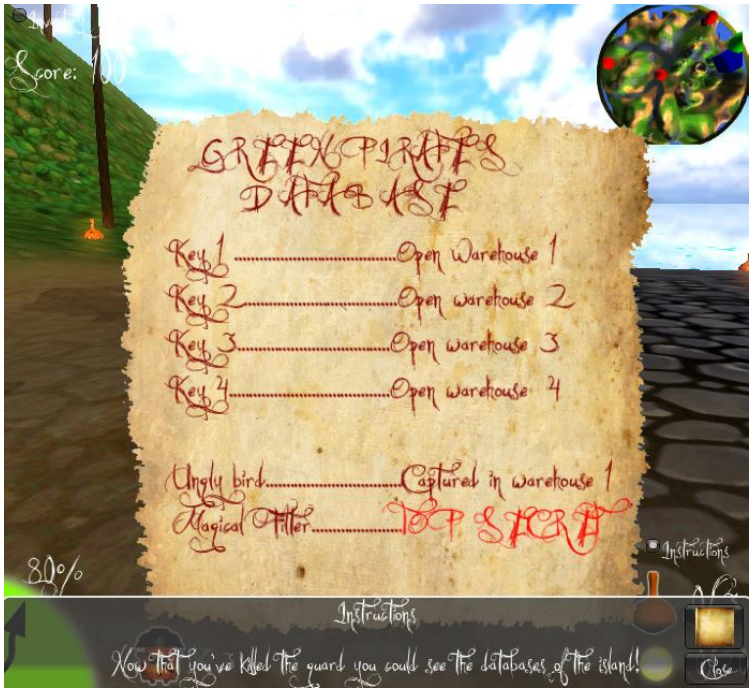
Εικόνα 37: Παρουσίαση καταλόγου αντικειμένων του παίκτη

Ο παίκτης μπορεί να δει όλα τα αντικείμενα που έχει στην κατοχή του κάνοντας κλικ στην επιλογή «Inventory» πάνω δεξιά στην οθόνη. Πατώντας ανοίγει στο κάτω μέρος της οθόνης ο κατάλογος των αντικειμένων. Κάνοντας κλικ πάνω σε ένα αντικείμενο εμφανίζεται η χρησιμότητά του. Για να γυρίσει στο προηγούμενο παράθυρο πρέπει ο χρήστης να πατήσει το κουμπί «Back» κάτω δεξιά. Τέλος, για να κλείσει τον κατάλογο αντικειμένων κάνει κλικ πάνω δεξιά, την ίδια ενέργεια που έκανε για να το εμφανίσει.

Σε συγκεκριμένη στιγμή στο παιχνίδι εμφανίζεται και μια επιπλέον επιλογή στην κάτω δεξιά μεριά της οθόνης του παιχνιδιού (βλ. Εικόνα 38) . Κάνοντας ο χρήστης κλικ μπορεί να πληροφορηθεί για το τι χρειάζεται να κάνει στην παρούσα φάση. Εμφανίζονται στο παράθυρο δύο κουμπιά. Ένα που εμφανίζει ένα έγγραφο (τις πληροφορίες για αντικείμενα του παιχνιδιού) και ένα που κλείνει το παράθυρο των οδηγιών.

Εικόνα 38: Παρουσίαση των οδηγιών κατά την διάρκεια του παιχνιδιού

Πατώντας το κουμπί που εμφανίζει το έγγραφο εμφανίζεται η παρακάτω εικόνα στην οθόνη:



Εικόνα 39: Παρουσίαση εγγράφου πληροφοριών

Ο χρήστης μπορεί να δει πληροφορίες για την χρηστικότητα του κάθε κλειδιού και των άλλων αντικειμένων που υπάρχουν στην πίστα. Κάνοντας κλικ στο κουμπί «Close» γυρνάει στο προηγούμενο παράθυρο.

Κατά την διάρκεια του παιχνιδιού εμφανίζονται και άλλα παράθυρα που παρέχουν πληροφορίες για το τι χρειάζεται να κάνει ο παίκτης ώστε φτάσει στον στόχο του και η χρήση τους είναι προφανής, διότι έχουν μόνο κουμπιά «Close» και «Back».

Τέλος, κατά την διάρκεια του παιχνιδιού εμφανίζονται μηνύματα στην οθόνη που ξεθωριάζουν και χάνονται για να πληροφορήσουν τον παίκτη για την κατάσταση που βρίσκεται την δεδομένη στιγμή (βλ. Εικόνα 40).

Εικόνα 40: Παρουσίασης συστήματος μηνυμάτων στον χρήστη

4.6 Περίληψη

Στο κεφάλαιο αυτό παρουσιάσαμε τον σχεδιασμό του παιχνιδιού, την υλοποίηση του σεναρίου και την γραφική διεπαφή της εφαρμογής μας. Παρουσιάστηκε αναλυτικά τι χρειάστηκε να χρησιμοποιηθεί για τον σχεδιασμό του παιχνιδιού, πώς υλοποιήσαμε το σενάριο και τέλος ο τρόπος που ο εκάστοτε χρήστης χρησιμοποιεί τις επιλογές που έχει για να εκτελέσει τις απαραίτητες ενέργειες στο παιχνίδι.

ΚΕΦΑΛΑΙΟ 5
ΥΛΟΠΟΙΗΣΗ ΠΑΙΧΝΙΔΙΟΥ

5.1 Εισαγωγή

Στην ενότητα αυτή θα εξετάσουμε την τεχνική αποτίμηση των διαφορετικών υπομονάδων της μηχανής που χρησιμοποιήθηκαν για την υλοποίηση του παιχνιδιού, θα παρουσιαστούν αναλυτικά τα στάδια της υλοποίησης καθώς και παραδείγματα που θα κάνουν την κατανόησή τους ευκολότερη.

5.2 Κίνηση κάμερας

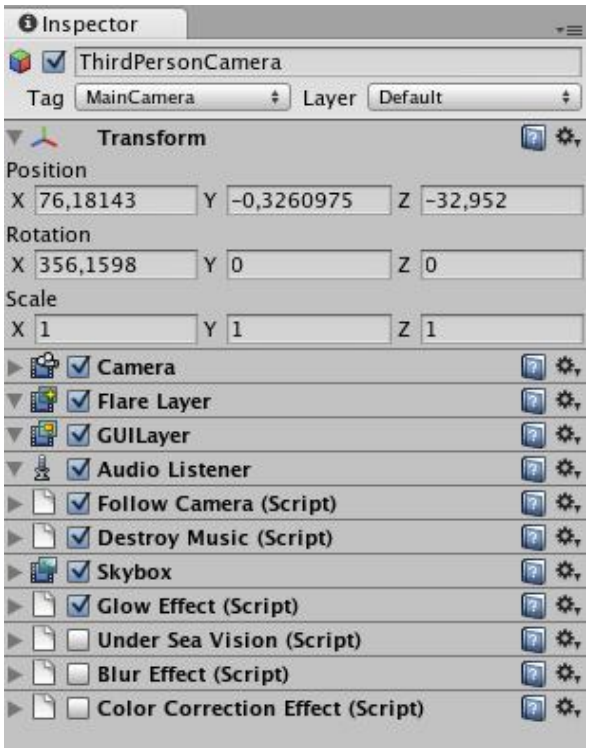
Αρχικά, αξίζει να σημειωθεί ότι για το συγκεκριμένο είδος παιχνιδιού η κίνηση της κάμερας είναι αρκετά σημαντική για την ροή του. Για την υλοποίησή της κίνησης παρατηρήθηκαν διάφορα παιχνίδια του είδους και καταλήξαμε στην χρησιμοποίηση της κίνησης κάμερας που είναι παρόμοια με αυτήν του παιχνιδιού Jak & Dexter (βλ. Εικόνα 41). Οι σχεδιαστές της κάμερας του συγκεκριμένου παιχνιδιού αντιμετωπίσανε το πρόβλημα του να μπορεί η κάμερα να ελίσσεται σε ένα πολύπλοκο τρισδιάστατο περιβάλλον και ταυτόχρονα, να συμπεριφέρεται διακριτικά και έξυπνα. Ο χαρακτήρας τους θα μπορούσε να πηδάει σε λόφους, δέντρα, θάμνους, να περνάει κάτω απο γέφυρες, να κρύβεται πίσω απο βράχους και όλα αυτά χωρίς να εμποδίζει την δράση απο το πεδίο θέασης του παίκτη [11] .

Αυτό μπορεί να ακουστεί αστείο, αλλά ένα σημαντικό χαρακτηριστικό της κάμερας είναι να μην ζαλίζει τον παίκτη. Αυτό είναι ένα σοβαρό πρόβλημα που έχουν κάμερες από άλλα παιχνίδια που εξετάστηκαν. Για να αποφευχθεί το φαινόμενο αυτό μειώσαμε όσο γίνεται τις περιστροφικές κινήσεις και αφήσαμε σε ένα βαθμό τον παίκτη να τις ελέγχει.

Εικόνα 41: Στιγμιότυπο απο το παιχνίδι Jak& Dexter

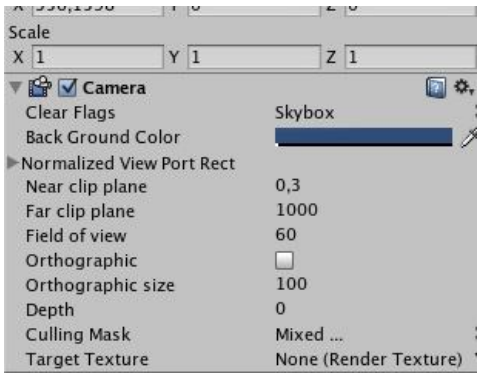
Προχωρώντας στην υλοποίηση της κάμερα, θα παρουσιαστεί το GameObject που ονομάζεται "Third Person Camera" (βλ. Εικόνα 42) και περιλαμβάνει:

- την κύρια υπομονάδα της κάμερας
- έναν AudioListener
- επίπεδο τύπου Flare
- επίπεδο τύπου GUI
- ένα Skybox
- 6 υλοποιήσεις που ελέγχουν την συμπεριφορά της



Εικόνα 42: Γενική εικόνα του GameObject της κάμερας

Η **κύρια υπομονάδα της κάμερας** (βλ. Εικόνα 43) έχει παραμετροποιηθεί ώστε να εμφανίζει το ορισμένο, απο τον σχεδιαστή, κουτί ουρανού («skybox») στα σημεία που ο χώρος είναι κενός, να μην είναι ορθογραφικού τύπου και να μην εμφανίζει κάποια επίπεδα που χρησιμεύουν για την εμφάνιση του χάρτη πραγματικού χρόνου της πίστας.



Εικόνα 43: Κύρια υπομονάδα κάμερας

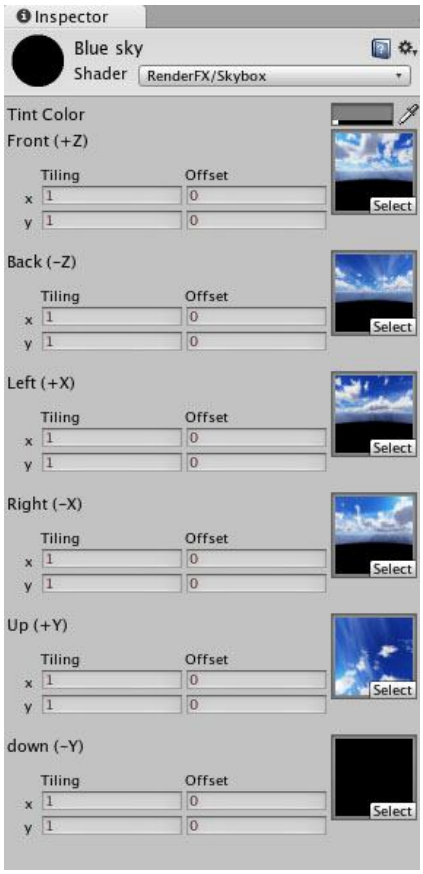
Ο AudioListener, το επίπεδο Flare, το επίπεδο GUI, έχουν τις προεπιλεγμένες ρυθμίσεις.

Ο **Audio Listener** συμπεριφέρεται σαν μια συσκευή μικρόφωνου. Μπορεί να δεχτεί είσοδο απο κάθε δεδομένη πηγή ήχου (Audio Source) στην σκηνή και να την αναπαράγει απο τα ηχεία του υπολογιστή. Είναι συνδεδεμένη με την κύρια κάμερα του παιχνιδιού.

Το **επίπεδο τύπου Flare** είναι συνδεδεμένο με την κάμερα για μπορεί να εμφανίζει εφφέ τύπου Lens Flare. Απο τις προεπιλεγμένες ρυθμίσεις, οι κάμερες έχουν ήδη συνδεδεμένο ένα τέτοιου τύπου.

Το **επίπεδο τύπου GUI** είναι συνδεδεμένο με την κάμερα και επιτρέπει την εμφάνιση των δυσδιάστατων γραφικών διεπαφών του χρήστη. Όταν ένα επίπεδο GUI είναι συνδεδεμένο με μια κάμερα έχει την δυνατότητα να σχεδιάζει τις GUI υφές και τα GUI κείμενα σε μια σκηνή.

Στην **υπομονάδα Skybox** έχουμε ορίσει το υλικό που θα χρησιμοποιεί η κάμερα ώστε να υλοποιήσει την τεχνική skybox και να γεμίζει το κενό χώρο της κάμερας με αυτό το υλικό. Αποτελείται απο 6 δυσδιάστατες εικόνες για κάθε πλευρά του κύβου (βλ. Εικόνα 44).



Εικόνα 44: Οι εικόνες που έχουν εισαχθεί για το skybox

Υλοποίηση ακολουθίας κάμερας (Follow Camera Implementation)

Η υλοποίησης ακολουθίας κάμερας που καθορίζει την κύρια συμπεριφορά της κάμερας του παιχνιδιού, εμπεριέχει 7 συναρτήσεις:

- Awake
- LateUpdate
- ApplySnapping
- AdjustLineOfSight
- ApplyPositionDamping
- SetUpRotation
- AngleDistance

Η **Awake** καλείται όταν φορτώνεται το παιχνίδι, μια και μοναδική φορά και ορίζει τον στόχο που έχει η κάμερα. Ο στόχος είναι τύπου "Character Controller", δηλαδή η υπομονάδα που ελέγχει τον χειρισμό του παίκτη.

Έπειτα, η **LateUpdate** καλείται σε κάθε καρέ μετά απο το κάλεσμα της Update συνάρτησης. Αυτή η σειρά παίζει ρόλο διότι χρειάζεται να εντοπιστούν τα αντικείμενα που έχουν μετακινηθεί μέσα στην Update συνάρτηση. Έτσι, ορίζετε το κέντρο και το κεφάλι του στόχου (κύριος χαρακτήρας). Έπειτα, ελέγχουμε τι ενέργεια γίνεται στο καρέ. Εάν κάνει άλμα τότε ορίζεται ξανά το ύψος του στόχου. Ελέγχουμε εάν ο χρήστης έχει ζητήσει να περιστραφεί η κάμερα και αναλόγως περιστρέφουμε ή μετακινούμε την κάμερα ομαλά. Τέλος, τοποθετείτε στην επιθυμητή τοποθεσία ο άξονας περιστροφής της κάμερας.

Οι συναρτήσεις ApplySnapping, ApplyPositionDamping και SetUpRotation καλούνται απο την LateUpdate.

Στην **ApplySnapping** γίνεται η μετακίνηση και η περιστροφή της κάμερας μέσω ομαλής κίνησης όταν η κάμερα χρειάζεται περιστροφή. Υλοποιείται με τις συναρτήσεις Math.SmoothDampAngle και Math.SmoothDamp που υπάρχει στο API της Unity3Dκαι είναι βασισμένες στο κεφάλαιο 1.10 του Game Programming Gems 4 [12]. Αλλάζουν την γωνία της κάμερας σταδιακά στον χρόνο, που έχει δοθεί σε μοίρες, μέχρι να φτάσει την επιθυμητή γωνία. Αντίστοιχα πράττουμε και με την θέση της κάμερας. Σταματάει να εκτελείται, όταν η κάμερα έχει ξεπεράσει την ελάχιστη απόσταση που επιτρέπεται να πλησιάσει στον στόχο. Τέλος, καλείται η **AdjustLineOfSight** και γίνεται έλεγχος ώστε να έχουμε οπτική επαφή με τον παίκτη ακόμα και όταν υπάρχουν εμπόδια μεταξύ κάμερας και στόχου.

Η συνάρτηση **ApplyPositionDamping** είναι υπεύθυνη για την ομαλή κίνηση της κάμερας σε μια σταθερή απόσταση απο τον στόχο της. Βασίζεται και αυτή στην συνάρτηση Math.SmoothDamp που εφαρμόζεται και στους τρεις άξονες. Τέλος, και εδώ καλείται η **AdjustLineOfSight**.

Η συνάρτηση **SetUpRotation** είναι υπεύθυνη για την τοποθέτηση του άξονα περιστροφής στο επιθυμητό μέρος, οποιαδήποτε στιγμή. Για παράδειγμα, όταν ο χαρακτήρας χοροπηδάει δεν κεντράρεται στην οθόνη, διότι θα πρέπει να δοθεί η αίσθηση του πόσο ψηλά πηδάει. Την ίδια στιγμή, θέλουμε οι περιστροφές του να είναι ομαλές και σε καμία περίπτωση να μην βγει ο χαρακτήρας εκτός οθόνης. Τοποθετείται η περιστροφή γύρω απο τον άξονα y. Όταν ο χαρακτήρας βρίσκεται στο

έδαφος είναι πάντα στο κέντρο της οθόνης και όταν είναι στον αέρα δεν μετακινείται η κάμερα εκτός εάν περάσει το κατώφλι που ορίζει το πεδίο θέασης της κάμερας. Τέλος όταν προσγειώνεται τότε περιστρέφουμε την κάμερα κεντράροντας τον πάλι.

Η συνάρτηση **AdjustLineOfSight** χρησιμοποιείται για να υπάρχει οπτική επαφή με τον παίκτη ακόμα και όταν βρίσκεται εμπόδιο μεταξύ αυτού και της κάμερας. Βασίζεται στην συνάρτηση Physics.Linecast. Η Physics.Linecast επιστρέφει αλήθεια όταν ένας συγκρουστής (collider) παρεμβάλλεται στην γραμμή μεταξύ των ορισμάτων αρχής και τέλους. Εάν επιστρέψει αλήθεια, γυρίσει πληροφορίες για τις συντεταγμένες που έγινε η σύγκρουση. Τέλος, μπορούμε να ορίσουμε και επίπεδα τα οποία δεν θέλουμε να ανιχνευτεί σύγκρουση.

Τέλος, η **AngleDistance** βρίσκει την διαφορά μεταξύ δύο γωνιών και χρησιμοποιείται για να σταματήσουμε την μετακίνηση και την περιστροφή της κάμερας. Καλείται απο την ApplySnapping.

Παρακάτω φαίνεται ο κώδικας της συνάρτησης **LateUpdate**:

```
function LateUpdate () {
	var targetCenter = target.position + centerOffset;
	var targetHead = target.position + headOffset;

	if (controller.IsJumping ())
	{
		var newTargetHeight = targetCenter.y + height;
		if (newTargetHeight < targetHeight || newTargetHeight - 
targetHeight > 5)
			targetHeight = targetCenter.y + height;
	}
	else
	{
		targetHeight = targetCenter.y + height;
	}

	if (Input.GetKeyDown("/") && !isSnapping)
	{
		velocity = Vector3.zero;
		isSnapping = true;
	}

	if (isSnapping)
	{
		ApplySnapping (targetCenter);
	}
	else
	{
		ApplyPositionDamping (Vector3(targetCenter.x, targetHeight, 
targetCenter.z));
	}

	SetUpRotation(targetCenter, targetHead);
}
```

Υλοποίηση καταστροφής μουσικής (Destroy Music Implementation)

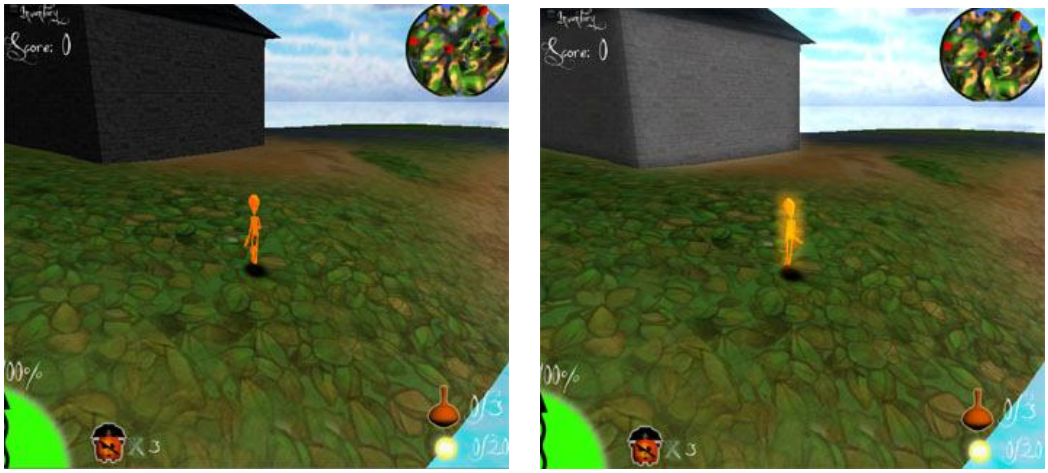
Όταν ξεκινάει το κύριο παιχνίδι πρέπει να σταματήσουμε οποιαδήποτε μουσική υπάρχει από τις οθόνες του μενού. Αυτό γίνεται με την υλοποίηση καταστροφής μουσικής. Έχει μια συνάρτηση Awake που καλείται στην πρώτη φορά που φορτώνεται το παιχνίδι και καταστρέφει το αντικείμενο, που περιέχει το αρχείο μουσικής για τις οθόνες του μενού.

Υλοποίηση όρασης κάτω απο το νερό

Στην εν λόγω υλοποίηση, υλοποιούμε την ορατότητα που θα έχει ο παίκτης όταν είναι κάτω απο το επίπεδο της θάλασσας. Ο χαρακτήρας ελέγχεται εάν θα περάσει ένα κατώφλι, που ορίζει το επίπεδο της θάλασσας. Όταν το περάσει τότε γνωρίζουμε ότι βρίσκεται κάτω απο την θάλασσα, θολώνουμε την κάμερα και «διορθώνουμε» το χρώμα ώστε να έχει ένα τόνο μπλε. Αυτό επιτυγχάνεται με το να ενεργοποιήσουμε τις αντίστοιχες υλοποιήσεις (Blur Effect, Color Correction Effect). Όταν ο παίκτης είναι πάνω απο την θάλασσα απενεργοποιούμε οι παραπάνω υλοποιήσεις και ο παίκτης βλέπει κανονικά.

Υλοποίηση εφφέ τύπου «λάμψης» (Glow Effect Implementation)

Η υλοποίηση είναι προεγκατεστημένη στην Unity3D και χρησιμοποιήθηκε για να δώσει μια πιο βελτιωμένη εμφάνιση στο παιχνίδι μας κάνοντας τις φωτεινές περιοχές της οθόνης να λάμπουν (βλ. Εικόνα 45).



Εικόνα 45: Χωρίς Glow Effect (αριστερή εικόνα), με Glow Effect (δεξιά εικόνα)

Η υλοποίησης της λάμψης χρησιμοποιεί το alpha κανάλι της τελικής εικόνας για αντιπροσωπεύσει την φωτεινότητα του χρώματος. Όλα τα χρώματα συμπεριφέρονται σαν RGB, και πολλαπλασιάζονται με το alpha κανάλι.

Υλοποίηση εφφέ θολώματος (Blur Effect Implementation)

Το εφφέ θολώματος (Blur effect) είναι προεγκατεστημένο στην Unity3D και χρησιμοποιήθηκε σε συνδυασμό με το εφφέ διόρθωσης χρώματος (Color Correction effect) για να δώσει την εντύπωση της υποθαλάσσιας όρασης.

Η υλοποίησης παίρνει τέσσερα δείγματα υφών και παίρνει την μέση τιμή όλων αυτών. Τα εφαρμόζει επανειλημμένα, διασκορπίζει τις τοποθεσίες των δειγμάτων και παίρνουμε την εκτίμηση Gaussian θολώματος.

Υλοποίησης εφφέ διόρθωσης χρώματος (Color Correction Effect Implementation)

Το εφφέ διόρθωσης χρώματος (Color Correction effect) είναι προεγκατεστημένο στην Unity3D και χρησιμοποιήθηκε σε συνδυασμό με το εφφέ θολώματος (Blur effect) για να δώσει την εντύπωση της υποθαλάσσιας όρασης. Γενικά, χρησιμοποιείται για να εφαρμοστεί διόρθωση χρώματος στην σκηνή.

Η διόρθωση χρώματος δουλεύει κάνοντας επαναχαρτογράφιση (remapping) της αρχικής εικόνας μέσω μια εικόνας τύπου ράμπας διαστάσεων 256x1 :

1. result.red = η τιμή κόκκινου του εικονοστοιχείου στην εικόνα τύπου ράμπα (original.red + RampOffsetR) του ευρετηρίου
2. result.green = η τιμή πράσινου του εικονοστοιχείου στην εικόνα τύπου ράμπα (original.green + RampOffsetG) του ευρετηρίου
3. result.blue = η τιμή μπλε του εικονοστοιχείου στην εικόνα τύπου ράμπα (original.blue + RampOffsetB) του ευρετηρίου

Για παράδειγμα, για να αναστρέψουμε τα χρώματα σε μια εικόνα το μόνο που χρειάζεται, είναι να αντιστρέψουμε την αρχική χρωματική ράμπα οριζοντίως (οπότε όπου βρισκόταν το λευκό θα είναι μαύρο και αντιστρόφως).

5.3 Σύστημα φυσικής

Όλο το παιχνίδι βασίζεται στην φυσική, λόγω του ελέγχου συγκρούσεων που έχουν τα αντικείμενα μεταξύ τους. Σε περίπτωση που δεν συνέβαινε αυτό, θα υπήρχαν αντικείμενα που το ένα θα εισχωρούσε στο άλλο χωρίς κανένα πρόβλημα. Για την υλοποίηση στην εφαρμογή μας χρησιμοποιούμε σε όλα τα αντικείμενα συγκρουστές. Οι συγκρουστές είναι είτε προσαρμοσμένοι στο αντικείμενο για μεγαλύτερη ακρίβεια υπολογισμών, είτε είναι απλά γεωμετρικά στέρεα (σφαίρα, κύβος, κύλινδρος).

Τα δέντρα φυσικής

Τα δέντρα φυσικής βρίσκονται στο παιχνίδι για να μπορεί ο παίκτης να βρίσκει ειδικά αντικείμενα. Η υλοποίησή του βασίζεται σε υπομονάδες της μηχανής φυσικής. Το κάθε δέντρο αποτελείται απο το κύριο μοντέλο του και μια σειρά που σφαίρες που παριστάνουν σχοινιά που κρέμονται στα κλαδιά του δέντρου. Κάθε σφαίρα μπορούμε να την δούμε ως ένα κρίκο αλυσίδας. Όταν χτυπηθεί το δέντρο απο τον παίκτη τα σχοινιά αρχίζουν να κινούνται λόγω της εφαρμογής μια δύναμης που έχει συνδεθεί με την ενέργεια της γροθιάς. Όταν δοθεί η απαιτούμενη δύναμη ο σύνδεσμος σπάει και το αντικείμενο αποκόπτεται απο το δέντρο και ο παίκτης μπορεί να το συλλέξει. Οι σφαίρες έχουν το δικό τους υλικό φυσικής (physic material). Το υλικό φυσικής έχει οριστεί έτσι ώστε η σφαίρα να μπορεί να χοροπηδά όταν πέφτει στο έδαφος, χωρίς η

κίνηση να φαίνεται υπερβολική. Για την δημιουργία υλικού φυσικής ορίζονται η στατική τριβή, η δυναμική τριβή και πόσο αναπηδητική είναι η επιφάνειά του.

Πιο αναλυτικά, το κάθε σχοινί είναι μια ομάδα απο σφαίρες που ενώνονται μεταξύ τους. Η κάθε σφαίρα έχει τις εξής υπομονάδες:

- Σφαιρικό συγκρουστή
- Άκαμπτο σώμα (rigidbody)
- Σύνδεσμος τύπου στρόφιγγα (hinge joint)

Το υλικό φυσικής έχει προστεθεί μόνο στους σφαιρικούς συγκρουστές που θα μπορούν να αποκολληθούν, για την αποφυγή περιττών υπολογισμών.

Αρχικά, το άκαμπτο σώμα δεν έχει καμία βραδύτητα, αλλά καθώς ο χρόνος περνάει. προστείθετω μια μικρή βραδύτητα στην κίνηση και στην περιστροφή.

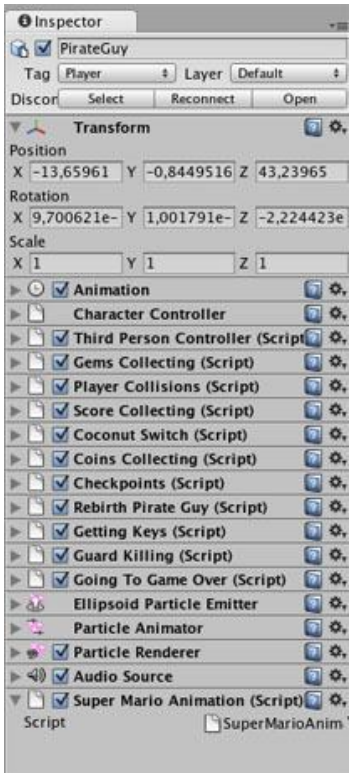
Ο σύνδεσμος τύπου στρόφιγγα έχει οριστεί έχοντας άξονα περιστροφής τον Z και σημείο εφαρμογής το κέντρο του συνδεδεμένου σώματος. Στις σφαίρες που μπορούν να αποκολληθούν απο το σχοινί έχει οριστεί η δύναμη που χρειάζεται, ώστε να γίνει η αποκόλληση.

Γενικά, η φυσική είναι ένα πολύ ενδιαφέρον κομμάτι στα σημερινά παιχνίδια, διότι δίνει στον παίκτη την εντύπωση ότι αλληλεπιδρά πραγματικά με ότι βλέπει στην οθόνη του. Κάθε φορά που εισάγεται ένα στοιχείο φυσικής, συνήθως χρειάζεται μεγάλη βελτίωση σε πολλούς τομείς ώστε να έχει την επιθυμητή συμπεριφορά.

5.4 Χειρισμός παίκτη

Η υλοποίηση του χειρισμού του παίκτη, γίνεται μέσω του GameObject που ονομάζεται “Third Person Controller” (βλ. Εικόνα 46) και περιλαμβάνει:

- Επίπεδο για την κίνηση του παίκτη
- Επίπεδο χειρισμού του χαρακτήρα
- Επίπεδο για τους ήχους που προέρχονται απο τον παίκτη
- Επίπεδο για τον εκπομπό σωματιδίων
- Επίπεδο για την κίνηση των σωματιδίων
- Επίπεδο για την σχεδίαση των σωματιδίων
- Οι υλοποιήσεις για τον χειρισμό του παίκτη και τις αλληλεπιδράσεις με τον περιβάλλον του.



Εικόνα 46: Γενική εικόνα του GameObject του χειρισμού του παίκτη

Η **υπομονάδα για την κίνηση (animation)** του χαρακτήρα χρησιμοποιείται για την εισαγωγή και την διαχείριση των κινήσεων του χαρακτήρα. Θα εξηγηθεί περαιτέρω στο 5.8.

Η **υπομονάδα χειρισμού του χαρακτήρα (character controller)** χρησιμοποιείται, κυρίως, για την υλοποίηση του είδος χειρισμού «τρίτου προσώπου» ή του «πρώτου προσώπου» και δεν κάνει χρήση της φυσικής των δύσκαμπτων σωμάτων (rigidbody physics) . Μέσω της εν λόγω υπομονάδας γίνεται η παραμετροποίηση του χειριστή ώστε να ταιριάζει με το δικό μας χαρακτήρα. Έτσι, επιτυγχάνεται η σωστή αλληλεπίδραση με το περιβάλλον του και ο έλεγχος των συγκρούσεων.

Η **υπομονάδα ελλειψοειδούς εκπομπού σωματιδίων (particle emitter)** χρησιμοποιείται για την παραγωγή σωματιδίων εσωτερικά μιας σφαίρας. Αλλάζοντας την παράμετρο «Ελλειψοειδές» μπορούμε να μεγεθύνουμε και να τεντώσουμε την σφαίρα. Μπορούμε να ελέγχουμε τον εκπομπό εάν θα είναι ενεργός ή όχι, το μέγεθος, τον αριθμό, την ενέργεια των σωματιδίων, την ταχύτητα και διάφορες άλλες παραμέτρους που προσάπτουν τυχαιότητα στην εκπομπή των σωματιδίων.

Η **υπομονάδα του σωματιδιακού κινητή (particle animator)** χρειάζεται για να μετακινηθούν τα σωματίδια κατά την διάρκεια του χρόνου και να εφαρμόσουμε δυνάμεις σε αυτά και να εφαρμόσουμε χρωματικούς κύκλους**.**

Για να ολοκληρωθεί η εισαγωγή σωματιδίων θα πρέπει να σχεδιαστούν και στην οθόνη. Συνεπώς, θα χρειαστεί και μια **υπομονάδα σχεδιαστή σωματιδίων (particle renderer).** Σε αυτήν την υπομονάδα εισάγουμε την μορφή που θα έχουν τα σωματίδια μας, με ποιο τρόπο θα γίνει η σχεδίασή τους και εάν οι υφές, που έχουν τα υλικά μας είναι κινούμενες ή όχι.

Η **υπομονάδα AudioSource** χρησιμοποιείται για την αναπαραγωγή ηχητικών αρχείων απο μια συγκεκριμένη τοποθεσία. Ελέγχεται πότε θα γίνει η αναπαραγωγή τους, η έντασή τους, ο τόνος, πόσο θα εξασθενεί και αν θα επαναλαμβάνεται ή όχι.

Υλοποίηση χειρισμού τρίτου προσώπου (Third Person Controller Implementation)

Εδώ υλοποιείται ο χειρισμός του χαρακτήρα. Η πιο σημαντική υλοποίηση για τον χειρισμό του παίκτη. Αποτελείται απο τις εξής συναρτήσεις:

- Update
- UpdateSmoothedMovementDirection
- DidPunch
- cameraShake

Η **συνάρτηση Update** έχει ως εξής: Πρώτα, γίνονται ο ορισμός της κατεύθυνσης και των αξόνων που θα πραγματοποιηθεί η κίνηση, μέσω της συνάρτησης UpdateSmoothedMovementDirection (βλ. παρακάτω). Έπειτα, ελέγχουμε εάν ο παίκτης είναι στο έδαφος για να ορίσουμε την κάθετη ταχύτητα σε αυτόν τον έλεγχο, γίνεται και δεύτερος εμφωλευμένος για το εάν ο παίκτης μπορεί να κάνει άλμα.

Έπειτα, εφαρμόζουμε την βαρύτητα στην κάθετη ταχύτητα ώστε να προσγειωθεί απο το άλμα του. Όταν φεύγουμε απο αυτούς του ελέγχους υπολογίζουμε την κίνηση του. Για τον υπολογισμό της κίνησης πολλαπλασιάζουμε την κατεύθυνση της κίνησης επί την ταχύτητα και το αποτέλεσμα το προσθέτουμε στην όποια κάθετη ταχύτητα έχει, σε περίπτωση που είναι σε άλμα. Έπειτα, περνάμε ως όρισμα το διάνυσμα της κίνησης στην συνάρτηση controller.Move. Η συνάρτηση Move κάνει το περιορισμό της κίνησης ανάλογα με τις σημαίες συγκρούσεων (collision flags). Έπειτα, περιστρέφεται ο χαρακτήρας προς την κατεύθυνση που μετακινείται.

Ακολουθεί ο κώδικας της Update :

```
function Update() {

	UpdateSmoothedMovementDirection();

	if (grounded) {
		verticalSpeed = 0.0;

		// Jump
		if (canJump && Input.GetButton ("Jump")) {
			verticalSpeed = jumpSpeed;
			jumping = true;
			SendMessage("DidJump",
SendMessageOptions.DontRequireReceiver);
		}
	}
	// Apply gravity
	verticalSpeed -= gravity * Time.deltaTime;

	var movement = moveDirection * moveSpeed + Vector3 (0,
verticalSpeed, 0);
	movement *= Time.deltaTime;
```

```
        // Move the controller
        var controller : CharacterController =
GetComponent(CharacterController);
        var flags = controller.Move(movement);
        grounded = (flags & CollisionFlags.CollidedBelow) != 0;

        // Set rotation to the move direction
        transform.rotation = Quaternion.LookRotation(moveDirection);

        // We are in jump mode but just became grounded
        if (grounded && jumping)
        {
                jumping = false;
                SendMessage("DidLand",
SendMessageOptions.DontRequireReceiver);
        }
}
```

Στην συνάρτηση **UpdateSmoothedMovementDirection** ορίζουμε τις κατευθύνσεις ως προς την κάμερα. Ορίζεται η κατεύθυνση και η ταχύτητα ανάλογα με την είσοδο που δίνει ο χρήστης απο το πληκτρολόγιο. Και εξομαλύνεται η κίνηση και περιστροφή του παίκτη σε σχέση με την κατεύθυνση της κάμερας.

Με την **συνάρτηση cameraShake** υλοποιούμε την κίνηση της κάμερας όταν ο παίκτης χτυπήσει ένας αντίπαλό του. Μπορούμε να κάνουμε όσες μετακινήσεις στην κάμερα θέλουμε και για μεγάλη απόσταση.

Όταν, ο παίκτης χτυπήσει τον αντίπαλο προσθέτετε περιοδική κίνηση στον άξονα Υ με κάποια συχνότητα. Όταν τελειώσει μια περίοδος μικραίνουμε την περίοδο και ελαττώνουμε τις φορές που θα κινηθεί η κάμερα. Στο τέλος επαναφέρουμε την κάμερα στην αρχική θέση της.

Η **συνάρτηση DidPunch** εκτελείται κάθε φορά που ο παίκτης δίνει είσοδο το πλήκτρο της γροθιάς. Αρχικοποιούμε δύο λίστες που περιλαμβάνουν τα αντικείμενα των δέντρων φυσικής και των εχθρών. Έχουμε ορίσει την ακτίνα που θα καταλαμβάνει η γροθιά του παίκτη. Όταν ο παίκτης έρθει αρκετά κοντά στο είδος των αντικείμενων που έχουμε βάλει στις λίστες, τότε στέλνουμε τα μηνύματα προς αυτά. Για παράδειγμα στο αντικείμενο του αντίπαλου λέμε να κάνουμε ελάττωση στην ενέργεια του και στο δέντρο φυσικής να εφαρμοστεί μια δύναμη πάνω του.

Ο κώδικας για την DidPunch συνάρτηση:

```
function DidPunch()
{       yield WaitForSeconds(punchHitTime);
        var i=0;
        var pos = transform.TransformPoint(punchPosition);
        var enemies : GameObject[] =
GameObject.FindGameObjectsWithTag("EnemyGuy");
        var
trees:GameObject[]=GameObject.FindGameObjectsWithTag("PhysicTree"); //for
the trees
```

```
        for (var go : GameObject in enemies)
        {
                var enemy = go.GetComponent(EnemyDamage);
                print(Vector3.Distance(enemy.transform.position, pos));

                if (enemy == null)
                 continue;

                if (Vector3.Distance(enemy.transform.position, pos) <
punchRadius)
                {
                        enemy.SendMessage("ApplyDamage", punchHitPoints);
                        cameraShake();
                        // Play sound.
                        if (punchSound)
                                audio.PlayOneShot(punchSound);
                }
        }

                for (var gmobject : GameObject in trees)
        {
                var tree = gmobject.GetComponent(ApplyForceTree);

                if (tree == null)
                 continue;
                if (Vector3.Distance(tree.transform.position, pos) <
punchRadius)
                {
                        tree.SendMessage("ApplyFrcTree");
                        cameraShake();
                }
        }
        yield WaitForSeconds(punchTime - punchHitTime);
        busy = false;

}
```

Υλοποίηση συγκρούσεων παίκτη (Player Collisions Implementation)

Με αυτή την υλοποίηση καθορίζεται η συμπεριφορά που έχει ο χαρακτήρας όταν συγκρούεται με συγκεκριμένα αντικείμενα. Έχει τις παρακάτω συναρτήσεις:

- Awake
- OnControllerColliderHit
- showMalamoGui
- showMalamoInDangerWindow
- showInstructions
- openDoor

Οι συναρτήσεις showMalamoGui, showMalamoInDangerWindow και showInstructions είναι για την γραφική διεπαφή του χρήστη και θα εξηγηθούν στο κεφάλαιο 5.9.

Στην **συνάρτηση Awake** γίνεται η αρχικοποίηση των μεταβλητών ώστε κάθε φορά που ξεκινάει η σκηνή να μην έχουν κρατηθεί προηγούμενες τιμές. Σημαντική λειτουργία για κάθε επανεκκίνηση του παιχνιδιού.

Η **συνάρτηση OnControllerColliderHit** καλείται κάθε φορά που ο ελεγκτής (controller) χτυπήσει ένα συγκρουστή ενώ εκτελεί μετακίνηση (Controller.Move ( )). Επιστρέφει το σημείο που έγινε η σύγκρουση και τι είδους ήταν το αντικείμενο που χτυπήθηκε. Τα είδη των αντικειμένων τα ορίζουμε με «tags». Στην συγκεκριμένη συνάρτηση ελέγχετε εάν έχει καταστραφεί ο συγκεκριμένος αριθμός φρουρών και έπειτα εάν έχουμε τα κλειδιά για να ανοίξουμε τις πόρτες των αποθηκών. Ακόμα εάν έχουμε το αντικείμενο «Μαγικό φίλτρο» τότε σταματάει να κυλάει η αντίστροφη μέτρηση και στέλνεται μήνυμα ότι ο παίκτης έχει κερδίσει το παιχνίδι.

Ακολουθεί ο κώδικάς του:

```
function OnControllerColliderHit(hit:ControllerColliderHit)
{
    if(GuardKilling.killedGuards)
     {
        if((hit.gameObject.tag=="warehouse1door")&&(hasKey1))
              {
                    if(!door1Opened)
                    {
                          openDoor1(hit.gameObject);
                          SendMessage("showMalamoGui");
                          closeButtonInWindow=false;
                          doWindowMalamoInDanger=true;
                          print("Warehouse 1 door opened");
                    }
              }
        if((hit.gameObject.tag=="warehouse2door")&&(hasKey2))
              {
                    if(!door2Opened)
                    {
                    openDoor2(hit.gameObject);
                    print("Warehouse 2 door opened");
                    }
              }
        if((hit.gameObject.tag=="warehouse3door")&&(hasKey3))
              {     if(!door3Opened)
                    {
                    openDoor3(hit.gameObject);
                    print("Warehouse 3 door opened");
                    }
              }
        if((hit.gameObject.tag=="warehouse4door")&&(hasKey4))
              {     if(!door4Opened)
                    {
                          openDoor4(hit.gameObject);
                          print("Warehouse 4 door opened");
                    }
              }
        if((hit.gameObject.tag=="Malamo")&&(GettingKeys.magicalFilter))
              {
```

```
                TimeCountdown.stopCountDown();
                SendMessage("winningGame");
        }
    }
```

Στην **συνάρτηση openDoor** παίζεται η κίνηση της πόρτας, για να γίνει ανοικτή, και ο ήχος που κάνει όταν ανοίγει. Τέλος, χρησιμοποιούμε μία λογική μεταβλητή ώστε να γνωστοποιηθεί ότι η πόρτα έχει ανοίξει.

Υλοποίηση καταγραφής της βαθμολογίας (Score Collecting Implementation)

Υλοποίηση καταγραφής της βαθμολογίας του παίκτη. Χρησιμοποιείται και εδώ η συνάρτηση OnControllerColliderHit. Μια μεταβλητή αποθηκεύει τους πόντους του παίκτη. Έχουμε εισάγει ήχους για να παίζουν όταν συλλέγουμε ένα αντικείμενο. Έχουμε ορίσει να παίρνουμε 150 πόντους για τα αντικείμενα τύπου «ρούμι» και 50 πόντους για τα αντικείμενα τύπου «νόμισμα».

Υλοποίησης συγκομιδής και καταγραφής των «νομισμάτων» (Coins Collecting Implementation)

Υλοποίηση συγκομιδής και καταγραφής των αντικειμένων που βρίσκονται στο περιβάλλον. Η υλοποίηση γίνεται με την συνάρτηση OnControllerColliderHit. Κάθε φορά που ο παίκτης έρχεται σε επαφή με το αντικείμενο τύπου «Νόμισμα» (Coin) εκπέμπονται σωματίδια απο αυτό και μετά καταστρέφεται. Πριν βγει απο την συνάρτηση, προσθέτει και μια μονάδα στην τιμή της μεταβλητής που αποθηκεύεται ο αριθμός των νομισμάτων.

Ο κώδικας της OnControllerColliderHit της CoinsCollecting υλοποίησης:

```
function OnControllerColliderHit (hit:ControllerColliderHit) {
    if(hit.gameObject.tag=="coins1")
    {   CountCoins++;
        hit.gameObject.GetComponent(ParticleEmitter).emit=true;
        hit.gameObject.GetComponent(MeshRenderer).enabled=false;
        hit.gameObject.GetComponent(MeshCollider).isTrigger=true;

    hit.gameObject.transform.Find("CoinGlowPlane").GetComponent(MeshRe
nderer).enabled=false;
        hit.gameObject.GetComponent(MeshCollider).isTrigger=true;
        yield WaitForSeconds(0.5);
        hit.gameObject.GetComponent(ParticleEmitter).emit=false;
        yield WaitForSeconds(2);
        Destroy(hit.gameObject,1);
        coinsUp=1;
        resetTime=1;
        hittingTime=Time.time;

    }
}
```

Υλοποίηση σημείων ελέγχου (Checkpoints Implementation)

Υλοποίηση δυνατότητας του παίκτη να εμφανίζεται σε συγκεκριμένα σημεία στο περιβάλλον όταν «χάνει» μια ζωή. Τα σημεία ελέγχου είναι αντικείμενα που μετακινούνται προς τα κάτω όταν χαρακτήρας πατήσει πάνω τους για πρώτη φορά. Το τελευταίο σημείο που θα πατηθεί θα είναι το σημείο που θα αναγεννηθεί ο χαρακτήρας. Στην υλοποίηση σημεία ελέγχου (checkpoints implementation) υπάρχουν οι παρακάτω συναρτήσεις:

- Awake
- Update
- OnControllerColliderHit
- SetActive
- SetInactive
- Rebirth

Στην **συνάρτηση Awake** γίνεται η αρχικοποίηση της πρώτης θέσης που θα γίνει η αναγέννηση του παίκτη, ορίζουμε τον εκπομπό σωματιδίων να είναι ανενεργός και το φως που είναι πάνω απο το σημείο ελέγχου (checkpoint) να είναι και αυτό ανενεργό.

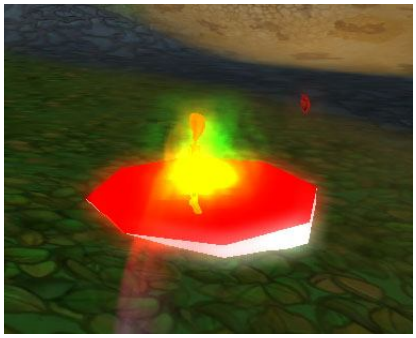
Στην **συνάρτηση Update** που καλείται σε κάθε καρέ ελέγχουμε εάν το σημείο ελέγχου χρειάζεται να αλλάξει την τιμή του Z άξονα, δηλαδή να «πατηθεί» και εάν ο χαρακτήρας έχει χάσει ζωή. Εάν έχει χάσει ζωή τότε τον μεταφέρουμε στο τελευταίο ενεργό σημείο ελέγχου και αναγεννάτε, καλώντας την συνάρτηση rebirth.

Στην **συνάρτηση OnControllerColliderHit** όταν ο παίκτης έρθει σε επαφή με το αντικείμενο τύπου «Σημείο Ελέγχου» (checkpoint) και το αντικείμενο είναι διαφορετικό απο αυτό που είχε περάσει πριν, τότε απενεργοποιούμε το προηγούμενο και ενεργοποιούμε το τρέχον.

Ο κώδικας της OnControllerColliderHit:

```
function OnControllerColliderHit (hit:ControllerColliderHit)
{
	if(hit.gameObject.tag=="checkpoint")
	{
		if (check!=hit.gameObject)
		{
			SetInactive ();
			check = hit.gameObject;
			SetActive ();
		}
	}
}
```

Η **συνάρτηση SetActive** είναι υπεύθυνη για την ενεργοποίηση ενός σημείο ελέγχου. Περιλαμβάνει την ενεργοποίηση του εκπομπού σωματιδίων, του φωτός πάνω απο αυτό το σημείο (βλ. Εικόνα 47) για συγκεκριμένη χρονική διάρκεια και το ορίζει ως ενεργό.



Εικόνα 47: Ο εκπομπός σωματιδίων πάνω απο ένα σημείο ελέγχου

Στην **συνάρτηση rebirth** μεταφέρουμε τον παίκτη στο συγκεκριμένο σημείο ελέγχου, παίζεται το ηχητικό κομμάτι και τέλος, ενεργοποιούμε τα σωματίδια και το φως που προέρχονται απο τον παίκτη για συγκεκριμένη χρονική διάρκεια.

Υλοποίηση εξουδετέρωσης φρουρών (Guard Killing Implementation)

Χρησιμοποιείται για την υλοποίηση ενός μέρος του σεναρίου. Ορίζονται δύο φύλακες που είναι στην είσοδο του νησιού. Στην συνάρτηση Update όταν σκοτωθούν οι φύλακες απο τον παίκτη ενημερώνεται η αντίστοιχη λογική μεταβλητή και εμφανίζεται το μήνυμα στον χρήστη ότι τους έχει σκοτώσει και μπορεί να προχωρήσει.

Ο κώδικας της συνάρτησης Update( ) :

```
function Update()
{
	if(!guard1&&!guard2)
	{
		killedGuards=true;
		if(!MessageShowed)
		{
			SendMessage("ShowGuardsGui");
			MessageShowed=true;
		}
	}
}
```

Υλοποίηση ικανότητας συλλογής κλειδιών (Getting Keys Implementation)

Εδώ υλοποιείται η ικανότητα του παίκτη να συλλέγει τα κλειδιά. Στο παιχνίδι υπάρχει συγκεκριμένος αριθμός κλειδιών. Ο παίκτης όταν έρχεται σε επαφή, με κάποιο από αυτά θέτει σε αναπαραγωγή τον ήχο που αντιστοιχεί, θέτει σε λειτουργία τον σωματιδιακό εκπομπό και τέλος κάνει αλήθεια μια λογική μεταβλητή που θα χρησιμοποιηθεί αργότερα για την ανανέωση του ευρετηρίου.

Υλοποίηση Going To Game Over

Υλοποίηση της ικανότητας του παίκτη να χάνει ενέργεια απο τους αντιπάλους όταν τον χτυπάνε. Ο χαρακτήρας χάνει ενέργεια όταν συγκρούεται με τα αντικείμενα τύπου «Καρύδα» και όταν εισέρχεται στο ηφαίστειο. Έχει τις εξής συναρτήσεις:

- OnCollisionEnter
- Update
- GameOver
- LosingLife

Στην **συνάρτηση OnCollisionEnter** εισέρχεται κάθε φορά που ένας συγκρουστής ξεκινήσει να έρχεται σε επαφή με ένα άλλο αντικείμενο ή άκαμπτο σώμα. Εάν είναι τύπου «Καρύδα» το αντικείμενο, τότε η τιμή της μεταβλητής που κρατάει την ενέργεια του παίκτη μειώνεται κατά ένα συγκεκριμένο αριθμό.

Στην **συνάρτηση Update** που καλείται σε κάθε καρέ ελέγχεται εάν η μεταβλητή που κρατάει την ενέργεια του παίκτη πέσει κάτω απο το μηδέν, τότε ο αριθμός των προσπαθειών (ζωές) μειώνεται κατά μια μονάδα. Τέλος, εάν ο παίκτης δεν έχει άλλη προσπάθεια έχουμε τερματισμό του παιχνιδιού (Game Over).

Στην **συνάρτηση GameOver** επιλέγουμε να μην καταστρέψουμε το όνομα του παίκτη στην επόμενη σκηνή και έπειτα, κάνουμε την εναλλαγή. Αυτό γίνεται διότι, κανονικά όλα τα αντικείμενα καταστρέφονται στην εναλλαγή απο την μία σκηνή στην άλλη.

Οι παραπάνω ενέργειες γίνονται με τις συναρτήσεις DontDestroyOnLoad ( ) και την Application.LoadLevel ( ).

Η λειτουργικότητα της **συνάρτησης LosingLife** είναι να γνωστοποιήσει στην «Checkpoints» υλοποίηση ότι ο παίκτης έχει χάσει μια «ζωή». Έπειτα, η υλοποίηση «Checkpoints» αναλαμβάνει τα υπόλοιπα που εξηγήθηκαν παραπάνω.

5.5 Τεχνητή νοημοσύνη

Η Unity3D δεν έχει κάποια υπομονάδα για την διαχείριση τεχνητής νοημοσύνης των αντιπάλων. Στο παιχνίδι, βάσει σχεδιασμού χρειάζονται κάποιοι αντίπαλοι για την βελτίωση του gameplay και την αύξηση της δυσκολίας. Έτσι, οι αντίπαλοι θα πρέπει να μπορούν να κινηθούν στον χώρο και να έχουν μια προσαρμοζόμενη συμπεριφορά, για τις καταστάσεις που προκύπτουν κατά την διάρκεια του παιχνιδιού.

5.5.1 Τρόπος προσέγγισης και τελική υλοποίηση

Η προσέγγιση που έγινε στο πρόβλημα της τεχνητής νοημοσύνης των αντιπάλων είναι μια αρκετά γνωστή τεχνική που χρησιμοποιείται στην ανάπτυξη παιχνιδιών. Υλοποιήθηκε η εισαγωγή σημείων διαδρομής (waypoints) για να μπορούν οι αντίπαλοι χαρακτήρες να κινούνται στο χώρο και ένα μοντέλο συμπεριφοράς για την συμπεριφορά τους, ανάλογα με τα γεγονότα που συμβαίνουν στο παιχνίδι.

5.5.2 Μοντέλο συμπεριφοράς του αντίπαλου χαρακτήρα

Ο αντίπαλος μπορεί να είναι σε τρεις κύριες καταστάσεις:

1. Ανενεργή

2. Υποψιασμένη
3. Συναγερμού

Αυτές οι καταστάσεις αλλάζουν ανάλογα με την απόσταση του στόχου -τον χαρακτήρα του παίκτη- απο τον αντίπαλο. Έπειτα, έχουν τοποθετηθεί ξεχωριστοί συγκρουστές στον κάθε αντίπαλο για τον εντοπισμό του παίκτη. Έτσι, δεν υπάρχει σύγκρουση μεταξύ του στόχου και του αντιπάλου η κατάσταση που είναι ενεργή είναι η «ανενεργή». Όταν υπάρχει σύγκρουση και η απόσταση είναι μικρότερη απο ένα κατώφλι τότε βρισκόμαστε στην υποψιασμένη κατάσταση. Τέλος, όταν ο αντίπαλος είναι πολύ κοντά στον στόχο τότε η κατάσταση που είναι ενεργή είναι η κατάσταση συναγερμού.

Στην **ανενεργή κατάσταση** ο αντίπαλος έχει την δυνατότητα να κάνει τρεις ενέργειες. Αυτές οι ενέργειες εναλλάσσονται με συγκεκριμένη πιθανότητα βασισμένη στο gameplay. Για παράδειγμα, η πιο σημαντική ενέργεια που είναι η περιπολία συμβαίνει εφτά στις δέκα φορές, η ενέργεια του καπνίσματος δυο στις δέκα και τέλος, μια στις δέκα η ενέργεια του ύπνου.

Κατά την ενέργεια της περιπολίας έχουμε την εξής λογική: Έχουν οριστεί τα σημεία διαδρομής και με ποια σειρά έχουν συνδεθεί. Είναι σαν μια συνδεδεμένη λίστα. Ξεκινώντας από την αρχή των σημείων διαδρομής της περιπολίας, ο κάθε αντίπαλος θα ακολουθεί την διαδρομή του. Κάθε φορά που ένας αντίπαλος φτάνει στο σημείο που κατευθύνεται, τότε ορίζεται καινούριο.

Κατά την διάρκεια της ενέργειας του καπνίσματος και του ύπνου, ο αντίπαλος παραμένει ακίνητος και προβάλλεται το εικονίδιο αυτής.

Παράδειγμα κώδικα για την ανενεργή κατάσταση:

```
function idle(){
	SendMessage("StopThrowingCoconut");
	if (actionPos>=0&&actionPos<0.6&&!locked)// 70 % of times
patroling
	{
		SendMessage("StartPatrol");
		if(!doingPatrol)
		{
			doPatrol();
			Patrol();
		}
		doingPatrol=true;
		locked=true;
	}
	else if(actionPos>0.6&&actionPos<0.8&&!locked)// 20% of times
smoking
	{	SendMessage("StopPatroling");
		doSmoking();
		locked=true;

	}
	else if (actionPos>0.8&&!locked)// 10% of times sleeping
	{
		SendMessage("StopPatroling");
```

```
            doSleeping();
            locked=true;
        }
}
```

Στην **υποψιασμένη κατάσταση** ο αντίπαλος θα ακολουθήσει και θα προσπαθήσει να τον φτάσει. Ο κάθε αντίπαλος έχει ένα συγκρουστή που είναι για την ανίχνευση του στόχου του. Όταν ο παίκτης βρεθεί μέσα στην περιοχή που ορίζεται απο τον συγκρουστή, η υποψιασμένη κατάσταση γίνεται ενεργή. Στην υποψιασμένη κατάσταση ο αντίπαλος θα αρχίσει να κυνηγά τον στόχο του. Αυτό γίνεται ορίζοντας την θέση του στόχου ως το σημείο προς το οποίο θα κινηθεί ο παίκτης. Χρησιμοποιείται η ίδια συνάρτηση που χρησιμοποιήθηκε και για την κίνηση προς ένα σημείο διαδρομής.

Κώδικας συνάρτησης για κίνηση προς ένα σημείο:

```
function MoveTowards (position : Vector3) {
        var direction = position - transform.position;
        direction.y = 0;

        // Rotate towards the target
        transform.rotation = Quaternion.Slerp (transform.rotation,
Quaternion.LookRotation(direction), rotationSpeed * Time.deltaTime);
        transform.eulerAngles = Vector3(0, transform.eulerAngles.y, 0);

        // Modify speed
        var forward = transform.TransformDirection(Vector3.forward);
        var speedModifier = Vector3.Dot(forward, direction.normalized);
        speedModifier = Mathf.Clamp01(speedModifier);

        // Move the character
        direction = forward * speed * speedModifier;
        GetComponent (CharacterController).SimpleMove(direction);
        SendMessage("SetSpeed", speed * speedModifier,
SendMessageOptions.DontRequireReceiver);
}
```

Στην **κατάσταση συναγερμού** ο αντίπαλος θα ρίχνει βολές κατά του χαρακτήρα του παίκτη. Όταν ο παίκτης φτάσει σε αρκετά κοντινή απόσταση απο τον αντίπαλο τότε ενεργοποιείται η κατάσταση του συναγερμού. Σε αυτήν κατάσταση ο αντίπαλος περιστρέφεται προς το σημείο που είναι ο παίκτης και αρχίζει να του ρίχνει βολές με συγκεκριμένη ταχύτητα. Η κάθε βολή είναι ένα αντικείμενο που έχει την υπομονάδα του άκαμπτου σώματος (rigidbody), οπότε η συμπεριφορά της είναι σύμφωνη με νόμους της φυσικής. Οι βολές αυτές εάν έρθουν σε επαφή με τον παίκτη του προκαλούν μείωσης ενέργειας.

Τέλος, κάθε φορά που ο στόχος του αντιπάλου απομακρύνεται, ο αντίπαλος γυρνάει στην ανενεργή κατάσταση και στα σημεία που έκανε την περιπολία του.

5.6 Γραφικά

5.6.1 Σκιαστές που χρησιμοποιήθηκαν

Στην πλειονότητα των περιπτώσεων χρησιμοποιήθηκαν οι σκιαστές που έρχονται προεγκατεστημένοι με την Unity. Σε μερικές περιπτώσεις χρειάστηκε και οι χρησιμοποίηση σκιαστών οι οποίοι είναι δεν είναι προεγκατεστημένοι αλλά έχουν κατασκευαστεί απο χρήστες ως πρόσθετοι.

Αναφορά στους προεγκατεστημένους σκιαστές που χρησιμοποιήθηκαν

Σκιαστής διάχυσης (Diffuse)

Ο σκιαστής διάχυσης υπολογίζει ένα απλό φωτιστικό μοντέλο (Lambertian). Ο φωτισμός σε μια επιφάνεια ελαττώνεται καθώς η γωνία μεταξύ αυτού και του φωτός ελαττώνεται. Ο φωτισμός εξαρτάται μόνο απο αυτήν την γωνία και δεν αλλάζει όταν η κάμερα μετακινείται ή περιστρέφεται. Είναι σκιαστής φωτισμένος ανά εικονοστοιχείο και το πιο φτηνό μοντέλο σκιαστή σε αυτήν την κατηγορία.

Σκιαστής διαφάνειας φωτισμένος ανά κορυφή

Αυτός ο σκιαστής μπορεί να κάνει το δικτύωμα της γεωμετρίας να είναι μερικώς ή εντελώς διαφανές διαβάζοντας το κανάλι άλφα της κύριας υφής. Το μαύρο είναι εντελώς διάφανο και το άσπρο εντελώς αδιαφανές. Εάν η κύρια υφή δεν έχει άλφα κανάλι το αντικείμενο θα είναι εντελώς αδιαφανές. Όντας φωτισμένος ανά κορυφή είναι απο τους πιο φτηνούς και πιο απλούς σκιαστές. Όποια φώτα φωτίζουν πάνω του σχεδιάζονται με ένα πέρασμα για αυτό και είναι γρηγορότερος απο τον σκιαστή φωτισμένος ανά εικονοστοιχείο. Ακόμα και τα φώτα εικονοστοιχείου θα σχεδιαστούν σαν φώτα κορυφής όταν χρησιμοποιείται αυτός ο σκιαστής.

Σκιαστής φύσης τύπου ομαλής έμφραξης φύλλων και ομαλής έμφραξης κορμού (Soft occlusion leaves, Soft occlusion bark)

Ο σκιαστής φύσης τύπου ομαλής έμφραξης φύλλων και κορμών χρησιμοποιείται απο την μηχανή του εδάφους και μπορεί να κάνει τα φύλλα και το κορμό των δέντρων να λυγίζουν για την προσομοίωση του αέρα. Ανάλογα με τον συντελεστή καμπής μπορούν τα φύλλα και ο κορμός να λυγίζουν. Η υφή των φύλλων έχουν άλφα κανάλι για να σχεδιάζονται διάφανα όπου υπάρχει το μαύρο.

Σκιαστής διάχυσης ανώμαλης επιφάνειας (Bumped Diffuse)

Ο σκιαστής διάχυσης ανώμαλης επιφάνειας υπολογίζει την διάχυση όπως στον σκιαστή διάχυσης. Για την ανώμαλη επιφάνεια χρησιμοποιεί μια υφή που ονομάζεται χάρτης ανώμαλης επιφάνειας (Bump map or Normal Map) για να δημιουργήσει το εφφέ της ανώμαλης επιφάνειας χωρίς την αλλαγή της γεωμετρίας. Στον χάρτη ανώμαλης επιφάνειας, κάθε τιμή χρώματος του εικονοστοιχείου αντιπροσωπεύει την

γωνία του διανύσματος που είναι κάθετο στην επιφάνεια του πολυγώνου (surface normal) . Έπειτα χρησιμοποιώντας αυτήν την τιμή υπολογίζεται ο φωτισμός αντί τις γεωμετρίας. Ο χάρτης ανώμαλης επιφάνειας αντιπαρέρχεται την γεωμετρία του δικτυώματος (mesh geometry) όταν υπολογίζεται ο φωτισμός του αντικειμένου.

Σκιαστής νερού (water shader)

Ο σκιαστής νερού χρησιμοποιεί μια επίπεδη επιφάνεια, τοποθετημένη οριζόντια σαν το δικτύωμα νερού. Αλλάζοντας τις ιδιότητες του σκιαστή μπορούμε να το προσαρμόσουμε στις ανάγκες μας. Πιο αναλυτικά:
Ελέγχουμε την κλίμακα των κυμάτων
Αλλάζοντας τις διαστάσεις του χάρτη ανώμαλης επιφάνειας (bumpmap).

Ελέγχουμε την παραμόρφωση που οφείλεται στην διάθλαση και στην ανάκλαση
Αλλάζοντας τον βαθμό λεπτομέρειας που βλέπουμε απο τον χάρτη ανώμαλης επιφάνειας.

Ενεργοποιούμε την περιβαλλοντική διάθλαση και ανάκλαση
Σχεδιάζοντας σε υφές το περιβάλλον και προβάλλοντας τες κατάλληλα στην επιφάνεια του νερού.

Παράγουμε τα κύματα
Ορίζοντας τους δύο χάρτες ανώμαλης επιφάνειας (bumpmaps) που συνδυάζονται και κινούνται σε διαφορετικές κατευθύνσεις τα κύματα αλλάζουν κλίμακα και ταχύτητα. Ο ένας από τους δύο χάρτες θα πρέπει να είναι διπλάσιος σε μέγεθος απο τον πρώτο ώστε να φανεί σωστά το τέχνασμα του κύματος.

Ελέγχουμε την ταχύτητα των κυμάτων
Ελέγχουμε την ταχύτητα που οι χάρτες ανώμαλης επιφάνειας θα κινούνται.

Ελέγχουμε τον συντελεστή Fresnel
Ο συντελεστής Fresnel είναι αυτός που επιτρέπει να αντανακλάται απο διάφορες γωνίες το περιβάλλον της επιφάνειας. Μπορούμε να πούμε ότι είναι το ποσοστό διάθλασης και ανάκλασης που θα είναι εμφανές.

Αναφορά στους σκιαστές που έχουν κατασκευαστεί απο χρήστες

Σκιαστής τύπου δέρματος (Skin Shader)

Ο σκιαστής τύπου δέρματος χρησιμοποιείται για την καλύτερη σχεδίαση του δέρματος στην οθόνη. Χρησιμοποιεί υφές για την δημιουργία της αίσθησης του φωτισμού απο περισσότερες απο μία πηγές δίνοντας μια κοκκινωπή απόχρωση στο εξωτερικό του αντικειμένου και μια μπλε απόχρωση στις μικρότερες γωνίες σε σχέση με την επιφάνεια (grazing angles). Δίνει μία πιο απαλή εμφάνιση και περισσότερες λεπτομέρειες στις σκιασμένες πλευρές του αντικειμένου.

Είναι φωτισμένος ανά εικονοστοιχείο σκιαστής και χρειάζεται προγραμματισμό και για τις κορυφές και για τα κομμάτια. Έτσι σε περίπτωση που μια κάρτα γραφικών δεν υποστηρίζει το κατάλληλο μοντέλο σκιαστή χρησιμοποιούνται οι προεγκατεστημένοι σκιαστές.

Για την εφαρμογή του έχουμε δύο υφές. Μία είναι η υφή τύπου ράμπας για το περίβλημα του μοντέλου και για τον συντελεστή frensel. Μέσω αυτής μπορούμε να ελέγχουμε το χρώμα του δέρματος στις μικρές γωνίες. Τα RGB κανάλια περιλαμβάνουν την αλλαγή του χρώματος και το άλφα κανάλι χρησιμοποιείται για την μείωση του συντελεστή frensel στην κατοπτρική επιφάνεια. Η δεύτερη είναι μια υφή τύπου Wrap Ramp. Χρησιμοποιείται για τον χρωματισμό και τον έλεγχο του ψεύτικου φωτισμού δέρματος.

5.7 Σωματιδιακά τεχνάσματα εντυπωσιασμού (Particles effects)

5.7.1 Υλοποίηση σωματιδιακών τεχνασμάτων εντυπωσιασμού

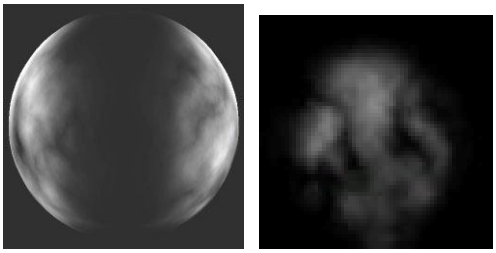
Τα σωματιδιακά τεχνάσματα εντυπωσιασμού χρησιμοποιούνται κατά κόρον στα σημερινά παιχνίδια και αυτό διότι απο τη μία μεριά δεν είναι ιδιαίτερα ακριβά από άποψη υπολογισμών για την σχεδίαση τους και απο την άλλη εντυπωσιάζουν την οπτική του θεατή. Στην δική μας υλοποίηση τα τεχνάσματα χρησιμοποιήθηκαν κατά τις εξής καταστάσεις:

- Συγκομιδή αντικειμένων
- Αναγέννηση του παίκτη σε σημείο ελέγχου
- Χτύπημα του δέντρου «φυσικής»
- Κίνηση κύριου χαρακτήρα

Για την **συγκομιδή αντικειμένων και την αναγέννηση του παίκτη** σε ένα σημείο ελέγχου έχουμε χρησιμοποιήσει υλικό που έχει τον **σκιαστή τύπου προσθετικό**ς (Additive). Αυτός ο σκιαστής έρχεται προεγκατεστημένος απο την Unity3D και εδώ θα γίνει μια επεξήγηση της λειτουργίας του. Έχει δύο ιδιότητες που μπορούν να παραμετροποιηθούν. Μια είναι η απόχρωσή του και δεύτερη είναι η υφή που θα έχει. Η υφή έχει ένα κανάλι άλφα για να γίνει η ανάμειξη κατά την σχεδίαση. Όπου υπάρχει μαύρο θα είναι διάφανο και όπου υπάρχει άλλη διαβάθμιση του γκρίζου θα σχεδιάζεται το χρώμα σύμφωνα με αυτή. Το χρώμα που παράγεται δίνεται απο τον τύπο :

$$BlendColor = SrcFactor + DstFactor \quad (1)$$

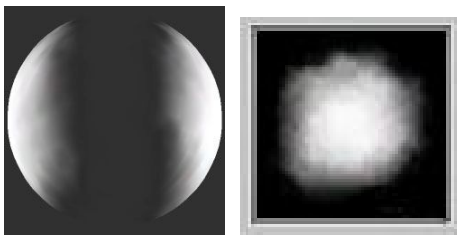
Όπου SrcFactor είναι το παραγόμενο χρώμα πολλαπλασιασμένο επί την τιμή του άλφα καναλιού της εικόνας (alpha value). Όπου DstFactor είναι η τιμή του χρώματος που υπάρχει ήδη στην οθόνη και εισέρχεται ακριβώς όπως είναι, δηλαδή έχει την τιμή 1.



Εικόνα 48: Προσθετικός σκιαστής άλφα (Alpha additive shader) και η υφή που χρησιμοποιήθηκε

Τα **σωματίδια ενός δέντρου «φυσικής»** έχουν σωματίδια που το υλικό του έχει σκιαστή (shader) τύπου **άλφα αναμεμειγμένο** (Alpha blended) . Η εμφάνισή του πριν γίνει χρωματισμός της επιφάνεια φαίνεται στην εικόνα.

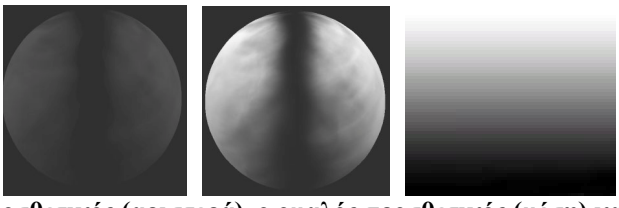
Ο σκιαστής έρχεται προεγκατεστημένος απο την Unity3D και η διαφορά του απο τον προηγούμενο σκιαστή είναι στην διαδικασία ανάμειξης του χρώματος. Στην σχέση (1) έχουμε αλλαγή στο DstFactor. Όπου DstFactor είναι η τιμή του χρώματος που υπάρχει ήδη στην οθόνη πολλαπλασιασμένο επί την τιμή (1-alpha value).



Εικόνα 49: Αναμεμειγμένος σκιαστής άλφα (Alpha Blended Shader) και η υφή που χρησιμοποιήθηκε

Το οπτικό τους αποτέλεσμα θα είναι ο προσθετικός σκιαστής να έχει ένα πιο φωτεινό τέχνασμα εντυπωσιασμού απο το άλφα αναμεμειγμένο σκιαστή. Στον προσθετικό τα χρώματα θα προστεθούν στην προσωρινή μνήμη καρέ ενώ στον άλφα αναμεμειγμένο θα αναμειχτούν μεταξύ τους, αντικαθιστώντας τα ήδη υπάρχοντα χρώματα.

Στο τρέξιμο του χαρακτήρα έχει τοποθετηθεί ένας σχεδιαστής τροχιάς (trail renderer). Ο εν λόγω σχεδιαστής αποτελείται απο ένα υλικό του οποίου ο σκιαστής είναι **ομαλός προσθετικός** (additive smooth). Στην σχέση ανάμειξης (1) ο SrcFactor, δηλαδή το παραγόμενο χρώμα παραμένει ως έχει, με τιμή 1 και ο DstFactor, δηλαδή το χρώμα που βρίσκεται στην οθόνη πολλαπλασιάζεται επί την τιμή (1- χρώμα της πηγής (source color)). Το οπτικό αποτέλεσμα είναι ότι ο ομαλός προσθετικός σκιαστής είναι λίγο πιο φωτεινός από ότι ο απλός προσθετικός.

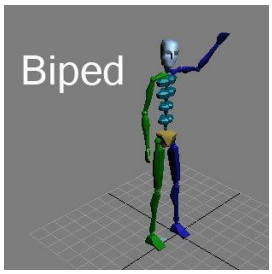


Εικόνα 50: Ο προσθετικός (αριστερά), ο ομαλός προσθετικός (μέση) και η υφή που έχει χρησιμοποιηθεί (δεξιά)

5.8 Κίνηση (animation)

5.8.1 Εισαγωγή κίνησης μοντέλου

Για την κίνηση (animation) του παίκτη χρησιμοποιήθηκε η τεχνική σύλληψη κίνησης (Motion Capture) σε ένα βοηθητικό μοντέλο που ορίστηκε στο πρόγραμμα τρισδιάστατης σχεδίασης, 3DS Max. Η διαδικασία έχει ως εξής: Δημιουργείται ένα μοντέλο τύπου «δίποδου» (biped, βλ. Εικόνα 51) και εισάγεται η κίνηση σε αυτό. Το δίποδο έχει συγκεκριμένη ονομασία για τα μέλη του. Εισάγουμε το αρχείο που περιέχει τα δεδομένα της κίνησης στο δίποδο και κάνουμε επαναπροσδιορισμό της κίνησης στο μοντέλο μας. Η κίνηση συνήθως έπρεπε να διορθωθεί σε λεπτομέρειες, λόγω της διαφοράς του σκελετού του μοντέλου μας, με τον σκελετό των μοντέλων που χρησιμοποιεί η τεχνική σύλληψης κίνησης. Για παράδειγμα, το μοντέλο μας είχε δάκτυλα, ενώ ο σκελετός που έχει τα δεδομένα κίνησης δεν είχε. Έτσι, έγινε διόρθωση της κίνησης των δακτύλων κατά την διάρκεια του χρόνου.



Εικόνα 51: Εικόνα δίποδου

5.8.2 Κύκλος κίνησης του μοντέλου

Όλες οι δυνατές κινήσεις που μπορεί να κάνει το μοντέλο έχουν εισαχθεί σε ένα αρχείο. Στην Unity3D έχουμε την δυνατότητα να διαχωρίζουμε την κάθε κίνηση, επιλέγοντας τα καρέ που αναπαράγεται αυτή. Έτσι, έχουμε εφτά κινήσεις: «περπάτημα», «τρέξιμο», «πήδημα», «αριστερή γροθιά»,«δεξιά γροθιά», «θάνατος», «χορός breakdance» (βλ. Εικόνα 52).

Εικόνα 52: Οι κινήσεις του μοντέλου

5.8.3 Κίνηση βασισμένη στον χειρισμό του παίκτη

Η κίνηση του παίκτη βασίζεται στα κουμπιά εισόδου του παίκτη. Έχουν οριστεί κουμπιά για το περπάτημα, το άλμα και τις γροθιές. Η εναλλαγή μεταξύ της κίνησης που έχει οριστεί για το περπάτημα και για την κίνηση που έχει οριστεί για την ανενεργή κατάσταση γίνεται ομαλά. Εξασθενεί η μια κίνηση και ξεκινάει η άλλη. Η εντύπωση που δίνει αυτή η ενέργεια είναι περισσότερο ρεαλιστική. Τέλος, έχουν οριστεί επίπεδα για την ανάμειξη των ξεχωριστών κινήσεων. Αυτή η τεχνική χρησιμοποιήθηκε για την αποφυγή της χειροκίνητης εισαγωγής βάρους σε κάθε κίνηση. Για παράδειγμα, στην εναλλαγή μεταξύ των κινήσεων του περπατήματος, αδράνειας και γροθιάς, θα πρέπει η κίνηση της γροθιάς να έχει μεγαλύτερη προτεραιότητα (δηλαδή μεγαλύτερο επίπεδο) απο τις άλλες δύο. Έτσι, θα παιχτεί μόνο αυτή. Όταν ξεκινά η γροθιά θα έχει κάποιο βάρος ανάμειξης απο το περπάτημα ή απο την κίνηση της αδράνειας, αλλά στην αντίθετη περίπτωση δεν θα λαμβάνουν βάρος ανάμειξης απο την κίνηση της γροθιάς.

5.9 Γραφική διεπαφή χρήστη (GUI)

Αρχικά, θα πρέπει να σημειωθεί ότι η γραφική διεπαφή χρήστη έχει φτιαχτεί έτσι ώστε να υπάρχει συνέπεια στα χρώματα και στην διαρρύθμισή της.

Η γραφική διεπαφή χρήστη αποτελείται απο ξεχωριστά στοιχεία. Για την δημιουργία των περισσοτέρων έχει χρησιμοποιηθεί η UnityGUI υπομονάδα. Η UnityGUI υπομονάδα αποτελείται απο δύο κομμάτια. Το κομμάτι GUI Style και το κομμάτι GUI Skin. Το κομμάτι GUI Skin είναι μια συλλογή απο κομμάτια GUI Styles και εφαρμόζονται στη γραφική διεπαφή. Κάθε τύπος ελέγχου έχει τον δικό του ορισμό για το Style.

Έτσι, έχει εισαχθεί μια υλοποίηση που μέσω αυτού ορίζουμε το κομμάτι GUI Skin και στον συντάκτη γίνεται ο ορισμός των εικόνων και των χρωμάτων που θα

χρησιμοποιηθούν. Για οτιδήποτε αφορά την γραφική διεπαφή του χρήστη στο παιχνίδι υπάρχει η συνάρτηση **OnGUI** η οποία καλείται για να σχεδιάσει και να χειριστεί τα στοιχεία της. Καλείται για κάθε καρέ.

5.9.1 Υλοποίηση αρχικής εικόνας παιχνιδιού

Για την υλοποίηση της αρχικής εικόνας παιχνιδιού ορίζουμε μια εικόνα που θα είναι το φόντο, ορίζουμε ένα ορθογώνιο σχήμα που εσωτερικά του θα τοποθετηθεί ο τίτλος του παιχνιδιού και τέλος τοποθετούνται τα κουμπιά έναρξης του παιχνιδιού, οδηγιών, της εμφάνισης της ιστορίας και του τερματισμού του παιχνιδιού. Για τον ορισμό των κουμπιών πρέπει να καθοριστεί το μέγεθός τους και το κείμενο που θα αναγράφεται πάνω τους. Η συνάρτηση για τον ορισμό του κουμπιού είναι λογική συνάρτηση.

Δείγμα κώδικα για το κουμπί έναρξης παιχνιδιού:

```
        if (GUI.Button(Rect( Screen.width/2.5, Screen.height/1.334,
Screen.width/5.5, Screen.height/10), "Play"))
        {
                isLoading = true;
                introMusic=gameObject.Find("introMusic");
                DontDestroyOnLoad(introMusic);
                Application.LoadLevel("nickNameScreen");
        }
```

Όταν πατηθεί το κουμπί της εμφάνισης της ιστορίας του παιχνιδιού, εμφανίζεται ένα παράθυρο που έχει μετακινητές δεξιά και κουμπιά για συνέχεια ή επιστροφή στην προηγούμενη οθόνη. Στο παράθυρο που έχουμε ορίσει τις διαστάσεις και την διάταξη εμφανίζεται το κείμενο του σεναρίου. Έχει οριστεί ακόμα και ένα παράθυρο για τις οδηγίες του παιχνιδιού.

Παρακάτω ο κώδικας για την ενεργοποίηση αυτών των δύο:

```
        if(GUI.Button(  Rect( Screen.width/6.8, Screen.height/1.334,
        Screen.width/5.5, Screen.height/10,), "How to play")||windowOpen)
        {
                windowOpen=true;
                windowRect0 = GUI.Window (0,Rect (Screen.width/4, Screen.height/6,
Screen.width/2, Screen.height/2),doMyInstructions,  "How to play");
        }

        if(GUI.Button( Rect( Screen.width/1.56, Screen.height/1.334,
Screen.width/5.5, Screen.height/10,), "The Story")||windowOpen1)
        {
                windowOpen1=true;
                windowRect1= GUI.Window (0,Rect (Screen.width/10, Screen.height/6,
Screen.width/1.2, Screen.height/1.85),doTheStoryWindow,  "The Story of Zaharias");

        }
```

Και ο κώδικας για την υλοποίησή τους:

```
function doMyInstructions()
{
```

```
        GUI.Label(Rect (30, 30, Screen.width/2-10, Screen.width/2-10)
,"Arrows for the movement of the player\nPress \"Right Click\" for breaking of the
vases\nPress \"Left Click\"for the hitting your opponent ","small");
        if(GUI.Button(Rect (Screen.width/2.7,Screen.height/2.4, 80,
25),"close"))
                windowOpen=false;
}

function doTheStoryWindow(){
        if(GUI.Button(Rect (Screen.width/1.377,Screen.height/2.04, 70,
25),"close"))
                windowOpen1=false;
                // Begin the ScrollView
        scrollViewVector = GUI.BeginScrollView (Rect( Screen.width/86,
Screen.height/32, Screen.width/1.22,Screen.height/2.2),
scrollViewVector, Rect (0, 0, 600, 1800));

        // Put something inside the ScrollView
        GUI.Label (Rect (0, 0, 600, 1800), innerText,"small");

        // End the ScrollView
        GUI.EndScrollView();
}
```

Γενικά, οι υλοποιήσεις των στοιχείων της γραφικής διεπαφής χρήστη απαιτούν τον ορισμό του χώρου που θα πιάνουν, την ονομασία του και έπειτα τις λειτουργίες που θα γίνονται όταν πατηθεί κάποιο κουμπί ή επιλεχτεί ένας διθέσιος διακόπτης (toggle). Με παρόμοιο τρόπο υλοποιούνται και οι οθόνες για την εισαγωγή ονόματος του παίκτη, ενημέρωσης φορτώματος του παιχνιδιού, τερματισμού του παιχνιδιού και παρουσίαση των υψηλότερων βαθμολογιών.

5.9.2 Υλοποίηση ευρετηρίου του παίκτη

Το ευρετήριο αποτελείται απο ένα διθέσιο διακόπτη που ενεργοποιεί το παράθυρο. Εσωτερικά σε αυτό υπάρχουν κουμπιά που έχουν εικόνες σαν περιεχόμενο. Τέλος, όταν πατάμε ένα κουμπί εμφανίζεται κείμενο στο παράθυρο και εξαφανίζονται τα περιεχόμενα του ευρετηρίου. Μπορούμε να γυρίσουμε πίσω με το κουμπί της επιστροφής. Η ενημέρωση για τα περιεχόμενα του ευρετηρίου κατά την διάρκεια του παιχνιδιού γίνεται μέσω από την υλοποίηση «GettingKeys», που βρίσκεται στον χειρισμό του χαρακτήρα μας.

Δείγμα κώδικα εμφάνισης κουμπιών ευρετηρίου:

```
function showMap (windowID : int) {
if(GettingKeys.key1||GettingKeys.key2||GettingKeys.key3||GettingKeys.key4||
GettingKeys.fork||GettingKeys.magicalFilter)
{       if(GettingKeys.key1)
         {
        mauveButton=GUI.Button(Rect(10,40,Screen.width/15,Screen.height/15),
        GUIContent(mauveIcon));
         }

         if(GettingKeys.key2)
```

```
        {greenButton=GUI.Button(Rect(10+(Screen.width/15),40,Screen.width/15,
        Screen.height/15), GUIContent(greenIcon));}


        if(GettingKeys.key3)
        {blueButton=GUI.Button(Rect(10+2*(Screen.width/15),40,Screen.width/15,
        Screen.height/15), GUIContent(blueIcon));}

        if(GettingKeys.key4)
        {yellowButton=GUI.Button(Rect(10+3*(Screen.width/15),40,
        Screen.width/15,Screen.height/15), GUIContent(yellowIcon));}
        if(GettingKeys.fork)
        {forkButton=GUI.Button(Rect(10+4*(Screen.width/15),40,Screen.width/15,
Screen.height/15), GUIContent(forkIcon));}
        if(GettingKeys.magicalFilter)
         {magicalFilterButton=GUI.Button(Rect(10+5*(Screen.width/15),40,
        Screen.width/15,Screen.height/15), GUIContent(magicalFilterIcon));
        }
}
else
{
        GUI.Label ( Rect( (Screen.width/2.25),Screen.height/15, Screen.width/4,
        Screen.height/8), "Nothing in the inventory yet");
}
}
```

Εμφάνιση ευρετηρίου και οδηγιών για τα κουμπιά:

```
function OnGUI () {
        if(gSkin)
                GUI.skin = gSkin;

doWindow0 = GUI.Toggle (Rect (10,0,100,40), doWindow0, "Inventory");

if (doWindow0){
GUI.Window (0, Rect (0,Screen.height-
(Screen.height/7),Screen.width,Screen.height/7),showMap ,"My Inventory");
}

if(mauveButton&&GuardKilling.killedGuards)
{
GUI.Window(0,Rect (0,Screen.height-
(Screen.height/7),Screen.width,Screen.height/7),showInformation,"Key
One\r\nOpen Warehouse 1 with this key..");
}
if(greenButton&&GuardKilling.killedGuards)
{
GUI.Window(0,Rect (0,Screen.height-
(Screen.height/7),Screen.width,Screen.height/7),showInformation,"Key
Two\nOpen Warehouse 2 with this key..");
}
if(blueButton&&GuardKilling.killedGuards)
{
GUI.Window(0,Rect (0,Screen.height-
(Screen.height/7),Screen.width,Screen.height/7),showInformation,"Key
Three\nOpen Warehouse 3 with this key..");
}}
```

5.9.3 Υλοποίηση μικρού χάρτη πίστας πραγματικού χρόνου

Για την υλοποίηση του μικρού χάρτη πραγματικού χρόνου χρησιμοποιήθηκε η δυνατότητα της Unity3D να σχεδιάζει την θέαση της κάμερας σε υφή. Έτσι δημιουργήθηκε μια κάμερα που σχεδίαζε σε μια υφή ότι γινόταν στο παιχνίδι και τοποθετήθηκε ψηλά. Το πρόβλημα που δημιουργήθηκε ήταν ότι, λόγω έλλειψης άλφα καναλιού, το έδαφος δεν εμφανιζόταν σωστά στην υφή. Έτσι, τραβήχτηκε ένα στιγμιότυπο του εδάφους απο την γωνία θέασης της κάμερας.

Πρώτα, ορίστηκε η περιοχή που θα εμφανίζεται ο μικρός χάρτης πραγματικού χρόνου. Εμφανίσαμε τα δεδομένα που παίρνουμε απο τον σχεδιασμό της κάμερας σε υφή και προστέθηκε το στιγμιότυπο του εδάφους ως φόντο.

Για την εμφάνιση των εχθρών, του χαρακτήρα, των αποθηκών και αντικειμένων δημιουργήθηκαν καινούρια επίπεδα. Με μία μάσκα, αυτά τα επίπεδα εμφανίζονταν στην κάμερα που έκανε σχεδιασμό θέασης στην υφή και όχι στην κύρια κάμερα του παιχνιδιού.

5.9.4 Υλοποίηση βοηθητικών στοιχείων οθόνης

Για την υλοποίηση βοηθητικών στοιχείων οθόνης εισήχθησαν υφές και προστέθηκε όπου χρειαζόταν κείμενο. Πιο συγκεκριμένα, για την εμφάνιση της βαθμολογίας του παίκτη, της αντίστροφης μέτρησης χρόνου αποστολής, μετρητών αντικειμένων έχουν δημιουργηθεί GameObject με την υπομονάδα GUI Text. Το αποτέλεσμα που θέλουμε το παίρνουμε βάζοντας ένα πρόθεμα, ενημερώνοντας το κείμενο του GameObject και τοποθετώντας το στην επιθυμητή θέση.

Δείγμα κώδικα για την εμφάνιση της βαθμολογίας:

```
function Update () {
var prefix ="Score: ";

guiText.pixelOffset.x=-20;
guiText.pixelOffset.y=(Screen.height/2)+160;
guiText.text=prefix+ScoreCollecting.Points;

GameManagerStart.playerScore=ScoreCollecting.Points;

}
```

Η εμφάνιση της κατάστασης της υγείας του παίκτη και εμφάνιση των εικονιδίων υλοποιούνται στην **HUD** υλοποίηση. Εκεί, τοποθετούνται οι υφές στα σημεία της οθόνης και στην κλίμακα που θέλουμε. Επίσης, γίνεται η εναλλαγή των εικόνων για την ενέργεια του παίκτη ανάλογα με την κατάσταση της ενέργειας που γνωστοποιείτε μέσω της υλοποίησης GoingToGameOver (χειρισμός παίκτη).

5.9.5 Υλοποίηση βοηθητικών μηνυμάτων κατάστασης παιχνιδιού

Τα βοηθητικά μηνύματα κατάστασης του παιχνιδιού είναι δύο ειδών. Πρώτον, είναι αυτά που είναι εικόνες και εξασθενούν με το πέρασμα του χρόνου ή την μεταβολή της απόστασης απο την κάμερα. Δεύτερον, είναι αυτά που περιέχονται σε ένα παράθυρο.

Τα πρώτα έχουν υλοποιηθεί ως εξής: Έχουμε ορίσει ένα βοηθητικό αόρατο GameObject στο κεφάλι του αντίπαλου χαρακτήρα και μετράμε την απόσταση του απο την κάμερα. Τέλος, αφαιρούμε το τρισδιάστατο διάνυσμα απόστασης, μικρότερο κατά αρκετές φορές, από το διάνυσμα του χρώματος.

Δείγμα κώδικα για την υλοποίηση εξασθένισης με βάση την απόσταση:

```
function OnGUI(){
//talking  Bubbles
position = thirdPersonCamera.WorldToScreenPoint( head.position );
position = Vector2( position.x, Screen.height - position.y );

if( currentIcon != null )
{
  if(!enemyGuyHideBubble)
  {
        GUI.color = Color( 1.0, 1.0, 1.0, 1.0 -
      Vector3.Distance(thirdPersonCamera.transform.position,
      head.position ) / 80);

      GUI.Label( Rect( position.x - 35.0, (position.y - 95.0)-
      Vector3.Distance( thirdPersonCamera.transform.position,
      head.position ), 170, 170 ), talkBubble );

      GUI.Label(  Rect( position.x+20 , (position.y - 60)-
      Vector3.Distance( thirdPersonCamera.transform.position,
      head.position ), 60, 60), currentIcon );

      GUI.color = Color.white;
    }
   else
      GUI.color.a=0;
}
}
```

Το δεύτερο είδος μηνυμάτων, με την εμφάνιση παραθύρων γίνεται όπως και στην αρχική οθόνη με την **συνάρτηση OnGUI**. Εμφανίζονται όταν συγκεκριμένες λογικές μεταβλητές γίνονται αλήθεια.

5.10 Καταγραφή βαθμολογιών

Η καταγραφή βαθμολογιών γίνεται απο πλευράς εξυπηρετητή (server-side high scores). Οι ενέργειες που έχουν γίνει στέλνουν, το όνομα και την βαθμολογία του παίκτη στον εξυπηρετητή για να αποθηκευτούν σε μια βάση δεδομένων MySQL και

έπειτα να εμφανιστούν στην οθόνη του παιχνιδιού. Έχει δημιουργηθεί μια βάση δεδομένων, δυο υλοποιήσεις για την διαχείριση της βάσης. Το ένα παίρνει τα ονόματα και τις βαθμολογίες που του δίνουμε και τις προσθέτει στην βάση MySQL και το άλλο είναι υπεύθυνο για την μεταφορά των πέντε πρώτων βαθμολογιών και ονομάτων σε ένα αντικείμενο τύπου GUI Text για την εμφάνισή τους.

Ο κώδικας για την αποστολή και φόρτωση των βαθμολογίων στην βάση δεδομένων:

```
var addScoreUrl="http://localhost/unity_test/add_scores.php?";
var highscoreUrl="http://localhost/unity_test/display.php";

function OnLevelWasLoaded() {
  postScore(GameManagerStart.playerName,GameManagerStart.playerScore);
      gameObject.guiText.text = "Loading Scores";
  yield WaitForSeconds(1);
  getScores();
}

function postScore(name, score) {
   //Add the nickname and score to  MySQL DB.
   var highscore_url = addScoreUrl + "name=" + WWW.EscapeURL(name) +
"&score=" + score; //+ "&hash=" + hash;
   // Post the URL to the site and create a download object to get the result.
   hs_post = WWW(highscore_url);
       print(hs_post);
   yield hs_post; // Wait until the download is done
  if(hs_post.error)
  {
     print("There was an error posting the high score: " + hs_post.error);
   }
}

// Get the scores from the MySQL DB to display in a GUIText.
function getScores() {
   hs_get = WWW(highscoreUrl);
   yield hs_get;

   if(hs_get.error)
       {
     print("There was an error getting the high score: " + hs_get.error);
   }
       else {
     gameObject.guiText.text = hs_get.data; // this is a GUIText that will
display the scores in game.
       }
}
```

ΚΕΦΑΛΑΙΟ 6
ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

6.1 Μέθοδος

Για την αξιολόγηση της εργασίας, έγινε χρήση ερωτηματολογίων χρηστικότητας QUIS (Questionnaire for User Interaction Satisfaction) [13]. Τα ερωτηματολόγια συμπληρώθηκαν από τους χρήστες, οι οποίοι απάντησαν σε ερωτήσεις που είχαν να κάνουν με την εμφάνιση, την λειτουργικότητα, την ευκολία εκμάθησης, τις γενικές δυνατότητες της εφαρμογής, τα πολυμέσα και την εγκατάσταση του προγράμματος.

Στην διαδικασία αξιολόγησης χρηστικότητας συμμετείχαν δέκα χρήστες (έξι άνδρες και τέσσερις γυναίκες), με ηλικίες από 20 έως 35 χρόνων, από διαφορετικούς επαγγελματικούς χώρους, οι οποίοι έχουν διαφορετική εξοικείωση με ψηφιακές εφαρμογές και ήταν τουλάχιστον απόφοιτοι λυκείου.

Το ερωτηματολόγιο QUIS περιλαμβάνει 39 ερωτήσεις, χωρισμένες σε πέντε ενότητες, και οι απαντήσεις δίνονται σε κλίμακα από ένα (1) έως εννιά (9), όπως φαίνεται στο παρακάτω παράδειγμα :

Ερωτηματολόγιο Αξιολόγησης Διαλογικής Χρήσης Συστήματος (QUIS - Πανεπιστήμιο του Maryland, 1997)

Αριθμός ερ/γίου : _____ Σύστημα : _____ Ηλικία : _____ Φύλο : ___ άρρεν ___ θήλυ___

ΜΕΡΟΣ 1: Γενική εντύπωση του χρήστη

Παρακαλώ κυκλώστε τους αριθμούς που αντικατοπτρίζουν ακριβέστερα τις εντυπώσεις σας από τη χρήση αυτού του υπολογιστικού συστήματος. Δεν ξέρω/ δεν απαντώ = NA.

1.1. Γενική αντίδραση του Συστήματος :

απαράδεκτη υπέροχη
1 2 3 4 5 6 7 8 9 NA

1.2. Γενική αντίδραση του Συστήματος :

μπερδεύει ικανοποιεί
1 2 3 4 5 6 7 8 9 NA

1.3. Γενική αντίδραση του Συστήματος :

χαζό ευχάριστο
1 2 3 4 5 6 7 8 9 NA

1.4. Γενική αντίδραση του Συστήματος :

δύσκολο εύκολο
1 2 3 4 5 6 7 8 9 ΝΑ

1.5. Γενική αντίδραση του Συστήματος :

άκαμπτο ευέλικτο
1 2 3 4 5 6 7 8 9 ΝΑ

ΜΕΡΟΣ 2: Οθόνη

2.1. Ο σχεδιασμός της οθόνης βοήθησε :

2.1.1. Ο σχεδιασμός της οθόνης βοήθησε : ποτέ πάντα
1 2 3 4 5 6 7 8 9 ΝΑ

2.1.2. Ποσότητα πληροφορίας που εμφανίζονταν στην οθόνη :
ανεπαρκής επαρκής
1 2 3 4 5 6 7 8 9 ΝΑ

2.1.3. Δόμηση της πληροφορίας που εμφανίζονταν στην οθόνη :
χαοτική οργανωμένη
1 2 3 4 5 6 7 8 9 ΝΑ

2.2. Αλληλουχία οθονών :

2.2.1. Αλληλουχία οθονών : μπερδεμένη σαφής
1 2 3 4 5 6 7 8 9 ΝΑ

2.2.2. Επόμενη οθόνη στη σειρά : απρόβλεπτη προβλέψιμη
1 2 3 4 5 6 7 8 9 ΝΑ

2.2.3. Επιστροφή στην προηγούμενη οθόνη : αδύνατη εύκολη
1 2 3 4 5 6 7 8 9 ΝΑ

Παρακαλώ γράψτε τα σχόλια σας για τη δομή των οθονών εδώ :

_____
_____
_____

ΜΕΡΟΣ 3: Ορολογία και επικοινωνία με το Σύστημα

3.1. Τα μηνύματα εμφανίζονται στην οθόνη με :

3.1.1. Τα μηνύματα εμφανίζονται στην οθόνη με :
ασυνέπεια συνέπεια
1 2 3 4 5 6 7 8 9 ΝΑ

3.2. Τα μηνύματα που εμφανίζονται στην οθόνη είναι :

3.2.1. Τα μηνύματα που εμφανίζονται στην οθόνη είναι :

μπερδεμένα σαφή
1 2 3 4 5 6 7 8 9 NA

3.3. Ο υπολογιστής σας ενημερώνει για το τι κάνει :

3.3.1. Ο υπολογιστής σας ενημερώνει για το τι κάνει :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

3.3.2. Εκτελώντας μια κίνηση οδηγούμαστε σε προβλέψιμο αποτέλεσμα :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

3.3.3. Ο έλεγχος της ποσότητας της ανάδρασης του Συστήματος :

αδύνατος εύκολος
1 2 3 4 5 6 7 8 9 NA

3.3.4 Καθυστερήσεις : απαράδεκτες παραδεκτές
1 2 3 4 5 6 7 8 9 NA

3.4. Μηνύματα λαθών :

3.4.1.Μηνύματα λαθών : δεν βοηθούν πολύ βοηθητικά
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την ορολογία και την επικοινωνία με το Σύστημα εδώ :

_____
_____
_____

ΜΕΡΟΣ 4: Εκμάθηση χρήσης

4.1. Η εκμάθηση χρήσης του συστήματος είναι :

4.1.1. Η εκμάθηση χρήσης του συστήματος είναι :

δύσκολη εύκολη
1 2 3 4 5 6 7 8 9 NA

4.1.2. Χρόνος για την εκμάθηση χρήσης του συστήματος :

λίγος πολύς
1 2 3 4 5 6 7 8 9 NA

4.2. Η εξερεύνηση των δυνατοτήτων με τη μέθοδο "προσπάθειας και λάθους (trial and error)" :

4.2.1. Η εξερεύνηση των δυνατοτήτων με τη μέθοδο "προσπάθειας και λάθους (trial and error)" :

απογοητεύει αποδίδει
1 2 3 4 5 6 7 8 9 NA

4.3. Οι εργασίες γίνονται σε λογική αλληλουχία :

4.3.1. Οι εργασίες γίνονται σε λογική αλληλουχία :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

4.3.2. Τα βήματα για την ολοκλήρωση της εργασίας ακολουθούν μια λογική σειρά

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

4.3.3 Ανάδραση όταν ολοκληρωθεί η εργασία :

ασαφής σαφής
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την εκμάθηση της χρήσης εδώ :

_____
_____
_____

ΜΕΡΟΣ 5: Δυνατότητες του Συστήματος

5.1. Η ταχύτητα του Συστήματος είναι :

5.1.1. Η ταχύτητα του Συστήματος είναι : πολύ αργή ικανοποιητική
1 2 3 4 5 6 7 8 9 NA

5.2. Το Σύστημα είναι σταθερό :

5.2.1. Το Σύστημα είναι σταθερό : ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

5.2.2. Χειρισμοί – Λειτουργίες αναξιόπιστα αξιόπιστα
1 2 3 4 5 6 7 8 9 NA

5.3. Η ευκολία χειρισμού εξαρτάται από την εμπειρία του χρήστη :

5.3.1. Η ευκολία χειρισμού εξαρτάται από την εμπειρία του χρήστη :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για τις δυνατότητες του Συστήματος εδώ :

_____
_____
_____

ΜΕΡΟΣ 6: Πολυμέσα

6.1. Η ποιότητα των εικόνων/ φωτογραφιών είναι :

κακή καλή
1 2 3 4 5 6 7 8 9 NA

6.1.1.Εικόνες/ Φωτογραφίες :

θολές καθαρές
1 2 3 4 5 6 7 8 9 NA

6.1.2. Η φωτεινότητα της εικόνας/ φωτογραφίας :

σκοτεινή φωτεινή
1 2 3 4 5 6 7 8 9 NA

6.2. Η ηχητική απόδοση είναι :

6.2.1. Η ηχητική απόδοση είναι :

ασταθής εύρυθμη
1 2 3 4 5 6 7 8 9 NA

6.2.2. Η ηχητική απόδοση είναι :

αλλοιωμένη ξεκάθαρη
1 2 3 4 5 6 7 8 9 NA

6.3. Τα χρώματα που χρησιμοποιούνται είναι :

αφύσικα φυσικά
1 2 3 4 5 6 7 8 9 NA

6.3.1. Η ποσότητα των χρωμάτων που είναι διαθέσιμη είναι :

ανεπαρκής επαρκής
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για τα πολυμέσα εδώ :

_____
_____
_____

ΜΕΡΟΣ 7: Εγκατάσταση συστήματος

7.1. Η ταχύτητα του εγκατάστασης του συστήματος είναι :

αργή γρήγορη
1 2 3 4 5 6 7 8 9 NA

7.2. Η εξατομίκευση είναι :

δύσκολη εύκολη
1 2 3 4 5 6 7 8 9 NA

7.3. Πληροφορείται ο χρήστης για την πρόοδό της :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

7.4. Δίνει εποικοδομητικές εξηγήσεις όταν συμβαίνουν αποτυχίες :

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την εγκατάσταση του συστήματος εδώ :

_____

_____

6.2 Στόχος

Ο στόχος της παραπάνω διαδικασίας ήταν διττός. Τα αποτελέσματα που προέκυψαν, από την μια μεριά έδιναν στοιχεία χρηστικότητας για την ίδια την εφαρμογή και από την άλλη βοήθησαν, στο να εξαχθούν συμπεράσματα για την μελλοντική βελτίωση της εφαρμογής αλλά και για την βελτίωση της υλοποίησης διαδραστικών εφαρμογών διασκεδάσεως.

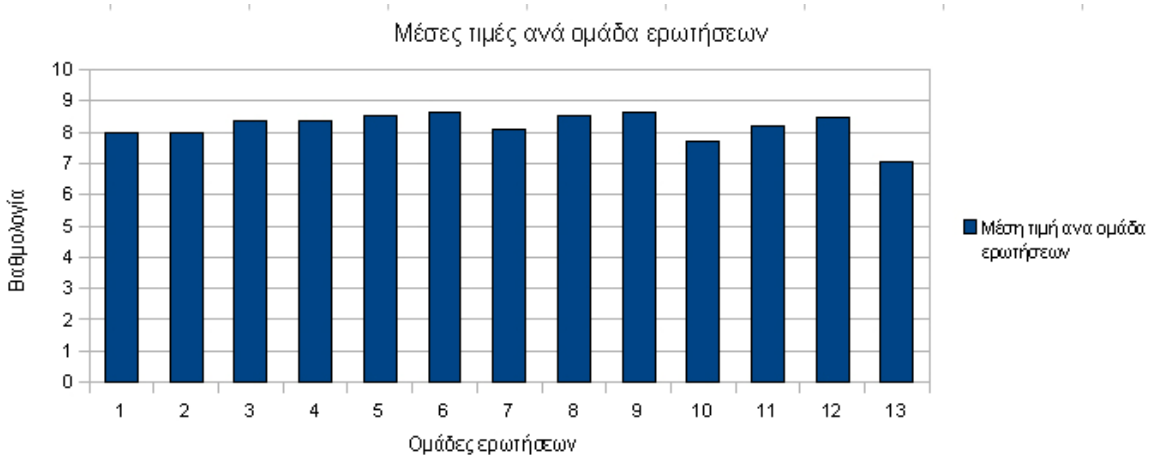
6.3 Αποτελέσματα ερωτηματολογίων

Στα παρακάτω ιστογράμματα παρουσιάζονται οι απαντήσεις των χρηστών που χρησιμοποίησαν την εφαρμογή και συμπλήρωσαν το ερωτηματολόγιο.

Αρχικά παραθέτουμε τα ιστογράμματα μέσης τιμής ανά ερώτηση και μέσης τιμής ανά ομάδα για μια γενική εικόνα και έπειτα τις απαντήσεις των χρηστών ανά ερώτηση.

Μέση τιμή απαντήσεων ανά ερώτηση

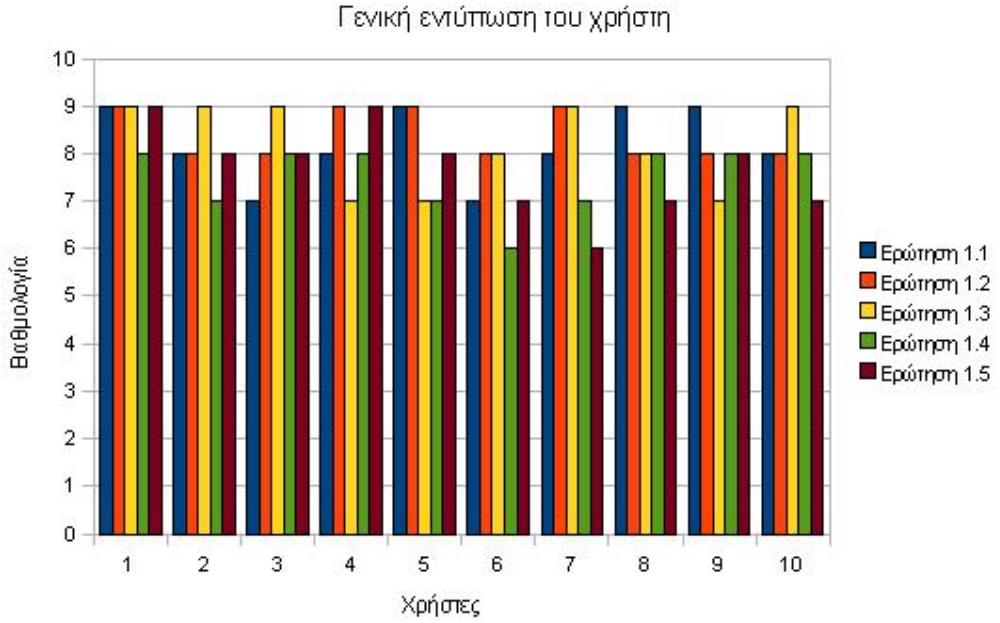
Μέση τιμή απαντήσεων ανά ομάδα ερωτήσεων



Οι ομάδες ερωτήσεων ήταν οι εξής:

1. Γενική εντύπωση του χρήστη ερωτήσεις: 1.1 έως 1.5
2. Σχεδιασμός της οθόνης: 2.1.1 έως 2.1.3
3. Αλληλουχία οθονών: 2.2.1 έως 2.2.3
4. Μηνύματα που εμφανίζονται στην οθόνη: 3.1.1 έως 3.2.1
5. Ο υπολογιστής ενημερώνει για το τι κάνει: 3.3.1 έως 3.4.1
6. Εκμάθηση χρήσης: 4.1.1, 4.1.2
7. Εξερεύνηση των δυνατοτήτων με τη μέθοδο «προσπάθειας και λάθους»: 4.2.1
8. Οι εργασίες γίνονται σε λογική αλληλουχία: 4.3.1 έως 4.3.3
9. Το σύστημα είναι γρήγορα και σταθερό: 5.1.1 έως 5.2.2
10. Ευκολία χειρισμού εξαρτάται απο πείρα χρήστη: 5.3.1
11. Εικόνα: 6.1.1, 6.1.2
12. Ήχος: 6.2.1 έως 6.3.1
13. Ταχύτητα εγκατάστασης συστήματος 7.1 έως 7.4

Μέρος πρώτο – Απαντήσεις χρηστών

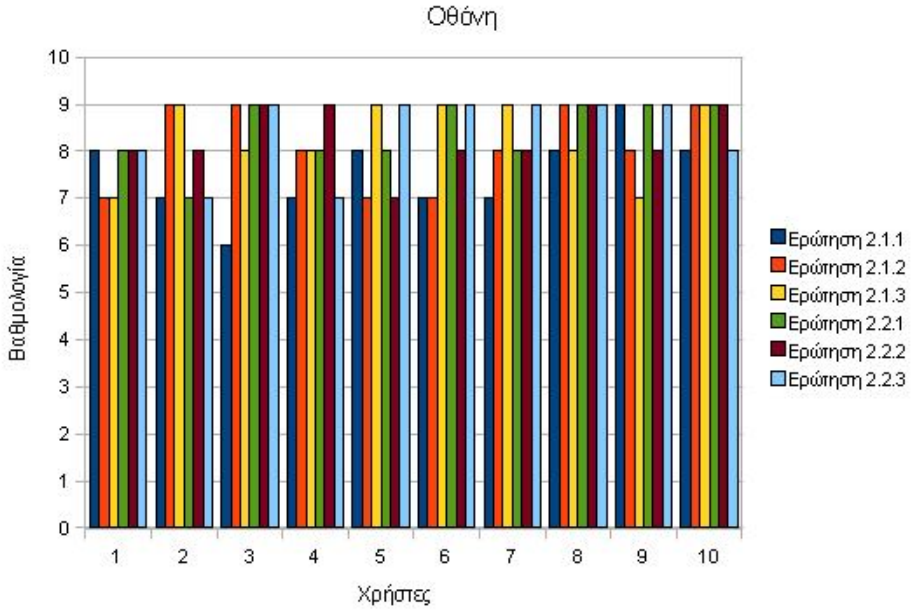


Εικόνα 53: Διάγραμμα γενικής εντύπωσης χρήστη

Σχόλια χρηστών για την ενότητα:

«Ευχάριστο παιχνίδι για ένα ευχάριστο διάλειμμα!!»

Μέρος δεύτερο – Απαντήσεις χρηστών

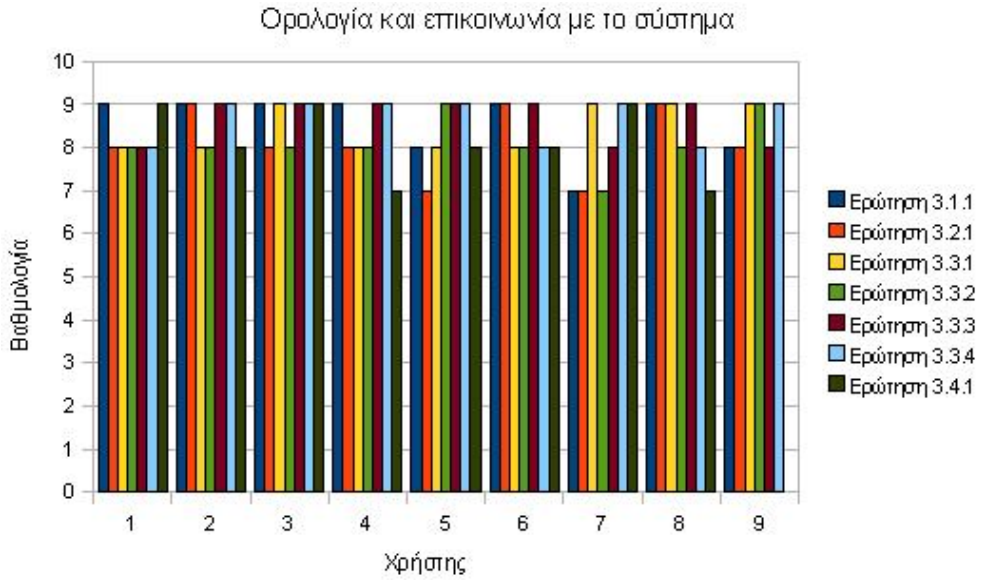


Εικόνα 54: Διάγραμμα απαντήσεων χρηστών για την οθόνη

Σχόλια χρηστών για την δεύτερη ενότητα:

«Αρκετά καλή, παρουσιάζουν συνέπεια και οργάνωση.»
«Οι οθόνες είναι συνδεδεμένες καλά και εύκολα μπορούμε να γυρνάμε στην προηγούμενη οθόνη»

Μέρος τρίτο – Απαντήσεις χρηστών

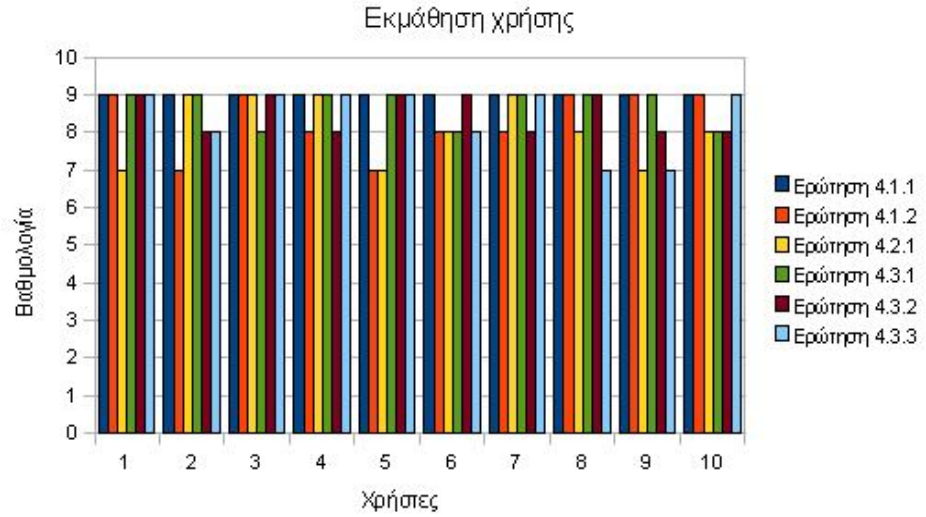


Εικόνα 55: Διάγραμμα ορολογίας και επικοινωνίας με το Σύστημα

Σχόλια χρηστών για την τρίτη ενότητα:

«Η ορολογία σωστή και βοηθητική, απλά δεν ήταν ευανάγνωστα τα μηνύματα λόγω της γραμματοσειράς που επιλέχθηκε.»
«Θα ήθελα να μου λεει περισσότερες πληροφορίες για το που να βρω συγκεκριμένα αντικείμενα στο παιχνίδι.»

Μέρος τέταρτο – Απαντήσεις χρηστών

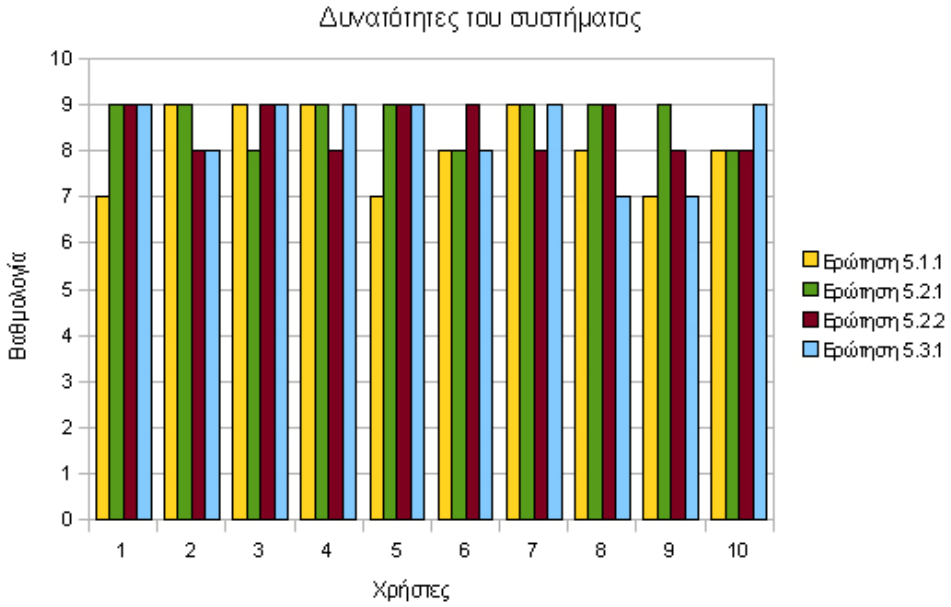


Εικόνα 56: Διάγραμμα εκμάθησης χρήσης

Σχόλια χρηστών για την τέταρτη ενότητα:

«Πολύ εύκολη η εκμάθηση. Άλλωστε παρέχεται από το σύστημα η απαιτούμενη βοήθεια.»
«Έμαθα απο τα λάθη και μετά κέρδισα άνετα.»
«Δεν παίζω καθόλου παιχνίδια κι όμως τα κατάφερα. Έχει εύκολη εκμάθηση.»

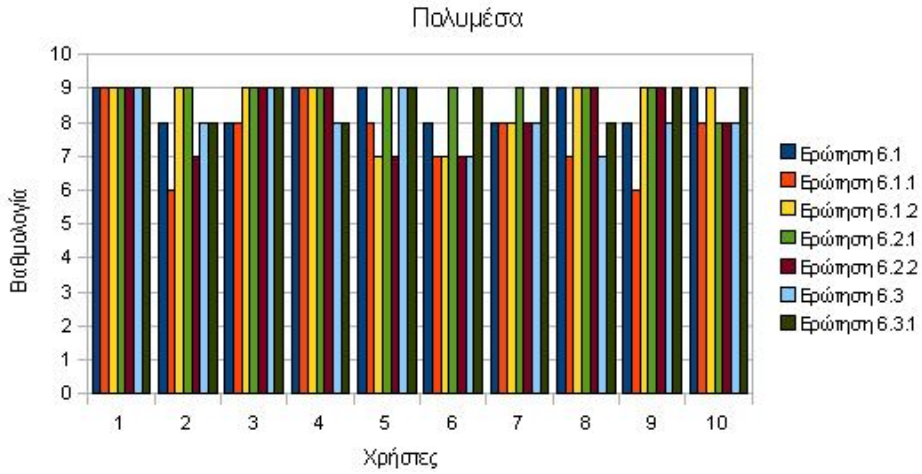
Μέρος πέμπτο – Απαντήσεις χρηστών



Εικόνα 57: Διάγραμμα δυνατοτήτων του συστήματος

Σχόλια χρηστών για την πέμπτη ενότητα:
«Πολύ ευχάριστο παιχνίδι με αρκετές δυνατότητες οι οποίες μπορούν να επεκταθούν μελλοντικά.»

Μέρος έκτο – Απαντήσεις χρηστών

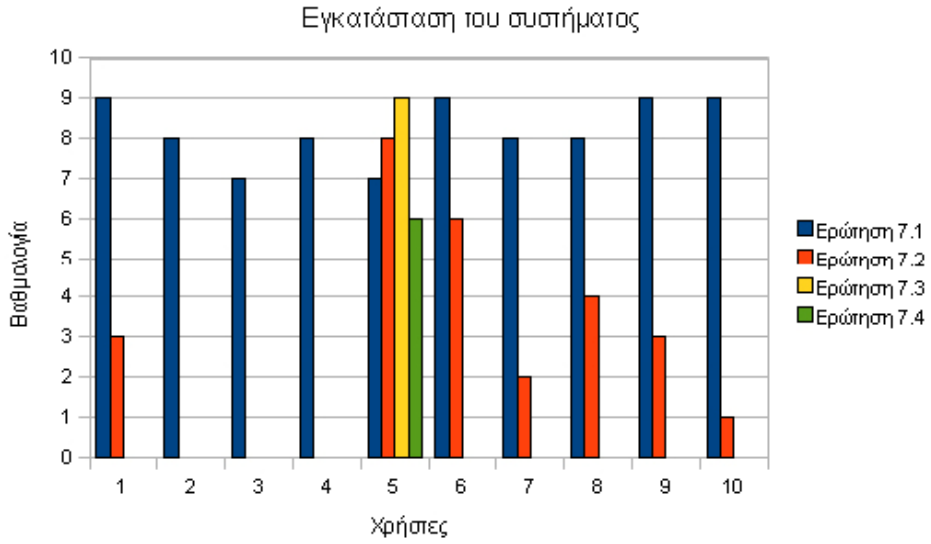


Εικόνα 58: Διάγραμμα πολυμέσων

Σχόλια χρηστών για την έκτη ενότητα:

«Πολύ καλή επιλογή χρωμάτων, ξεκάθαρος ήχος και εικόνα. Λίγο κουραστική η επιλογή του ήχου.»
«Ο ήχος του παιχνιδιού θα μπορούσε να περιέχει περισσότερα από ένα τραγούδια.»

Μέρος έβδομο – Απαντήσεις χρηστών



Εικόνα 59: Διάγραμμα εγκατάστασης του συστήματος

Σχόλια χρηστών για την έβδομη ενότητα:

«Πολύ εύκολη η διαδικασία της εγκατάστασης.»
«Είναι εύκολη, καθώς απαιτείται μονάχα η επικόλληση του εκτελέσιμου αρχείου στον υπολογιστή.»

6.4 Συμπεράσματα

1ο μέρος
Από την εξέταση των απαντήσεων των χρηστών, μπορεί κάποιος να συμπεράνει πως οι χρήστες του παιχνιδιού, ήταν ικανοποιημένοι με την «γενική εντύπωση της εφαρμογής». Οι χρήστες θεώρησαν πως το παιχνίδι είναι αρκετά ευχάριστο, εύκολο και ευέλικτο, δίνοντας γενικά βαθμολογία πάνω από 6 σε όλες τις ερωτήσεις της πρώτης ενότητας.

2ο μέρος
Οι απαντήσεις των χρηστών έδειξαν ικανοποίηση για στοιχεία όπως, «η ποσότητα πληροφορίας που εμφανίζεται στην οθόνη», «η δόμηση της πληροφορίας», «ο σχεδιασμός της οθόνης» και «η αλληλουχία οθονών». Ενώ έδειξαν ότι περίμεναν περισσότερα από την ερώτηση «επιστροφή στην προηγούμενη οθόνη». Γενικά η βαθμολόγηση ήταν πάνω από το έξι στο σύνολο σχεδόν των ερωτήσεων. Η ερώτηση 2.1.2 για την «δόμηση της πληροφορίας που εμφανίζεται στην οθόνη», είχε την μεγαλύτερη διακύμανση στην δεύτερη ενότητα.

3ο μέρος
Γενικά στους χρήστες, άρεσε ο τρόπος επικοινωνίας του παιχνιδιού και έβαλαν υψηλή βαθμολογία σε ερωτήσεις που αφορούσαν τα μηνύματα που εμφανιζόταν στην οθόνη εάν εξαιρέσουμε την γραμματοσειρά. Υψηλή βαθμολογία πήραν επίσης και ερωτήσεις όπως «αν εκτελώντας μια κίνηση οδηγούμαστε σε προβλέψιμο αποτέλεσμα» και ήταν ικανοποιημένοι από τις πιθανές «καθυστερήσεις» στην απόκριση της εφαρμογής. Και στην τρίτη ενότητα οι βαθμολογήσεις των χρηστών, ήταν πάνω από το έξι στο σύνολό τους. Οι ερωτήσεις 3.3.3 για τον έλεγχο της ποσότητας ανάδρασης του συστήματος και 3.3.4 για τις καθυστερήσεις στην απόκριση της εφαρμογής, πήραν τις καλύτερες βαθμολογίες. Ενώ η ερώτηση 3.4.1 για το «αν βοηθούν τα μηνύματα λαθών» είχε την μεγαλύτερη διακύμανση.

4ο μέρος
Οι χρήστες υποστήριξαν πως «η εκμάθηση χρήσης του συστήματος» ήταν εύκολη και «ο χρόνος για την εκμάθηση» ήταν λίγος. Επίσης, θεωρούν πως «η εξερεύνηση της εκμάθησης με τη μέθοδο δοκιμής και λάθους» αποδίδει. Αρκετά καλές ήταν επίσης οι κρίσεις τους για το αν «οι εργασίες γίνονται σε λογική αλληλουχία», ενώ θετικές για το αν «τα βήματα για την ολοκλήρωση της εργασίας ακολουθούν μια λογική σειρά» και για την «ανάδραση» όταν ολοκληρωθεί μια εργασία. Η ερώτηση 4.1.1 για την ευκολία της εκμάθησης χρήσεως της εφαρμογής πήρε την μεγαλύτερη βαθμολογία, ενώ η ερώτηση 4.2.1 για την «εξερεύνηση της εκμάθησης με τη μέθοδο δοκιμής και λάθους», είχε την μεγαλύτερη διακύμανση. Και στην τέταρτη ενότητα οι απαντήσεις των χρηστών ήταν πάνω από το 6 σε όλες τις ερωτήσεις.

5ο μέρος
Οι χρήστες πιστεύουν πως η «ταχύτητα του συστήματος» είναι ικανοποιητική και η εφαρμογή «σταθερή» και ότι «οι χειρισμοί - λειτουργίες» γίνονται αρκετά αξιόπιστα. Τέλος το αν «η ευκολία χειρισμού εξαρτάται απο την εμπειρία του χρήστη», οι απαντήσεις περιείχαν μεγάλο φάσμα από τις πιθανές απαντήσεις. Στην πέμπτη ενότητα, όλες οι απαντήσεις ήταν ξανά πάνω από το πέντε, ενώ η ερώτηση 5.3.1, είχε την μεγαλύτερη διακύμανση.

6ο μέρος
Οι χρήστες ήταν αρκετά ικανοποιημένοι απο τα πολυμέσα του παιχνιδιού καθώς έδωσαν υψηλή βαθμολογία για όλες τις ερωτήσεις. Την μεγαλύτερη βαθμολογία πήρε η ερώτηση 6.2.1 που οι χρήστες απάντησαν σχεδόν ομόφωνα ότι η ηχητική απόδοση είναι εύρυθμη. Μικρότερη βαθμολογία συγκέντρωσε η ερώτηση για το εάν η εικόνα είναι θολή ή καθαρή. Όλες οι απαντήσεις ήταν πάνω απο το έξι. Μεγαλύτερη διακύμανση είχε η ερώτηση 6.2.2 για το εάν η ηχητική απόδοση ήταν ξεκάθαρη ή αλλοιωμένη.

7ο μέρος
Τέλος, στο 7ο μέρος οι χρήστες δυσκολεύτηκαν να απαντήσουν διότι το παιχνίδι, δεν χρειάζεται εγκατάσταση. Χρειάζεται επικόλληση του φακέλου στην τοποθεσία που επιθυμεί ο χρήστης και με ένα διπλό κλικ στο εικονίδιο του παιχνιδιού, αρχίζει. Έτσι, η ερώτηση 7.1 που αφορά την ταχύτητα της εγκατάστασης είχε υψηλή βαθμολογία. Εξατομίκευση της εγκατάστασης δεν υπάρχει, οπότε έχουμε πολύ χαμηλή βαθμολογία σε αυτήν την ερώτηση. Στις ερωτήσεις 7.3 για την πρόοδο της

εγκατάστασης και 7.4 για δυνατότητα να δίνει «εποικοδομητικές απαντήσεις όταν συμβαίνουν αποτυχίες», οι περισσότεροι χρήστες δεν ήξεραν να απαντήσουν.

Συμπερασματικά, οι απαντήσεις των χρηστών είχαν υψηλή βαθμολογία και από ότι φάνηκε δεν υπήρχε χρήστης που να είχε ιδιαίτερη συμπεριφορά.

ΚΕΦΑΛΑΙΟ 7

ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

7.1 Εισαγωγή

Το παρόν κεφάλαιο περιέχει μια σύντομη αναφορά των όσων έχουν υλοποιηθεί στα πλαίσια της διπλωματικής εργασίας. Επίσης περιλαμβάνει αναφορά των αποτελεσμάτων που εξήχθησαν και κριτική τους ώστε να φανεί ο βαθμός επίτευξης των αρχικών στόχων. Τέλος, αναφέρονται βελτιώσεις και πιθανές μελλοντικές προεκτάσεις της συγκεκριμένης εφαρμογής.

7.2 Στόχος και υλοποίηση της εφαρμογής

Στόχος της παρούσας εργασίας, όπως έχει αναφερθεί, είναι η σχεδίαση και η υλοποίηση ενός τρισδιάστατου παιχνιδιού βασισμένο στην φυσική. Η υλοποίηση της εφαρμογής έγινε με την χρησιμοποίηση της μηχανής παιχνιδιών Unity3D. Ο σχεδιασμός χαρακτήρων και ότι έχει να κάνει τα τρισδιάστατα μοντέλα έγινε στο 3D Studio Max 2008. Οι δυσδιάστατες εικόνες έχουν επεξεργαστεί στο πρόγραμμα επεξεργασίας Adobe Photoshop CS3. Τα δεδομένα κίνησης είναι απο το εργαστήριο MoCap του πανεπιστήμιου Carnegie Mellon. Όλοι οι ήχοι προέρχονται απο την ιστοσελίδα www.soundsnap.com. Το μοντέλου του κύριου χαρακτήρα του παιχνιδιού είναι απο δημιουργημένο απο τον Daniel Martinez Lara.

7.3 Αποτελέσματα

Τα αποτελέσματα απο την χρήση του παιχνιδιού βοήθησαν στην βελτίωση του σε διάφορους τομείς.

Για παράδειγμα, οι χρήστες συχνά χάνονταν στις περιπλανήσεις τους στο κόσμο. Για να βελτιωθεί αυτή η παράμετρος δημιουργήθηκε ένα μονοπάτι με την χρήση του συντάκτη εδάφους (Terrain Editor) ώστε η περιπλάνησή τους να είναι πιο εύκολη. Ακολουθώντας το μονοπάτι μπορούν να βρεθούν στις αποθήκες όπου βρίσκονται διάφορα αντικείμενα που είναι σημαντικά για την συνέχιση του παιχνιδιού.

Οι χρήστες συνήθως κουράζονταν απο τις μεγάλες περιόδους που απαιτούνταν για να φτάσουν από την μία αποθήκη στην άλλη για να βρουν τα αντικείμενα. Έτσι, τοποθετήθηκαν αντικείμενα στο μονοπάτι που ο χρήστης μπορεί να μαζεύει. Επιπλέον προστέθηκε η λειτουργία του τρεξίματος κατά την πλοήγηση.

Υπήρξαν προβλήματα με την κίνηση του παίκτη όταν περπατούσε ή έτρεχε και συγχρόνως έκανε την ενέργεια της γροθιάς. Απομονώθηκε η κίνηση του σώματος απο την μέση και κάτω ώστε να τρέχει ενώ απο την μέση και πάνω να εκτελεί την ενέργεια της γροθιάς. Αυτές οι δύο «μισές» κινήσεις προστέθηκαν σε μία καινούρια που θα καλείται κάθε φορά που ο παίκτης τρέχει και εκτελεί την ενέργεια της γροθιάς ταυτόχρονα.

Δεν ήταν εύκολο να αντιληφθεί ο χρήστης πότε δεχόταν επίθεση απο τον αντίπαλό του. Για την αντιμετώπιση αυτού προβλήματος προστέθηκαν οπτικά εφφέ και ηχητικά εφφέ κάθε φορά που ο παίκτης δεχόταν επίθεση και έχανε ενέργεια.

Η μουσική του παιχνιδιού δεν ήταν δυνατό να κλείνει απο τον χρήστη με αποτέλεσμα να ενοχλείται μετά το πέρας ενός χρονικού διαστήματος. Έτσι, προστέθηκε και αυτή η δυνατότητα. Με τον πάτημα ενός κουμπιού μπορεί ο χρήστης να ενεργοποιεί και να απενεργοποιεί την μουσική στο παιχνίδι. Υλοποιήθηκε μηδενίζοντας την ένταση της πηγής που προέρχεται η μουσική.

Απο τους χρήστες παρατηρήθηκε ότι τα δέντρα που είχαν τοποθετηθεί απο τον συντάκτη εδάφους, βρίσκονταν στον αέρα. Μετά απο έρευνα φάνηκε ότι είναι γνωστό πρόβλημα του συντάκτη και οφείλεται στον ορισμό των σημείων περιστροφής και του σκιαστή που έχει ώστε να κάνει τα δέντρα να λυγίζουν σαν να τα φυσά ο αέρας. Ενέργειες που κάναμε για να παρακάμψουμε αυτό το γνωστό πρόβλημα είναι είτε η μετακίνηση του σημείου περιστροφής, είτε η δημιουργία δεύτερου εδάφους που θα τοποθετηθεί κάτω απο το αρχικό. Η πρώτη ενέργεια δημιουργεί πρόβλημα με τον σκιαστή που χρησιμοποιεί ο συντάκτης εδάφους. Δεν γίνεται σωστό λύγισμα του δέντρου με αποτέλεσμα να φαίνεται μη ρεαλιστικό. Στην δεύτερη ενέργεια, η μνήμη που κατανάλωνε το δεύτερο έδαφος ήταν αντιστρόφως ανάλογη απο την αναγκαιότητά του. Έτσι, αποφασίστηκε να προσθέσουμε τα δέντρα χειροκίνητα και να δημιουργηθεί ένα εργαλείο για την γρήγορη τοποθέτηση τους. Αυτό το εργαλείο θα επιλέγει όλα τα δέντρα και θα τα τοποθετεί στο έδαφος. Το μειονέκτημα απο αυτήν ενέργεια που κάναμε, είναι ότι δεν εφαρμόζεται σε αυτά ο σκιαστής του εδάφους που μας παρείχε το λύγισμα.

7.4 Μελλοντικές προεκτάσεις και βελτιώσεις

Κάποιες πιθανές μελλοντικές προεκτάσεις και βελτιώσεις της συγκεκριμένης εφαρμογής μπορεί να είναι οι ακόλουθες:

• *Ανάπτυξη πολυπλοκότερης τεχνητής νοημοσύνης για τους αντιπάλους*
Η ανίχνευση του στόχου του αντιπάλου σε πολυπλοκότερα περιβάλλοντα, αποφυγή πιθανών εμποδίων που βρίσκονται στο δρόμο και εισαγωγή πρόσθετων καταστάσεων συμπεριφοράς στον αντίπαλο. Για την αποφυγή του παγώματος της κίνησης θα πρέπει να γίνει ανίχνευση για εμπόδια μπροστά του.

• *Ανάπτυξη περισσότερων χαρακτηριστικών που να βασίζονται στην φυσική*
Θα μπορούσαμε να εισαχθεί ένα όχημα με τροχούς στο παιχνίδι μας. Βάζοντας συγκρουστές σε κάθε τροχό και τους κατάλληλους συνδέσμους ώστε να

συμπεριφέρεται φυσικά κατά την διάρκεια της οδήγησης και των συγκρούσεων με σημεία της πίστας.

• *Επέκταση των εργαλείων της Unity3D για πιο γρήγορη ανάπτυξη παιχνιδιών*
Χρησιμοποιώντας το UnityAPI να δημιουργηθούν εργαλεία που θα μπορούν να ξαναχρησιμοποιηθούν για την πιο γρήγορη ανάπτυξη παιχνιδιών. Για παράδειγμα, πολύ αποδοτικό θα ήταν ένα σύστημα για την παραγωγή σημείων ελέγχου διαδρομής και η διαχείρισή τους. Η απεικόνιση τους στην οθόνη ώστε να διευκολύνεται ο σχεδιαστής του παιχνιδιού κατά την διάρκεια της αποσφαλμάτωσης.

• *Εισαγωγή δυνατότητας παιχνιδιού για πολλαπλούς παίκτες*
Χρησιμοποιώντας την υπομονάδα δικτύου της Unity3D να μπορούν να συνδεθούν πολλοί παίκτες στον κόσμο του παιχνιδιού και να έχουν την δυνατότητα να αναλαμβάνουν αποστολές είτε ως σύμμαχοι, είτε ως αντίπαλοι.

7.5 Επίλογος

Με την παραπάνω εργασία μου δόθηκε η ευκαιρία να εξερευνήσω πώς μπορούν να σχεδιαστούν και να αναπτυχθούν τρισδιάστατα παιχνίδια, καθώς επίσης, και το πώς μπορούν διαφορετικές εφαρμογές να συνδυαστούν για την επίτευξη ενός συγκεκριμένου στόχου. Ήταν μια διαδικασία χρονοβόρα, επίπονη αλλά αρκετά ενδιαφέρουσα. Ακόμη, μου δόθηκε η ευκαιρία να ασχοληθώ με τον προγραμματισμός παιχνιδιών και τον προγραμματισμό σκιαστών (shaders).

Ο τομέας των ηλεκτρονικών παιχνιδιών είναι ένας τομέας που έχει κινήσει το ενδιαφέρον μια μεγάλης γκάμας επιστημόνων και ερευνητών. Το αποτέλεσμα είναι συνεχώς να αναπτύσσονται εφαρμογές και υλικό υπολογιστών που έχουν σκοπό την βελτίωση της εμπειρίας του χρήστη βελτιώνοντας τα γραφικά και την σχεδίασή τους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]:Fundamentals of Game Design, Ernest Adams, Rolling Andrew 2006,Prentice Hall

[2]: http://source.valvesoftware.com/, The source project (2008)

[3]:http://fifengine.de/, The FIFE project (2008)

[4]: http://spring.clan-sy.com/, The Spring Project (2008)

[5]: http://unity3d.com/support/documentation/Components/index.html, Reference Manual of Unity3D

[6]: Panda 3D Website, Entertainment Technology Center, Carnegie Mellon University, http://www.etc.cmu.edu/panda3d.

[7]: Goslin M.,"Postmortem: Disney Online's Toontown by Mike Goslin [01.28.04]", http://www.gamasutra.com/features/20040128/goslin_01.shtml.

[8]: Designing Interactive Theme Park Rides: Lessons Learned Creating Disney's Pirates of the Caribbean: Battle for the Buccaneer Gold, Jesse Schell and Joe Shochet, Game Developer's Conference 2001

[9]: http://www.python.org/community/pycon/dc2004/papers/29/

[10]: The 3dsMax-friendly BVH release of CMU's motion capture database, http://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/3dsmax-friendly-release-of-cmu-motion-database

[11] : Postmortem: Naughty Dog's Jak and Daxter: the Precursor Legacy, Stephen White, http://www.gamasutra.com/features/20020710/white_02.htm (2002)

[12]: Game Programming Gems 4, Andrew Kirmse, 2004, Charles River Media

[13] : http://www.unifycommunity.com, Unify Community