ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη παιχνιδιού τριών διαστάσεων με χρήση μηχανής απόδοσης γραφικών

(3D game development using a rendering engine)

Διπλωματική Εργασία

Νίκος Ζερβουδάκης

Εξεταστική επιτροπή

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

Χανιά 2010

Στην οικογένεια μου
στην κοπέλα μου, Τατιάνα
και στους φίλους μου

Περιεχόμενα

Κατάλογος σχημάτων

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας ήταν η ανάπτυξη ενός παιχνιδιού τριών διαστάσεων για ηλεκτρονικούς υπολογιστές, κάνοντας χρήση μηχανής απόδοσης τρισδιάστατων γραφικών. Η ανάπτυξη τρισδιάστατων διαδραστικών παιχνιδιών αποτελεί μια πολυσύνθετη και χρονοβόρα διαδικασία. Η χρήση λογισμικού ανάπτυξης παιχνιδιών διευκολύνει την υλοποίησή τους και επιτρέπει την επικέντρωση του δημιουργού στο περιεχόμενο και στην εμπειρία του παιχνιδιού που προσφέρεται στον χρήστη (gameplay). Μία μηχανή απόδοσης γραφικών (rendering engine) είναι λογισμικό το οποίο αναλαμβάνει την απεικόνιση, τη διαχείριση τρισδιάστατων σκηνών, την φόρτωση και την κίνηση (animation) πλεγμάτων (meshes), την επικοινωνία με το υλικό και άλλες λειτουργίες που βοηθάνε την ανάπτυξη τρισδιάστατων εφαρμογών γενικότερα. Μία Μηχανή Παιχνιδιών (game engine) είναι λογισμικό το οποίο προσφέρει εξειδικευμένες προγραμματιστικές δυνατότητες για την υλοποίηση αποκλειστικά τρισδιάστατων παιχνιδιών.

Παρότι θα μπορούσαμε να χρησιμοποιήσουμε μια ολοκληρωμένη μηχανή παιχνιδιών για την υλοποίηση του παιχνιδιού το οποίο παρουσιάζεται σε αυτή τη διπλωματική, επιλέξαμε τη χρήση μίας μηχανής απόδοσης γραφικών καθώς προσφέρει πλεονεκτήματα όπως έλεγχο χαμηλότερου επιπέδου, μη χρησιμοποίηση έτοιμων “game templates”, ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον αντί κάποιου γραφικού editor και υλοποίηση με γλώσσα προγραμματισμού, μεγαλύτερη ταχύτητα στις δημιουργηθείσες εφαρμογές, δυνατότητα ενσωμάτωσης μεγαλύτερου αριθμού ενοτήτων λογισμικού (modules) και μεγαλύτερη μεταφερσιμότητα σε διαφορετικές πλατφόρμες.

Η μηχανή απεικόνισης που επιλέξαμε είναι η “Irrlicht”, η οποία είναι ανοιχτού κώδικα (open source) και χρησιμοποιήσιμη ανεξαρτήτου πλατφόρμας (cross-platform). Προσφέρει έλεγχο χαμηλού επιπέδου με γλώσσα C++ και πληθώρα δυνατοτήτων τελευταίας τεχνολογίας, ως προς την γραφική απόδοση. Το παιχνίδι που αναπτύξαμε είναι μια μείξη των κατηγοριών δράσης, πρώτου προσώπου βολών και περιπέτειας. Στο τομέα των γραφικών υλοποιήσαμε πολλά σύγχρονα οπτικά εφέ, όπως σωματιδιακά τεχνάσματα για νερό, καπνό και φωτιά, κινούμενες υφές, σκιαστές (shaders), δυναμικές σκιές, δυναμικό φωτισμό, εναλλαγή υφών, διαχωρισμός οθόνης (split screen). Το παιχνίδι διαδραματίζεται σε ποικιλόμορφους εικονικούς κόσμους. Εκεί, ο χρήστης ελέγχει έναν χαρακτήρα ο οποίος έρχεται σε επαφή με άλλους, ελεγχόμενους από το σύστημα. Υπάρχουν χαρακτήρες φιλικοί, εχθρικοί και επίσης περαστικοί και αυτοκίνητα που περιφέρονται στον κόσμο. Υλοποιήσαμε εύρεση συντομότερης διαδρομής, προς κάποιο σημείο, με αναζήτηση A* και ακολούθηση αυτού του μονοπατιού. Τέλος εξασφαλίσαμε επεκτασιμότητα ως προς την κατασκευή νέων επιπέδων, μέσω ενός συστήματος διαχείρισης περιεχομένου των επιπέδων, φορτώνοντας παραμέτρους από αρχεία δεδομένων τύπου XML, αναπτύσσοντας μία μηχανή παιχνιδιών (game engine).

Ευχαριστίες

Ευχαριστώ ιδιαίτερα την επιβλέπουσα καθηγήτριά μου κ. Κατερίνα Μανιά για την καθοδήγησή της στην εκπόνηση αυτής της διπλωματικής εργασίας. Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές κ. Λαγουδάκη Μιχάλη και κ. Δεληγιαννάκη Αντώνιο για τη βοήθεια τους και για το χρόνο που θα αφιερώσουν στην ανάγνωση του κειμένου. Τέλος, ένα μεγάλο ευχαριστώ στους ανθρώπους που με στήριξαν και με βοήθησαν με όλους τους τρόπους, όλα αυτά τα χρόνια.

Ιανουάριος 2010
Νίκος Ζερβουδάκης

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Στόχος αυτής της διπλωματική εργασίας είναι η ανάπτυξη ενός παιχνιδιού για ηλεκτρονικούς υπολογιστές με τριών διαστάσεων γραφικά, κάνοντας χρήση μηχανής απόδοσης τρισδιάστατων γραφικών (rendering engine), το οποίο έχει χαρακτηριστικά αντίστοιχα με αυτά που συναντούμε στα ηλεκτρονικά παιχνίδια του εμπορίου. Επιδιώκουμε σύγχρονα τρισδιάστατα γραφικά με οπτικά τεχνάσματα που προσδίδουν ρεαλισμό, εξομοίωση νόμων της φυσικής σε εικονικό περιβάλλον, τεχνητή νοημοσύνη, όπως εύρεση συντομότερου μονοπατιού με ευριστική αναζήτηση A* και τέλος επεκτασιμότητα μέσω ενός συστήματος διαχείρισης περιεχομένου.

Μία Μηχανή Παιχνιδιών (game engine) είναι λογισμικό το οποίο προσφέρει εξειδικευμένες προγραμματιστικές δυνατότητες για την υλοποίηση αποκλειστικά τρισδιάστατων παιχνιδιών. Παρότι θα μπορούσαμε να χρησιμοποιήσουμε μια ολοκληρωμένη μηχανή παιχνιδιών για την υλοποίηση του παιχνιδιού μας, επιλέξαμε τη χρήση μίας μηχανής απόδοσης γραφικών καθώς προσφέρει πλεονεκτήματα, όπως έλεγχο χαμηλότερου επιπέδου, μη χρησιμοποίηση έτοιμων “game templates”, ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον αντί κάποιου γραφικού editor και υλοποίηση με γλώσσα προγραμματισμού, δυνατότητα ενσωμάτωσης μεγαλύτερου αριθμού ενοτήτων λογισμικού (modules) και άλλα.

Το παιχνίδι που υλοποιούμε είναι μια μείξη διαφόρων ειδών και συγκεκριμένα των κατηγοριών δράσης, πλατφόρμας και βολών πρώτου προσώπου. Ο χρήστης χειρίζεται έναν χαρακτήρα, ο οποίος έχει συγκεκριμένη αποστολή για κάθε επίπεδο του παιχνιδιού. Σε κάθε επίπεδο ο χαρακτήρας αυτός έρχεται σε επαφή με άλλους ελεγχόμενους από το σύστημα, οι οποίοι του δίνουν στοιχεία για την αποστολή του και ενίοτε έχουν εχθρική συμπεριφορά απέναντι του. Σε αυτή τη περίπτωση ο χρήστης πρέπει να τους εξουδετερώσει στοχεύοντας και πυροβολώντας τους. Το παιχνίδι τελειώνει όταν ο χαρακτήρας ολοκληρώσει όλα τα βήματα της αποστολής. Όταν ο χρήστης φέρει σε πέρας την αποστολή του, μεταφέρεται στη σελίδα με τα αποτελέσματα του παιχνιδιού.

Για την ανάπτυξη της εφαρμογής μας, κάνουμε έρευνα στο υπάρχον γνωσικό υπόβαθρο στο **Κεφάλαιο 2**, εξετάζουμε τις διαθέσιμες επιλογές μηχανών γραφικών, αλλά και λογισμικού χρήσιμου για ανάπτυξη παιχνιδιών και κατόπιν αναλύουμε τις επιλογές μας στο **Κεφάλαιο 3**. Στο **Κεφάλαιο 4** σχεδιάζουμε το παιχνίδι μας και καταγράφουμε τις απαιτήσεις του και ακολούθως, σχεδιάζουμε την αρχιτεκτονική του συστήματος που αναπτύσσουμε στο **Κεφάλαιο 5**. Η υλοποίησή μας γίνεται στο **Κεφάλαιο 6.** Αναφερόμαστε σε ένα επίπεδο παιχνιδιού, αν και έχουμε αναπτύξει και άλλα, καθώς η υλοποίηση επιπλέον επιπέδων δεν διαφέρει. Στο **Κεφάλαιο 7** αξιολογούμε το σύστημα με μέθοδο ερωτηματολογίων χρηστικότητας QUIS (Questionnaire for User Interaction Satisfaction[13]) του πανεπιστημίου Maryland. Τέλος μελετάμε τα αποτελέσματα της εργασίας και αναφερόμαστε σε μελλοντικές επεκτάσεις στο **Κεφάλαιο 8**.

Κεφάλαιο 2

Σχετική έρευνα

2.1 Τρισδιάστατα γραφικά υπολογιστών

Τα τρισδιάστατα γραφικά υπολογιστών αποτελούν προσπάθειες απεικόνισης γραφικών τριών διαστάσεων στην απεικόνισης δύο διαστάσεων, οθόνη μιας ψηφιακής συσκευής. Το γεγονός ότι η απεικόνιση χρησιμοποιεί τρεις διαστάσεις τα καθιστά ιδιαίτερα ρεαλιστικά. Τέτοιου είδους γραφικά χρησιμοποιούνται συνήθως από προγράμματα όπως παιχνίδια υπολογιστών και εικονικούς κόσμους.

Στις μέρες μας συναντάμε τρισδιάστατα γραφικά συχνά στον κινηματογράφο για τη δημιουργία σκηνών εικονικών κόσμων με χαρακτηριστικό παράδειγμα οι ταινίες του κύκλου "Ο Άρχοντας των Δαχτυλιδιών", εμπλουτισμένες με ειδικά εφέ που είναι αδύνατον να γυριστούν ως πραγματικές σκηνές, όπως επίσης και σε όλες τις συσκευές με ψηφιακή οθόνη κινητά τηλέφωνα, palmtop, κ.α..

Οι πολλές εντυπωσιακές εφαρμογές των γραφικών τριών διαστάσεων καθιστούν την σχεδίαση τους μια από τις πιο ενδιαφέρουσες σύγχρονες τέχνες. Βέβαια, το πρόβλημα της παρουσίασης γραφικών τριών διαστάσεων σε οθόνη δύο διαστάσεων εξακολουθεί να υφίσταται. Λύση σε αυτά τα προβλήματα δίνει η χρήση εικονικής πραγματικότητας (virtual reality). Η Εικονική Πραγματικότητα αναφέρεται σε "αλληλεπιδραστικά γραφικά πραγματικού χρόνου (real-time) με τρισδιάστατα μοντέλα, συνδυασμένα με μια τεχνολογία απεικόνισης η οποία δίνει τη δυνατότητα στο χρήστη για εμβύθιση στον μοντελοποιημένο κόσμο και τη δυνατότητα για απευθείας χειρισμό"[15].

2.2 Γραφικά υπολογιστών πραγματικού χρόνου

Σε αυτή τη διπλωματική εργασία θα ασχοληθούμε με τριών διαστάσεων (3D) γραφικά υπολογιστών πραγματικού χρόνου (Real Time Computer Graphics[3]). Τα γραφικά υπολογιστών πραγματικού χρόνου είναι διαδοχικές εικόνες (frames), οι οποίες παράγονται από υπολογισμούς σε πραγματικό χρόνο.

Μια από τις πιο κύριες εφαρμογές αυτού του τύπου γραφικών είναι τα 3D διαδραστικά παιχνίδια. Σε αυτά ο χρήστης δίνει εντολές με κάποια συσκευή εισόδου, συνήθως πληκτρολόγιο, ποντίκι ή χειριστήριο παιχνιδιών, και με αυτό το τρόπο αλληλεπιδρά με τον τρισδιάστατο εικονικό κόσμο του παιχνιδιού σε πραγματικό χρόνο.

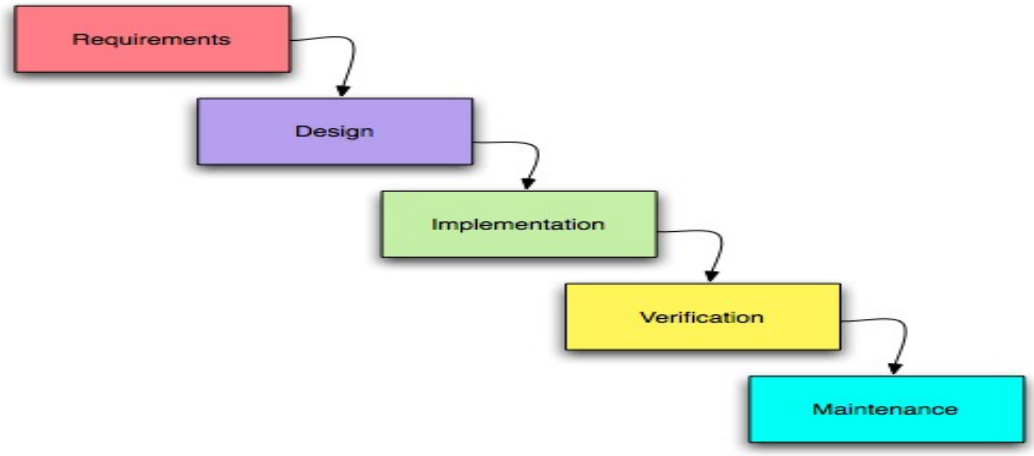
Για την δημιουργία 3D γραφικών πραγματικού χρόνου χρησιμοποιούνται εξισώσεις γραφικού σχεδιασμού, γεωμετρίας και φωτισμού που υπολογίζουν τη μορφή της παραγόμενης στιγμιαίας εικόνας. Οι επιθυμητοί υπολογισμοί δύνανται να προγραμματιστούν απευθείας από τον δημιουργό, όπως επίσης είναι δυνατό να χρησιμοποιηθεί λογισμικό που προσφέρει στον προγραμματιστή έτοιμες γραφικές συναρτήσεις και παραμετροποιήσιμες λειτουργίες απεικόνισης, δηλαδή αναλαμβάνει το τεχνικό κομμάτι της απεικόνισης, το οποίο ονομάζεται μηχανή 3D απόδοσης γραφικών ή απεικόνισης (3D rendering engine[10]). Ο όρος "Απόδοση" (Rendering) στην επιστήμη υπολογιστών, είναι ανάλογος με την απόδοση μιας σκηνής από έναν "καλλιτέχνη". Rendering είναι η διαδικασία της δημιουργίας μιας εικόνας από ένα μοντέλο, μέσω προγραμμάτων ηλεκτρονικών υπολογιστών. Το μοντέλο είναι μια περιγραφή των τρισδιάστατων αντικειμένων σε αυστηρά καθορισμένη γλώσσα.

Περιλαμβάνει γεωμετρία, οπτική γωνία, υφές, φωτισμό και πληροφορίες σκίασης. Σε αυτή την εργασία, για τη παραγωγή γραφικών πραγματικού χρόνου, χρησιμοποιούμε μηχανή 3D απεικόνισης.

2.3 Ανάπτυξη παιχνιδιών

Για την κατασκευή ενός διαδραστικού τρισδιάστατου παιχνιδιού δεν αρκεί μόνο η δημιουργία real time computer graphics αλλά η ανάπτυξη μιας ολοκληρωμένης εφαρμογής, με σενάριο, συγκεκριμένο τρόπο και κανόνες παιχνιδιού, τρισδιάστατους χαρακτήρες και κόσμους, τεχνητή νοημοσύνη των χαρακτήρων του παιχνιδιού, ήχους και μουσική, διεπαφή χρήστη και άλλα. Γι' αυτό, ένα παιχνίδι υψηλών απαιτήσεων απαιτεί την σύμπραξη πολλών επιστημών και πολλών τεχνών πράγμα που σημαίνει ομαδική δουλειά, αφοσίωση και συνεργασία. Η διεπιστημονική ομάδα ανάπτυξης παιχνιδιών πρέπει να αποτελείται από άτομα με γνώσεις άλγεβρας, διανυσματικής ανάλυσης, απειροστικού λογισμού, φυσικής, αρχιτεκτονικής υπολογιστών, γραφικών, δικτύων, τεχνητής νοημοσύνης, επικοινωνίας ανθρώπου μηχανής κ.α. αλλά και σύγχρονων τεχνών (3d modeling, μείξη ήχου, μοντάζ βίντεο, σεναριογραφία κ.α.). Επίσης, κρίσιμος παράγοντας είναι το λογισμικό υλοποίησης που χρησιμοποιείται.

Η διαδικασία ανάπτυξης ενός παιχνιδιού, από την βασική ιδέα μέχρι την τελική μορφή του παιχνιδιού, ονομάζεται κύκλος ανάπτυξης παιχνιδιού (game development cycle[1]). Ένα μοντέλο που χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών, το οποίο θα ακολουθήσουμε σε αυτή την εργασία, είναι το "μοντέλο κύκλου ζωής καταρράκτης" (waterfall life cycle model).



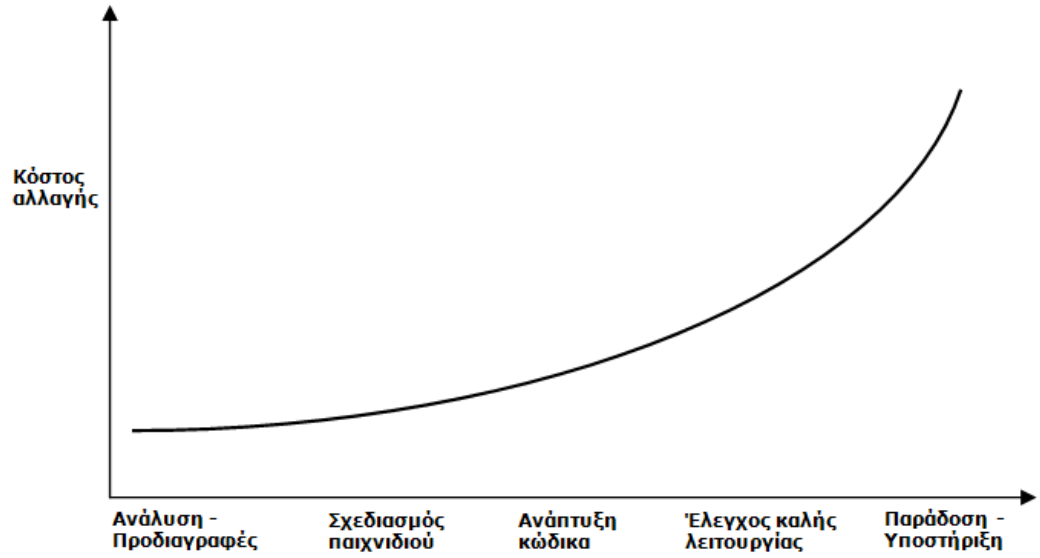
Εικόνα 1: «Το μη-μορφοποιημένο "waterfall model". Η διαδικασία κυλάει από την κορυφή προς τα κάτω, σαν καταρράκτης.»[3]

Τα βήματα του μοντέλου είναι η ανάλυση και προσδιορισμός των απαιτήσεων, δηλαδή των υπηρεσιών, στόχων και περιορισμών του συστήματος. Ο σχεδιασμός του συστήματος και του λογισμικού, όπου καθορίζεται μια γενική αρχιτεκτονική του συστήματος. Η υλοποίηση, κατά το οποίο στάδιο το λογισμικό υλοποιείται ως ένα σύνολο από προγράμματα και ενότητες προγραμμάτων και η ολοκλήρωση και ο έλεγχος του συστήματος, όπου το σύστημα χτίζεται από τις επιμέρους ενότητες και ελέγχεται ως ολοκληρωμένο πλέον σύστημα. Τέλος, η συντήρηση, όπου έχουμε τη διόρθωση των λαθών που δεν είχαν ανακαλυφθεί σε προηγούμενα στάδια του σχεδιασμού και βελτίωση των υπηρεσιών του συστήματος.

Πλεονεκτήματα του μοντέλου καταρράκτη είναι η ευκολία διαχείρισης του σχεδιασμού, ο περιορισμός του κόστους όλης της διαδικασίας παραγωγής του λογισμικού, οι πληροφορίες που παίρνουμε μπορούν να επιστραφούν στα προηγούμενα στάδια (ανάδραση) και τέλος έχουμε έλεγχο του ολοκληρωμένου συστήματος[16].

Εικόνα 2: Οι τρεις συνιστώσες της ανάπτυξης[10]

Ένα βασικό αξίωμα της ανάπτυξης λογισμικού είναι, ότι οι τρεις συνιστώσες, χρόνος, κόστος και ποιότητα, δεν είναι ανεξάρτητες μεταξύ τους. Για το "μοντέλο κύκλου ζωής καταρράκτης" το κόστος της ανάπτυξης περιγράφεται από το ακόλουθο γράφημα.



Εικόνα 3: Το κόστος της ανάπτυξης για το "μοντέλο κύκλου ζωής καταρράκτης"[10].

Όπως παρατηρούμε το κόστος αλλαγής αυξάνεται σε προχωρημένα στάδια ανάπτυξης, άρα το μοντέλο είναι ευαίσθητο σε μεγάλες αλλαγές στις συνθήκες ανάπτυξης.

2.4 Λογισμικό ανάπτυξης παιχνιδιών

Για την ανάπτυξη ενός τρισδιάστατου παιχνιδιού πρέπει να χρησιμοποιηθεί το κατάλληλο λογισμικό. Για ένα εμπορικό παιχνίδι υψηλών απαιτήσεων, συνηθέστερα, η εταιρία αναπτύσσει το λογισμικό που θα χρησιμοποιηθεί για την δημιουργία του

παιχνιδιού ή χρησιμοποιεί λογισμικό που έχει αναπτύξει παλαιότερα η ίδια, αποφεύγοντας να χρησιμοποιήσει ξένο, έτοιμο λογισμικό. Η δημιουργία λογισμικού ανάπτυξης παιχνιδιών είναι εκτός του θέματος αυτής της εργασίας. Για το λόγο αυτό θα αρκεστούμε σε έτοιμο λογισμικό.

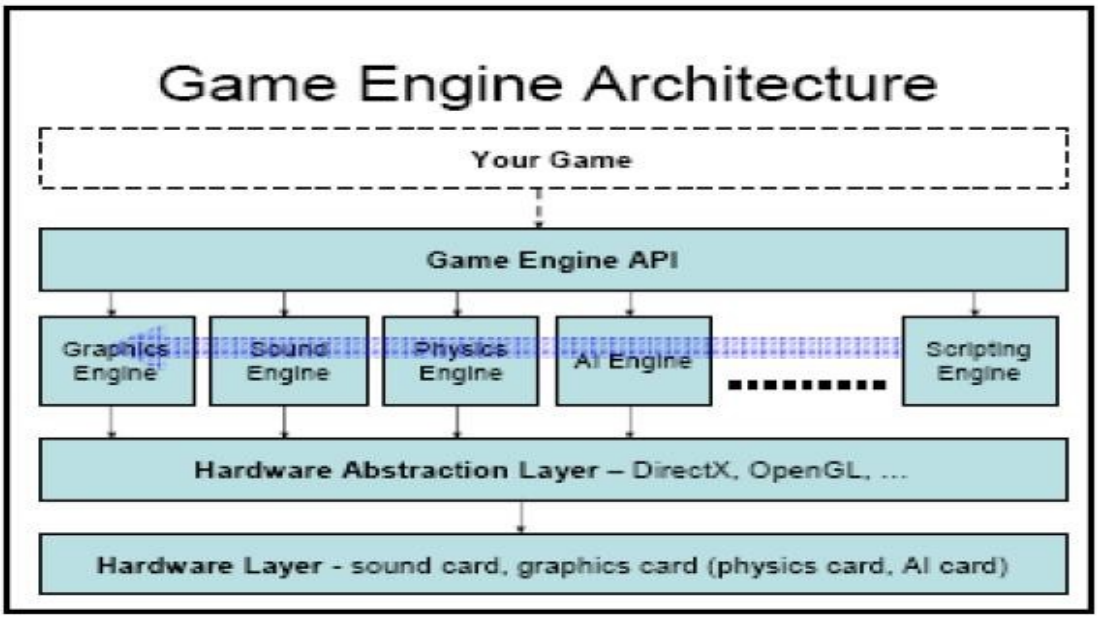
Έτοιμο λογισμικό για game development μπορεί να είναι ένα ολοκληρωμένο περιβάλλον εργασίας (framework[2]) που προσφέρει ένα σύνολο προγραμματιστικών εργαλείων, όπως το XNA framework της Microsoft για τις πλατφόρμες Microsoft XBOX και Microsoft Windows, είτε μεμονωμένα προγραμματιστικά και σχεδιαστικά πακέτα λογισμικού.

Ειδικότερα, όσον αφορά τα γραφικά του παιχνιδιού, όπως αναφέραμε νωρίτερα, χρησιμοποιούνται 3D μηχανές απεικόνισης (rendering engines). Κάθε μηχανή απεικόνισης έχει ουσιαστικές διαφορές ως προς την ευελιξία και τις δυνατότητες που προσφέρει, οπότε είναι αναγκαίο να γίνει έρευνα και σύγκριση των μηχανών πριν γίνει η επιλογή της κατάλληλης για τις απαιτήσεις του εκάστοτε παιχνιδιού.

Αυτές οι μηχανές εκτός από το να είναι αποκλειστικά απεικόνισης, δηλαδή να παρέχουν στον προγραμματιστή εργαλεία μόνο για την απεικόνιση γραφικών, ενδέχεται να είναι μέρος μιας μηχανής παιχνιδιών (game engine) η οποία προσφέρει επιπλέον λειτουργίες, όπως υποστήριξη γλώσσας σεναρίου παιχνιδιών (game script), ενσωματωμένα σχεδιαστικά προγράμματα, βιβλιοθήκες για ήχο, φυσική (physics) όπως αναγνώριση συγκρούσεων (collision detection), υποστήριξη συστήματος δικτύου (networking system) και άλλα.

Μια μηχανή παιχνιδιών[5] είναι μια συλλογή αυτόνομων ενοτήτων συστήματος (modules) και διεπαφών (interfaces) που επιτρέπουν στον δημιουργό να επικεντρωθεί στο παραγόμενο περιεχόμενο για καλύτερη εμπειρία παιχνιδιού που παρέχεται στον χρήστη (gameplay[6]) και όχι σε τεχνικά ζητήματα.

Από τον ορισμό παρατηρούμε ότι ο συνδυασμός μιας μηχανής γραφικών με ενότητες λογισμικού που παρέχουν δυνατότητες χρήσιμες για την ανάπτυξη παιχνιδιών, οδηγούν στην ανάπτυξη μιας μηχανής παιχνιδιών. Στην επόμενη εικόνα βλέπουμε την θέση της μηχανής παιχνιδιών στην επικοινωνία του παιχνιδιού με το υλικό.



Εικόνα 4: Η θέση της μηχανής παιχνιδιών, στην επικοινωνία του παιχνιδιού με το υλικό[10]

Το λογισμικό που χρησιμοποιείται σε αυτή την εργασία είναι ανοιχτού κώδικα με ελεύθερη άδεια, ώστε να μπορούμε να το χρησιμοποιήσουμε χωρίς πολλούς περιορισμούς και παράλληλα να εξετάσουμε τον πηγαίο κώδικα του λογισμικού που κάνουμε χρήση.

Για να "χτίσουμε" την game engine του παιχνιδιού μας έχουμε την ανάγκη λογισμικού για τις ακόλουθες λειτουργίες: 1)απεικόνιση τρισδιάστατων γραφικών, 2)αναπαραγωγή ήχων και μουσικής, 3)απόδοση φυσικής, 4)game scripting και 4)τεχνητή νοημοσύνη.

2.5 Γιατί μηχανή απόδοσης γραφικών?

Για την απεικόνιση γραφικών επιλέγουμε τη λύση κάποιας μηχανής απόδοσης γραφικών. Συγκρίνοντας μια μηχανή γραφικών με μια έτοιμη μηχανή παιχνιδιών, διακρίνουμε τα εξής πλεονεκτήματα των μηχανών γραφικών:

- Έλεγχος χαμηλότερου επιπέδου
- Δυνατότητα ενσωμάτωσης πολύ μεγαλύτερου αριθμού βιβλιοθηκών και ενοτήτων λογισμικού
- Δεν χρησιμοποιούμε έτοιμα "game templates", άρα έχουμε μεγαλύτερη ελευθερία δημιουργίας
- Υλοποίηση με γλώσσα προγραμματισμού και όχι απλώς σε game script
- Υλοποίηση σε κανονικό προγραμματιστικό περιβάλλον και όχι σε κάποιο γραφικό editor, άρα και περισσότερες δυνατότητες
- Δημιουργία δικιάς μας μηχανής παιχνιδιού
- Συνηθέστερα, μεγαλύτερη ταχύτητα στις δημιουργηθείσες εφαρμογές
- Μεγαλύτερη μεταφερσιμότητα σε λειτουργικά συστήματα και πλατφόρμες
- Η γνώση που αποκτάμε κατά την ανάπτυξη ενός παιχνιδιού, μπορεί να είναι χρήσιμη στο μέλλον για ανάπτυξη μιας άλλης τρισδιάστατης εφαρμογής με την ίδια μηχανή γραφικών, που όμως δεν είναι παιχνίδι.

Μια τέτοια μηχανή δεν είναι ολοκληρωμένη game engine, όμως συχνά παρέχει κάποιες από τις πολύ απαραίτητες δυνατότητες για κατασκευή παιχνιδιών, όπως φυσική, ήχους, scripting και τεχνητή νοημοσύνη , οπότε αρχικά πρέπει να γίνει επιλογή της μηχανής απόδοσης γραφικών και εν συνεχεία των υπολοίπων συστατικών, που θα χρησιμοποιήσουμε για την υλοποίηση του παιχνιδιού.

2.6 Σύγκριση μηχανών απόδοσης γραφικών

Ακολούθως, θα εξετάσουμε κάποιες επιλογές μηχανών απεικόνισης γραφικών (rendering engines). Οι μηχανές που εξετάζουμε είναι ανοιχτού κώδικα και παρέχουν ανεξαρτησία πλατφόρμας.

Η Ogre[8] (Αντικειμενοστραφής μηχανισμός απόδοσης γραφικών), είναι μια ευέλικτη 3D μηχανή προσανατολισμένη στις σκηνές, γραμμένη σε C++ που αποσκοπεί να καταστήσει ευκολότερη την ανάπτυξη εφαρμογών που χρησιμοποιούν

3D επιτάχυνση υλικού (3D Hardware Acceleration). Για άλλες δυνατότητες, όπως ήχο, δικτύωση, AI, σύγκρουση, φυσική κλπ, θα πρέπει να ενσωματωθούν συμπληρωματικές βιβλιοθήκες υποστήριξης των παραπάνω δυνατοτήτων.

Η Ogre προσφέρει συνολικά περισσότερες δυνατότητες στην απεικόνιση γραφικών από όλες τις άλλες μηχανές απόδοσης γραφικών, όπως οπτικά εφέ, σκιαστές, υποστήριξη τύπων αρχείων, ταχύτητα. Χρησιμοποιείται από αρκετά πανεπιστήμια παγκοσμίως για ερευνητικούς και εκπαιδευτικούς σκοπούς και υποστηρίζεται από μεγάλη κοινότητα χρηστών. Επίσης έχουν εκδοθεί βιβλία για την εκμάθηση της, λόγου χάριν το "Pro OGRE 3D Programming"[17]. Υπάρχουν πολλές εμπορικές και μη εφαρμογές που κάνουν χρήση της, όπως τα "Zero Gear" από την "NibleBit", και το "Torchlight" από την "Runic Games". Το κύριο μειονέκτημά της είναι ότι εκτός από γραφική απόδοση, δεν προσφέρει άλλα χαρακτηριστικά για ανάπτυξη παιχνιδιών και σε συνδυασμό με την πληθώρα δυνατοτήτων της καθιστά δύσκολη τη χρήση της από αρχάριους, στον τομέα των παιχνιδιών, προγραμματιστές.

Εικόνα 5: "Torchlight" με χρήση Ogre

Η Crystal space[7] είναι μια μηχανή γραφικών που στοχεύει κυρίως στη κατασκευή παιχνιδιών. Γι' αυτό το λόγο οι δημιουργοί του ανέπτυξαν και ένα πλαίσιο εργασίας (framework) για την ανάπτυξη 3D παιχνιδιών που ονομάζεται CEL (Crystal entity Layer) και είναι ένας σύντροφος της Crystal space, προσθέτοντας διαχείριση οντοτήτων και παρέχοντας έτσι μια μηχανή παιχνιδιού .

Είναι προγραμματισμένη σε C++ χρησιμοποιώντας αντικειμενοστραφή προγραμματισμό. Συνήθως χρησιμοποιείται ως μηχανή του παιχνιδιού, αλλά το πλαίσιο εργασίας που προσφέρει είναι περισσότερο γενικό και μπορεί να χρησιμοποιηθεί για κάθε τύπο 3D απεικόνισης.

Παρέχει υψηλή φορητότητα και υποστηρίζει Microsoft Windows, Linux, UNIX, Mac και άλλα. Ο κώδικας της είναι ανοιχτός και διανέμεται υπό άδεια LGPL. Μπορεί να χρησιμοποιήσει Opengl σε όλες τις πλατφόρμες, SDL σε όλες τις πλατφόρμες και X11 σε Unix. Μπορεί επίσης να χρησιμοποιήσει τις ρουτίνες

συναρμολόγησης NASM και MMX.

Είναι επικεντρωμένη σε μεγάλο βαθμό στη δημιουργία παιχνιδιών και αποτελεί την ταχύτατη λύση στην ανάπτυξη παιχνιδιού. Αυτό οφείλεται στο γεγονός ότι η ανάπτυξή του βασίστηκε στις βελτιστοποιήσεις για τους ανθρώπους που θέλουν να δημιουργήσουν ένα παιχνίδι χωρίς πολύ κόπο.

Η Crystal Space σε συνεργασία με τον οργανισμό Blender foundation, δημιούργησαν μια ομάδα εργασίας που ανέπτυξε ένα ολοκληρωμένο παιχνίδι ανοιχτού κώδικα που κυκλοφόρησε εμπορικά, με όνομα"Yo Frankie". Για την ιστορία, παράλληλα με το "Yo Frankie" η Blender ανέπτυξε την "Blender Game Engine – BGE".

Παρόλα αυτά, η Crystal Space σήμερα δεν είναι τόσο αναπτυσσόμενη όσο παλαιότερα και η κοινότητα χρηστών που την υποστηρίζει έχει αποδυναμωθεί αισθητά. Για αυτό το λόγο δεν αποτελεί επιλογή μας η Crystal Space.

Εικόνα 6: "yoFrankie" με χρήση Crystal space

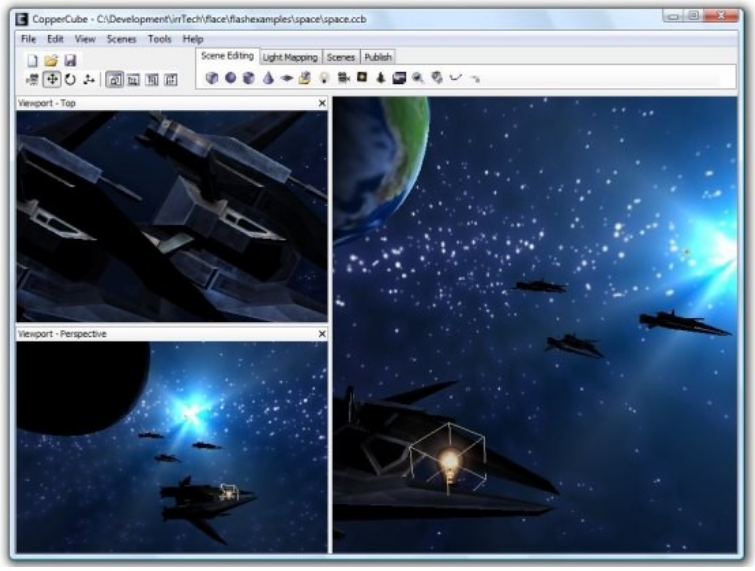
Η Irrlicht Engine[4] είναι μια μηχανή απόδοσης 3d γραφικών, γραμμένη σε γλώσσα C++. Είναι ανοιχτού κώδικα και διαθέσιμη για πολλές πλατφόρμες και με μεταφορά σε πολλές γλώσσες προγραμματισμού (bindings).

Υποστηρίζει και DirectX, Opengl και επίσης διαθέτει τον δικό της λογισμικό για απόδοση γραφικών (software renderer). Οι δημιουργοί της, προσφέρουν συνεχώς νέες εκδόσεις της μηχανής, που διορθώνουν τυχόν προβλήματα και επίσης προσθέτουν σύγχρονα χαρακτηριστικά που συναντώνται στις εμπορικές μηχανές. Υποστηρίζεται από μια μεγάλη κοινότητα χρηστών και υπάρχουν πολλές εργασίες υπό ανάπτυξη που την χρησιμοποιούν.

Η Irrlicht είναι αποκλειστικά μηχανή απόδοσης γραφικών. Δεν παρέχει βοηθήματα και λειτουργίες για ανάπτυξη παιχνιδιών, με εξαίρεση βασική μηχανή φυσικής και υποστήριξη εισόδου χρήστη. Παρόλα αυτά, λόγω της μεγάλη κοινότητας που την υποστηρίζει, έχουν αναπτυχθεί βοηθήματα για την Irrlicht που μπορούν να χρησιμοποιηθούν παράλληλα ή και αυτόνομα. Παραδείγματος χάριν, η εταιρία "Ambiera", έχει αναπτύξει μια βιβλιοθήκη ήχου με το όνομα "IrrKlang" και μια βιβλιοθήκη για ανάγνωση XML αρχείων με το όνομα "IrrXML". Υπάρχουν και

άλλες βοηθητικές βιβλιοθήκες και εφαρμογές δημιουργημένες από την κοινότητα.

Τα **κύρια πλεονεκτήματα της Irrlicht** είναι, η υποστήριξη των αρκετά σύγχρονων δυνατοτήτων που προσφέρουν οι τελευταίας γενιάς κάρτες γραφικών, η άριστη τεκμηρίωση της βιβλιοθήκης της, αλλά και του πηγαίου κώδικα της, η αντικειμενοστραφής μορφή της, η μεγάλη και ενεργή κοινότητα χρηστών, η σταθερότητα και η ταχύτητά της, και τέλος, οι εφαρμογές που την χρησιμοποιούν είναι εύκολα μεταφέρσιμες σε άλλες πλατφόρμες



Εικόνα 7: "Coppercube" με χρήση Irrlicht

2.7 Επιλογή λογισμικού ανάπτυξης παιχνιδιών

Η έρευνα που κάναμε για τις μηχανές απόδοσης γραφικών, αλλά και γενικότερα για λογισμικό ανάπτυξης παιχνιδιών είναι πολύ μεγαλύτερη από αυτή που παρουσιάσαμε. Ήταν αναγκαίο να γίνουν πολλές δοκιμές και έρευνα εις βάθος ώστε ο σχεδιασμός που θα κάναμε να είναι υλοποιήσιμος από ένα άτομο μόνο, δίχως εμπειρία στην ανάπτυξη εφαρμογών αυτού του είδους.

Το πρώτο στάδιο των δοκιμών περιελάμβανε αρχικά την εγκατάσταση δεκάδων μηχανών απεικόνισης και κατόπιν τη δημιουργία δοκιμαστικών εφαρμογών. Στη συνέχεια, απορρίψαμε κάποιες λόγω προβλημάτων στην εγκατάσταση ή στη δημιουργία δοκιμαστικών εφαρμογών. Για παράδειγμα, με την μηχανή "Crystal Space" παρατηρήθηκαν προβλήματα σταθερότητας και συμβατότητας με τον υπολογιστή που έγιναν οι δοκιμές, καθώς έτοιμες δοκιμαστικές εφαρμογές που παρέχονται από την επίσημη διανομή δεν λειτουργούσαν με το υλικό μας.

Στο δεύτερο στάδιο οι εφαρμογές που δημιουργήσαμε προηγουμένως, εμπλουτίστηκαν με χαρακτηριστικά που θέλαμε να έχει το παιχνίδι μας, όπως τρισδιάστατοι χαρακτήρες. Για να γίνει αυτό, αναζητήσαμε τρισδιάστατα πλέγματα (3d meshes) που μπορούν να μπουν στο παιχνίδι μας. Αυτά έπρεπε να υποστηρίζονται από κάποια από τις μηχανές, να υπάρχει δυνατότητα για κίνηση των πλεγμάτων μέσω της μηχανής και επίσης να έχουμε άδεια να τα χρησιμοποιήσουμε.

Στο τελικό στάδιο, στην πιο σύνθετη εφαρμογή μας, προσθέσαμε επιπλέον βιβλιοθήκες, ώστε να έχουμε ήχο, είσοδο-έξοδο σε αρχεία, βοηθητικές ρουτίνες για τεχνητή νοημοσύνη και άλλα.

Βάσει των αποτελεσμάτων που είχαμε και της έρευνας των μηχανών απόδοσης γραφικών, αποφασίσαμε να δομήσουμε τη μηχανή του παιχνιδιού μας χρησιμοποιώντας τα ακόλουθα στοιχεία: **Irrlicht** για απόδοση δισδιάστατων και τρισδιάστατων γραφικών πραγματικού χρόνου, δημιουργία διεπαφής χρήστη, έλεγχο εισόδου χρήστη, είσοδο και έξοδο αρχείων, απόδοση βασικής φυσικής, την ενότητα **IrrKlang** για διαχείριση και αναπαραγωγή ήχων, την ενότητα **IrrXML** για προσπέλαση σε αρχεία δεδομένων τύπου XML και την ενότητα **IrrAI** για βοηθητικές λειτουργίες, σχετικές με τεχνητή νοημοσύνη.

Από τα παραπάνω API **(**Application programming interface**)**, χρησιμοποιήσαμε τις βιβλιοθήκες τους σε γλώσσα C++ και δημιουργήσαμε τον πυρήνα της εφαρμογής μας, σε προγραμματιστικό περιβάλλον Microsoft Visual Studio Express 2008 σε πλατφόρμα Microsoft Windows XP 32bit. Έτσι η τελική μορφή του παιχνιδιού μας ως εφαρμογή, είναι μια Windows 32bit application, με απαίτηση για να εκτελεστεί τις δυναμικές βιβλιοθήκες διασύνδεσης (DLL) των Irrlicht και IrrKlang.

Αξίζει να σημειωθεί πως τα παραπάνω API, παρότι έχουν όλα το πρόθεμα "Irr" και ενώ ο βασικός σκοπός τους είναι να χρησιμοποιηθούν μαζί με την Irrlicht για την κατασκευή παιχνιδιών, μπορούν να χρησιμοποιηθούν η κάθε μία ξεχωριστά και αυτόνομα. Επίσης, δεν έχουν δημιουργηθεί από τις ίδιες ομάδες εργασίας και η κάθε μια αποτελεί ξεχωριστή διανομή λογισμικού.

Κεφάλαιο 3

Τεχνολογική βάση

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε τα επιμέρους δομικά στοιχεία της μηχανής του παιχνιδιού μας, ως προς την τεχνολογία και τις λύσεις που προσφέρουν. Κάθε υποενότητα του κεφαλαίου ασχολείται με ένα στοιχείο.

3.2 Irrlicht

Όπως προαναφέραμε η Irrlicht Engine είναι μια μηχανή απόδοσης 3D γραφικών γραμμένη σε γλώσσα C++. Είναι ανοιχτού κώδικα και διαθέσιμη για πολλές πλατφόρμες και με μεταφορά σε πολλές γλώσσες προγραμματισμού. Θα χρησιμοποιηθεί για απόδοση δισδιάστατων και τρισδιάστατων γραφικών πραγματικού χρόνου, δημιουργία διεπαφής χρήστη, έλεγχο εισόδου χρήστη, είσοδο και έξοδο αρχείων, απόδοση βασικής φυσικής.

3.2.1 Πλατφόρμες

Η αρχιτεκτονική της είναι ανεξάρτητη πλατφόρμας, που σημαίνει στην προκειμένη περίπτωση ότι οι εφαρμογές που την χρησιμοποιούν είναι εκτελέσιμες σε περισσότερες από μία πλατφόρμα. Υποστηρίζει τις εξής πλατφόρμες:

- Windows 98, ME, NT 4, 2000, XP, XP64, Vista, CE
- Linux
- OSX
- Sun Solaris/SPARC
- Όλες τις πλατφόρμες που χρησιμοποιούν SDL

3.2.2 API απόδοσης γραφικών

Η Irrlicht υποστηρίζει έξι API απόδοσης γραφικών που προσφέρουν hardware abstraction layers (HAL), για την επικοινωνία με το υλικό:

- Direct3D 8
- Direct3D 9
- Opengl 2-3
- Η Irrlicht software renderer
- Η Burningsvideo software renderer
- Μια null συσκευή.

Όταν χρησιμοποιείται η μηχανή Irrlicht, ο προγραμματιστής δεν χρειάζεται να γνωρίζει ποιο από τα παραπάνω API η μηχανή χρησιμοποιεί, καθώς οι λειτουργίες της είναι αφηρημένες (abstract). Υπάρχουν τρεις λόγοι για τους οποίους η μηχανή δεν εστιάζεται μόνο σε ένα API: Πρώτος λόγος είναι η **απόδοση**, καθώς ορισμένοι

προσαρμογείς γραφικών αποδίδουν καλύτερα με Opengl, ενώ άλλοι με Direct3D. Το API Direct3D δεν υπάρχει σε Mac ή Linux αντίθετα με το Opengl . Επίσης υπάρχουν οι Irrlicht software renderers, οι οποίοι θα λειτουργούν πάντα σε οποιαδήποτε πλατφόρμα. Με τον τρόπο αυτό οι εφαρμογές που χρησιμοποιούν την μηχανή λειτουργούν σε πολλές πλατφόρμες και είναι συμβατές με μεγάλη ποικιλία υλικού οπότε έχουμε **ανεξαρτησία πλατφόρμας** . Τέλος αντιμετωπίζουμε τα **προβλήματα οδηγών**, τα οποία είναι ένα κοινό πρόβλημα που αντιμετωπίζει ένας χρήστης κατά τη χρήση 3D λογισμικού. Υπάρχουν χιλιάδες διαμορφώσεις υλικού και συχνά παιχνίδια και εφαρμογές 3D δεν λειτουργούν επειδή υπάρχει ένα μη συμβατό πρόγραμμα οδήγησης υλικού. Αυτό λύνεται δίνοντας στο χρήστη τη δυνατότητα να επιλέξει το οδηγό.

3.2.3 Οπτικά εφέ

Η Irrlicht έχει την δυνατότητα να προσθέτει στην σκηνή ρεαλιστικά οπτικά εφέ και τεχνάσματα. Κάποια από αυτά είναι: ρεαλιστικές υδάτινες επιφάνειες, δυναμικός φωτισμός, δυναμικές σκιές με χρήση stencil buffer, πινακίδες, bump χαρτογράφηση, σφαιρική χαρτογράφηση, διαφανή αντικείμενα, εφέ τριχώματος σε αντικείμενο, εφέ για χιόνι, εφέ για φωτιά και καπνό, απεικόνιση ουρανού με sky-box, ομίχλη.

3.2.4 Υποστηριζόμενοι τύποι αρχείων

Η Irrlicht υποστηρίζει πληθώρα μορφών αρχείων. Τα αρχεία, είναι σε θέση να φορτωθούν και εν μέρει επίσης να εγγραφούν απευθείας από την μηχανή. Με τον τρόπο αυτό τα δεδομένα δεν χρειάζονται μετατροπή για τη χρήση τους με την Irrlicht εξοικονομώντας χρόνο ανάπτυξης. Ο κατάλογος των υποστηριζόμενων μορφών συνεχώς αυξάνεται κατά την εξέλιξη της μηχανής.

Υποστηριζόμενοι τύποι αρχείων για υφές (textures):

Όνομα (κατάληξη, ανάγνωση (r) / εγγραφή (w))

- JPEG File Interchange Format (.jpg, r/w)
- Portable Network Graphics (.png, r/w)
- Truevision Targa (.tga, r/w)
- Windows Bitmap (.bmp, r/w)
- Adobe Photoshop (.psd, r)
- Zsoft Paintbrush (.pcx, r/w)
- Portable Pixmaps (.ppm, r/w)
- Quake 2 textures (.wal, r)

Υποστηριζόμενοι τύποι αρχείων για τρισδιάστατα πλέγματα (meshes):

Όνομα (κατάληξη, ανάγνωση (r) / εγγραφή (w))

- Irrlicht scenes (.irr, r/w)
- Irrlicht static meshes (.irrmesh, r/w)
- 3D Studio meshes (.3ds, r)
- B3D files (.b3d, r)
- Alias Wavefront Maya (.obj, r/w)
- Lightwave Objects (.lwo, r)

- COLLADA 1.4 (.xml, .dae, r/w)
- Microsoft DirectX (.x, r) (binary & text)
- Milkshape (.ms3d, r)
- OGRE meshes (.mesh, r)
- My3DTools 3 (.my3D, r)
- Pulsar LMTools (.lmts, r)
- Quake 3 levels (.bsp, r)
- Quake 2 models (.md2, r)
- Quake 3 models (.md3, r)
- DeleD (.dmf, r)
- FSRad oct (.oct, r)
- Cartography shop 4 (.csm, r)
- STL 3D files (.stl, r/w)

3.2.5 Κίνηση χαρακτήρων

Επί του παρόντος υπάρχουν δύο τύποι κίνησης (animation) χαρακτήρων που εφαρμόζονται: **Morph target animation**: Οι χαρακτήρες παρεμβάλλονται γραμμικά από το ένα καρέ (frame) στο επόμενο. Αυτό γίνεται στη σειρά παιχνιδιών Quake, ενώ με τον ίδιο τρόπο λειτουργεί και η μηχανή Irrlicht κατά την εισαγωγή MD2 και Md3 αρχείων. **Σκελετικό animation**: Ένα πλέγμα χειραγωγείται από κινούμενες αρθρώσεις. Η Irrlicht ακολουθεί αυτή τη τακτική κατά τη φόρτωση Ms3d, X και B3d αρχείων. Είναι εύκολο να συνδεθούν τα αντικείμενα σε τμήματα του animated mesh. Είναι, για παράδειγμα, δυνατό να επισυναφθεί ένα όπλο στο χέρι ενός χαρακτήρα, το οποίο μπορεί να μετακινηθεί όπως οι κινήσεις στο χέρι του χαρακτήρα με μία μόνο γραμμή κώδικα.

3.2.6 Διαχείριση σκηνών

Η απεικόνιση στην Irrlicht γίνεται χρησιμοποιώντας ένα ιεραρχικό γράφο σκηνών. Οι κόμβοι σκηνών συνδέονται μεταξύ τους και ακολουθούν τις κινήσεις των άλλων, μπορούν να διαλέγουν τα παιδιά τους που βλέπουν τον οπτικό τους κώνο, και είναι σε θέση να κάνουν ανίχνευση σύγκρουσης. Ένας κόμβος σκηνής μπορεί για παράδειγμα να είναι μια κάμερα, ένα εσωτερικό ή εξωτερικό επίπεδο, ένας σκελετικός κινούμενος χαρακτήρας, κινούμενο νερό ή οτιδήποτε άλλο.

Με τον τρόπο αυτό η μηχανή μπορεί να κάνει μείξη εσωτερικού και εξωτερικού χώρου χωρίς να τους ενώσει δίνοντας στον προγραμματιστή πλήρη έλεγχο πάνω σε οτιδήποτε συμβαίνει στην σκηνή. Αυτή η διαχείριση είναι πολύ εύκολα επεκτάσιμη αφού μπορούν να προστεθούν οι επιθυμητοί σκηνικοί κόμβοι όπως πλέγματα, φορτωτές υφών και στοιχεία διεπαφής χρήστη.

3.2.7 Υλικά και σκιαστές

Για να είναι σε θέση να δημιουργηθούν ρεαλιστικά περιβάλλοντα με ταχύτητα υπάρχουν πολλά κοινά υλικά που διατίθεται ενσωματωμένα στη μηχανή. Μερικά υλικά στηρίζονται στον σταθερής λειτουργίας αγωγό (fixed function pipeline), χαρτογραφημένα με γεωμετρία φωτός για παράδειγμα, και μερικά στηρίζονται στον προγραμματιζόμενο αγωγό (programmable pipeline), χαρτογραφημένα ανά pixel για παράδειγμα, καθώς το σύγχρονο 3d υλικό προσφέρει αυτές τις δυνατότητες και η

Irrlicht τις αξιοποιεί.

Είναι δυνατόν να συνδυαστούν αυτοί οι τύποι υλικών σε μια σκηνή χωρίς προβλήματα ενώ όταν χρησιμοποιηθεί υλικό που δεν παρέχει τις απαραίτητες προϋποθέσεις τότε η μηχανή παρέχει υλικά οπισθοχώρησης. Ωστόσο, εάν τα ενσωματωμένα υλικά δεν είναι αρκετά, είναι δυνατόν να προστεθούν νέα υλικά κατά το χρόνο εκτέλεσης, χωρίς την ανάγκη τροποποίησης και μεταγλώττισης της μηχανής.

Επί του παρόντος, οι γλώσσες σκιαστών που υποστηρίζονται είναι: ARB Fragment and Vertex Programs, Pixel and Vertex Shaders 1.1 to 3.0, HLSL και GLSL

3.2.8 Χρήση της Irrlicht

Η ενσωμάτωση της μηχανής απόδοσης γραφικών Irrlicht, στο περιβάλλον εργασίας μας, το οποίο είναι Microsoft Visual C++ 2008 Express Edition σε λειτουργικό σύστημα Microsoft Windows 32 bit, γίνεται με την ενσωμάτωση του αρχείου ενσωμάτωσης (include file), τη σύνδεση με το αρχείο βιβλιοθήκης (library file) και την δυναμική διασύνδεση κατά την εκτέλεση, με την βιβλιοθήκη δυναμικής διασύνδεσης (dll – Dynamic Link Library).

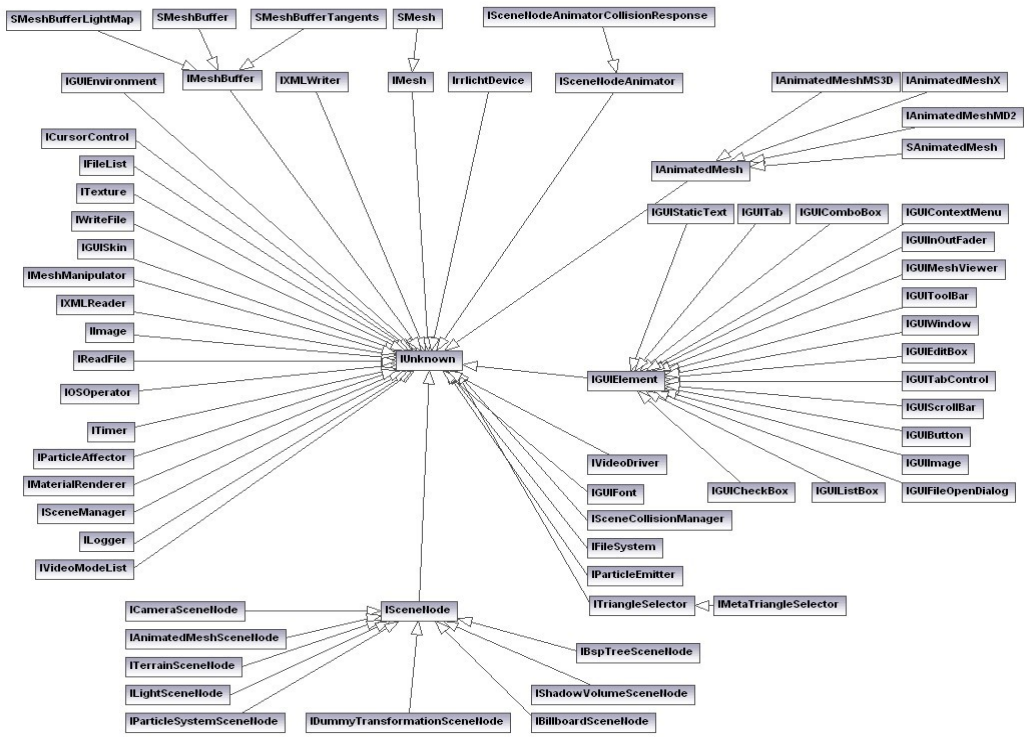
Η χρήση της Irrlicht είναι απλή και εύκολα κατανοητή. Όλες οι λειτουργίες της μηχανής, παρέχονται από την οντότητα της, **IrrlichtDevice.** Παρακάτω είναι η δήλωση αυτής της οντότητας με όλες τις λειτουργίες της.

```
// Copyright (C) 2002-2008 Nikolaus Gebhardt
class IrrlichtDevice : public virtual IReferenceCounted{
public:
        virtual bool run() = 0;
        virtual void yield() = 0;
        virtual void sleep(u32 timeMs, bool pauseTimer=false) = 0;
        virtual video::IVideoDriver* getVideoDriver() = 0;
        virtual io::IFileSystem* getFileSystem() = 0;
        virtual gui::IGUIEnvironment* getGUIEnvironment() = 0;
        virtual scene::ISceneManager* getSceneManager() = 0;
        virtual gui::ICursorControl* getCursorControl() = 0;
        virtual ILogger* getLogger() = 0;
        virtual video::IVideoModeList* getVideoModeList() = 0;
        virtual IOSOperator* getOSOperator() = 0;
        virtual ITimer* getTimer() = 0;
        virtual void setWindowCaption(const wchar_t* text) = 0;
        virtual bool isWindowActive() const = 0;
        virtual bool isWindowFocused() const = 0;
        virtual bool isWindowMinimized() const = 0;
        virtual bool isFullscreen() const = 0;
        virtual video::ECOLOR_FORMAT getColorFormat() const = 0;
        virtual void closeDevice() = 0;
        virtual const c8* getVersion() const = 0;
        virtual void setEventReceiver(IEventReceiver* receiver) = 0;
        virtual IEventReceiver* getEventReceiver() = 0;
        virtual bool postEventFromUser(const SEvent& event) = 0;
        virtual void setInputReceivingSceneManager(scene::ISceneManager* sceneManager) = 0;
        virtual void setResizeAble(bool resize=false) = 0;
        virtual bool activateJoysticks(core::array<SJoystickInfo> & joystickInfo) = 0;
};
```

Οι παραπάνω λειτουργίες προσφέρουν πρόσβαση σε σημαντικές οντότητες της μηχανής όπως IvideoDriver, IfileSystem, IGUIEnvironment, IsceneManager, IcursorControl, Ilogger, IOSOperator, Itimer, IeventReceiver. Οι προαναφερθείσες οντότητες προσφέρουν έλεγχο και λειτουργίες, πάνω στους εξής τομείς :

- **IvideoDriver :** στην κάρτα γραφικών,
- **IfileSystem:** στο σύστημα αρχείων,
- **IGUIEnvironment :** στη γραφική διεπαφή χρήστη,
- **IsceneManager :** στην τρισδιάστατη σκηνή,
- **IcursorControl:** στον κέρσορα του ποντικιού,
- **Ilogger :** στο αρχείο καταγραφής της Irrlicht
- **IOSOperator :** σε στοιχεία του λειτουργικού συστήματος,
- **Itimer :** στο χρονομετρητή της Irrlicht
- **IeventReceiver :** στο διαχειριστή γεγονότων της Irrlicht

Οι παραπάνω οντότητες της Irrlicht είναι διακριτά καθορισμένες, και με αυτό το τρόπο είναι εύκολη και ξεκάθαρη η χρήση των επιμέρους στοιχείων της μηχανής και των μεθόδων αυτών. Παρακάτω παραθέτουμε ένα διάγραμμα οντοτήτων της Irrlicht σε γλώσσα UML. Δημιουργήθηκε κάνοντας χρήση reverse engineering με την εφαρμογή Altova Umodel.



Εικόνα 8: UML διάγραμμα οντοτήτων της Irrlicht

3.3 IrrKlang

Η IrrKlang θα χρησιμοποιηθεί για αναπαραγωγή μουσικής και ήχων με εκμετάλλευση των δυνατοτήτων του υλικού ώστε να προσφέρει ηχητικά εφέ και τρισδιάστατο ήχο. Ο τρισδιάστατος ήχος δίνει μια χωρική διάσταση στον ήχο που σημειοδοτεί την κατεύθυνση και την απόσταση μιας πηγής ήχου, μέσω στοιχείων του ακουστικού περιβάλλοντος όπως περιτύλιξη, κατευθυντικότητα, πλάτος και διαμόρφωση του χώρου.

3.3.1 Εισαγωγή

Η IrrKlang είναι μια 2D και 3D μηχανή ήχου και βιβλιοθήκη (Windows, Mac OS X, Linux) η οποία υποστηρίζει τους πιο σημαντικούς τύπους αρχείων ήχου WAV, MP3, OGG, FLAC, MOD, XM, IT, S3M, και μπορεί να χρησιμοποιηθεί σε # C + + και όλες τις .NET γλώσσες (C, VisualBasic.NET, κλπ).

Έχει όλα τα γνωστά χαρακτηριστικά από τις χαμηλού επίπεδου βιβλιοθήκες ήχου καθώς και πολλές χρήσιμες λειτουργίες, όπως μια εξελιγμένη μηχανή πραγματικού χρόνου, επεκτάσιμη ανάγνωση ήχου, single και multithreading υποστήριξη, εξομοίωση 3d ήχου για παλιό υλικό που δεν υποστηρίζει κάτι τέτοιο, ένα plugin σύστημα, πολλαπλά μοντέλα rolloff και άλλα. Όλα αυτά μπορούν να προσεγγιστούν μέσω ενός εξαιρετικά απλού API. Η irrKlang διατίθεται δωρεάν για μη-εμπορική χρήση.

3.3.2 Ηχητικά εφέ

Εκτός από το φαινόμενο Doppler για 3D ήχους η irrKlang αυτή τη στιγμή υποστηρίζει τα εξής εφέ, τόσο για 2D όσο και για 3D ήχους, χορωδίες, compressor, παραμόρφωση, echo, flanger, gargle, interactive 3d audio επιπέδου 2 αντήχηση, επιπτώσεις parametric equalizer, κύματα reverb.

Είναι δυνατόν να απενεργοποιηθούν και να τροποποιηθούν αυτές οι επιπτώσεις κατά τη διάρκεια αναπαραγωγής για κάθε ήχο ξεχωριστά και επίσης να προσαρμοστούν οι παράμετροι των επιπτώσεων εύκολα αν είναι ήδη ενεργοί. Επί του παρόντος οι επιπτώσεις αυτές είναι διαθέσιμες μόνο όταν χρησιμοποιείται το πρόγραμμα οδήγησης DirectSound8.

3.3.3 Πλατφόρμες

Η IrrKlang είναι υλοποιημένη με abstract μορφή λειτουργιών, ώστε ανεξάρτητα από την πλατφόρμα που λειτουργεί να μην χρειάζεται αλλαγές ο κώδικας των προγραμμάτων. Οι ακόλουθες πλατφόρμες υποστηρίζονται προς το παρόν μαζί με το αντίστοιχα backends εξόδου ήχου:

- Windows 98, ME, NT 4, 2000, XP, Vista, Windows 7
  - DirectSound 3
  - DirectSound 8
  - WinMM
- Linux / * nix
  - ALSA
- Mac OS X
  - CoreAudio

Η μηχανή λειτουργεί με βιβλιοθήκη για όλες τις υποστηριζόμενες πλατφόρμες με τον ίδιο τρόπο, έτσι ο προγραμματιστής έχει να γράψει τον κώδικα της εφαρμογής μόνο μια φορά και θα τρέχει σε όλες τις υποστηριζόμενες πλατφόρμες.

3.3.4 Υποστηριζόμενοι τύποι αρχείων

Η irrKlang υποστηρίζει τα ακόλουθα πρότυπα ήχου:

- Ogg Vorbis (*.ogg)
- RIFF WAVE (*.wav)
- MPEG-1 Audio Layer 3 (*.mp3)
- Free Lossless Audio Codec (*.flac)
- Amiga Modules (*.mod)
- Impulse Tracker (*.it)
- Scream Tracker 3 (*.s3d)
- Fast Tracker 2 (*.xm)

3.4 IrrXML

Η XML είναι μια εξαιρετικά ελαστική δομή κωδικοποίησης δεδομένων, είναι κατάλληλη για την αποθήκευση και ανταλλαγή κάθε είδους δεδομένων που μπορεί να κωδικοποιηθεί σε μορφή κειμένου. Προσφέρει διαχειρίσιμα δεδομένα (portable data) και επιτρέπει την επαναχρησιμοποίηση των δεδομένων.

Η ενότητα (module) IrrXML, θα χρησιμοποιηθεί για την προσπέλαση σε αρχεία δεδομένων τύπου XML, όπου θα έχουμε αποθηκευμένες παραμέτρους για τα επίπεδα του παιχνιδιού μας. Οι παράμετροι έχουν δυναμική μορφή, καθώς κάποιοι είναι προαιρετικοί ή και μπορούν να υπάρχουν παραπάνω από μια φορές.

3.4.1 Εισαγωγή

Ο IrrXML είναι ένας ανοικτού κώδικα, XML parser σε γλώσσα C ++. Τα πλεονεκτήματα του IrrXML είναι η ταχύτητά και η απλότητα του. Ταιριάζει ιδανικά για πραγματικού χρόνου εφαρμογές που πρέπει να διαβάσουν δεδομένα XML, όπως παιχνίδια. Ο IrrXML γράφτηκε αρχικά ως μέρος της μηχανής απόδοσης γραφικών Irrlicht αλλά αργότερα έγινε αρκετά ώριμος και αποτελεί πια ξεχωριστό έργο.

3.4.2 Χαρακτηριστικά

Ο IrrXML παρέχει προς τα εμπρός μόνο πρόσβαση, δηλαδή μόνο για ανάγνωση σε ένα ρεύμα δεδομένα XML. Χαρακτηριστικά του είναι η υψηλή ταχύτητα και η πολύ χαμηλή χρήση μνήμης, καθώς αναπτύχθηκε με την πρόθεση να χρησιμοποιηθεί σε παιχνίδια 3D, όπως και έχει ήδη γίνει. Είναι με πιστοποιημένο OSI license (zlib). Αποτελείται από μόνο 60 KB κώδικα και μπορεί εύκολα να προστεθεί σε άλλα έργα, παρέχει ανεξαρτησία πλατφόρμας και λειτουργεί με πολλούς μεταγλωττιστές. Είναι σε θέση να αναλύσει ASCII, UTF-8, UTF-16 και UTF-32 αρχεία κειμένου, τόσο σε μικρή όσο και σε μεγάλη endian μορφή. Ανεξάρτητα από τη μορφή του αρχείου εισόδου, ο parser μπορεί να επιστρέψει όλες τις λέξεις σε ASCII, UTF-8, UTF-16 και UTF-32 format. Με την αφαιρετική (abstract) πρόσβαση αρχείων έχει το πλεονέκτημα, όταν η ανάγνωση από αρχείο δεν είναι δυνατή να μπορεί να πραγματοποιήσει ανάγνωση από οποιοδήποτε είδος δεδομένων (μνήμη,

δίκτυο, και άλλα). Για παράδειγμα, όταν χρησιμοποιείται με την Irrlicht Engine, μπορεί να διαβάζει απευθείας από συμπιεσμένα αρχεία. Συνοδεύεται από εμπεριστατωμένη τεκμηρίωση. Τέλος, δεν έχει εξωτερικές εξαρτήσεις, δεν χρειάζεται STL (Standard Template Library).

3.5 IrrAI

Η ενότητα (module) IrrAI, θα χρησιμοποιηθεί στον έλεγχο χαρακτήρων που χειρίζεται το σύστημα (N.P.C. - Non Playing Characters), όπως εύρεση συντομότερου μονοπατιού από το σημείο που βρίσκεται ο χαρακτήρας προς ένα άλλο και ακολούθηση αυτού του μονοπατιού. Επίσης θα χρησιμοποιήσουμε μεθόδους της, για την δημιουργία αλληλεπιδράσεων μεταξύ χαρακτήρων, παραδείγματος χάριν, σταμάτημα της κίνησης ενός χαρακτήρα όταν εντοπίσει έναν άλλο χαρακτήρα, εχθρικό προς αυτόν.

3.5.1 Εισαγωγή

Η IrrAI είναι μια ξεχωριστή ενότητα (module) φτιαγμένη για χρήση παράλληλα με της Irrlicht η οποία παρέχει βοηθητικές λειτουργίες και δυνατότητες τεχνητής νοημοσύνης και λογικής παιχνιδιού.

3.5.2 Δυνατότητες

Το **οπτικό πεδίο** ενός NPC (Non playing character) παρουσιάζεται ως ένα κόκκινο τρίγωνο προεξέχει από το NPC το οποίο αντιπροσωπεύει το οπτικό πεδίο. Οι NPC ελέγχουν τον τομέα του οπτικού πεδίου για τους εχθρούς ή αντικείμενα που παρουσιάζουν ενδιαφέρον, ώστε να μπορούν να αντιδράσουν σε αυτές. Στην εργασία μας, χρησιμοποιήσαμε την δυνατότητα οπτικού πεδίου, για την αποφυγή επικείμενων συγκρούσεων μεταξύ περαστικών και αυτοκινήτων. Επίσης τα αυτοκίνητα αποφεύγουν τη σύγκρουση με τους χαρακτήρες του παιχνιδιού μας. Περιγράφουμε την υλοποίηση, στη παράγραφο 6.7.1.

Η **εύρεση μονοπατιών** στην IrrAI βασίζεται σε ένα σύστημα σημείων αναφοράς. Τα σημεία αναφοράς ορίζονται και συνδέονται, χρησιμοποιώντας το IrrAI Editor που διανέμεται παράλληλα με την ενότητα. Η αναζήτηση συντομότερου μονοπατιού πάνω στα σημεία αναφοράς (waypoints), δύναται να πραγματοποιηθεί είτε με έναν αλγόριθμο αναζήτησης σε πλάτος (breadth first search algorithm) είτε με ευριστικό αλγόριθμο αναζήτησης A* (A star), οι οποίοι βρίσκουν το συντομότερο δρομολόγιο μεταξύ των σημείων εκκίνησης και προορισμού. Αναφερόμαστε εκτενέστερα στην αναζήτηση συντομότερου μονοπατιού, στην παράγραφο 6.7.3.

Οι **εχθροί** NPCs προδιαγράφονται με μια ομάδα NPCEnemyGroup η οποία είναι απλώς ένας κατάλογος των NPCs που θεωρούνται εχθροί. Οι σύμμαχοι αντίστοιχα προδιαγράφονται σε μια ομάδα τύπου συμμάχων. Όταν ένας εχθρός ή ένας σύμμαχος βρεθεί στην οπτική εμβέλεια ενός χαρακτήρα, τότε συμβαίνει κάποιο γεγονός (event), το οποίο μπορεί να χρησιμοποιηθεί, για να τροφοδοτήσει κάποια δράση, όπως την αλλαγή κατάστασης του NPC, την αδρανοποίηση του, και άλλα.

Κεφάλαιο 4

Σχεδιασμός

4.1 Καταγραφή της βασικής ιδέας

Ο Fonz, ο κεντρικός χαρακτήρας, είναι ένας επί πληρωμή πράκτορας με πολλές επιτυχημένες αποστολές στη ζωή του. Αυτή τη φορά έχει ως αποστολή να εντοπίσει και να απελευθερώσει έναν όμηρο από τα δίχτυα μιας συμμορίας στη Νότια Ιταλία. Στην αποστολή του αυτή δέχεται εντολές από τον αρχηγό της μεγαλύτερης συμμορίας της περιοχής ενώ μέλη μιας άλλης μεγάλης μαφιόζικης οργάνωσης τον καταδιώκουν.

4.2 Σενάριο

Ο κεντρικός ήρωας της κύριας αποστολής που έχουμε υλοποιήσει είναι ο Fonz. Ο Fonz είναι ένας τύπος γύρω στα τριάντα, μεγαλόσωμος, αρκετά ψηλός και γεροδεμένος. Από παιδί ακόμα ήταν ένας μοναχικός και κλειστός χαρακτήρας που πάντα είχε το στοιχείο του τυχοδιωκτισμού και της εξερεύνησης. Είναι ένας ασυμβίβαστος και κλειστός άνθρωπος που διατηρεί τις συνήθειες του σταθερές εδώ και χρόνια. Χόμπι του είναι κυρίως η γυμναστική και η σκοποβολή, ενώ του αρέσουν πολύ τα ταξίδια και οι ασυνήθιστοι προορισμοί.

Το δύσκολο παρελθόν του, εγκαταλελειμμένος από τους γονείς του, μεγάλωσε σε ιδρύματα και οι συναναστροφές του τον οδήγησαν σε μια ζωή χωρίς φραγμούς όπου μόνη σταθερή αξία ήταν τα χρήματα. Γι' αυτά μπορούσε πλέον να κάνει τα πάντα χωρίς να έχει ενδοιασμούς για το όποιο κόστος. Αυτή τη στιγμή είναι ένας επί πληρωμή πράκτορας ο οποίος δουλεύει για λογαριασμό της μαφίας και άλλων οργανώσεων που δρουν σε αρκετές χώρες στην Ευρώπη.

Σε αυτή την ιστορία, ο Fonz, βρίσκεται στη Νότια Ιταλία και ο στόχος του είναι να εντοπίσει και να ελευθερώσει κάποιον που κρατείται από μια μαφιόζικη οργάνωση. Εντολές παίρνει από τον αρχηγό της αντίπαλης συμμορίας στην περιοχή και μέσω πληροφοριοδοτών που συναντά στην πόλη, οι οποίοι τον στέλνουν όλο και πιο κοντά στο στόχο. Σε αυτή του την αποστολή εμποδίζεται από τα μέλη της εχθρικής οργάνωσης η οποία και κρατά τον όμηρο και προσπαθούν να τον βγάλουν από τη μέση.

Ξεκινώντας από κεντρικό σημείο της πόλης, ο Fonz κατευθύνεται στο μέρος όπου έχουν καθορίσει ως σημείο συνάντησης με τον αρχηγό της συμμορίας που τον κάλεσε, τον Mr. Coat. Ο Fonz βλέπει μπροστά του μια υπερπολυτελή βίλα ενώ πίσω από τους φοίνικες ξεπροβάλλει μια σκοτεινή φιγούρα με ένα μακρύ, δερμάτινο παλτό. Τον πλησιάζει και εκείνος του δίνει οδηγίες για τη νέα του αποστολή...

4.3 Τρόπος παιχνιδιού

Σε αυτή την εργασία υλοποιείται κατηγορία παιχνιδιού για ένα χρήστη. Μελλοντικά υπάρχει δυνατότητα υλοποίησης κατηγορίας παιχνιδιού πολλαπλών χρηστών. Παρακάτω παρουσιάζονται τα επιμέρους συστατικά του τρόπου παιχνιδιού για τη κατηγορία παιχνιδιού ενός χρήστη.

4.3.1 Κεντρικός χαρακτήρας

Ο κεντρικός χαρακτήρας είναι ένα τρισδιάστατο πλέγμα (3D mesh[7]), το οποίο με την είσοδο του χρήστη περιφέρεται στον τρισδιάστατο κόσμο και εκτελεί διάφορες ενέργειες όπως άλμα, περιστροφή γύρω από τον εαυτό του και πυροβολισμό. Επίσης μπορεί να είναι διαφορετικός εικονικός χαρακτήρας, αν αυτό επιβάλει η πλοκή της αποστολής που επιλέγει στο αρχικό μενού ο χρήστης.

4.3.2 Στόχος του παιχνιδιού

Ο στόχος του παιχνιδιού , είναι η ολοκλήρωση κάποιας αποστολής. Ο ήρωας έχει συγκεκριμένη αποστολή σε κάθε επίπεδο, η οποία διαδραματίζεται με μικρά βήματα, όπως η πλοκή μιας ταινίας. Όταν ο χαρακτήρας ολοκληρώσει τα βήματα της αποστολής, ολοκληρώνεται το παιχνίδι. Στην πορεία προκύπτουν γρίφοι ή ερωτήματα και άλλες φορές έρχεται αντιμέτωπος με εχθρικούς χαρακτήρες. Σε αυτή τη περίπτωση, ο χαρακτήρας πρέπει να εξουδετερώσει τους εχθρούς που βρίσκονται στον τρισδιάστατο κόσμο, δηλαδή να αφαιρέσει όλους τους πόντους ζωής τους με εύστοχες βολές του, πριν αυτός χάσει όλους τους πόντους ζωής του, τότε είναι νικητής.

4.3.3 Κάμερα

Η κύρια κάμερα του παιχνιδιού είναι τρίτου προσώπου, δηλαδή ακολουθεί τον χαρακτήρα από πίσω σε σταθερή απόσταση και ύψος. Επίσης υπάρχει διαθέσιμη επιλογή για κάμερα πρώτου προσώπου, δηλαδή η κάμερα έχει την οπτική γωνία του χαρακτήρα Η εναλλαγή κάμερας γίνεται στη διάρκεια του παιχνιδιού με πάτημα του κουμπιού αλλαγής κάμερας από το χρήστη.

4.3.4 Χειρισμός

Ο χειρισμός του ήρωα γίνεται με το πληκτρολόγιο και το ποντίκι. Με τα βέλη του πληκτρολογίου ο χαρακτήρας κινείται μπροστά, πίσω, αριστερά και δεξιά, με το κουμπί Ε πηδάει, με την κίνηση του ποντικιού περιστρέφεται γύρω από τον εαυτό του και με το αριστερό κουμπί του ποντικιού πυροβολεί στο κέντρο της οπτικής γωνίας του χαρακτήρα. Με το κουμπί space δρομολογείται αλληλεπίδραση με τον κόσμο ή τους άλλους χαρακτήρες του κόσμου ανάλογα τη θέση του χαρακτήρα. Με το πλήκτρο 'V' η κάμερα εναλλάσσεται από πρώτου προσώπου σε τρίτου και αντίστροφα.

4.3.5 Τρισδιάστατος κόσμος

Ο τρισδιάστατος εικονικός κόσμος, ή αλλιώς επίπεδο, περιλαμβάνει ένα πολυγωνικό χάρτη, δηλαδή ένα τρισδιάστατο πλέγμα με μορφή εσωτερικού ή εξωτερικού χώρου, πάνω στον οποίο περπατάνε οι χαρακτήρες. Ο τρισδιάστατος κόσμος συμπληρώνεται από διάφορα τρισδιάστατα αντικείμενα, και το "κουτί ουρανού" (sky-box) γύρω από το χάρτη, το οποίο είναι ένα κουτί με πλευρές του δισδιάστατες εικόνες που δίνουν την ψευδαίσθηση ουρανού. Για μεγαλύτερη ποικιλομορφία προστέθηκαν στατικά και κινούμενα τρισδιάστατα αντικείμενα όπως δέντρα και αυτοκίνητα. Επίσης είναι εμπλουτισμένος με ειδικά εφέ που κάνουν το τοπίο πιο ρεαλιστικό, όπως ομίχλη, υδάτινες επιφάνειες, διαφανή αντικείμενα και άλλα.

4.3.6 Χαρακτήρες

Υπάρχουν δύο κατηγορίες χαρακτήρων, αυτοί που προσφέρουν αλληλεπίδραση με τον χαρακτήρα, όπως συνομιλία και μάχη, και αυτοί που δεν προσφέρουν και απλά προσθέτουν ρεαλισμό στον εικονικό κόσμο. Ο κάθε χαρακτήρας είναι ένα ανθρωπόμορφο τρισδιάστατο πλέγμα με κίνηση. Στον τρισδιάστατο κόσμο που υλοποιούμε υπάρχουν περίπου δέκα εχθροί και αρκετοί άλλοι χαρακτήρες που δεν αλληλεπιδρούν με τον χρήστη.

4.3.7 Όπλο

Όλοι οι χαρακτήρες που μπορούν να επιτεθούν κρατάνε όπλο ίδιου τύπου και επιτίθενται με ακριβώς τον ίδιο τρόπο. Η βολή του όπλου καταγράφει ευθύγραμμη κίνηση και συμβολίζεται από μια πύρινη μπάλα. Όταν η βολή πετύχει ένα χαρακτήρα αυτός χάνει ζωή.

4.3.8 Ζωή χαρακτήρα και εχθρών:

Ο χαρακτήρας ξεκινά το παιχνίδι με 100 πόντους ζωής. Οι εχθροί ξεκινούν με 20 έως 100 πόντους ζωής ο καθένας. Μετά από χτύπημα, από πυροβολισμό από οποιαδήποτε χαρακτήρα, οι πόντοι ζωής του χαρακτήρα μειώνονται κατά 5 και όταν μηδενιστούν πεθαίνει.

4.4 Χαρακτηριστικά παιχνιδιού

Παρακάτω αναλύουμε τα χαρακτηριστικά παιχνιδιού (features), χωρισμένα σε κατηγορίες.

4.4.1 Επιλογές πριν την έναρξη του παιχνιδιού

Το μενού επιλογών πριν την έναρξη του παιχνιδιού παρέχει τις εξής επιλογές:

- Έναρξη κατηγορίας παιχνιδιού ενός χρήστη.
- Καταχώρηση ονόματος χρήστη για καταγραφή ρεκόρ.
- Επιλογή επιπέδου. Το κάθε επίπεδο έχει δικό του σενάριο, πλοκή, κόσμο, και χαρακτήρες.
- Επιλογή λεπτομερειών γραφικών, ανάλυση οθόνης και προγράμματος οδήγησης γραφικών μεταξύ των Direct3D, Opengl και Software renderer.
- Εμφάνιση καταγεγραμμένων αποτελεσμάτων.
- Εμφάνιση βοήθειας και πληροφοριών της εφαρμογής.
- Έξοδος από το παιχνίδι

4.4.2 Τρισδιάστατα πλέγματα εικονικών χαρακτήρων

Ο κάθε εικονικός χαρακτήρας, είναι ένα τρισδιάστατο πλέγμα με κίνηση (animation). Η μοναδική διαφορά κάθε χαρακτήρα είναι η εμφάνιση του, δηλαδή τα πολύγωνα και οι υφές που το αποτελούν. Η εμφάνιση δεν επηρεάζει τον τρόπο παιχνιδιού.

4.4.3 Οπτικά εφέ

Τα γραφικά υλοποιούνται κάνοντας χρήση της μηχανής απεικόνισης γραφικών πραγματικού χρόνου Irrlicht, που επιλέχθηκε βάσει των κριτηρίων που έχουμε αναλύσει. Σημαντικό κομμάτι του ρεαλισμού του παιχνιδιού, είναι η υλοποίηση ρεαλιστικών ειδικών εφέ, αξιοποιώντας πλήρως τις δυνατότητες που προσφέρουν οι σύγχρονες τεχνολογίες γραφικών.

Τα εφέ που υλοποιούνται στο παιχνίδι μας είναι: ρεαλιστικές υδάτινες επιφάνειες, δυναμικός φωτισμός, δυναμικές σκιές, διαφανή αντικείμενα, εφέ για χιόνι, εφέ για φωτιά και καπνό, απεικόνιση ουρανού και ομίχλη.

4.4.4 Λεπτομέρειες στην οθόνη κατά τη διάρκεια παιχνιδιού

Στην οθόνη κατά τη διάρκεια παιχνιδιού εμφανίζονται λεπτομέρειες που βοηθούν την ενημέρωση του χρήστη για τη πορεία του στο παιχνίδι. Αυτές είναι, ενημέρωση ζωής, που εμφανίζει τους εναπομείναντες πόντους ζωής του χαρακτήρα, ραντάρ που εντοπίζει τη παρουσία εχθρών στο χάρτη και στο κέντρο του ραντάρ βρίσκεται ο χαρακτήρας του χρήστη. Ακόμα εμπεριέχεται ειδική σελίδα που παρέχει πληροφορίες για την κατάσταση της αποστολής, το επόμενο βήμα και το πλήθος των εχθρών που απομένουν για την ολοκλήρωση του στόχου της νίκης. Τα ονόματα των εχθρικών χαρακτήρων είναι γραμμένα σε πινακίδες πάνω από το κεφάλι τους και τέλος, ειδικά εικονίδια ενημερώνουν τον χρήστη ότι ο χαρακτήρας που ελέγχει κατέχει κάποια ειδική δύναμη.

4.4.5 Μουσική και ήχοι:

Η μουσική και οι ήχοι υλοποιούνται κάνοντας χρήση βοηθητικών βιβλιοθηκών για αναπαραγωγή μουσικής και ήχων. Οι βιβλιοθήκες που χρησιμοποιούνται επιλέχθηκαν βάσει της ανάλυσης απαιτήσεων.

Υπάρχει παρασκηνιακή μουσική κατά τη διάρκεια του παιχνιδιού. Για κάθε τρισδιάστατο κόσμο υπάρχει διαφορετικό μουσικό κομμάτι. Επίσης ακούγονται ηχητικά εφέ για τους πυροβολισμούς, το άλμα του κεντρικού χαρακτήρα, ήχος πόνου όταν κάποιος δεχτεί βολή, και διάφορα ηχητικά εφέ όπως κορναρίσματα και άλλα.

4.4.6 Νοημοσύνη χαρακτήρων

Όπως έχουμε προαναφέρει, υπάρχουν δύο κατηγορίες χαρακτήρων, αυτοί που προσφέρουν αλληλεπίδραση με τον χαρακτήρα, όπως συνομιλία και μάχη, και αυτοί που δεν προσφέρουν και απλά προσθέτουν ρεαλισμό στο εικονικό περιβάλλον.

Οι χαρακτήρες της πρώτης κατηγορίας, έχουν είτε εχθρική στάση απέναντι στον χαρακτήρα του χρήστη, είτε ουδέτερη. Όταν ένας χαρακτήρας είναι εχθρικός και εντοπίσει τον χαρακτήρα του χρήστη, δηλαδή όταν ο δεύτερος βρεθεί στην οπτική εμβέλεια του εχθρού, κινείται προς τον στόχο μέχρις ότου να προσεγγίσει απόσταση βολής. Όταν αυτό πραγματοποιηθεί, ξεκινάει διαδικασία στόχευσης και όταν ο στόχος επιτευχθεί, επιτίθεται στον χαρακτήρα του χρήστη, με πυροβολισμό. Ένας χαρακτήρας με ουδέτερη στάση, απλώς βρίσκεται αδρανής σε κάποιο σημείο του επιπέδου και προσφέρει συνομιλία με τον χαρακτήρα του χρήστη.

Η δεύτερη κατηγορία απλά περιφέρεται στον κόσμο. Ένας τέτοιος χαρακτήρας κινείται πάνω σε μια συγκεκριμένη ομάδα σημείων (way-point group). Η κίνηση από σημείο σε σημείο, γίνεται μόνο όταν τα σημεία αυτά είναι συνδεδεμένα, και μόνο προς την κατεύθυνση της σύνδεσης. Επίσης αυτή η κατηγορία χαρακτήρων,

είναι χωρισμένη σε ομάδες (περαστικοί, αυτοκίνητα), οι οποίες ομάδες είναι αντίπαλες και αποφεύγουν τις συγκρούσεις. Περισσότερα στο Κεφάλαιο 6, όπου γίνεται η υλοποίηση.

4.5 Ανάλυση απαιτήσεων

Οι προδιαγραφή απαιτήσεων λογισμικού (software requirements specification) παρέχει μια λίστα με την περιγραφή των λειτουργικών και μη προδιαγραφών, για τα στοιχεία του λογισμικού που θα αναπτύξουμε. Οι απαιτήσεις αυτές προέρχονται από τον σχεδιασμό που κάναμε στο προηγούμενο κεφάλαιο. Η ανάλυση απαιτήσεων θα πραγματοποιηθεί με τις παρακάτω όψεις:

- Ο κατάλογος των λειτουργικών απαιτήσεων παρουσιάζεται αριθμητικά
- Ομαδοποίηση των απαιτήσεων
- Ανάλυση των περιπτώσεων χρήσης (use cases)
- Διαγράμματα περιπτώσεων χρήσης

Αυτή η ενότητα, έχει σκοπό να καθορίσει τις απαιτήσεις για την διαδικασία ανάπτυξης του παιχνιδιού μας.

4.5.1 Λειτουργικές απαιτήσεις

Θα καταγράφουμε τις λειτουργικές απαιτήσεις σε μια αριθμημένη λίστα. Αργότερα θα τις ομαδοποιήσουμε για βελτιωμένη χρηστικότητα στην ανάλυση απαιτήσεων που κάνουμε.

1. Η εφαρμογή ελέγχει την είσοδο χρήστη από πληκτρολόγιο και ποντίκι.
2. Υποστηρίζει πάνελ, κουμπιά, μενού και γραφικά.
3. Το λογισμικό πρέπει να έχει ένα κύριο μενού με τις εξής επιλογές:
    - επιλογή εικονικού κόσμου (επιπέδου)
    - επιλογή λεπτομερειών γραφικών
    - εμφάνιση των καλύτερων χρόνων
    - κουμπί για έναρξη του παιχνιδιού
    - κουμπί για έξοδο από την εφαρμογή
4. Υποστηρίζονται fullscreen mode και windowed mode.
5. Το παιχνίδι διαδραματίζεται σε τρισδιάστατη σκηνή, στο επίπεδο που έχει επιλέξει ο χρήστης στο αρχικό μενού.
6. Η εφαρμογή έχει δικό της φορτωτή επιπέδων, ο οποίος:
    - φορτώνει τον εικονικό κόσμο από αρχείο
    - φορτώνει χαρακτήρες από αρχείο
    - φορτώνει σενάριο, πλοκή και παραμέτρους από αρχείο
7. Το παιχνίδι φορτώνεται με τις ρυθμίσεις που επέλεξε ο χρήστης στο κύριο μενού.
8. Το λογισμικό υποστηρίζει διάφορους τύπους τρισδιάστατων χαρακτήρων.
9. Το λογισμικό υποστηρίζει διάφορους τύπους υφών.
10. Το λογισμικό υποστηρίζει διάφορα οπτικά εφέ:
    - δυναμικές σκιές
    - σκιαστές
    - υδάτινες επιφάνειες
    - φωτιά

- εκρήξεις
- δυναμικό φωτισμό
- αλλαγή υφών

11. Το λογισμικό υποστηρίζει διάφορους τύπους αρχείων ήχου.
12. Υπάρχουν δύο κάμερες πλοήγησης, πρώτου και τρίτου προσώπου.
13. Ο χρήστης ελέγχει την πλοήγηση και τις ενέργειές του χαρακτήρα του.
14. Οι κάμερες ακολουθούν το χαρακτήρα.
15. Η πλοήγηση των χαρακτήρων, γίνεται εντός προκαθορισμένων ορίων, που ορίζονται από το κάθε επίπεδο.
16. Τα υποκείμενα του εικονικού κόσμου, υπόκεινται σε βαρύτητα.
17. Οι υπόλοιποι χαρακτήρες, ελέγχονται από το σύστημα με χρήση τεχνητής νοημοσύνης.
18. Οι εχθρικοί χαρακτήρες έχουν οπτική εμβέλεια 100 μέτρων, για να εντοπίζουν τον χαρακτήρα του χρήστη.
19. Οι εχθρικοί χαρακτήρες επιτίθενται στο χαρακτήρα του χρήστη όταν τον εντοπίσουν.
20. Οι φιλικοί χαρακτήρες βρίσκονται σταθεροί σε ένα σημείο του κόσμου και παρέχουν αλληλεπίδραση με τον χαρακτήρα του χρήστη, πχ. συνομιλία.
21. Υπάρχουν περαστικοί και αυτοκίνητα, που περιφέρονται στον κόσμο.
22. Οι περαστικοί και τα αυτοκίνητα ελέγχουν και αποφεύγουν τις συγκρούσεις.
23. Τα αυτοκίνητα ελέγχουν και για τον χαρακτήρα του χρήστη και σταματάνε όταν πρόκειται να συμβεί σύγκρουση.
24. Ο χαρακτήρας του χρήστη και οι εχθρικοί χαρακτήρες έχουν όπλο και μπορούν να πυροβολήσουν.
25. Κάθε χαρακτήρας έχει συγκεκριμένους πόντους ζωής.
26. Όταν ένας χαρακτήρας δεχθεί βολή, δέχεται μείωση πόντων ζωής κατά 5.
27. Όταν τελειώσουν οι πόντοι ζωής ενός χαρακτήρα, αυτός πεθαίνει.
28. Οι συγκρούσεις αντικειμένων ελέγχονται από το σύστημα.
29. Η πλοκή του σεναρίου εκτελείται μόνο με προκαθορισμένη σειρά.
30. Όταν εκτελεστούν όλα τα βήματα της αποστολής, η αποστολή είναι επιτυχής και ο χρήστης μεταβαίνει στο κεντρικό μενού στη σελίδα αποτελεσμάτων.
31. Η σελίδα αποτελεσμάτων έχει τους καλύτερους χρόνους, τα ονόματα των χρηστών και τα ονόματα των αποστολών.
32. Κατά τη διάρκεια του παιχνιδιού, ο χρήστης μπορεί να ενημερώνεται για τη κατάσταση της αποστολής, με πάτημα ενός πλήκτρου.
33. Οι πόντοι ζωής του παίχτη αναγράφονται, στο πάνω αριστερό κομμάτι της οθόνης.
34. Οι διαθέσιμες βολές του παίχτη αναγράφονται, στο πάνω αριστερό κομμάτι της οθόνης.
35. Κατά τη διάρκεια του παιχνιδιού, ο χρήστης μπορεί να ενεργοποιήσει χάρτη στο κάτω αριστερό κομμάτι της οθόνης, με πάτημα ενός πλήκτρου.
36. Οι εχθροί έχουν μια πινακίδα πάνω από το κεφάλι τους που αναγράφει το όνομά τους και τους πόντους ζωής τους.
37. Στο κεντρικό μενού ακούγεται ένα μουσικό κομμάτι
38. Ο ήχος μέσα στο παιχνίδι είναι τρισδιάστατος, δηλαδή δίνει την αίσθηση του χώρου.
39. Υπάρχουν διάφορα ηχητικά εφέ, για πυροβολισμούς, φωνητικά και άλλα.
40. Κατά την φόρτωση του παιχνιδιού, εμφανίζεται οθόνη φόρτωσης
41. Οποιαδήποτε στιγμή ο χρήστης μπορεί να τερματίσει την εφαρμογή με το κουμπί 'esc'.

4.5.2 Ομαδοποίηση των απαιτήσεων

Θα ομαδοποιήσουμε τις λειτουργικές απαιτήσεις που παραθέσαμε προηγουμένως, σε κατηγορίες. Δημιουργούμε τις εξής βασικές κατηγορίες:

- Κατηγορία 1: Επιλογές παιχνιδιού, διεπαφή χρήστη εκτός παιχνιδιού και εμφάνιση αποτελεσμάτων.
- Κατηγορία 2: Γραφικά
- Κατηγορία 3: Ηχητικά εφέ και μουσική
- Κατηγορία 4: Διεπαφή χρήστη εντός παιχνιδιού.
- Κατηγορία 5: Πλοήγηση και έλεγχος χαρακτήρα.
- Κατηγορία 6: Λογική παιχνιδιού
- Κατηγορία 7: Διαχείριση δεδομένων και κατάστασης εφαρμογής

Ακολούθως κατηγοριοποιούμε τις λειτουργικές απαιτήσεις στις κατηγορίες που δημιουργήσαμε. Μια απαίτηση μπορεί να ανήκει σε παραπάνω από μια κατηγορίες.

Κατηγορία 1: Επιλογές παιχνιδιού, διεπαφή χρήστη εκτός παιχνιδιού και εμφάνιση αποτελεσμάτων:
1.Η εφαρμογή ελέγχει την είσοδο χρήστη από πληκτρολόγιο και ποντίκι.
2.Υποστηρίζει πάνελ, κουμπιά, μενού και γραφικά.
3.Το λογισμικό πρέπει να έχει ένα κύριο μενού με επιλογές
4.Υποστηρίζονται fullscreen mode και windowed mode.
7.Το παιχνίδι φορτώνεται με τις ρυθμίσεις που επέλεξε ο χρήστης στο κύριο μενού.
30. Όταν εκτελεστούν όλα τα βήματα της αποστολής, η αποστολή είναι επιτυχής και ο χρήστης μεταβαίνει στο κεντρικό μενού στη σελίδα αποτελεσμάτων.
31. Η σελίδα αποτελεσμάτων έχει τους καλύτερους χρόνους, τα ονόματα των χρηστών και τα ονόματα των αποστολών.

Κατηγορία 2: Γραφικά:
2.Υποστηρίζει πάνελ, κουμπιά, μενού και γραφικά.
4.Υποστηρίζονται fullscreen mode και windowed mode.
5.Το παιχνίδι διαδραματίζεται σε τρισδιάστατη σκηνή, στο επίπεδο που έχει επιλέξει ο χρήστης στο αρχικό μενού.
8.Το λογισμικό υποστηρίζει διάφορους τύπους τρισδιάστατων χαρακτήρων.
9.Το λογισμικό υποστηρίζει διάφορους τύπους υφών.
10.Το λογισμικό υποστηρίζει διάφορα οπτικά εφέ.

Κατηγορία 3: Ηχητικά εφέ και μουσική:
11.Το λογισμικό υποστηρίζει διάφορους τύπους αρχείων ήχου.
37.Στο κεντρικό μενού ακούγεται ένα μουσικό κομμάτι
38.Ο ήχος μέσα στο παιχνίδι είναι τρισδιάστατος, δηλαδή δίνει την αίσθηση του χώρου.
39.Υπάρχουν διάφορα ηχητικά εφέ, για πυροβολισμούς, φωνητικά και άλλα.

Κατηγορία 4: Διεπαφή χρήστη εντός παιχνιδιού:
32.Κατά τη διάρκεια του παιχνιδιού, ο χρήστης μπορεί να ενημερώνεται για τη κατάσταση της αποστολής, με πάτημα ενός πλήκτρου.
33.Οι πόντοι ζωής του παίχτη αναγράφονται, στο πάνω αριστερό κομμάτι της οθόνης.

34.Οι διαθέσιμες βολές του παίχτη αναγράφονται, στο πάνω αριστερό κομμάτι της οθόνης.
35.Κατά τη διάρκεια του παιχνιδιού, ο χρήστης μπορεί να ενεργοποιήσει χάρτη στο κάτω αριστερό κομμάτι της οθόνης, με πάτημα ενός πλήκτρου.
36.Οι εχθροί έχουν μια πινακίδα πάνω από το κεφάλι τους που αναγράφει το όνομα τους και τους πόντους ζωής τους.

Κατηγορία 5: Πλοήγηση και έλεγχος χαρακτήρα:
12.Υπάρχουν δύο κάμερες πλοήγησης, πρώτου και τρίτου προσώπου.
13.Ο χρήστης ελέγχει τη πλοήγηση και τις ενέργειες του χαρακτήρα του.
14.Οι κάμερες ακολουθούν το χαρακτήρα.
15.Η πλοήγηση των χαρακτήρων, γίνεται εντός προκαθορισμένων ορίων, που ορίζονται από το κάθε επίπεδο.
16.Τα υποκείμενα του εικονικού κόσμου, υπόκεινται σε βαρύτητα.

Κατηγορία 6: Λογική παιχνιδιού και Τεχνητή νοημοσύνη:
17.Οι υπόλοιποι χαρακτήρες, ελέγχονται από το σύστημα με χρήση τεχνητής νοημοσύνης.
18.Οι εχθρικοί χαρακτήρες έχουν οπτική εμβέλεια 100 μέτρων, για να εντοπίζουν τον χαρακτήρα του χρήστη.
19.Οι εχθρικοί χαρακτήρες επιτίθενται στον χαρακτήρα του χρήστη, όταν τον εντοπίσουν.
20.Οι φιλικοί χαρακτήρες βρίσκονται σταθεροί σε ένα σημείο του κόσμου και παρέχουν αλληλεπίδραση με τον χαρακτήρα του χρήστη, πχ. συνομιλία.
21.Υπάρχουν περαστικοί και αυτοκίνητα, που περιφέρονται στον κόσμο.
22.Οι περαστικοί και τα αυτοκίνητα ελέγχουν και αποφεύγουν τις συγκρούσεις.
23.Τα αυτοκίνητα ελέγχουν και για τον χαρακτήρα του χρήστη και σταματάνε όταν πρόκειται να συμβεί σύγκρουση.
24.Ο χαρακτήρας του χρήστη και οι εχθρικοί χαρακτήρες έχουν όπλο και μπορούν να πυροβολήσουν.
25.Κάθε χαρακτήρας έχει συγκεκριμένους πόντους ζωής.
26.Όταν ένας χαρακτήρας δεχθεί βολή, δέχεται μείωση πόντων ζωής κατά 5.
27.Όταν τελειώσουν οι πόντοι ζωής ενός χαρακτήρα, αυτός πεθαίνει.
28.Οι συγκρούσεις αντικειμένων ελέγχονται από το σύστημα.
29.Η πλοκή του σεναρίου εκτελείται μόνο με προκαθορισμένη σειρά.
30.Όταν εκτελεστούν όλα τα βήματα της αποστολής, η αποστολή είναι επιτυχής και ο χρήστης μεταβαίνει στο κεντρικό μενού στη σελίδα αποτελεσμάτων.

Κατηγορία 7: Διαχείριση δεδομένων και κατάστασης εφαρμογής:
6.Η εφαρμογή έχει δικό της φορτωτή επιπέδων
7.Το παιχνίδι φορτώνεται με τις ρυθμίσεις που επέλεξε ο χρήστης στο κύριο μενού.
30.Όταν εκτελεστούν όλα τα βήματα της αποστολής, η αποστολή είναι επιτυχής και ο χρήστης μεταβαίνει στο κεντρικό μενού στη σελίδα αποτελεσμάτων.
42.Κατά την φόρτωση του παιχνιδιού, εμφανίζεται οθόνη φόρτωσης.
43.Οποιαδήποτε στιγμή ο χρήστης μπορεί να τερματίσει την εφαρμογή με το 'esc'.

4.5.3 Περιπτώσεις χρήσης

Στη συνέχεια αναλύουμε τις περιπτώσεις χρήσης (use cases).Στόχος της ανάλυσης των περιπτώσεων χρήσης είναι:

- Να καθορίσει και να περιγράψει τις λειτουργικές απαιτήσεις του συστήματος
- Να δώσει μια σαφή και συνεπή περιγραφή για το τι θα πρέπει να κάνει το σύστημα
- Να παρέχει την κατάλληλη βάση για να γίνονται έλεγχοι για επαλήθευση του συστήματος.
- Να παρέχει την ικανότητα να ανιχνεύονται οι λειτουργικές απαιτήσεις μέσα στις κλάσεις και τις λειτουργίες του συστήματος

Αναλύουμε τις περιπτώσεις μελετώντας τρεις χρήσιμες σκοπιές: του χρήστη, της εφαρμογής δηλαδή του συστήματος, και της λογικής του παιχνιδιού (game logic). Οι περιπτώσεις χρήσης παρουσιάζονται σε μορφή πίνακα (table template):

Use Case 1	Εκκίνηση παιχνιδιού	
Requirements explored	**Κατηγορία 1:** Επιλογές παιχνιδιού, διεπαφή χρήστη εκτός παιχνιδιού και εμφάνιση αποτελεσμάτων. # 1,2,3,4,7,30,31	
Scope & Level	Χρήστη, εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Το παιχνίδι ξεκινά	
Failed End Condition	Δεν υπάρχουν διαθέσιμα επίπεδα. Δεν υποστηρίζεται το υλικό.	
Primary, Secondary Actors	Χρήστης, Σύστημα φόρτωσης επιπέδων, Σύστημα διαχείρισης επιλογών	
Trigger	Εκτέλεση εφαρμογής	
Description	**Step**	**Action**
	1	Ο χρήστης έχει μόλις εκτελέσει την εφαρμογή και βρίσκεται στο κεντρικό μενού.
	2	Επιλέγει το επίπεδο που επιθυμεί
	3	Πατάει το κουμπί έναρξης του παιχνιδιού
	4	Το παιχνίδι ξεκινά με τις προεπιλεγμένες ρυθμίσεις
Extensions	**Step**	**Branching Action**
	3α	3α1Το επίπεδο δεν υπάρχει
		3α2 Το παιχνίδι δεν εκκινεί
Sub-Variations	1α	1α1α Ο χρήστης έχει μόλις ολοκληρώσει μια αποστολή και μεταβαίνει στη σελίδα αποτελεσμάτων
		1α1β Μετά μεταβαίνει στην αρχική σελίδα του μενού.
		1β1α Ο χρήστης έχει μόλις αποτύχει σε μια αποστολή και μεταβαίνει στο κεντρικό μενού.
	2α	2α1 Ο χρήστης μεταβαίνει στη σελίδα επιλογών και τροποποιεί τις ρυθμίσεις
		2α2 Το παιχνίδι εκκινεί με τις τροποποιημένες ρυθμίσεις

Use Case 2	Πλοήγηση χαρακτήρα.	
Requirements explored	#5,8,12,13,14,15,16	
Scope & Level	Χρήστη, Λογική παιχνιδιού	
Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Ο χρήστης πλοηγεί τον χαρακτήρα του	
Failed End Condition	Υπάρχει κάποιο εμπόδιο.	
Primary, Secondary Actors	Χρήστης, Χαρακτήρας, Εικονικός κόσμος, Σύστημα πλοήγησης	
Trigger	Είσοδος χρήστη με πλήκτρο πλοήγησης.	
Description	**Step**	**Action**
	1	Ο χρήστης πατάει κάποιο πλήκτρο πλοήγησης
	2	Το σύστημα πλοήγησης ελέγχει την είσοδο του χρήστη και αναγνωρίζει προς τα ποια κατεύθυνση θέλει να κινηθεί.
	3	Το σύστημα ελέγχει την ταχύτητα του παιχνιδιού και δημιουργεί έναν πολλαπλασιαστή ταχύτητας.
	4	Το σύστημα πολλαπλασιάζει το διάνυσμα κατεύθυνσης με τον πολλαπλασιαστή.
	5	Το σύστημα μετακινεί τον χαρακτήρα κατά την υπολογιζόμενη απόσταση.
Extensions	**Step**	**Branching Action**
	5α	5α1Υπάρχει εμπόδιο που εμποδίζει την κίνηση
		5α2 Η κίνηση δεν πραγματοποιείται

Use Case 3	Πυροβολισμός	
Requirements explored	#8,9,10,39,13,24,25,26,27,28	
Scope & Level	Χρήστη, Λογική παιχνιδιού	
Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Ο χαρακτήρας του χρήστη πυροβολεί	
Failed End Condition	Δεν υπάρχουν σφαίρες διαθέσιμες	
Primary, Secondary Actors	Χρήστης, Χαρακτήρας, Εικονικός κόσμος, Εχθροί, Σύστημα βολών	
Trigger	Είσοδος χρήστη με πλήκτρο βολής.	
Description	**Step**	**Action**
	1	Ο χρήστης πατάει το πλήκτρο βολής.
	2	Το σύστημα βολών υπολογίζει την κατεύθυνση της βολής, αναλόγως με την περιστροφή του χαρακτήρα. Θέτει σημείο εκκίνησης της βολής το όπλο του χαρακτήρα.
	3	Το σύστημα δημιουργεί μια μπάλα φωτιάς που θα κινηθεί από το σημείο εκκίνησης μέχρι το τέλος ενός μοναδιαίου διανύσματος, με την προϋπολογισθείσα κατεύθυνση.
	4	Το σύστημα ελέγχει την ταχύτητα του παιχνιδιού και δημιουργεί έναν πολλαπλασιαστή ταχύτητας.

	5	Το σύστημα κινεί τη βολή, αναλόγως τον πολλαπλασιαστή ταχύτητας.
	6	Το σύστημα ανιχνεύει συγκρούσεις, διαμήκους της γραμμής.
	7	Αν δεν υπάρχει σύγκρουση, το σύστημα θέτει σημείο εκκίνησης το τέλος της γραμμής και επιστρέφει στο βήμα 3.
	8	Η βολή εξαφανίζεται, χωρίς να υπάρχουν συγκρούσεις μετά από 1000μέτρα.
Extensions	**Step**	**Branching Action**
	1α	1α1 Δεν υπάρχουν σφαίρες διαθέσιμες
		1α2 Δεν πυροβολεί
Sub-Variations	1α	1α1 Το σύστημα δίνει εντολή σε έναν εχθρό να πυροβολήσει
	7α	7α1 Υπάρχει σύγκρουση με άλλο χαρακτήρα.
		7α2 Οι πόντοι ζωής του χτυπημένου μειώνονται κατά 5.
		7α3 Η βολή εξαφανίζεται, και εμφανίζεται μια έκρηξη στο σημείο σύγκρουσης.
	7β	7β1 Υπάρχει σύγκρουση με κάποιο αντικείμενο.
		7β2 Η βολή εξαφανίζεται, και εμφανίζεται μια έκρηξη στο σημείο σύγκρουσης.

Use Case 4	Ολοκλήρωση αποστολής	
Requirements explored	#7,30,5,32,13,15,20,27,29,30	
Scope & Level	Χρήστη, Λογική παιχνιδιού	
Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Ο χαρακτήρας του χρήστη ολοκληρώνει την αποστολή	
Failed End Condition	Ο χαρακτήρας του χρήστη αποτυγχάνει, έχοντας χάσει τους πόντους ζωής του	
Primary, Secondary Actors	Χρήστης, Χαρακτήρας, Εικονικός κόσμος, Εχθροί, Σύστημα βολών	
Trigger	Είσοδος χρήστη με πλήκτρο δράσης.	
Description	**Step**	**Action**
	1	Ο χρήστης έχει ήδη ενημερωθεί για την αποστολή, με παράθυρο ενημέρωσης.
	2	Ο χρήστης πλοηγεί τον χαρακτήρα του στο σημείο που πρέπει να πάει.
	3	Ο χρήστης πατάει το πλήκτρο δράσης.
	4	Μια σκηνή διαδραματίζεται. Ίσως και κάποιος διάλογος. Ίσως κάποια μάχη.
	5	Το βήμα της αποστολής ολοκληρώνεται.
	6	Αν δεν είναι το τελευταίο βήμα υπάρχει ενημέρωση για το επόμενο βήμα και μεταβαίνει στο βήμα 2 ξανά.
	7	Αν είναι το τελευταίο βήμα, η αποστολή ολοκληρώνεται και ο χρήστης μεταβαίνει στη σελίδα αποτελεσμάτων του κεντρικού μενού.
Extensions	**Step**	**Branching Action**

	5α	5α1 Ο χαρακτήρας πεθαίνει σε μια μάχη.
		5α2 Η αποστολή δεν ολοκληρώνεται και ο χρήστης μεταβαίνει στο κεντρικό μενού.

Use Case 5	Επίθεση από εχθρό	
Requirements explored	#8,9,10,39,13,24,25,26,27,28,17,18,19	
Scope & Level	Λογική παιχνιδιού	
Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Ο εχθρός επιτίθεται πυροβολεί, στοχεύοντας τον χαρακτήρα.	
Failed End Condition	Δεν υπάρχουν σφαίρες διαθέσιμες	
Primary, Secondary Actors	Εχθροί, Σύστημα διαχείρισης χαρακτήρων, Χαρακτήρας, Εικονικός κόσμος, Σύστημα βολών	
Trigger	Εντοπισμός του χαρακτήρα του χρήστη, από εχθρό	
Description	**Step**	**Action**
	1	Ο εχθρός εντοπίζει τον χαρακτήρα (100 μέτρα οπτική εμβέλεια).
	2	Ο εχθρός κινείται προς τα εμπρός και περιστρέφεται προς την κατεύθυνση του στόχου του.
	3	Όταν ο εχθρός προσεγγίσει εμβέλεια βολής (80 μέτρα) σταματάει την κίνησή του και στοχεύει.
	4	Όταν επιτευχθεί η στόχευση, πυροβολεί.
	5	→ Use case 3 / sub variation 1a
Extensions	**Step**	**Branching Action**
	3α	3α1 Ο εχθρός χάνει τον χαρακτήρα.
		3α2 Δεν επιτίθεται

Use Case 6	Περιήγηση αυτοκινήτου ή περαστικού	
Requirements explored	#5,8,9,10,21,22,23,28	
Scope & Level	Λογική παιχνιδιού	
Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Περιήγηση αυτοκινήτου ή περαστικού σε κάποιο μονοπάτι	
Failed End Condition	Αδρανοποίηση	
Primary, Secondary Actors	Περαστικοί, Αυτοκίνητα, Σύστημα διαχείρισης περαστικών και αυτοκινήτων, Εικονικός κόσμος,	
Trigger	-	
Description	**Step**	**Action**
	1	Ο περαστικός/Το αυτοκίνητο βρίσκεται σε ένα σημείο του μονοπατιού του.
	2	Με τυχαία τρόπο επιλέγει το επόμενο σημείο που θα μεταβεί, από τα διαθέσιμα σημεία προς μετάβαση.
	3	Κινείται προς το επόμενο σημείο.

	4	Ανιχνεύει συγκρούσεις με αντίπαλη ομάδα (ομάδα1= περαστικοί / ομάδα2= αυτοκίνητα)
	5	Άμα ανιχνευτεί σύγκρουση, σταματάει για 5 δευτερόλεπτα μέχρι να περάσει το άλλο αντικείμενο.
	6	Μόλις περάσει, συνεχίζει μέχρι να φτάσει στο στόχο του -> βήμα3.
Extensions	**Step**	**Branching Action**
	6α	6α1 Για κάποιο λόγο περνάνε 5 δευτερόλεπτα χωρίς να περάσει το δεύτερο αντικείμενο.
		6α2 Ακολουθείται σχέδιο αποφυγής DEADLOCK
		6α3 Με τυχαία τρόπο επιλέγει διαφορετικό επόμενο σημείο που θα μεταβεί, από τα διαθέσιμα σημεία προς μετάβαση.
		6α4 Αν είναι όμοιο με το προηγούμενο σημείο, κινείται χωρίς να ανιχνεύει συγκρούσεις.

Use Case 7	Ενημέρωση χρήστη για την αποστολή	
Requirements explored	**Κατηγορία 4:** Διεπαφή χρήστη εντός παιχνιδιού	
Scope & Level	Χρήστη, εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Ενημέρωση χρήστη για την αποστολή	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης gui	
Trigger	-	
Description	**Step**	**Action**
	1	Ο χρήστης έχει φορτώσει το παιχνίδι και παίζει
	2	Πατάει το κουμπί ενημέρωσης
	3	Το σύστημα εμφανίζει παράθυρο ενημέρωσης για την αποστολή
Sub-Variations	1α	1α1α Το παράθυρο ενημέρωσης για την αποστολή είναι ενεργό
		1α1β Ο χρήστης ενημερώνεται ήδη.
		1α1γ Πατάει το κουμπί ενημέρωσης και απενεργοποιεί το παράθυρο.

Use Case 8	Ενημέρωση χρήστη για την ζωή και τις σφαίρες
Requirements explored	**Κατηγορία 4:** Διεπαφή χρήστη εντός παιχνιδιού
Scope & Level	Χρήστη, εφαρμογής
Preconditions	Η εφαρμογή εκτελείται

Success End Condition	Ενημέρωση χρήστη για την ζωή και τις σφαίρες	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης gui	
Trigger	-	
Description	**Step**	**Action**
	1	Ο χρήστης έχει φορτώσει το παιχνίδι και παίζει
	2	Στο πάνω δεξιό κομμάτι της οθόνης αναγράφεται η ζωή και οι σφαίρες του χαρακτήρα του

Use Case 9	Ενημέρωση χρήστη με ραντάρ	
Requirements explored	**Κατηγορία 4:** Διεπαφή χρήστη εντός παιχνιδιού	
Scope & Level	Χρήστη, εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Ενημέρωση χρήστη με ραντάρ	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα καμερών, Σύστημα διαχείρισης gui	
Trigger	Κουμπί εμφάνισης του ραντάρ	
Description	**Step**	**Action**
	1	Ο χρήστης έχει φορτώσει το παιχνίδι και παίζει
	2	Πατάει το κουμπί εμφάνισης του ραντάρ
	3	Το σύστημα εμφανίζει κάτω αριστερά ένα μικρό ραντάρ
Sub-Variations	1α	1α1α Το ραντάρ είναι ενεργό
		1α1β Ο χρήστης ενημερώνεται ήδη.
		1α1γ Πατάει το κουμπί του ραντάρ και το απενεργοποιεί

Use Case 10	Ενημέρωση χρήστη για τα συμβάντα του παιχνιδιού
Requirements explored	**Κατηγορία 4:** Διεπαφή χρήστη εντός παιχνιδιού
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής
Preconditions	Η εφαρμογή εκτελείται
Success End Condition	Ενημέρωση χρήστη για τα συμβάντα του παιχνιδιού
Failed End Condition	-
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης γεγονότων, Σύστημα διαχείρισης gui
Trigger	-

Description	Step	Action
	1	Συμβαίνει γεγονός που χρήζει ενημέρωσης
	2	Στο πάνω αριστερό κομμάτι της οθόνης εμφανίζεται μια συνοπτική πρόταση για την ενημέρωση του γεγονότος

Use Case 11	Ενημέρωση χρήστη για τα αποτελέσματα του παιχνιδιού	
Requirements explored	**Κατηγορία 1:** Διεπαφή χρήστη εκτός παιχνιδιού	
Scope & Level	Χρήστη, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Ενημέρωση χρήστη για τα αποτελέσματα του παιχνιδιού	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης γεγονότων, Σύστημα διαχείρισης μενού	
Trigger	-	
Description	**Step**	**Action**
	1	Ο χρήστης έχει ολοκληρώσει μια αποστολή, και πατάει το κουμπί εμφάνισης αποτελεσμάτων
	2	Το σύστημα μεταβαίνει στο κεντρικό μενού, στη σελίδα αποτελεδμάτων

Use Case 12	Αποτυχία αποστολής	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Αποτυχία αποστολής	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης γεγονότων, Σύστημα διαχείρισης μενού	
Trigger	-	
Description	**Step**	**Action**
	1	Ο κεντρικός χαρακτήρας χάνει όλους τους πόντους ζωής του.
	2	Το σύστημα εμφανίζει μήνυμα αποτυχίας
	3	Ο χρήστης πατάει κουμπί για επιστροφή στο κεντρικό μενού
	4	Το σύστημα μεταβαίνει στο κεντρικό μενού

Use Case 13	Αλληλεπίδραση με αντικείμενο
Requirements explored	#7,30,5,32,13,15,20,27,29,30
Scope & Level	Χρήστη, Λογική παιχνιδιού

Preconditions	Το παιχνίδι έχει φορτωθεί και δεν είναι σταματημένο	
Success End Condition	Ο χαρακτήρας του χρήστη αλληλεπιδρά με αντικείμενο	
Failed End Condition	Ο χαρακτήρας του χρήστη δεν αλληλεπιδρά με αντικείμενο	
Primary, Secondary Actors	Χρήστης, Χαρακτήρας, Εικονικός κόσμος, Σύστημα διαχείρισης αλλιλεπιδράσεων	
Trigger	Είσοδος χρήστη με πλήκτρο δράσης.	
Description	**Step**	**Action**
	1	Ο χρήστης πλοηγεί τον χαρακτήρα του στο σημείο που προσφέρει αλληλεπίδραση
	2	Ο χρήστης πατάει το πλήκτρο δράσης.
	3	Το αντικείμενο προσφέρει κάποια αλληλεπίδραση.
Extensions	**Step**	**Branching Action**
	1α	1α1 Ο χρήστης πλοηγεί τον χαρακτήρα του στο σημείο που ΔΕΝ προσφέρει αλληλεπίδραση
		1α2 Ο χαρακτήρας του χρήστη δεν αλληλεπιδρά με κάποιο αντικείμενο

Use Case 14	Εγκατάλειψη παιχνιδιού	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Εγκατάλειψη παιχνιδιού	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Σύστημα διαχείρισης μενού	
Trigger	-	
Description	**Step**	**Action**
	1	Ο χρήστης πατάει το κουμπί εγκατάλειψης
	2	Η εφαρμογή τερματίζει

Use Case 15	Νίκη εχθρού	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Νίκη εχθρού	
Failed End Condition	-	
Primary, Secondary Actors	Χαρακτήρας, Εικονικός κόσμος, Σύστημα διαχείρισης χαρακτήρων	
Trigger	-	
Description	**Step**	**Action**

	1	Οι πόντοι ζωής του εχθρικού χαρακτήρα τελειώνουν
	2	Ο εχθρός πεθαίνει

Use Case 16	Συνομιλία με χαρακτήρα	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Συνομιλία με χαρακτήρα	
Failed End Condition	-	
Primary, Secondary Actors	Σκηνή, Χαρακτήρας, Εικονικός κόσμος, Σύστημα διαχείρισης σκηνών, Σύστημα διαχείρισης gui	
Trigger	-	
Description	**Step**	**Action**
	1	Μια σκηνή διαδραματίζεται
	2	Εμφανίζονται διάλογοι στο κάτω μέρος της οθόνης.
	3	Όταν μιλάει κάποιος χαρακτήρας κινεί το σώμα του αναλόγως
	4	Όταν ολοκληρωθεί η σκηνή, η συνομιλία έχει ολοκληρωθεί

Use Case 17	Όραση	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Όραση	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Εικονικός κόσμος, Σύστημα διαχείρισης καμερών	
Trigger	-	
Description	**Step**	**Action**
	1	Η οπτική της κάμερας είναι σταθεροποιημένη
	2	Όταν ο χρήστης μετατοπίσει το ποντίκι και η κάμερα μετατοπίζεται και κατόπιν σταθεροποιείται ξανά
	3	Η χρήστης βλέπει την οπτική γωνία της κάμερας

Use Case 18	Εμφάνιση υποτίτλων
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής
Preconditions	Η εφαρμογή εκτελείται

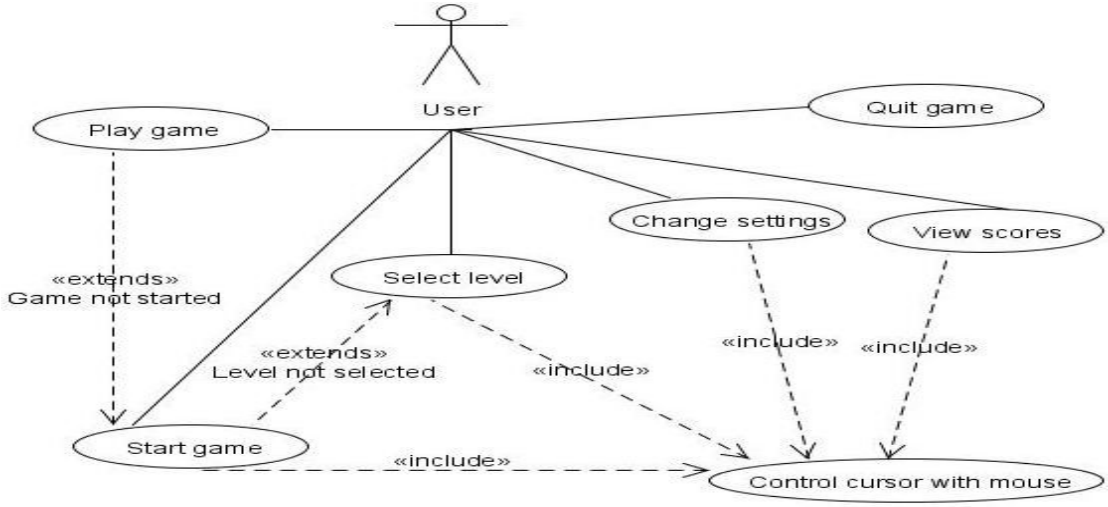
Success End Condition	Όραση	
Failed End Condition	-	
Primary, Secondary Actors	Σκηνή, Σύστημα διαχείρισης σκηνών, Σύστημα διαχείρισης gui	
Trigger	-	
Description	**Step**	**Action**
	1	Μια σκηνή διαδραματίζεται
	2	Το κάτω μέρος της οθόνης γίνεται μαύρο.
	3	Εμφανίζονται γράμματα σε λευκό χρώμα, που σχηματίζουν προτάσεις.
	4	Αριστερά εμφανίζεται το όνομα του ομιλητή

Use Case 19	Εμφάνιση άλλων χαρακτήρων	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Όραση	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, Χαρακτήρας, Εικονικός κόσμος, Σύστημα διαχείρισης καμερών	
Trigger	-	
Description	**Step**	**Action**
	1	Από την οπτική γωνία της κάμερας, ο χρήστης βλέπει τον εικονικό κόσμο
	2	Μέσα σε αυτόν παρατηρεί εικονικούς χαρακτήρες με την μορφή τρισδιάστατων πλεγμάτων

Use Case 20	Παρατήρηση ζωής εχθρών	
Requirements explored	**Κατηγορία 6, Κατηγορία 1, Κατηγορία 2**	
Scope & Level	Χρήστη, Λογική παιχνιδιού, Εφαρμογής	
Preconditions	Η εφαρμογή εκτελείται	
Success End Condition	Όραση	
Failed End Condition	-	
Primary, Secondary Actors	Χρήστης, εχθροί, Εικονικός κόσμος, Σύστημα διαχείρισης gui	
Trigger	-	
Description	**Step**	**Action**
	1	Πάνω από τους εχθρούς εμφανίζεται μια πινακίδα
	2	Η πινακίδα αναγράφει με αριθμό τους πόντους ζωής του εχθρού
	3	Όταν ο εχθρός βρεθεί στην οπτική γωνία της κάμερας, ο χρήστης παρατηρεί τη τον αριθμό της πινακίδας.

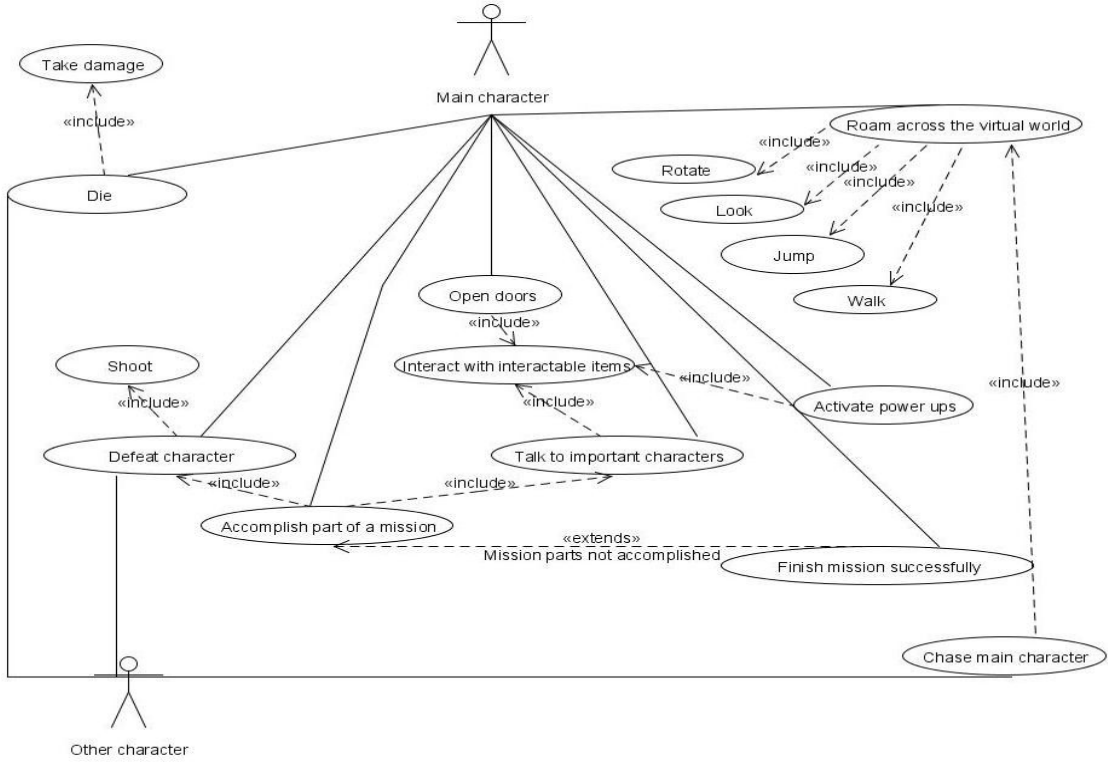
4.5.4 Διαγράμματα περιπτώσεων χρήσης

Παρακάτω παρουσιάζουμε τα διαγράμματα περιπτώσεων χρήσης σε γλώσσα UML2. Κατασκευάστηκαν με την εφαρμογή UMLet. Το πρώτο διάγραμμα περιπτώσεων χρήσης, περιγράφει το σύστημα διαχείρισης των δυνατοτήτων της εφαρμογής. Ηθοποιός είναι ο χρήστης της εφαρμογής.



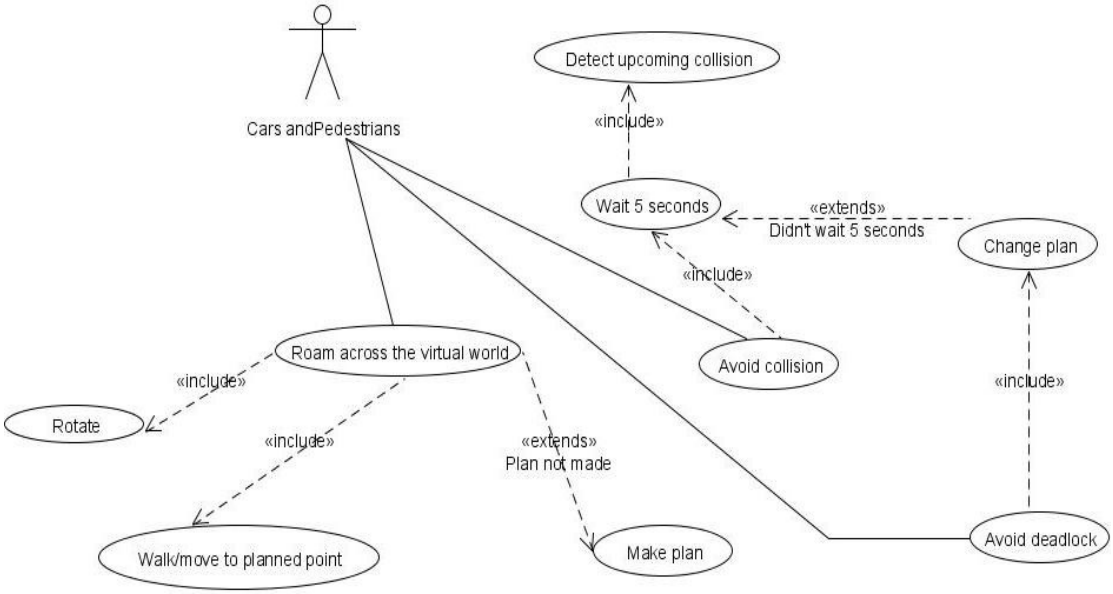
Εικόνα 9: UML use case diagram: Σύστημα διαχείρισης εφαρμογής

Το δεύτερο διάγραμμα περιπτώσεων χρήσης, περιγράφει το σύστημα διαχείρισης των χαρακτήρων του παιχνιδιού. Εδώ έχουμε δύο ηθοποιούς, τον κεντρικό χαρακτήρα, τον οποίο χειρίζεται ο χρήστης, και τους υπολοίπους χαρακτήρες, τους οποίους διαχειρίζεται το σύστημα.



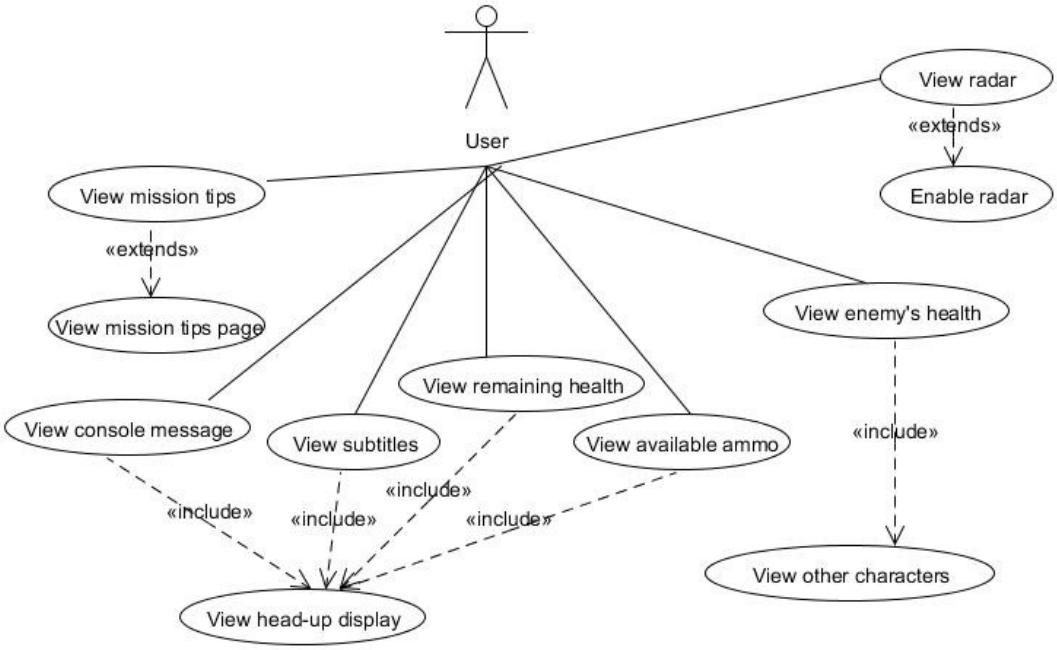
Εικόνα 10: UML use case diagram: Σύστημα διαχείρισης χαρακτήρων

Το τρίτο διάγραμμα περιπτώσεων χρήσης, περιγράφει το σύστημα διαχείρισης των περαστικών και των αυτοκινήτων. Έχουμε ένα ηθοποιό, καθώς τα αυτοκίνητα και οι περαστικοί έχουν τις ίδιες περιπτώσεις χρήσης.



Εικόνα 11: UML use case diagram: Σύστημα διαχείρισης περαστικών και αυτοκινήτων

Το τέταρτο διάγραμμα περιπτώσεων χρήσης, περιγράφει το σύστημα γραφικής διεπαφής χρήστη εντός παιχνιδιού.



Εικόνα 12: UML use case diagram: Σύστημα διαχείρισης γραφικής διεπαφής χρήστη εντός παιχνιδιού

Κεφάλαιο 5

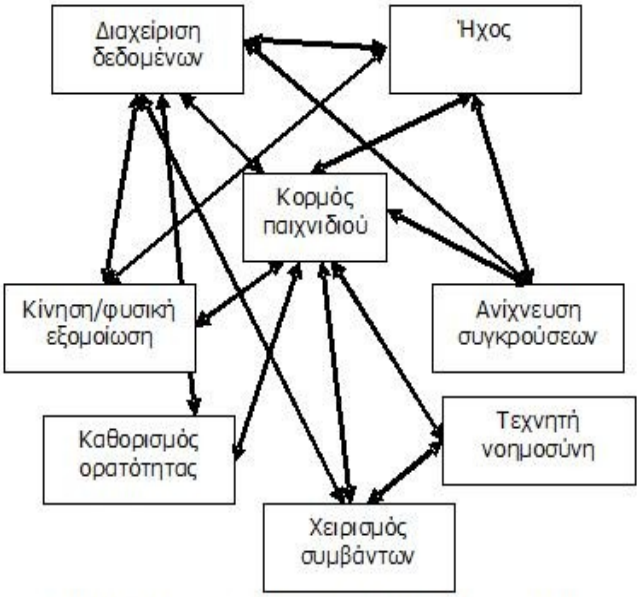
Σχεδιασμός αρχιτεκτονικής του συστήματος

5.1 Εισαγωγή

Η αρχιτεκτονική ενός παιχνιδιού καθορίζει τις επιμέρους οντότητες ενός παιχνιδιού και την μεταξύ τους αλληλεπίδραση και επικοινωνία.

Ένας τρόπος σχεδίασης της αρχιτεκτονικής του συστήματος, είναι η σχεδίαση με οντότητες και ανεξέλεγκτη επικοινωνία. Αυτή έχει τα εξής χαρακτηριστικά:

- Σαφής διαχωρισμός των λειτουργιών και υλοποίηση σε οντότητες
- Η μεταξύ τους επικοινωνία δεν καθορίζεται
- Ευκολότερος εντοπισμός σφαλμάτων
- Η ανεξέλεγκτη επικοινωνία δημιουργεί εξαρτήσεις

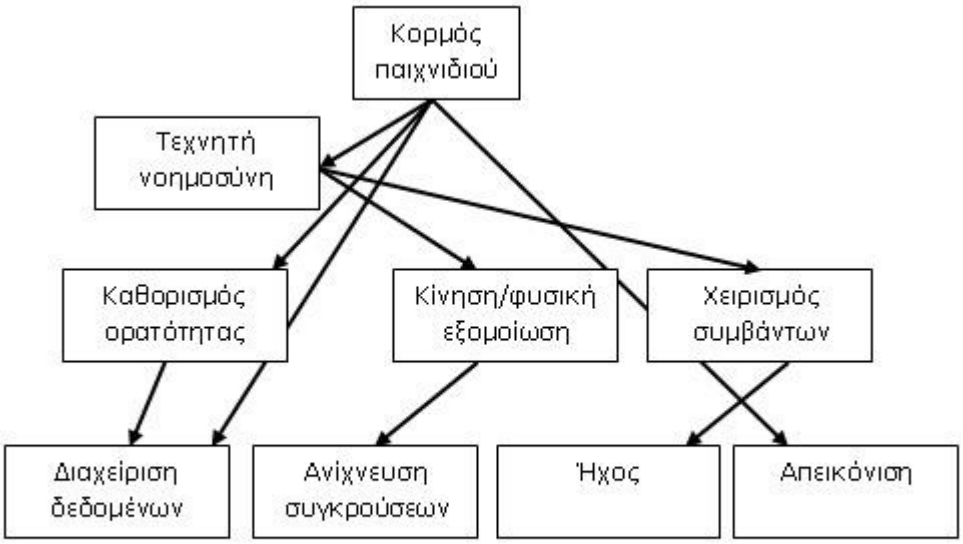


Εικόνα 2: Σχεδίαση παιχνιδιού με διακριτά υποσυστήματα

Εικόνα 13: Γενικευμένη μορφή αρχιτεκτονικής παιχνιδιού [10]

Όμως όπως προαναφέραμε, στο προηγούμενο μοντέλο, η ανεξέλεγκτη επικοινωνία δημιουργεί εξαρτήσεις. Για αυτό το λόγο προτιμάμε την ιεραρχική σχεδίαση. Η ιεραρχική αρχιτεκτονική, η οποία είναι παραλλαγή της σχεδίασης με οντότητες, έχει τα εξής χαρακτηριστικά:

- Διακριτά υποσυστήματα
- Περιορισμός της ενδοεπικοινωνίας με γράφο
- Διαχωρισμός σε υποσυστήματα υψηλού και χαμηλού επιπέδου
- Τα υψηλού επιπέδου υποσυστήματα μπορούν να επικοινωνήσουν με χαμηλού επιπέδου

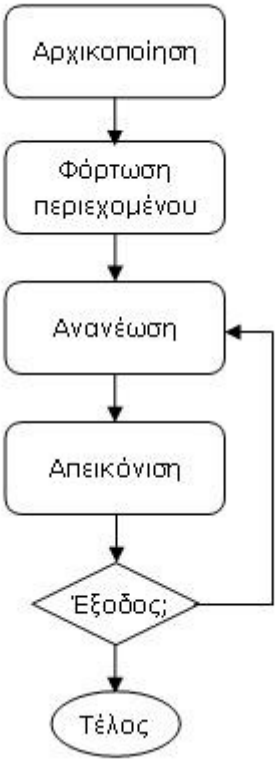


Εικόνα 3: Σχεδίαση παιχνιδιού με γράφο

Εικόνα 14: Ιεραρχική σχεδίαση με οντότητες [10]

5.2 Βρόγχος παιχνιδιού

Κάθε παιχνίδι περνά από μια σειρά συγκεκριμένων διαδικασιών προκειμένου να απεικονίσει κάθε καρέ παιχνιδιού. Τα βήματα που πρέπει να πραγματοποιηθούν σε κάθε καρέ για την απεικόνισή του ονομάζεται βρόγχος παιχνιδιού (game loop).



Εικόνα 15: Game loop [9]

Η αρχικοποίηση πραγματοποιεί εργασίες απαραίτητες για την εκκίνηση του παιχνιδιού. Το στάδιο της φόρτωσης μεταφέρει τα δεδομένα από το αποθηκευτικό μέσο στη μνήμη. Το στάδιο της ανανέωσης (Λογική), υλοποιεί κίνηση, ανίχνευση συγκρούσεων, εξομοίωση. Το στάδιο της απεικόνισης ζωγραφίζει τον εικονικό κόσμο στην οθόνη (ένα καρέ). Τα δύο τελευταία βήματα επαναλαμβάνονται μέχρι να κλείσουμε το παιχνίδι.

Για 60Hz ρυθμό ανανέωσης έχουμε 16.6 ms χρόνο για να ανανεώσουμε τον κόσμο και να ζωγραφίσουμε ένα καρέ.

Ο συνολικός αναλυτικότερος βρόγχος σε ψευδοκώδικα[9]:

Αρχικοποίηση
Φόρτωση περιεχόμενου
Αρχή βρόγχου
  Λογική
    Έλεγξε είσοδο από χειριστήριο
    Έλεγξε μηνύματα μέσω δικτύου
    Ενημέρωση χαρακτήρα παίκτη
    Δημιουργία λίστας αντικειμένων/χαρακτήρων προς ανανέωση
    Ενημέρωση αντικειμένων με απλή λογική
    Ενημέρωση αντικειμένων με σύνθετη λογική (τεχνητή νοημοσύνη)
    Αποστολή μηνυμάτων μέσω δικτύου
    Ενημέρωση ήχου
  Απεικόνιση
    Δημιουργία λίστας ορατών αντικειμένων/χαρακτήρων
    Απεικόνιση ορατού σκηνικού
    Απεικόνιση ορατών αντικειμένων/χαρακτήρων
    Απεικόνιση χαρακτήρα παίκτη
    Εφαρμογή ειδικών εφέ
    Απεικόνιση διεπαφής
    Αναπαραγωγή ήχου
Αν όχι έξοδος πήγαινε στην αρχή
Αποδέσμευση πόρων
Έξοδος

5.3 Υποσυστήματα αρχιτεκτονικής

Όπως προαναφέραμε στην εισαγωγή του κεφαλαίου, θα ακολουθήσουμε ιεραρχική σχεδίαση με οντότητες. Οι οντότητες είναι διακριτά υποσυστήματα της αρχιτεκτονικής. Ακολούθως θα καθορίσουμε ένα βασικό διαχωρισμό υποσυστημάτων και θα τα ορίσουμε, ως προς τις υπευθυνότητες τους και τον βαθμό στην ιεραρχία.

Στο υψηλό επίπεδο ιεραρχίας, υπάρχει το υποσύστημα του πυρήνα της μηχανής του παιχνιδιού μας. Το ονομάζουμε "Core". Είναι μια οντότητα, που αρχικοποιεί τη μηχανή του παιχνιδιού και τις υπόλοιπες χρήσιμες βιβλιοθήκες, και θα προσφέρει στις οντότητες που έχουν ανάγκη, τη χρήση των λειτουργιών της μηχανής.

Επίσης, υπάρχει το υποσύστημα διαχείρισης γεγονότων εισόδου χρήστη και το ονομάζουμε "Event Manager". Προσφέρει στις οντότητες που έχουν ανάγκη, προσαρμοσμένα για τις ανάγκες του παιχνιδιού σήματα εισόδου χρήστη.

Στο μεσαίο επίπεδο ιεραρχίας, έχουμε το κεντρικό σύστημα διαχείρισης του παιχνιδιού μας, που το ονομάζουμε "System" και το σύστημα αρχικοποίησης του

εικονικού κόσμου του παιχνιδιού που το ονομάζουμε “Level”.

Το πρώτο είναι ίσως η πιο σημαντική οντότητα της αρχιτεκτονικής μας, καθώς έχει πολλές υπευθυνότητες, όπως οι πραγματικού χρόνου υπολογισμοί γραφικών, λογικής, φυσικής, τεχνικής νοημοσύνης, η απόδοση γραφικών και γραφικής διεπαφής και η αρχικοποίηση του παιχνιδιού. Είναι η οντότητα που είναι υπεύθυνη για τον βρόγχο παιχνιδιού, που περιγράψαμε προηγουμένως. Στην περίπτωση που η πολυπλοκότητα του παιχνιδιού μας ήταν αρκετά μεγαλύτερη, θα ήταν προτιμότερο αυτό το σύστημα, να διαχωριστεί σε περισσότερα υποσυστήματα.

Το δεύτερο αρχικοποιεί τον εικονικό κόσμο, βάσει των παραμέτρων που έχει φορτώσει ο πυρήνας. Μόλις φορτωθεί ο κόσμος, δίνεται στο σύστημα για να ελέγχει αυτόν και τις εικονικές οντότητες που βρίσκονται σε αυτόν, βάσει των νόμων του παιχνιδιού μας.

Στο τελευταίο επίπεδο ιεραρχίας, έχουμε τις οντότητες που βρίσκονται στον εικονικό μας κόσμο. Αυτές είναι οι χαρακτήρες, οι περαστικοί, τα αυτοκίνητα, οι σκηνές της πλοκής του σεναρίου, τα αντικείμενα που μπορεί να πάρει ο χαρακτήρας του χρήστη και οι πόρτες που μπορεί να ανοίξει. Κάθε μια οντότητα έχει τις δικές της λειτουργίες και τα στιγμιότυπα τις αντιπροσωπεύουν, υποκείμενα ή αντικείμενα στον εικονικό κόσμο.

Επίσης υπάρχει ένα ξεχωριστό σύστημα, το οποίο είναι το κεντρικό μενού της εφαρμογής μας. Δεν αποτελεί κομμάτι της αρχιτεκτονικής του παιχνιδιού, αλλά αποτελεί κομμάτι της εφαρμογής.

Τα υποσυστήματα της αρχιτεκτονικής μας ικανοποιούν τις απαιτήσεις που έχουμε κατηγοριοποιήσει στο προηγούμενο κεφάλαιο ως εξής:

Κατηγορία 1:Επιλογές παιχνιδιού, διεπαφή χρήστη εκτός παιχνιδιού και εμφάνιση αποτελεσμάτων:
Οντότητες: Main menu

Κατηγορία 2: Γραφικά:
Οντότητες: Core, Level,System

Κατηγορία 3: Ηχητικά εφέ και μουσική:
Οντότητες: Core

Κατηγορία 4: Διεπαφή χρήστη εντός παιχνιδιού:
Οντότητες: System, Event manager

Κατηγορία 5: Πλοήγηση και έλεγχος χαρακτήρα:
Οντότητες: Character , System

Κατηγορία 6: Λογική παιχνιδιού:
Οντότητες: System , Entities

Κατηγορία 7: Διαχείριση δεδομένων και κατάστασης εφαρμογής:
Οντότητες: System

5.4 Οντότητες παιχνιδιού

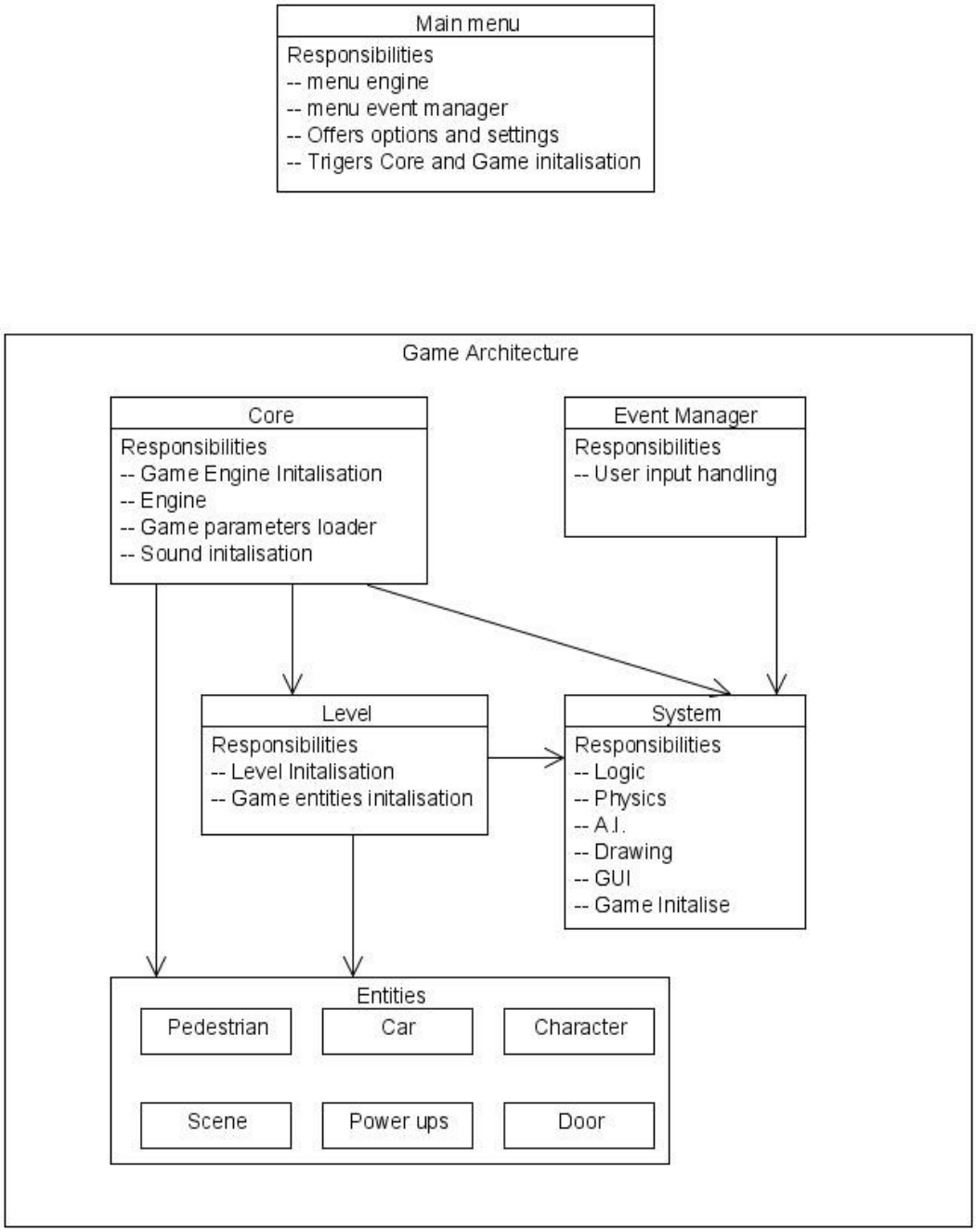
Βάσει της σχεδίασης και της ανάλυσης που κάναμε στα προηγούμενα κεφάλαια, καταλήξαμε στην ύπαρξη των παρακάτω οντοτήτων:

- **Player**: Αυτή η οντότητα, αντιπροσωπεύει τους χαρακτήρες του παιχνιδιού. Αυτό συμπεριλαμβάνει και τον χαρακτήρα που ελέγχει ο χρήστης και αυτούς που ελέγχει το σύστημα.
- **Pedestrian**: Οι περαστικοί, που περιφέρονται στο κόσμο. Η ύπαρξή τους σε κάθε επίπεδο είναι προαιρετική.
- **Car**: Τα αυτοκίνητα, που περιφέρονται στο κόσμο. Όπως και με τους περαστικούς, η ύπαρξη τους σε κάθε επίπεδο είναι προαιρετική.
- **Interactive**: Αυτή η οντότητα αντιπροσωπεύει όλα τα στοιχεία μέσα στο παιχνίδι που προσφέρουν αλληλεπίδραση. Οι παρακάτω οντότητες είναι παιδιά αυτής της κλάσης, καθώς όλες προσφέρουν αλληλεπίδραση.
- **Scene**: Αυτή η οντότητα αντιπροσωπεύει τις σκηνές που διαδραματίζονται κατά την εξέλιξη της πλοκή της υπόθεσης. Συνήθως είναι κάποιος διάλογος.
- **Powerup**: Κάποια αντικείμενα που προσφέρουν επιπλέον δυνατότητες στον χαρακτήρα του χρήστη.
- **Door**: Οι πόρτες που μπορούν να ανοιχθούν από τον χαρακτήρα που ελέγχει ο χρήστης.

Όλες οι παραπάνω οντότητες αρχικοποιούνται από την κλάση Level, καθώς όλες αποτελούν στοιχεία του κάθε επιπέδου του παιχνιδιού μας. Στη συνέχεια αναφερόμαστε στο σύνολο αυτών των οντοτήτων, ως entities, εννοώντας οντότητες επιπέδου παιχνιδιού.

5.5 Επίπεδα και επικοινωνία

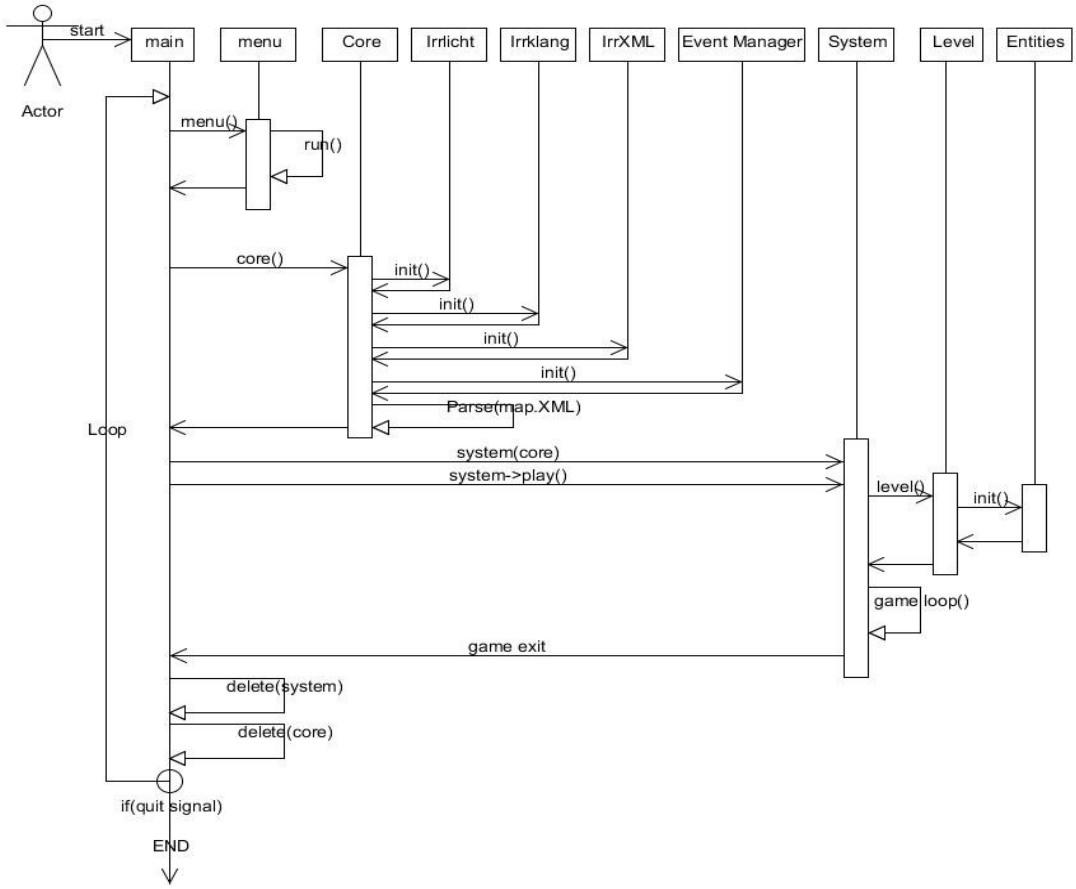
Η αρχιτεκτονική του παιχνιδιού μας, είναι χωρισμένη και ιεραρχημένη σε τρία επίπεδα. Στο πρώτο επίπεδο είναι ο πυρήνας της μηχανής του παιχνιδιού και το υποσύστημα διαχείρισης γεγονότων εισόδου χρήστη, στο δεύτερο επίπεδο είναι το κεντρικό σύστημα διαχείρισης του παιχνιδιού, και ο φορτωτής επιπέδου και οντοτήτων και στο τελευταίο επίπεδο βρίσκονται οι εικονικές οντότητες του επιπέδου του παιχνιδιού. Ξεχωριστό σύστημα αποτελεί το κεντρικό μενού της εφαρμογής μας.



Εικόνα 16: Η αρχιτεκτονική του παιχνιδιού μας, χωρισμένη σε τρία επίπεδα. Το μενού δεν σχετίζεται με την αρχιτεκτονική, αποτελεί όμως κομμάτι της εφαρμογής.

Έχουμε ιεραρχική αρχιτεκτονική, με διακριτά υποσυστήματα, υπάρχει περιορισμός της ενδοεπικοινωνίας με γράφο, διαχωρισμός σε υποσυστήματα υψηλού και χαμηλού επιπέδου. Τα βέλη περιγράφουν την επικοινωνία των υποσυστημάτων.

Παρακάτω παρουσιάζεται η ροή της εφαρμογής από το πιο υψηλό επίπεδο παρατήρησης, με UML διάγραμμα ροής (sequence diagram). Ακολούθως, σχολιάζουμε το διάγραμμα.



Εικόνα 17: Διάγραμμα ροής της εφαρμογής από το πιο υψηλό επίπεδο παρατήρησης

Αρχικά ο χρήστης εκτελεί το πρόγραμμα. Η **main**, δημιουργεί ένα στιγμιότυπο της οντότητας "**menu**", και εκτελείται η μέθοδος του, run(). Η εκτέλεση της μεθόδου run τερματίζεται όταν ο χρήστης επιλέξει την εκκίνηση του παιχνιδιού.

Όταν γίνει αυτό, το σύστημα αποθηκεύει τις ρυθμίσεις του χρήστη, και η main δημιουργεί ένα στιγμιότυπο της κλάσης "**Core**", με τιμές παραμέτρων, βασισμένες στις ρυθμίσεων του χρήστη. Το αντικείμενο της Core, εκτελεί αρχικοποίηση, των ενοτήτων της μηχανής του παιχνιδιού μας, και αποθηκεύει τις διευθύνσεις των αντικειμένων των ενοτήτων, σε δείκτες. Κατόπιν, το ίδιο αντικείμενο του "πυρήνα", ανοίγει το αρχείο παραμέτρων επιπέδου XML, και φορτώνει τις παραμέτρους του επιπέδου, σε κατάλληλες δομές δεδομένων.

Στη συνέχεια, η main, δημιουργεί ένα αντικείμενο της οντότητας "**System**" δίνοντας του ως όρισμα, το στιγμιότυπο του πυρήνα, ώστε να έχει τη δυνατότητα της χρησιμοποίησης των μεθόδων της Core.

Κατόπιν, η main εκτελεί την μέθοδο play() της κλάσης System. Σε αυτή τη μέθοδο το σύστημα αρχικοποιεί το επίπεδο του παιχνιδιού μας, δημιουργώντας ένα αντικείμενο της κλάσης "**Level**", βάσει των παραμέτρων που έχουν φορτωθεί στις δομές δεδομένων για τις παραμέτρους, από τον πυρήνα. Το στιγμιότυπο του επιπέδου **δημιουργεί όσα αντικείμενα των οντοτήτων παιχνιδιού** (player, pedestrian, door, κτλ.) αναφέρουν οι παράμετροι του επιπέδου. Όταν ολοκληρωθεί η δημιουργία του επιπέδου και των οντοτήτων του παιχνιδιού, η μέθοδος play(), του συστήματος, συνεχίζει την εκτέλεσή της, και μεταβαίνει στον **βρόγχο παιχνιδιού**. Όταν ο βρόγχος παιχνιδιού τερματιστεί, αποδεσμεύονται τα αντικείμενα και η εφαρμογή τερματίζει.

Κεφάλαιο 6

Υλοποίηση

6.1 Εισαγωγή

Η υλοποίηση γίνεται σε λειτουργικό σύστημα Microsoft Windows 32bit, σε περιβάλλον προγραμματισμού Microsoft Visual Studio C++ Express 2008, με γλώσσα προγραμματισμού C++. Η χρήση της Irrlicht , της IrrKlang, της IrrXML και της IrrAI, γίνεται με την συμπερίληψη των αρχείων ενσωμάτωσης (include files) τους, τη σύνδεση με τις βιβλιοθήκες (libraries) τους και την δυναμική διασύνδεση των βιβλιοθηκών δυναμικής διασύνδεσης (dll – Dynamic Link Libraries).

Παραθέτουμε συνολικά τις βιβλιοθήκες που χρησιμοποιούμε:

```
//include files for engine and modules
#include <irrlicht.h>
#include <irrKlang.h>
#include <irrXML.h>
#include <IrrAI.h>
//other include files
#include <string>
#include <math.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>
```

Παραθέτουμε τα namespaces που χρησιμοποιούμε για ευκολότερη υλοποίηση:

```
//namespaces for faster coding
using namespace irr;
using namespace irr::io;
using namespace irr::core;
using namespace irr::scene;
using namespace irr::video;
using namespace IrrAI;
using namespace irrklang;
using namespace std;
```

Στις επόμενες ενότητες του κεφαλαίου, παραθέτουμε κάποια κομμάτια της υλοποίησης μας, που όμως δεν είναι ολόκληρα για συντομία, όμως είναι αντιπροσωπευτικά της ολοκληρωμένης υλοποίησης.

6.2 Πυρήνας παιχνιδιού

Ο πυρήνας του παιχνιδιού είναι το σύστημα που ελέγχει τη μηχανή του παιχνιδιού μας . Το ονομάζουμε “Core”. Είναι μια οντότητα, που αρχικοποιεί τη μηχανή του παιχνιδιού και ενσωματώνει τις χρήσιμες βιβλιοθήκες και παράλληλα προσφέρει στα υπόλοιπα συστήματα τη χρήση των λειτουργιών της μηχανής. Συγκεντρωτικά, ο πυρήνας περιλαμβάνει το σύστημα απεικόνισης γραφικών, το σύστημα ήχου, το σύστημα διαχείρισης βάσεων δεδομένων XML, το σύστημα τεχνητής νοημοσύνης και τις παραμέτρους των συστημάτων. Επίσης, αρχικοποιεί και προσφέρει το υποσύστημα διαχείρισης γεγονότων εισόδου χρήστη.

```
class Core {
	private:
		IrrlichtDevice* device;
		EventReceiver* receiver;
		IAIManager* aimgr;
		std::string mapXmlPath;
		//...
	public :
		Core(bool fullscreen, bool music, bool shadows, bool vsync, bool aa, bool lowDetail,
		video::E_DRIVER_TYPE driver,dimension2d<s32> resolution,std::string MapPath);
		inline irr::IrrlichtDevice* getDevice() const { return this->device; }
		inline EventReceiver* getEventReceiver() const { return this->receiver; }
		inline IAIManager* getAIManager() const { return this->aimgr; }
		MapParameters* loadMapParameters() const;
		irrklang::ISoundEngine* irrKlang;
		//....
};
```

Η κλάση "πυρήνας" κατέχει τον έλεγχο των μονάδων απεικόνισης, ήχου, τεχνητής νοημοσύνης, εισόδου χρήστη και προσφέρει συναρτήσεις που επιτρέπουν στις υπόλοιπες οντότητες να αποκτούν μερικό έλεγχο και πρόσβαση στις παραπάνω μονάδες.

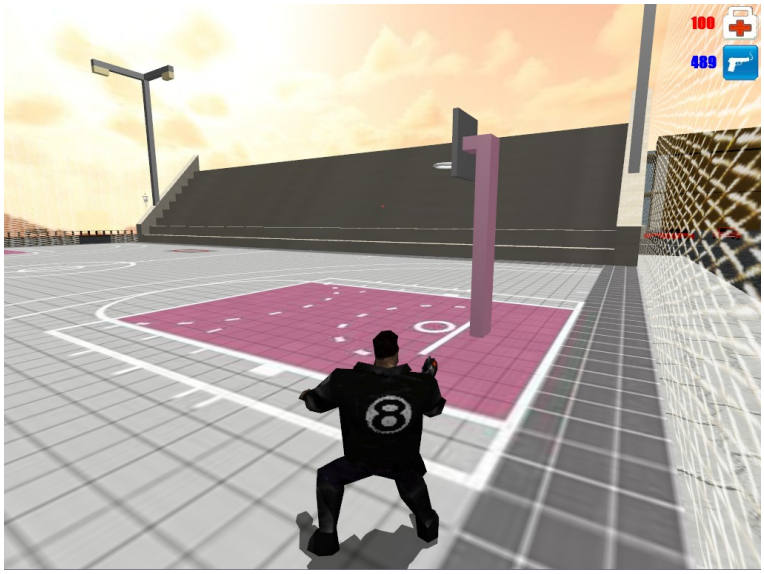
6.3 Σύστημα διαχείρισης περιεχομένου επιπέδων

Κάθε επίπεδο του παιχνιδιού περιέχει έναν εικονικό κόσμο, εικονικούς χαρακτήρες, αντικείμενα, σενάριο και πλοκή. Για να είναι το παιχνίδι μας επεκτάσιμο ως προς τα επίπεδα παιχνιδιού, αναπτύξαμε ένα σύστημα διαχείρισης περιεχομένου επιπέδων, το οποίο αντλεί παραμέτρους από αρχεία δεδομένων XML.

Συγκεκριμένα κάθε επίπεδο αποτελείται από: Αρχεία τρισδιάστατων πλεγμάτων (meshes) για τον εικονικό κόσμο, αρχεία τρισδιάστατων πλεγμάτων (meshes) με κίνηση (animation) για τους χαρακτήρες, αρχεία υφών (textures), αρχεία ήχων, αρχεία σημείων μονοπατιών (way-points) και τέλος αρχείο δεδομένων τύπου XML, το οποίο περιέχει πληροφορίες και παραμέτρους για όλα τα παραπάνω, και για πολλά άλλα, όπως οι σκηνές, οι διάλογοι, τοποθεσίες αντικειμένων, καμερών και φωτισμών, πληροφορίες.

Ο συνδυασμός των παραπάνω γίνεται μέσω δικιάς μας υλοποίησης. Αρχικά διαβάζουμε τα δεδομένα του XML αρχείου και κατόπιν τα φορτώνουμε σε δομές δεδομένων που έχουμε ορίσει για τις ανάγκες των παραμέτρων των οντοτήτων του παιχνιδιού. Στη συνέχεια δημιουργούμε αντικείμενα των οντοτήτων του παιχνιδιού μας, βάσει των παραμέτρων που έχουμε στις δομές δεδομένων που προηγουμένως φορτώσαμε.

Παρακάτω παραθέτουμε την πρώτη εικόνα του παιχνιδιού μας. Η εικόνα είναι από το επίπεδο "Fonz in Cicely". Το διαφανές συρματόπλεγμα που φαίνεται στην εικόνα, υπάρχει στο επίπεδο αυτό, αφού πρώτα φορτώθηκαν οι παράμετροί του από το αρχείο παραμέτρων.



Εικόνα 18: Εικόνα από ένα επίπεδο.

6.3.1 Αρχείο παραμέτρων επιπέδου

Οι παράμετροι κάθε επιπέδου είναι αποθηκευμένοι σε ένα ξεχωριστό αρχείο δεδομένων τύπου XML.

Η XML είναι μια εξαιρετικά ελαστική δομή κωδικοποίησης δεδομένων. Είναι κατάλληλη για την αποθήκευση και ανταλλαγή κάθε είδους δεδομένων που μπορεί να κωδικοποιηθεί σε μορφή κειμένου. Προσφέρει διαχειρίσιμα δεδομένα (portable data) και επιτρέπει την επαναχρησιμοποίηση των δεδομένων.

Παρακάτω παραθέτουμε ένα σύντομο παράδειγμα αρχείου παραμέτρων του επιπέδου και στη συνέχεια, αναλύουμε τα στοιχεία που περιέχει.

```
<?xml version="1.0"?>
<map name="City" path="maps/City.irr" endkey="8"
aipath="ai/city.irrai"heroname="Fonz">

        <playerposition x="-5000" y="50" z="-6100"/>
        <playerrotation x="0" y="270" z="0"/>
        <borders path="meshes/MapBorders.obj" />

        <pedestrians num="20"/>

        <cars num="20"/>

        <skybox up="media/irrlicht2_up.jpg"
        dn="media/irrlicht2_dn.jpg"
        lf="media/irrlicht2_lf.jpg"
        rt="media/irrlicht2_rt.jpg"
        ft="media/irrlicht2_ft.jpg"
        bk="media/irrlicht2_bk.jpg"
        />

        <object type="mountain">
                <position x="-7495" y="0" z="0"/>
                <rotation x="0" y="90" z="0"/>
        </object>
```

```
	<object type="sea">
		<position x="0" y="-200" z=" -17000"/>
		<rotation x="0" y="-90" z="0"/>
	</object>

	<object type="WaterPool">
		<position x="0" y="11" z="0"/>
		<rotation x="0" y="0" z="0"/>
	</object>

	<object type="Train">
		<position x="0" y="0" z="0"/>
		<rotation x="0" y="0" z="0"/>
	</object>

	<enemy name="Mr Coat" health="100">
		<position x="-5266" y="191" z="-5277"/>
		<rotation x="0" y="0" z="0"/>
	</enemy>

	<interaction type="door">
		<position x="-4975" y="6" z="3405"/>
		<rotation x="0" y="-90" z="0"/>
	</interaction>

	<interaction type="POWER1">
		<position x="-4900" y="5" z="-7000"/>
		<rotation x="0" y="0" z="0"/>
	</interaction>

	<scene key="0" endtime="36000" nextKey="1">
		<position x="-5266" y="191" z="-5277" />
		<camera_position x="-5000" y="191" z="-5500" />
		<camera_target x="-5266" y="191" z="-5277" />
		<dialog time="0" who="Fonz" what="Hi there. I am Fonz, and I'm supposed to meet
		someone right here." />
		<dialog time="6000" who="Mr Coat" what="Greetings.I've been expecting you." />
		...
		<move time="15000" who="Mr Coat" x="-5266" y="138" z="-5330" >
		<transfer who="Mr Coat" x="-106" y="51" z="-5234" />
		<setenemy who="Mr Coat" health="20" />
		<givekey who="Mr Coat" key="2" />
	</scene>
	<tip key="5" text="Enter the house, find Phoenix and get the diamonds." />
</map>
```

Τα στοιχεία (elements) έχουν χαρακτηριστικές ετικέτες (tags), και χαρακτηριστικά ονόματα στοιχείων (attributes). Παραδείγματος χάριν το στοιχείο :

```
<enemy name="Mr Coat" health="100">
	<position x="-5266" y="191" z="-5277"/>
	<rotation x="0" y="0" z="0"/>
</enemy>
```

Το παραπάνω κομμάτι κώδικα XML, αναφέρει ότι στο επίπεδό μας υπάρχει ένας εχθρός με το όνομα Mr Coat, ο οποίος έχει 100 πόντους ζωής, βρίσκεται στο σημείο (x,y,z)=(-5266,191,-5277) και αρχικά έχει περιστροφή 0 μοίρες προς όλους τους άξονες.

6.3.2 Άντληση δεδομένων

Η άντληση δεδομένων γίνεται με τη συνάρτηση **loadMapParameters()** του "πυρήνα" που προηγουμένως αναλύσαμε. Αυτή ανοίγει το αρχείο XML και διαβάζει τα περιεχόμενα του. Αποθηκεύει τις πληροφορίες που διαβάζει στη δομή δεδομένων MapParameters, την οποία δημιουργήσαμε για τον συγκεκριμένο σκοπό, δηλαδή την φόρτωση όλων των παραμέτρων του κάθε επιπέδου. Παρακάτω παραθέτουμε ένα κομμάτι της συνάρτησης φόρτωσης δεδομένων επιπέδου.

```
MapParameters* Core::loadMapParameters() const {

        IrrXMLReader* xml = createIrrXMLReader(mapXmlPath.c_str());
        MapParameters* m=new MapParameters();

        //parse the file until end reached
        bool done=false;

        while(xml && xml->read() && !done)
        {
                switch(xml->getNodeType())
                {
                        case EXN_ELEMENT:
                                if (!strcmp("map", xml->getNodeName()))
                                {
                                        m->name=xml->getAttributeValue("name");
                                        m->path=xml->getAttributeValue("path");
                                        m->aipath= xml->getAttributeValue("aipath");
                                        m->heroname= xml->getAttributeValue("heroname");
                                        m->endkey=xml->getAttributeValueAsInt("endkey");
                                        //.....
```

6.3.3 Δομές δεδομένων για τις παραμέτρους επιπέδου

Όπως αναφέραμε οι πληροφορίες του αρχείου XML φορτώνονται στη δομή δεδομένων **MapParameters** την οποία δημιουργήσαμε για το συγκεκριμένο σκοπό, δηλαδή για το παραμετρικό σύστημα επιπέδων.

```
struct MapParameters{
        vector3df playerposition,playerrotation;
        std::string name,path,BordersPath,aipath,heroname;
        irr::core::stringw up,dn,lf,rt,ft,bk;
        u32 carnum,pednum,endkey;
        core::array<InterParam> Inters;
        core::array<Object> Objects;
        core::array<SceneStr> Scenes;
        core::array<Enemy> Enemies;
        core::array<Tip> Tips;
};
```

Επιπλέον δημιουργήσαμε και τις δομές **InterParam, Object, SceneStr, Enemy, Tip,** οι οποίες αντιπροσωπεύουν τις πληροφορίες που χρειαζόμαστε για τις αλληλεπιδράσεις, τα αντικείμενα, τις σκηνές, τους εχθρούς και τις συμβουλές αντιστοίχως.

6.4 Υλοποίηση βρόγχου παιχνιδιού

Ο βρόγχος παιχνιδιού είναι το πιο σημαντικό και περίπλοκο καθώς περιέχει υπολογισμούς και διαδικασίες προκειμένου να απεικονιστεί το κάθε καρέ παιχνιδιού, ή το κάθε καρέ της κάθε σκηνής. Ο βρόγχος παιχνιδιού υλοποιείται στην οντότητα system. Παρακάτω παραθέτουμε ένα μικρό κομμάτι του βρόγχου παιχνιδιού, με τις βασικές διαδικασίες. Αυτές είναι, αρχικοποίηση επιπέδου, καθορισμός κομματιού της οθόνης που απεικονίζεται η λήψη της κάμερας, ενεργοποίηση της επιθυμητής κάμερας, εκκίνηση της σκηνής και εκκαθάριση του καρέ (frame), από τον οδηγό της κάρτας γραφικών, απεικόνιση της σκηνής, και καταγραφή χρόνου για το καρέ που απεικονίστηκε.

```
int System::play(){
//hide cursor
core->getDevice()->getCursorControl()->setVisible(false);

//init level
if(!initGame()) return -1;  //level NOT initialized

sceneStartTime = core->getDevice()->getTimer()->getTime();
//After initialising Irrlicht Device set cursor at the center
core->getDevice()->getCursorControl()->setPosition((Width/2),(Height/2));

//game loop
while(core->getDevice()->run()){
      if (core->getDevice()->isWindowActive()){
                              // load next scene if necessary
                              now = core->getDevice()->getTimer()->getTime();
                              if (now - sceneStartTime > timeForThisScene && timeForThisScene!=-1){
                                    currentScene++;
                                    //Set the viewpoint to the whole screen and begin scene
                                    driver->setViewPort(rect<s32>(0,0,Width,Height));

                                    if (!MovieScene){
                                          smgr->setActiveCamera(camera);
                                    }
                                    else{
                                          smgr->setActiveCamera(movieCamera);
                                    }
                                    sceneStartTime = core->getDevice()->getTimer()->getTime();
                                    driver->beginScene(true, true, SColor(255,200,200,200));
                                    switchToNextScene();

                                    smgr->drawAll();
                                    //drawScene, win caption and fps
                                    lastFPS=drawScene(lastFPS,driver->getFPS());
                              }
                              last = core->getDevice()->getTimer()->getTime();

      }
                  else{
                              core->getDevice()->yield();
                  }
}
return 1;
}
```

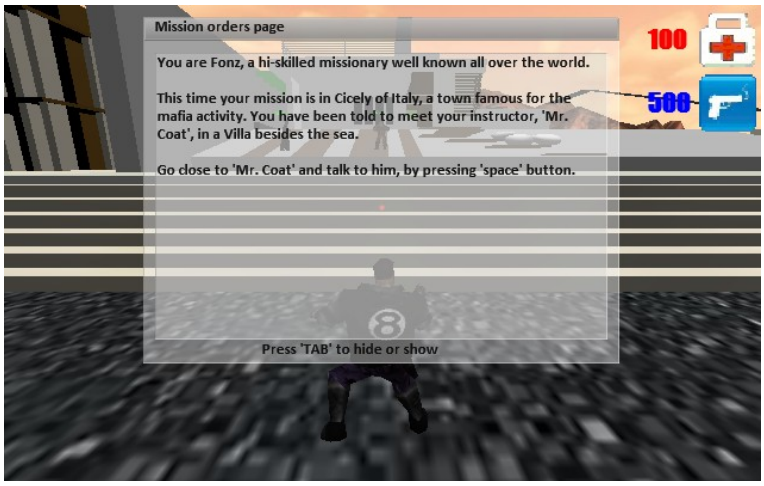
6.4.1 Σκηνή και κόμβοι σκηνών

Το παιχνίδι μας ουσιαστικά αποτελείται από σκηνές, που σε πραγματικό χρόνο υπολογίζονται και στη συνέχεια το σύστημα απεικονίζει τα καρέ της ενεργής σκηνής. Κάθε σκηνή αποτελείται αποκλειστικά από κόμβους σκηνών. Κόμβοι σκηνής μπορεί να είναι ένα τρισδιάστατο πλέγμα, όπως ένας χαρακτήρας, ένα τοπίο ή ένα αντικείμενο, μια κάμερα, μια πηγή φωτός, απλώς ένα σημείο και άλλα.

Στον βρόγχο παιχνιδιού οι υπολογισμοί και οι διαδικασίες αφορούν κόμβους σκηνής, όπως για παράδειγμα υπολογισμοί για το πού θα τοποθετηθεί ο χαρακτήρας, πού η κάμερα, αν θα εμφανιστεί ή θα εξαφανιστεί κάποιο εφέ, αν θα αλλάξει το τοπίο.

Η διαχείριση της σκηνής δίδεται από τον πυρήνα στο κεντρικό σύστημα μέσω της εντολής: **ISceneManager* smgr=core->getDevice()->getSceneManager().**

Με τον διαχειριστή σκηνής (Scene Manager), είναι δυνατό να προστεθούν/αφαιρεθούν και τροποποιηθούν, οι κόμβοι σκηνής.



Εικόνα 19: Εκκίνηση επιπέδου "Fonz in Cicely"

6.4.2 Κάμερες

Υπάρχουν τρεις κάμερες που εναλλάσσονται στο παιχνίδι. Η τρίτου προσώπου πίσω από τον κεντρικό χαρακτήρα και η πρώτου προσώπου είναι κάμερες που εναλλάσσονται ανάλογα την προτίμηση του χρήστη, με το πλήκτρο 'V'.

```
void System::setCamera(){
//switch camera
if(receiver->IsKeyDown(KEY_KEY_V) ){
        if(!cameraChanged){
                if(CameraON==TPS){
                        camera->setPosition(vector3df(6,20,6));
                        cameraChanged=true;
                        CameraON=FPS;
                }
                else{
                        camera->setPosition(vector3df(-75,50,0));
                        cameraChanged=true;
                        CameraON=TPS;
                }
        }
}else cameraChanged=false;
```

Εικόνα 20: Κάμερα τρίτου προσώπου

Παρακάτω, παραθέτουμε την υλοποίηση της μετακίνησης της κάμερας, ανάλογα με την κίνηση του χαρακτήρα, και ανάλογα με ποια κάμερα είναι ενεργοποιημένη.

```
//Set camera's new position
if(CameraON==TPS)
{
	//camera 3rd person  here
	vector3df p1=players[0]->getSN()->getPosition();
	f32* pzpx=players[0]->getPzPx();

	//move target forewords
	p1.Z-=pzpx[1]*200;
	p1.X+=pzpx[0]*200;
	camera->setTarget(p1+vector3df(0,players[0]->getAMSN()->getRotation().Z*5.5f,0));
}
else
{
	//camera fps here
	vector3df p1=players[0]->getSN()->getPosition();
	f32* pzpx=players[0]->getPzPx();

	//move target forewords
	p1.Z-=pzpx[1]*2000;
	p1.X+=pzpx[0]*2000;
	camera->setTarget(p1
	+vector3df(0,players[0]->getAMSN()->getRotation().Z*38,0));

	camera->setPosition(
	vector3df(6-players[0]->getAMSN()->getRotation().Z*(f32)0.39,20,6));
}
```

Εικόνα 21: Κάμερα πρώτου προσώπου

Με την εντολή if(receiver->IsKeyDown(KEY_KEY_V)), όταν ο χρήστης πατήσει το πλήκτρο 'V' αρχίζει η εναλλαγή της κάμερας. Στο πρώτο κομμάτι του κώδικα γίνεται η εναλλαγή της κάμερας, και στο δεύτερο υπολογίζεται η νέα θέση της κάμερας και ο νέος στόχος της.

Επίσης υπάρχει η κάμερα πλοκής movieCamera η οποία ενεργοποιείται όταν διαδραματίζεται κάποια σκηνή με την εντολή:

```
if (MovieScene) smgr->setActiveCamera(movieCamera);
```

Η τοποθεσία της και ο στόχος έχουν φορτωθεί από το αρχείο επιπέδου XML κατά την εκκίνηση του παιχνιδιού

6.4.3 Έλεγχος χαρακτήρα

Αρχικά η είσοδος του χρήστη ελέγχεται από τον EventReceiver. Με τις παρακάτω εντολές η εφαρμογή γνωρίζει ποιο πλήκτρο είναι πατημένο:

```
bool EventReceiver::OnEvent(const SEvent& event)
{
    // Remember whether each key is down or up
    if (event.EventType == irr::EET_KEY_INPUT_EVENT)
        KeyIsDown[event.KeyInput.Key] = event.KeyInput.PressedDown;
```

Τώρα ελέγχουμε την είσοδο του χρήστη μέσα στον βρόγχο παιχνιδιού και αναλόγως επεξεργαζόμαστε τον κόμβο σκηνής του κεντρικού χαρακτήρα:

```
if(receiver->IsKeyDown(KEY_KEY_W)){
        //move forwards
        players[0]->moveForward();
        //set animation
        players[0]->setAnimation(EMAT_RUN);
}
```

Και στην κλάση Player:

```
void Player::moveForward(){
	vector3df v = SN->getPosition();
	v.Z -= px*300*TimeMULT*speed;
	v.X += pz*300*TimeMULT*speed;
	SN->setPosition(v);
}

void Player::setAnimation(EMD2_ANIMATION_TYPE type){
	AMSN->setMD2Animation(type);
	gun->setMD2Animation(type);
}
```

Εικόνα 22: Κίνηση προς τα εμπρός

Για την κίνηση του χαρακτήρα, πολλαπλασιάζουμε τις συνιστώσες της θέσης του κόμβου, με το ημίτονο και το συνημίτονο της περιστροφής και με αρνητικό ή θετικό πρόσημο, αναλόγως προς τα ποια κατεύθυνση θέλουμε να κινηθεί.

Για την αλλαγή animation του χαρακτήρα, χρησιμοποιούμε τους έτοιμους τύπους animation που προσφέρουν τα τρισδιάστατα πλέγματα md2 που χρησιμοποιούμε, σε συνδυασμό με τους τύπους εντολών που έχει η Irrlicht.

6.4.4 Συγκρούσεις και βαρύτητα

Για τις συγκρούσεις και τη βαρύτητα αξιοποιήσαμε τις βασικές δυνατότητες φυσικής που προσφέρει η Irrlicht.

```
//Add collision to all players
for(u32 i=0;i<(playercount);i++){
	//create metaSelector and add mapSelector to it
	IMetaTriangleSelector* metaSelector=smgr->createMetaTriangleSelector();
	metaSelector->addTriangleSelector(mapSelector);
	//add other players to metaselector
	for(u32 j=0;j<(playercount);j++){
		if (j!=i){
			ITriangleSelector* en=
			smgr->createTriangleSelectorFromBoundingBox(players[j]->getAMSN());
			players[j]->getAMSN()->setTriangleSelector(en);
			metaSelector->addTriangleSelector(en);
```

```
                    en->drop();
                }
        }
        //Find mesh's ellipsoidRadius for collision
        const aabbox3d<f32>& box = players[i]->getAMSN()->getBoundingBox();
        vector3df radius = box.MaxEdge - box.getCenter();//radius
        radius.operator *=(pscale);             //scale of mesh
        vector3df gravity=vector3df(0,-8.0f,0);       //gravity

        if(metaSelector){
                ISceneNodeAnimatorCollisionResponse* anim;
                anim = smgr->createCollisionResponseAnimator(
                        metaSelector,
                        players[i]->getSN(),
                        radius,             //ellipsoidRadius
                        gravity);           //gravity

                metaSelector->drop();
                players[i]->getSN()->addAnimator(anim); //add collision management animator
                anim->drop();
        }
}
```

Για κάθε χαρακτήρα, αρχικά δημιουργούμε έναν ImetaTriangleSelector, στον οποίο τοποθετούμε κατά κάποιο τρόπο το σύνολο των πλεγμάτων, με τα οποία θα υπάρχει έλεγχος σύγκρουσης. Στη συνέχεια δημιουργούμε διανύσματα βαρύτητας και ακτίνας έλλειψης για το πλέγμα στο οποίο θέλουμε να προσθέσουμε νόμους της φυσικής. Και κατόπιν με τις παρακάτω εντολές, ολοκληρώνουμε τη διαδικασία.

```
ISceneNodeAnimatorCollisionResponse* anim = smgr->createCollisionResponseAnimator(
                metaSelector,
                players[i]->getSN(),
                radius,             //ellipsoidRadius
                gravity);           //gravity

players[i]->getSN()->addAnimator(anim); //add collision management animator
```

6.4.5 Σωματιδιακά τεχνάσματα

Σωματίδιο (particle) στα Γραφικά ονομάζουμε μια «οντότητα» που χαρακτηρίζεται από διάφορες παραμέτρους, κατ' ελάχιστον μια θέση στο χώρο, ένα χρόνο ζωής και ένα διάνυσμα ταχύτητας. Σε ένα σωματίδιο μπορούμε να αποδώσουμε φυσικές παραμέτρους όπως βαρύτητα, επιβράδυνση, μέγεθος, διαφάνεια. Επιπλέον μπορούμε να απεικονίσουμε μια υφή σε αυτό και να το φωτίσουμε με τα φώτα της σκηνής.

Αν θεωρήσουμε έναν πεπερασμένο αριθμό τέτοιων σωματιδίων και μια πηγή που τα εκπέμπει τότε έχουμε ένα σύστημα σωματιδίων (particle system). Ένα σύστημα σωματιδίων μπορεί να χρησιμοποιηθεί για να εξομοιωθούν διάφορες φυσικές διεργασίες όπως εκρήξεις, καπνός, φωτιά, νερό κλπ και η χρήση τους είναι ιδιαίτερα εμφανής στα παιχνίδια τελευταίας γενιάς.

Εικόνα 23: Σωματιδιακά τέχνασμα, για νερό στο συντριβάνι

Στο παιχνίδι μας, έχουμε εφέ καπνού, νερού και φωτιάς που κάνουν χρήση σωματιδίων. Παρακάτω παραθέτουμε κομμάτι κώδικα για τα σωματιδιακά τεχνάσματα καπνού, όταν υπάρχει σύγκρουση της βολής του όπλου κάποιου χαρακτήρα με κάποιο αντικείμενο.

```
// create smoke particle system
scene::IParticleSystemSceneNode* pas = 0;
pas = smgr->addParticleSystemSceneNode(false, 0, -1, Impacts[i].pos);


scene::IParticleEmitter* em = pas->createBoxEmitter(
        aabbox3d<f32>(-5,-5,-5,5,5,5),
        Impacts[i].outVector, 20,40, SColor(0,255,255,255),SColor(0,255,255,255),
        1200,1600, 20);

pas->setEmitter(em);
em->drop();


pas->setMaterialFlag(video::EMF_LIGHTING, false);
pas->setMaterialTexture(0, driver->getTexture("media/smoke.bmp"));
pas->setMaterialType(video::EMT_TRANSPARENT_VERTEX_ALPHA);

scene::ISceneNodeAnimator* anim = smgr->createDeleteAnimator(2000);
pas->addAnimator(anim);
anim->drop();
```

Σημαντικό ρόλο στα συστήματα σωματιδιακών τεχνασμάτων είναι ο εκπομπός (emitter). Στο συγκεκριμένο παράδειγμα δημιουργούμε δικό μας εκπομπό με τη συνάρτηση **scene::IParticleEmitter* em = pas->createBoxEmitter(..)**. Επίσης σημαντικές για το αποτέλεσμα είναι οι υφές. Παραπάνω κάνουμε χρήση διάφανης υφής, με απενεργοποιημένο τον φωτισμό για πιο "σταθερό" οπτικά αποτέλεσμα.

6.4.5 Εχθροί

Εικόνα 24: Δύο εχθροί

Ο έλεγχος των εχθρών, γίνεται αποκλειστικά από το σύστημα. Αρχικά, βρίσκονται σε κατάσταση αδράνειας, σε κάποιο σημείο στον χάρτη. Όταν ο χαρακτήρας του χρήστη έρθει σε κοντινή απόσταση, ο εχθρός τον εντοπίζει. Η οπτική εμβέλεια κάθε εχθρού, είναι εκατό μέτρα. Η απόσταση και η γωνία στόχευσης ανάμεσα στον εχθρό και τον χρήστη, υπολογίζεται σε πραγματικό χρόνο, δηλαδή πριν από κάθε απεικονιζόμενο καρέ.

```
// now set rotation coordinates beetween 0 and 360
f32 rot= r.Y;
if (rot>=0)         rot=fmod(rot,360.00f);
else if (rot<0)     rot=360+fmod(rot,360.00f);

// Get enemy's and user's-player positions
vector3df pos2=players[0]->getSN()->getPosition();
vector3df pos1=p->getSN()->getPosition();

//XZ AXIS//Calc distance
f32 Distance= sqrt(pow(pos2.X-pos1.X,2)+pow(pos2.Z-pos1.Z,2));
//Calc distance angle (fi) with asin and acon
f32 as2=asin((pos2.Z-pos1.Z)/Distance)* 180.00f / PI;
f32 ac2=acos((pos2.X-pos1.X)/Distance)* 180.00f / PI;

f32 fi;
if(as2>=0)          fi= ac2;
else                fi= (360.00f -ac2);
//Align fi with rot
fi=fmod((360.00f-fi),360.00f);

//Y AXIS//Calc distance
Distance= sqrt(pow(Distance ,2)+pow(pos2.Y-pos1.Y,2));
//Calc distance angle (fi2) with asin
float fi2=(asin((pos2.Y-pos1.Y)/Distance)* 180.00f / PI);
float prot=  p->getAMSN()->getRotation().Z;
```

Στη συνέχεια το σύστημα κινεί τον αντίπαλο προς τα εμπρός και ταυτόχρονα τον περιστρέφει προς την κατεύθυνση του στόχου του. Κώδικας για περιστροφή προς τα αριστερά:

```
// rotate around Y axis
if (abs(rot-fi)>(30*TimeMULT) && rot>fi && (rot-fi<180))
        p->getSN()->setRotation(r+ vector3df(0,-30*TimeMULT,0));//turn left
```

Όταν ο εχθρός προσεγγίσει εμβέλεια βολής, που είναι 80 μέτρα, σταματάει την κίνησή του προς τον στόχο και στοχεύει. Όταν επιτευχθεί η στόχευση πυροβολεί.

```
//Checked already that rotation around Y axis is set
//if player within shoot distance==800 and if rotation around Z axis is set and not shooting
 if( (!p->getStopAnim() || p->getAnimation()==EMAT_POINT )
&& (Distance<800) && rotZok && !justchase)
        p->shoot(mapSelector,&Impacts,&AllBullets);
```

Ανάμεσα στον εχθρό και στον χαρακτήρα του χρήστη υπάρχει ένα αόρατο ευθύγραμμο τμήμα που τους συνδέει. Όταν στο ευθύγραμμο τμήμα παρεμβληθεί κάποιο αντικείμενο, όπως πχ. ένας τοίχος, τότε ο εχθρός χάνει την οπτική επαφή με τον στόχο του. Ο έλεγχος γίνεται με την παρακάτω γραμμή κώδικα:

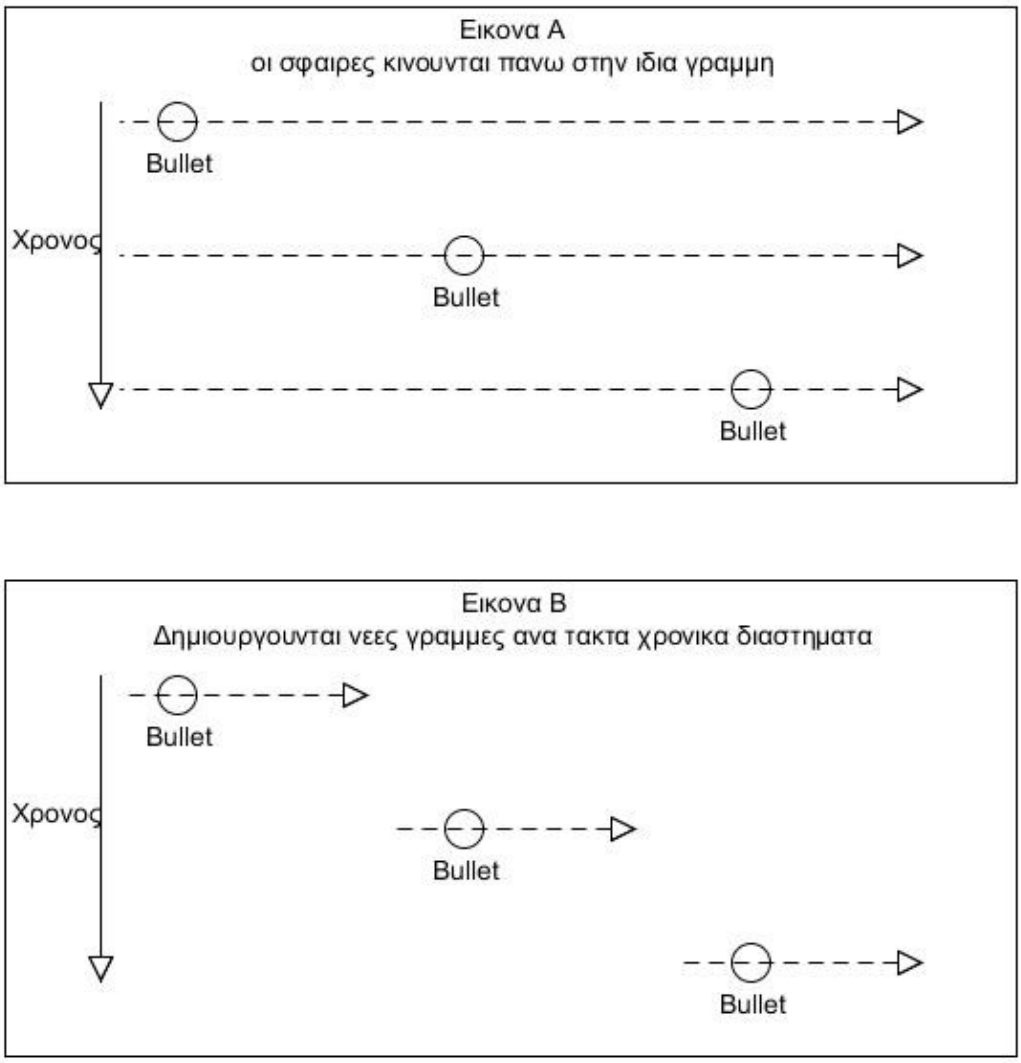
```
if (smgr->getSceneCollisionManager()->getCollisionPoint(line, mapSelector, targetEND, triangle))
// not visible target, because line collides with map!
```

6.4.6 Κίνηση σφαιρών

Η κίνηση των σφαιρών είναι ένα πιο περίπλοκο κομμάτι της υλοποίησης. Κατά τον πυροβολισμό πρέπει να συμβαίνουν δύο πράγματα, να ξεκινάει μια σφαίρα από το όπλο και να κατευθύνεται προς τη σωστή κατεύθυνση, και δεύτερον, στη πορεία της σφαίρας, να ανιχνεύονται πιθανές συγκρούσεις.

Για την ανίχνευση συγκρούσεων η Irrlicht, προσφέρει μονάχα τη δυνατότητα ανίχνευσης σύγκρουσης γραμμής με τρίγωνο, δηλαδή με πολύγωνο κάποιου πλέγματος. Όμως σε μια βολή, δεν μπορεί να ανιχνευτεί πιθανή σύγκρουση με μια μοναδική γραμμή, καθότι η σφαίρα κινείται σε πραγματικό χρόνο και επίσης δεν ξέρουμε πού θα σταματήσει, αφού υπάρχουν κινούμενα αντικείμενα στο χώρο. Σε ένα στατικό περιβάλλον, αυτό δεν αποτελεί πρόβλημα, αφού γνωρίζουμε από την αρχή, πού και πότε θα καταλήξει μια σφαίρα.

Για να λύσουμε το πρόβλημα και να ανιχνεύονται συγκρούσεις, σε πραγματικό χρόνο σε περιβάλλον με κίνηση, δημιουργήσαμε ίσα ευθύγραμμα τμήματα, που διανύουν την πορεία της σφαίρας με την ίδια ταχύτητα, με μέγεθος σχεδόν ίδιο με τη σφαίρα. Όταν κάποιο ευθύγραμμο τμήμα ανιχνεύσει σύγκρουση, σταματάει η πορεία της σφαίρας και η παραγωγή ευθυγράμμων τμημάτων και δημιουργείται σωματιδιακό τέχνασμα καπνού σε αυτό το σημείο.



Εικόνα 25: Ανίχνευση σύγκρουσης σε δυναμικό περιβάλλον, με τη λύση στην εικόνα Β

Όταν κάποιος χαρακτήρας πυροβολήσει, δημιουργείται μια δομή τύπου σφαίρας, με μεταβλητές για τον χρόνο, την κατεύθυνση, το σημείο εκκίνησης και τέλους.

```
struct Bullets{
        u32 time;
        vector3df direction,start,end;
};
```

Για την κίνηση των σφαιρών, υπάρχει η μέθοδος **moveBullets(),** η οποία υπολογίζει την ταχύτητα, το νέο τέλος και το μήκος του κάθε ευθύγραμμου τμήματος:

```
void System::moveBullets(){
for (s32 i=0; i<(s32)AllBullets.size(); ++i)
        if (AllBullets[i].time > core->getDevice()->getTimer()->getTime()){
                vector3df end=AllBullets[i].start + AllBullets[i].direction*1000.0f*TimeMULT;
                const f32 speed = 0.6f*TimeMULT*1000.0f;
                f32 length =end.getDistanceFrom(AllBullets[i].start);
                u32 time = (u32)(length / speed);
```

6.4.7 Γραφική διεπαφή χρήστη εντός παιχνιδιού

Η γραφική διεπαφή χρήστη εντός παιχνιδιού, ελέγχεται και αυτή μέσα στον βρόγχο παιχνιδιού από την οντότητα **system.** Ο έλεγχος της γραφικής διεπαφής, δίδεται και πάλι από τον πυρήνα του παιχνιδιού, με την παρακάτω εντολή:

IGUIEnvironment* gui=core->getDevice()->getGUIEnvironment();

Ένα σημαντικό κομμάτι της γραφικής διεπαφής χρήστη εντός παιχνιδιού είναι τα στοιχεία που εμφανίζονται σταθερά στην εικόνα, και δίνουν στον χρήστη τις πιο απαραίτητες πληροφορίες (HUD – Head Up Display). Στο παιχνίδι μας, τέτοια στοιχεία είναι οι πόντοι ζωής, οι διαθέσιμες σφαίρες, τα εικονίδια και ο χρόνος των ειδικών δυνάμεων και το ραντάρ όταν είναι ενεργοποιημένο. Ένα χαρακτηριστικό απόσπασμα κώδικα:

```
//gui
font =gui->getFont("media/font/guiright/GUIRIGHT.xml");
//add health points
health=gui->addStaticText(L"100",rect<s32>(Width - 96,15,Width -58,43),false,true,0,-1,false);
health->setOverrideFont(font);
health->setOverrideColor(SColor (255, 255,0, 0));
//add health icon
gui->addImage(driver->getTexture("media/health.png"),position2d<s32>(Width-53,5));
```

Εικόνα 26: Head up display

Επιπλέον στοιχεία της διεπαφής είναι: Σύντομα μηνύματα γεγονότων στο πάνω αριστερό κομμάτι της οθόνης. Κατά τη διάρκεια του παιχνιδιού, ο χρήστης μπορεί να ενημερώνεται για τη κατάσταση της αποστολής, με πάτημα ενός πλήκτρου. Οι εχθροί έχουν μια πινακίδα πάνω από το κεφάλι τους που αναγράφει το όνομά τους και τους πόντους ζωής τους και τέλος εμφανίζονται υπότιτλοι κατά την διάρκεια συνομιλίας.

6.4.8 Προσαρμογή ταχύτητας παιχνιδιού

Το παιχνίδι μας, πρέπει να έχει την ίδια ταχύτητα ανεξαρτήτως υπολογιστή. Όταν χρησιμοποιείται ένας αργός υπολογιστής, η ταχύτητα των χαρακτήρων θα ήταν αργή, ενώ χρησιμοποιώντας έναν γρήγορο θα ήταν γρήγορη. Γι' αυτό το σκοπό, στον

βρόγχο κάθε παιχνιδιού υπολογίζονται τα καρέ ανά δευτερόλεπτο, και με επιπλέον υπολογισμούς εξομαλύνεται η ταχύτητα του παιχνιδιού.

Στο παιχνίδι μας, υπολογίζουμε σε κάθε επανάληψη του βρόγχου παιχνιδιού, έναν πολλαπλασιαστή ταχύτητας που παίρνει τιμές, ανάλογα με τα καρέ ανά δευτερόλεπτο. Αυτός ο πολλαπλασιαστής ταχύτητας, **TimeMULT,** πολλαπλασιάζει όλα τα στοιχεία του παιχνιδιού, που με το πέρασμα του χρόνου κινούνται.

Υπολογισμός του πολλαπλασιαστή ταχύτητας:

```
int System::play(){
    s32 now = 0;
    s32 last = 0;
    s32 current_deltaT=0;
    TimeMULT=1;
    sceneStartTime = core->getDevice()->getTimer()->getTime();
   //game loop
   while(core->getDevice()->run()){
         if (core->getDevice()->isWindowActive()){
                                        now = core->getDevice()->getTimer()->getTime();
                                        current_deltaT = now -last;
                                        elapsedTime=current_deltaT;
                                        TimeMULT= current_deltaT/1000.0f;
                                        //game code.....
                                        last = core->getDevice()->getTimer()->getTime();
         }
          else   core->getDevice()->yield();
     }
     return 1;
}
```

6.5 Χαρακτήρες

Οι χαρακτήρες του παιχνιδιού μας αντιπροσωπεύονται από την οντότητα Player. Αυτή η κλάση αποτελείται από ένα κενό κόμβο σκηνής για την τοποθέτηση και την μετακίνηση του χαρακτήρα, ένα κόμβο σκηνής τύπου τρισδιάστατου πλέγματος για την εμφάνιση του χαρακτήρα, ένα κόμβο σκηνής τύπου μονάδας τεχνητής νοημοσύνης irrai για την αλληλεπίδραση με τους περαστικούς και τα αυτοκίνητα, και τα υπόλοιπα στοιχεία της οντότητας.

Εικόνα 27: Ένας χαρακτήρας σε εσωτερικό χώρο

6.5.1 Εμφάνιση

Το τρισδιάστατο πλέγμα που αντιστοιχεί σε κάθε χαρακτήρα είναι τύπου MD2. Αυτός ο τύπος έχει χρησιμοποιηθεί σε παλιά εμπορικά παιχνίδια που έχουν γνωρίσει μεγάλη επιτυχία, όπως το Quake2 και το Half-Life. Τα χαρακτηριστικά του είναι, η ομαδοποίηση συγκεκριμένων καρέ κινήσεων,η εύκολη χρησιμοποίηση συγκεκριμένων όπλων και αντικειμένων, το μικρό μέγεθος αρχείου, το μικρό πλήθος πολυγώνων και οι μικρές σε κατανάλωση μνήμης υφές.

Εικόνα 28: Τρείς χαρακτήρες τύπου MD2

Την εμφάνιση των χαρακτήρων συμπληρώνουν η σκιά που δημιουργείται με την βοήθεια της Irrlicht, και τα ειδικά εφέ που προσθέτουμε όταν ο χαρακτήρας αποκτήσει κάποια ειδική δύναμη.

6.5.2 Κίνηση

Όπως προαναφέραμε τα πλέγματα τύπου MD2 έχουν ομαδοποιημένα συγκεκριμένα καρέ κινήσεων. Αυτό το εκμεταλλεύεται η Irrlicht και προσφέρει στον προγραμματιστή, συναρτήσεις που κάνουν πολύ εύκολη την εισαγωγή κίνησης σε αυτά τα πλέγματα. Συγκεκριμένα με μια γραμμή κώδικα μπορούμε να κάνουμε έναν χαρακτήρα να έχει κίνηση τρεξίματος:

```
AMSN->setMD2Animation(EMAT_RUN)
```

Όπου AMSN σημαίνει Animated Mesh Scene Node, και στη δικιά μας περίπτωση τύπου MD2. Άλλες χρήσιμες μεθόδους που χρησιμοποιούμε για την κίνηση είναι η αλλαγή της ταχύτητας των καρέ και η επανάληψη συγκεκριμένων καρέ.

```
AMSN->setAnimationSpeed(16);
AMSN->setFrameLoop(184,184);
AMSN->setLoopMode(false);
```

6.5.3 Πυροβολισμός

Ο πυροβολισμός εκ μέρους κάποιου χαρακτήρα, πραγματοποιείται με την μέθοδο shoot της κλάσης player, δηλαδή της κλάσης των χαρακτήρων:

void Player::shoot(ITriangleSelector*,array<SParticleImpact>*,array<Bullets>*)

Σε αυτή τη μέθοδο, υπολογίζουμε το σημείο εκκίνησης της βολής και την κατεύθυνσή της, αποθηκεύουμε τον χρόνο εκκίνησης και τέλος δίνουμε ρεαλιστική ταχύτητα στη σφαίρα.

```
f32 length = (end - start).getLength();
const f32 speed = 0.6f;
u32 time = (u32)(length / speed);
//bullets
f32 D= sqrt(pow(end.X -start.X,2)+pow(end.Y -start.Y,2)+pow(end.Z -start.Z,2));
Bullets b;
b.time = time + core->getDevice()->getTimer()->getTime();
b.start= start;
b.direction= vector3df((end.X-start.X)/D,(end.Y-start.Y)/D,(end.Z-start.Z)/D);
AllBullets->push_back(b);
```

Όταν ολοκληρωθεί αυτή η μέθοδος, υπάρχει μια νέα σφαίρα στο διάνυσμα σφαιρών **core::array<Bullets>* AllBullets,** και τώρα ο βρόγχος παιχνιδιού αναλαμβάνει την κίνηση της με τον τρόπο που έχουμε περιγράψει.

6.5.4 Ειδικά εφέ

Έχουμε υλοποιήσει δύο ειδικές δυνάμεις, που μπορεί να ενεργοποιήσει ο χαρακτήρας μας, παίρνοντας δύο στοιχεία, που βρίσκονται σε διάφορα σημεία του επιπέδου. Η πρώτη προσφέρει τριπλάσια ταχύτητα και η δεύτερη προσφέρει άφθαρτη πανοπλία, και οι δύο ενεργοποιούνται για μικρό χρονικό διάστημα. Όταν είναι ενεργοποιημένες είναι ενεργοποιημένα και δυο ειδικά εφέ πάνω στο χαρακτήρα, αντίστοιχα για την κάθε μια δύναμη.

Για την τριπλάσια ταχύτητα, εμφανίζονται δύο φωτεινές μπάλες που περιστρέφονται γύρω από τον χαρακτήρα.

Εικόνα 29: Και οι δύο ειδικές δυνάμεις ενεργοποιημένες

```
// billboard
billPU2A = smgr->addBillboardSceneNode(SN, core::dimension2d<f32>(30, 30));
billPU2A->setMaterialFlag(video::EMF_LIGHTING, false);
billPU2A->setMaterialType(video::EMT_TRANSPARENT_ADD_COLOR);
billPU2A->setMaterialTexture(0, driver->getTexture("media/particlered.bmp"));

// add fly circle animator to bill
anim = smgr->createFlyCircleAnimator(core::vector3df(0,20,0),20.0f, 0.003f,
core::vector3df(0.41f, 0.4f, 0.0f));
billPU2A->addAnimator(anim);
anim->drop();

// let the lights follow the model...
anim = new CSceneNodeAnimatorFollowBoundingBox( SN, core::vector3df(0,-8,0), 2000, 0 );
anim->drop();
```

Η περιστροφή γύρω από τον χαρακτήρα μας γίνεται με τις μεθόδους **createFlyCircleAnimator** και **CsceneNodeAnimatorFollowBoundingBox.**

Για την άφθαρτη πανοπλία, αλλάζει το υλικό του πλέγματος του χαρακτήρα μας, και γίνεται γυαλιστερό μοβ. Όταν παρέλθει ο χρόνος της δύναμης, επανέρχεται το προηγούμενο υλικό του χαρακτήρα, καθώς το έχουμε προηγουμένως αποθηκεύσει.

```
AMSN->getMaterial(0).setTexture(0,
core->getDevice()->getVideoDriver()->getTexture("media/spheremap2.jpg"));
AMSN->setMaterialType(video::EMT_SPHERE_MAP);
AMSN->setAutomaticCulling ( scene::EAC_BOX );
```

Για το γυαλιστερό αποτέλεσμα έχουμε χρησιμοποιήσει υλικό τύπου σφαίρας με την τύπο video::EMT_SPHERE_MAP και για επίσης Automatic Culling με τον τύπο scene::EAC_BOX.

6.6 Αλληλεπιδράσεις

Με τον όρο αλληλεπιδράσεις, χαρακτηρίζουμε όλα τα στοιχεία που, όταν ο χαρακτήρας του χρήστη έρθει σε επαφή και πατήσει το πλήκτρο δράσης, ενεργοποιούν κάποια αλληλεπίδραση. Γι' αυτό το λόγο δημιουργήσαμε την κλάση **Interactive,** η οποία είναι ο γονέας όλων των παρακάτω οντοτήτων.

```
class Interactive{
        protected:
                //..
                vector3df position;
        public:
                //..
                virtual s32 action()=0;
                bool isReachable(vector3df p);
};
```

Οι παρακάτω κληρονομούν σημαντικές ιδιότητες και μεθόδους όπως τις **virtual s32 action()=0** και **bool isReachable(vector3df p).**

6.6.1 Ειδικές δυνάμεις

Όπως έχουμε πει υπάρχουν δύο ειδικές δυνάμεις, που μπορεί να ενεργοποιήσει ο χαρακτήρας μας, παίρνοντας δύο στοιχεία, που βρίσκονται σε διάφορα σημεία του επιπέδου. Έτσι και τα αντικείμενα αυτά προσφέρουν αλληλεπίδραση. Το ένα αντικείμενο υλοποιείται με χρήση κινούμενων υφών. Αρχικά δημιουργούμε τις υφές και τις τοποθετούμε σε ένα διάνυσμα και στη συνέχεια τους δίνουμε κίνηση με τον **texture animator.**

ISceneNodeAnimator* glow = smgr->createTextureAnimator(textures, 150);

Εικόνα 30: Το πρώτο αντικείμενο, ειδικής δύναμης

Το δεύτερο αντικείμενο υλοποιήθηκε δημιουργώντας μια σφαίρα **addSphereSceneNode(..)** και προσθέτοντας διάφανο σκιαστή σε αυτή **setMaterialType(video::EMT_TRANSPARENT_ADD_COLOR)**. Επιπλέον, εφαρμόσαμε στη σφαίρα το ίδιο εφέ που εφαρμόζεται στο χαρακτήρα όταν πάρει την δύναμη της τριπλάσιας ταχύτητας, δηλαδή εμφανίζονται δύο φωτεινές μπάλες που περιστρέφονται γύρω από το αντικείμενο. Για την υλοποίηση αυτή χρησιμοποιούμε τον κώδικα που περιγράψαμε νωρίτερα.

Εικόνα 31: Το δεύτερο αντικείμενο, ειδικής δύναμης

6.6.2 Πόρτες

Άλλο ένα αντικείμενο που προσφέρει αλληλεπίδραση είναι οι πόρτες. Όταν ο χαρακτήρας πατήσει το κουμπί δράσης, η πόρτα ανοίγει. Για να συγκρούονται οι χαρακτήρες σε αυτή, προσθέσαμε δυνατότητα σύγκρουσης, με τις εξής γραμμές κώδικα:

```
ITriangleSelector* TS= smgr->createTriangleSelectorFromBoundingBox(anode);
anode->setTriangleSelector(TS);
mapSelector->addTriangleSelector(TS);
```

Το τρισδιάστατο πλέγμα των πορτών είναι τύπου DirectX mesh οπότε για να του δώσουμε κίνηση, όταν ο χρήστης πατήσει το πλήκτρο δράσης, στον κώδικα του γεγονότος προσθέσαμε απλές εντολές κίνησης:

```
node->setAnimationSpeed(10);
node->setFrameLoop(0,20);
node->setLoopMode(false);
```

6.6.3 Σκηνές

Τελευταίο στοιχείο που προσφέρει αλληλεπίδραση είναι οι σκηνές. Αντίθετα με τα προηγούμενα, οι κόμβοι σκηνών δεν είναι ορατοί και επίσης δεν προσφέρουν πάντα κάποια αλληλεπίδραση, πάρα μόνο όταν ο χρήστης έχει το "κλειδί" για την σκηνή. Όταν ξεκινάει το παιχνίδι, ο χρήστης κατέχει το κλειδί 0. Άρα μπορεί να ενεργοποιήσει μόνο την σκηνή με κλειδί=0. Κάθε φορά που ολοκληρώνεται ένα κομμάτι πλοκής του σεναρίου, ο χρήστης αποκτά νέο κλειδί.

Κώδικας για ενεργοποίηση κάποιας σκηνής:

```
for (u32 i=0; i<level->getSceneNum(); ++i){
        if (Scenes[i]->isReachable(players[0]->getSN()->getPosition())){
                if(Scenes[i]->myscene.key==nextKey){
                        MovieScene=true;
                        Scenes[i]->action();
                        Scenes[i]->setStartTime(core->getDevice()->getTimer()->getTime());
                        movieCamera->setPosition(Scenes[i]->myscene.camera_position);
                        movieCamera->setTarget(Scenes[i]->myscene.camera_target);
```

Εικόνα 32: Μια σκηνή διαλόγου

6.7 Περαστικοί και αυτοκίνητα

Για να κάνουμε το περιβάλλον πιο ρεαλιστικό και ζωντανό, προσθέσαμε σε αυτό περαστικούς και αυτοκίνητα, που περιφέρονται στον εικονικό κόσμο.

Εικόνα 33: Ένα αυτοκίνητο

Στην παραπάνω εικόνα το αυτοκίνητο έχει σταματήσει και περιμένει τον χαρακτήρα να φύγει, γιατί τον έχει εντοπίσει και αποφεύγει τη σύγκρουση.

Εικόνα 34: Ένας περαστικός

6.7.1 Διαδρομές και σημεία διαδρομών

Οι περαστικοί και τα αυτοκίνητα εκτελούν συγκεκριμένες διαδρομές. Για να το κάνουμε αυτό χρησιμοποιήσαμε εργαλείο (editor) της IrrAI. Αυτό μας έδωσε τη δυνατότητα να αποθηκεύσουμε διαδρομές και σημεία διαδρομών (way-points). Τα

σημείο διαδρομών έχουν συγκεκριμένες συνδέσεις με άλλα σημεία, με μονή ή διπλή κατεύθυνση. Είχαμε τη δυνατότητα να αποθηκεύσουμε σύνολα σημείων σε ομάδες. Έτσι έχουμε μια ομάδα σημείων για τα αυτοκίνητα και μια για τους περαστικούς.

Η επιλογή νέου σημείου και η μετακίνηση της μονάδας irrai γίνεται από την irrai, αφού πρώτα έχουμε ορίσει την μονάδα NPC (Non playing character) της irrai και για αυτή τη μονάδα καθορίσαμε το way-point group που θα ακολουθήσει.

Έτσι όταν ένας περαστικός που είναι τύπου **SCombatNPCDesc** βρίσκεται σε ένα σημείο διαδρομής, έχει συγκεκριμένες επιλογές πορείας προς κάποιο άλλο σημείο διαδρομής. Η επιλογή λαμβάνεται με τυχαίο τρόπο.

Κομμάτι του ορισμού μονάδας NPC για τον περαστικό:

```
//Create the NPC
SCombatNPCDesc npcDesc;

npcDesc.Range = 5.0f;
npcDesc.FovDimensions = core::vector3df(50,250,0);
npcDesc.MoveSpeed = 0.05f;

npcDesc.StartWaypointID = desc.StartWaypointID;
npcDesc.UserData = this;
npcDesc.WaypointGroupName = desc.WaypointGroupName;
//Create tha bounding box of the NPC
core::aabbox3d<f32> box = AMSN->getTransformedBoundingBox();
npcDesc.Scale = box.MaxEdge - box.MinEdge;
npcDesc.Scale= npcDesc.Scale.operator *=(8);        //scale
npcDesc.Offset = core::vector3df(0,(box.MaxEdge.Y - box.getCenter().Y)*8,0);
```

Για την εργασία μας δημιουργήσαμε δύο διαδρομές, μια για τους περαστικούς και μια για τα αυτοκίνητα. Τα αυτοκίνητα, κινούνται αποκλειστικά στη σωστή πλευρά του δρόμου, χωρίς να εκτελούν παραβιάσεις του αντίθετου ρεύματος. Οι περαστικοί περπατάνε αποκλειστικά στα πεζοδρόμια και στις διαβάσεις των δρόμων

Προσθήκη περαστικών στο επίπεδο:

```
Pedestrian* character;

core::array<s32> usedIDs;
SWaypointGroup* group = aimgr->getWaypointGroupFromName("Group 0");

for (s32 i = 0 ; i < numNPCs ; ++i) {
        Pedestrian::SCharacterDesc desc;
        character = NULL;
        desc.SceneManager = smgr;
        desc.AIManager = aimgr;
        desc.WaypointGroupName = "Group 0";

        if (usedIDs.size() >= group->Waypoints.size()) {
                printf("Used up all available car waypoints\n");
                break;
        }
        s32 randID = 0;
        do {
                randID = rand() % group->Waypoints.size(); // Pick a random ID
        } while (contains(usedIDs, randID));

        desc.StartWaypointID = randID;
        usedIDs.push_back(randID);
```

```
        //desc.StartWaypointID = -1; // start from a random waypoint
        addPedestrian(vector3df(0,10000,0),vector3df(0,0,0),desc);
        A->addEntity(Peds[i]->getAIEntity());
        Peds[i]->getAIEntity()->setEnemyGroup(B);
        Peds[i]->getAIEntity()->setAllyGroup(A);
}
```

6.7.2 Έλεγχος επικείμενων συγκρούσεων

Για να είναι πιο ρεαλιστική η κίνηση των περαστικών και των αυτοκινήτων, προσθέσαμε έλεγχο επικείμενων συγκρούσεων και αποφυγή αυτών.

Τα αυτοκίνητα και οι περαστικοί χωρίστηκαν σε δύο αντίπαλες ομάδες με τις εντολές **setEnemyGroup** και **setAllyGroup.** Κάθε NPC έχει κάποιο εύρος ορατότητας (field of view) και κάποιο κουτί που περιβάλλει την οντότητα (bounding box). Όταν βρεθεί το bounding box κάποιου εχθρού στο field of view της οντότητας αυτή σταματάει για κάποιο χρονικό διάστημα για αποφυγή της σύγκρουσης.

Κλήση μετά από γεγονός:

```
void characterEventCallback(E_NPC_EVENT_TYPE event, void* userData, void* eventData) {
        case ENET_ENEMY_IN_RANGE: {
                // Halt, let him pass..
                character->halt();
```

Ακινητοποιήση του χαρακτήρα:

```
void Pedestrian::halt() {

        if (!AIEntity) return;

        EnemyVisible = true;
        ((INPC*)AIEntity)->setStayPut(true);

        StationaryTime = 0;

}
```

Κλήση μετά από αλλαγή κατάστασης:

```
static void characterStateChangedCallback(E_NPC_STATE_TYPE state, void* userData) {
        Pedestrian* character = (Pedestrian*)userData;
        if (!character) return;

        switch (state) {
                case ENST_WAITING:
                        character->getAMSN()->setAnimationSpeed(0);
```

6.7.3 Αποφυγή deadlocks

Για την αποφυγή επικείμενων συγκρούσεων, οι οντότητες σταματάνε την κίνηση τους. Όμως αν μια οντότητα περιμένει μια άλλη και αυτή περιμένει εξίσου, είναι πιθανό να συμβεί deadlock αφού η μια θα περιμένει την άλλη ατέρμονα.

Για την επίλυση αυτού του προβλήματος, οι οντότητες περιμένουν για συγκεκριμένη ώρα και κατόπιν σχεδιάζουν σχέδιο αποφυγής. Όταν ένας περαστικός περιμένει για παραπάνω από 4000ms σχεδιάζει αποφυγή:

```
bool Pedestrian::update(s32 elapsedTime) {
        if (!AIEntity) return false;
```

```
        SN->setPosition(AIEntity->getAbsolutePosition());
        SN->setRotation(AIEntity->getNode()->getRotation());
        if (!EnemyVisible)  ((INPC*)AIEntity)->setStayPut(false);
        EnemyVisible = false;
        if (((INPC*)AIEntity)->isStayingPut())  StationaryTime += elapsedTime;
        else                                    StationaryTime = 0;
        if (StationaryTime > 4000) makeAvoidancePlan();
        return false;
}
```

Στο σχέδιο αποφυγής, η οντότητα επιλέγει με τυχαίο τρόπο, κάποιο way-point από τα γειτονικά για να κινηθεί. Αν δεν υπάρχει κάποιο διαθέσιμο, τότε συνεχίζει την πορεία και περνάει μέσα από την αντίπαλη οντότητα, καθώς δεν υπάρχει άλλη επιλογή και θα είχαμε deadlock.

```
void Pedestrian::makeAvoidancePlan() {
        if (!AIEntity) return;
        // Pick a waypoint to go to furthest from NPC to be avoided
        IWaypoint* currentWaypoint = ((INPC*)AIEntity)->getCurrentWaypoint();
        if (!currentWaypoint) return;
        const core::list<SNeighbour*> neighbours = currentWaypoint->getNeighbours();
        core::list<SNeighbour*>::ConstIterator iter = neighbours.begin();
        iter += rand()%neighbours.getSize();
        ((INPC*)AIEntity)->setDestination((*iter)->Waypoint);
}
```

6.7.4 Εύρεση συντομότερου μονοπατιού με αναζήτηση A star

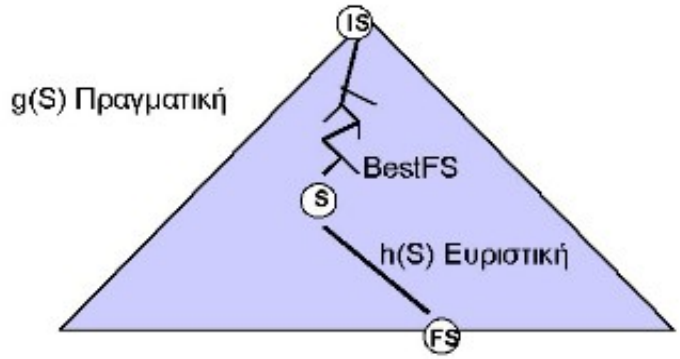
Στην επιστήμη των υπολογιστών, ο **A*** είναι ένας best-first search αλγόριθμος γραφήματος που βρίσκει το συντομότερο μονοπάτι από ένα δεδομένο αρχικό κόμβο σε κόμβο ενός στόχου (από έναν ή περισσότερους πιθανούς στόχους). Όπως όλοι οι αλγόριθμοι αναζήτησης με πληροφόρηση, η αναζήτηση γίνεται πρώτα στα δρομολόγια που φαίνεται να είναι πιο πιθανό να οδηγήσει την επίτευξη του στόχου. Η πληροφορία εξάγεται από την κατάσταση και η πληροφορημένη αναζήτηση (informed search[14]), κάνει χρήση ειδικής γνώσης για την επίλυση του προβλήματος. Στην απληροφόρητη αναζήτηση έχουμε περισσότερες επαναλήψεις άρα μεγαλύτερο κόστος.

Όπως προαναφέραμε η αναζήτηση **A*** ανήκει στην κατηγορία πρώτα στο καλύτερο (best-first search[14]), στην οποία έχουμε συνάρτηση αξιολόγησης (evaluation function) «καλύτερου». Πραγματοποιείται αξιολόγηση κάθε κόμβου n με την συνάρτηση αξιολόγησης f(n) και επεκτείνεται ο κόμβος με τη μικρότερη τιμή αξιολόγησης. Η υλοποίηση γίνεται με ουρά προτεραιότητας.

Μια αναζήτηση, η οποία ανήκει στις παραπάνω κατηγορίες, είναι η "Άπληστη Αναζήτηση Πρώτα στο Καλύτερο" (greedy BF search[14]). Η συνάρτηση αξιολόγησης f(n) είναι ίση με τη ευρετική (heuristic function) h(n), h(G)=0. Το εκτιμώμενο κόστος φθηνότερης διαδρομής από n σε στόχο, εξαρτάται μόνο από τον κόμβο n και το στόχο.

Ευριστικός μηχανισμός[18] είναι μια στρατηγική, βασισμένη στη γνώση για το συγκεκριμένο πρόβλημα, η οποία χρησιμοποιείται σαν βοήθεια για την γρήγορη επίλυσή του. Ο ευριστικός μηχανισμός υλοποιείται με ευριστική συνάρτηση.

Αυτό που διαφοροποιεί ένα A *, από έναν greedy best-first search είναι ότι λαμβάνει υπόψη επίσης την απόσταση που έχει ήδη ταξιδέψει. Το g (x) μέρος του heuristic είναι το κόστος από την αρχή, και όχι απλά το τοπικό κόστος από τον προηγουμένως επεκταμένο κόμβο.



Εικόνα 35: A διαφοροποίηση σε σχέση με BestFS[18]*

Ο A* στοχεύει στην ελαχιστοποίηση του ολικού εκτιμώμενου κόστους λύσης της αναζήτησης. Ο A* αξιολογεί τους κόμβους με την ακόλουθη συνάρτηση:
f(n) = g(n) + h(n), όπου g(n) είναι το πραγματικό κόστος μετάβασης στον κόμβο n και h(n), όπως είπαμε προηγουμένως, το εκτιμώμενο κόστος από τον κόμβο n στο στόχο[18].

Ο αλγόριθμος είναι πλήρης, αν ο παράγοντας διακλάδωσης είναι πεπερασμένος και βέλτιστος αν η ευρετική είναι αποδεκτή. Αποδεκτή είναι μια ευρετική, όταν σε κόμβο n δείχνει μικρότερο ή ίσο εκτιμώμενο κόστος από το πραγματικό κόστος της μετάβασης από τον n στο στόχο.

Η πολυπλοκότητα είναι όπως στον BestFS, δηλαδή $O(b^m)$, αλλά στην πράξη είναι μικρότερη. Ο A* είναι ένας από τους περισσότερο χρησιμοποιούμενους αλγορίθμους στην αναζήτηση με πληροφόρηση. Ο αλγόριθμος βελτιώνει την απόδοσή του με ευριστικές (heuristics) που είναι και συνεπείς και αποδεκτές, ενώ επίσης μπορεί να χρησιμοποιηθεί και κλειστό σύνολο για την αποφυγή επαναλήψεων. Η πολυπλοκότητα του χρόνου της A * εξαρτάται από την ευριστική. Στη χειρότερη περίπτωση, ο αριθμός των κόμβων που επεκτάθηκε είναι εκθετικός στη διάρκεια της λύσης, αλλά είναι πολυωνυμικός όταν ο χώρος αναζήτησης είναι δέντρο άρα υπάρχει μόνο μια κατάσταση στόχου, και η heuristic ανταποκρίνεται στον ακόλουθο όρο:

$$| h(x) - h^*(x) | = O(\log h^*(x))$$

Η ενότητα IrrAI που χρησιμοποιούμε μας δίνει έτοιμη την αναζήτηση A* Star πάνω σε γράφημα, με κόμβους σημεία διαδρόμων τύπου IrrAI. Παραθέτουμε την αναζήτηση A* παρμένη από τον πηγαίο κώδικα της IrrAI. Μετά το πέρας της υλοποίησης παρατηρήσαμε ότι ο αλγόριθμος χρειάζεται κάποιες διορθώσεις, τις οποίες παραθέτουμε με έντονη γραφή. Θεωρούμε ως ευριστική εκτίμηση της απόστασης την ευκλείδεια απόσταση.

```
long heuristicEstimateOfDistance(IWaypoint* A, IWaypoint* B){
        return (long)A->waypoint->getPosition().getDistanceFrom(B->waypoint->getPosition());
}

bool CAStarPathFinder::findPath(IWaypoint* startNode, IWaypoint* goalNode,
core::array<IWaypoint*>& path) {

core::array<SSearchNode*> lstOpen, lstClosed;
SAStarSearchNode* sNode = NULL;
// add start point to open list
lstOpen.push_back(new SAStarSearchNode(NULL, startNode));
```

```
startNode->CostG = 0;                    // Distance from start along optimal path.
startNode->Costh = heuristicEstimateOfDistance(startNode,goalNode); //heuristic culculation
startNode->CostF = startNode->Costh ;  //Estimated total distance from start to goal through y.

bool pFound = false;
while ((lstOpen.size() > 0) && !pFound) {
        // get first waypoint from open list
        sNode = (SAStarSearchNode*)lstOpen[0];
        if (sNode->Waypoint == goalNode) {
                pFound = getPath(sNode, path);
                break;
        }
        lstClosed.push_front(sNode);// move waypoint from open list to closed list
        lstOpen.erase(0);
        // add all now reachable waypoints to open list
        core::list<SNeighbour*>::ConstIterator iter = sNode->Waypoint->getNeighbours().begin();
        while (iter != sNode->Waypoint->getNeighbours().end()) {
                SAStarSearchNode* newSNode = new SAStarSearchNode(sNode, (*iter)-
>Waypoint);//foreach Y in neighbor_nodes of X
long distance=(long)sNode->waypoint->getPosition().getDistanceFrom(newSNode->waypoint-
>getPosition()); //(X,Y) distance
newSNode->CostG = sNode->CostG + distance; //tentative_g_score
newSNode->CostH = heuristicEstimateOfDistance(newSNode,goalNode);//(Y,goal) distance
newSNode->CostF = newSNode->CostG + newSNode->CostH;// F COST
                // add to open list (position in array depends on costF)
                s32 pInsert = 0, pExist = -1;
                // is new in closed list !?!?!
                for (u32 r = 0; r < lstClosed.size(); ++r) {
                        if (lstClosed[r]->Waypoint == newSNode->Waypoint) {
                                pInsert = -1;
                                break;
                        }
                }
                if (pInsert >= 0) {
                        // check for position and existance in open list
                        for (u32 r = 0; r < lstOpen.size(); ++r) {
                                // way point already in open list !!!
                                if (lstOpen[r]->Waypoint == newSNode->Waypoint) pExist = r;
                                // position for insert
                                if (((SAStarSearchNode*)lstOpen[r])->CostF < newSNode->CostF)
pInsert = r + 1;
                        }
                        if (pExist >= 0) {
                                // erase existing in open list if it has more costF !!!
                                if (((SAStarSearchNode*)lstOpen[pExist])->CostF > newSNode-
>CostF) {
                                        lstOpen.erase(pExist);
                                } else {
                                        pInsert = -1;// do not add to open list
                                }
                        }
                        if (pInsert >= 0) {// add as new to open list
                                // is new waypoint = destination, then path was found !!!
                                if (newSNode->Waypoint == goalNode) {
                                        pFound = getPath(newSNode, path);
                                        lstClosed.push_front(newSNode);
                                        break;
                                } else {
                                        if ((u32)pInsert >= lstOpen.size()) {
                                                lstOpen.push_back(newSNode);
```

```
                                        } else {
                                                lstOpen.insert(newSNode, pInsert);
                                        }
                                }
                        }
                }++iter;
        }
}
deleteSearchNodes(lstOpen);
deleteSearchNodes(lstClosed);
return pFound;
}
```

Εμείς χρησιμοποιήσαμε αυτόν τον αλγόριθμο, για να βρούμε το συντομότερο μονοπάτι προς κάποιο σημείο, για τους περαστικούς. Έτσι αρχικά βρίσκουμε το μονοπάτι και το αποθηκεύουμε στο διάνυσμα **path**. Επίσης αποθηκεύουμε το μέγεθος του μονοπατιού στη μεταβλητή **sizeOfPath** και στην **iter.**

```
void Pedestrian::goToPointWithAStar(s32 point){
        dontmove=false;
        IPathFinder* Astar=AIManager->createAStarPathFinder();

        IWaypoint* currentWaypoint = ((INPC*)AIEntity)->getCurrentWaypoint();
        IWaypoint* goal = ((INPC*)AIEntity)->getWaypointGroup()->Waypoints[point];

        followPath=AStar->findPath(currentWaypoint,goal,path);//retuns boolean

        if (!followPath) printf("Astar failed!!!!!!!!!!!!\n");
        sizeOfPath=path.size();
        iter=sizeOfPath;
}
```

Κατόπιν όταν ο περαστικός φτάνει στο επόμενο σημείο του μονοπατιού, του δίνουμε εντολή να προχωρήσει προς το επόμενο σημείο που έχουμε ήδη βρει, και επίσης μειώνουμε κατά ένα τη μεταβλητή **iter** η οποία αρχικά έχει τιμή ίση με το μέγεθος του μονοπατιού. Όταν ο **iter** φτάσει στο 0, ο περαστικός βρίσκεται στο στόχο του.

```
static void characterEventCallback(E_NPC_EVENT_TYPE event, void* userData, void* eventData) {
Pedestrian* character = (Pedestrian*)userData;
if (!character) return;

switch (event) {
case ENET_AT_GOAL: {
        INPC* npc = (INPC*)character->getAIEntity();
        if(character->followPath ){
                npc->setDestination(character->path[character->sizeOfPath-character->iter]);
                character->iter--;
                if(character->iter==0){ // Goal is achived!!
                        character->followPath=false; // End of path following
                }
        }else
                // Pick a random waypoint to go to
                npc->setDestination(rand()%(npc->getWaypointGroup()->Waypoints.size()-1));
        break;
}
```

Κεφάλαιο 7

Αξιολόγηση

7.1 Μέθοδος αξιολόγησης

Για την αξιολόγηση της εργασίας, έγινε χρήση ερωτηματολογίων χρηστικότητας QUIS (Questionnaire for User Interaction Satisfaction[13]) του πανεπιστημίου Maryland.

Τα ερωτηματολόγια συμπληρώθηκαν από τους χρήστες, που έπαιξαν το παιχνίδι για είκοσι περίπου λεπτά, 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 (πέντε άνδρες και πέντε γυναίκες), με ηλικίες από 23 έως 45 χρόνων με διαφορετική εξοικείωση σε εφαρμογές υπολογιστών και παιχνίδια.

Το ερωτηματολόγιο QUIS περιλαμβάνει 39 ερωτήσεις, χωρισμένες σε επτά ενότητες, και οι απαντήσεις δίνονται σε κλίμακα από ένα (1) έως εννιά (9) ή δεν γνωρίζω/ δεν απαντώ (NA). Παρακάτω, παραθέτουμε το ερωτηματολόγιο:

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 NA

1.5. Γενική αντίδραση του Συστήματος :

άκαμπτο ευέλικτο
1 2 3 4 5 6 7 8 9 NA

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 NA

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

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

2.2. Αλληλουχία οθονών :

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

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

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

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 NA

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 : ικανοποιητική πολύ αργή
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 :

_____

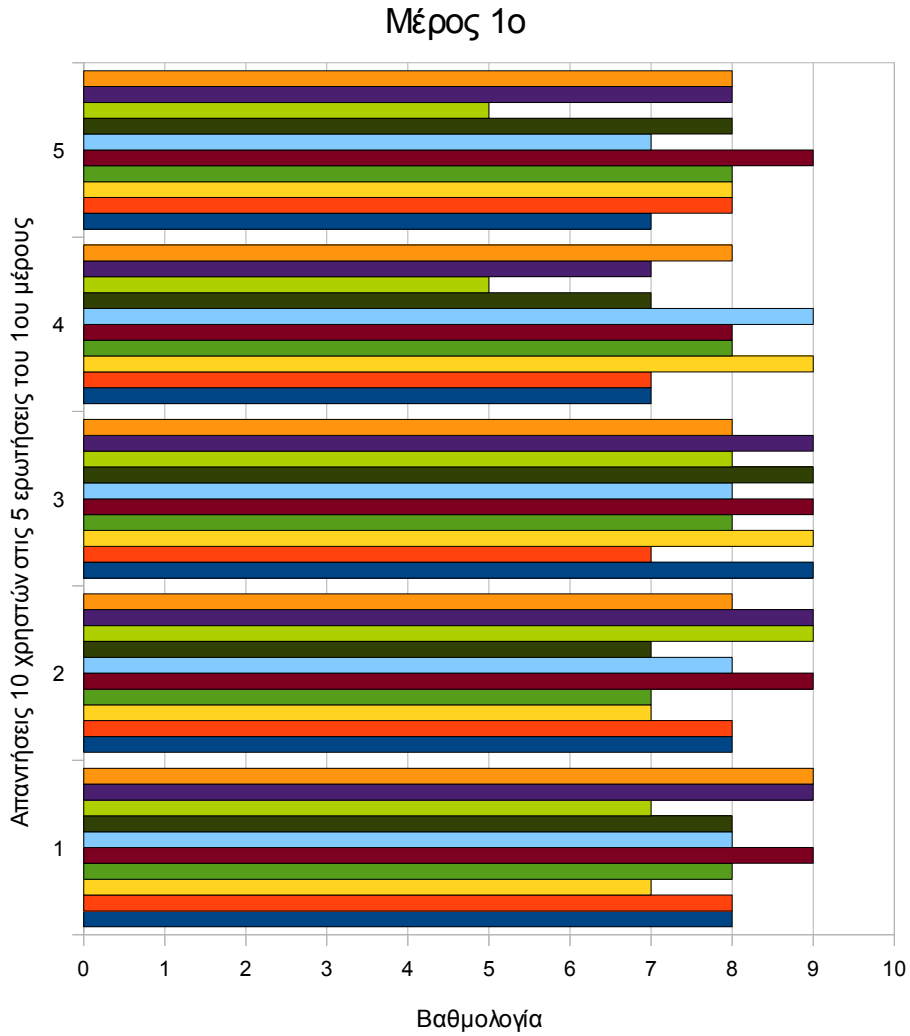
_____

7.2 Αποτελέσματα ερωτηματολογίων

Τα αποτελέσματα που προέκυψαν, δίνουν στοιχεία χρηστικότητας και ικανοποίησης για την ίδια την εφαρμογή και παράλληλα βοηθούν, στο να εξαχθούν συμπεράσματα για την μελλοντική βελτίωση και εργασία πάνω στην εφαρμογής μας.

Παραθέτουμε τα αποτελέσματα και τα σχόλια των χρηστών. Τα παρακάτω διαγράμματα έχουν οριζόντιο άξονα την βαθμολογία, και στον κάθετο άξονα ομαδοποιημένες τις απαντήσεις των δέκα χρηστών, ανά ερώτηση. Επίσης υπολογίζουμε τον μέσο όρο βαθμολογίας για κάθε ερώτηση, ώστε να εξετάσουμε καλύτερα τα αποτελέσματα μας.

Μέρος πρώτο – Γενική εντύπωση του χρήστη:

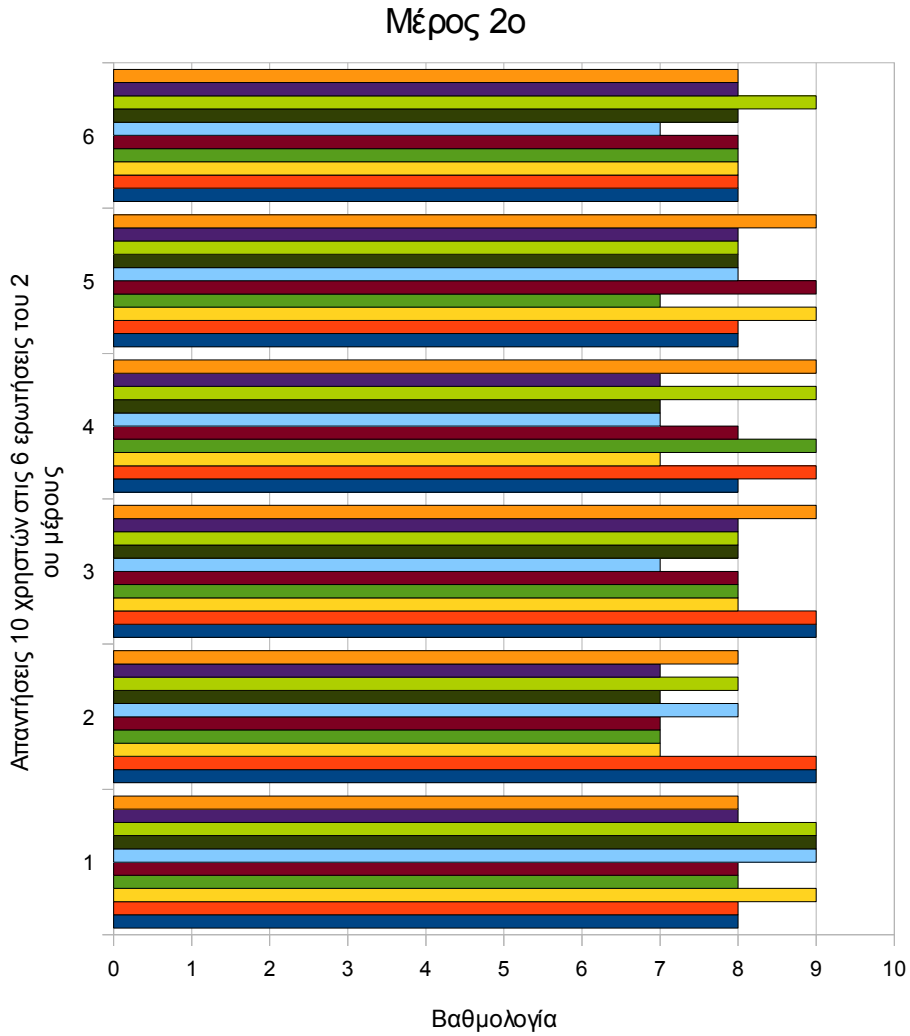


Εικόνα 36: Μέρος πρώτο – Γενική εντύπωση του χρήστη

Σχόλια χρηστών:
"Ευχάριστο παιχνίδι"
"Πολύ καλή γενική εντύπωση! Μόνο που έχασα, άρα είναι λιγάκι δύσκολο."

Μέσοι όροι απαντήσεων στο πρώτο μέρος:
Ερώτηση 1: 8,1
Ερώτηση 2: 8
Ερώτηση 3: 8,4
Ερώτηση 4: 7,5
Ερώτηση 5: 7,6

Μέρος δεύτερο – Οθόνη:

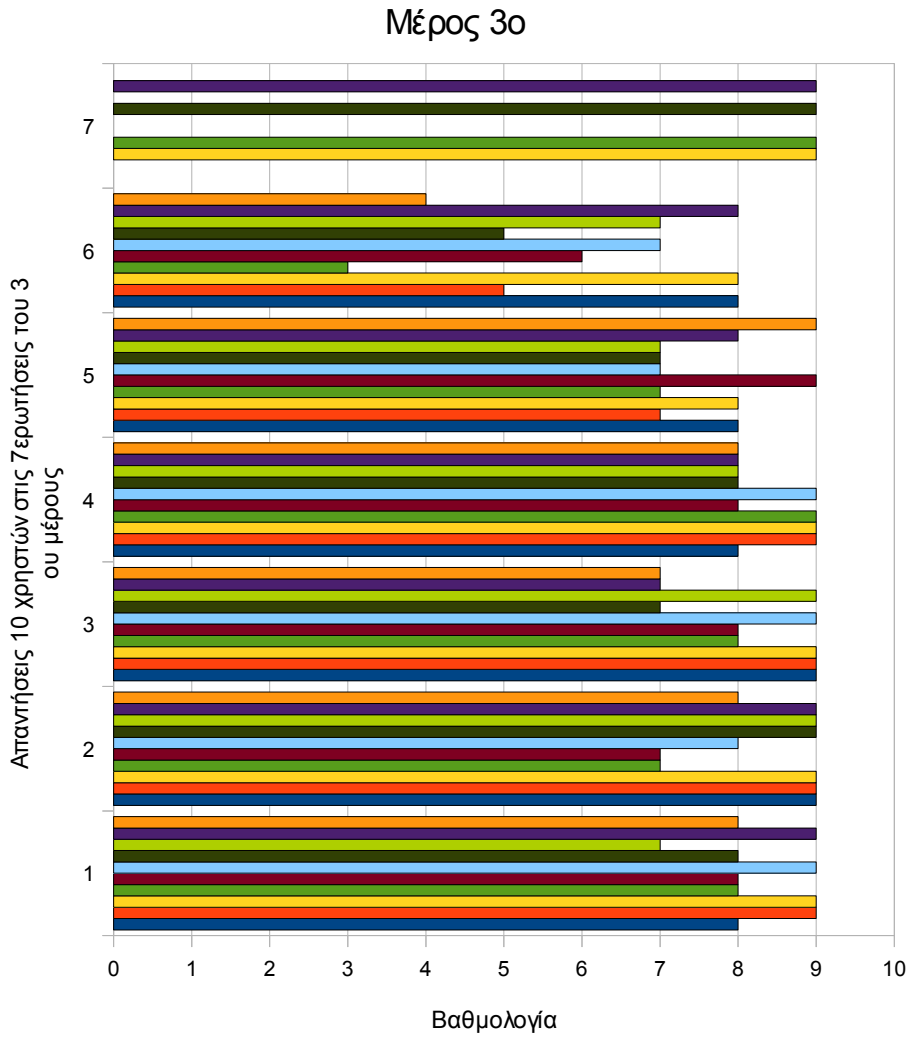


Εικόνα 37: Μέρος δεύτερο – Οθόνη

Σχόλια χρηστών:
"Οι πληροφορίες στην οθόνη είναι πολύ οργανωμένες"
"Ίσως να έπρεπε να υπάρχουν παραπάνω πληροφορίες μερικές φορές"

Μέσοι όροι απαντήσεων στο δεύτερο μέρος:
Ερώτηση 1: 8,4
Ερώτηση 2: 7,7
Ερώτηση 3: 8,2
Ερώτηση 4: 8
Ερώτηση 5: 8,2
Ερώτηση 6: 8

<u>Μέρος τρίτο – Ορολογία και επικοινωνία με το σύστημα:</u>

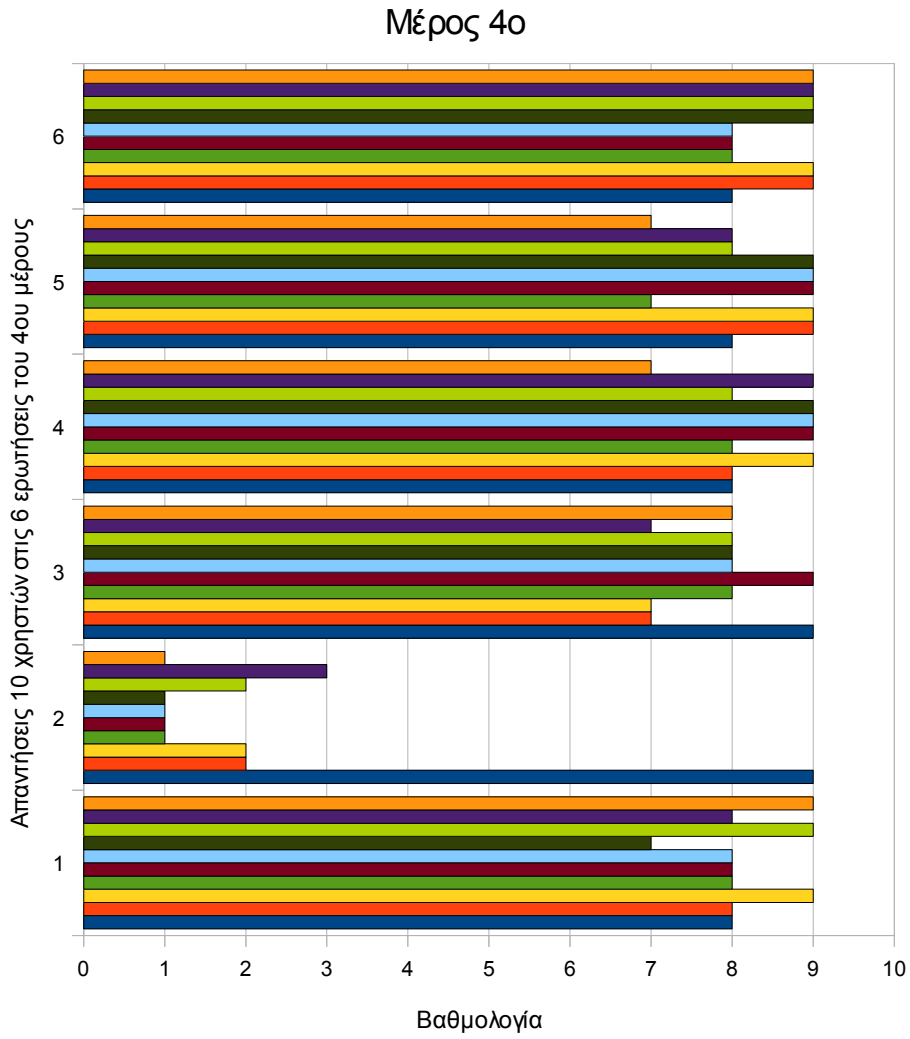


Εικόνα 38: Μέρος τρίτο – Ορολογία και επικοινωνία με το σύστημα

Σχόλια χρηστών:
"Κάπως αργό."
"Τα μηνύματα που εμφανίζονται στην οθόνη είναι πολύ σαφή!"
"Δεν είδα κανένα μήνυμα λάθους"

<u>Μέσοι όροι απαντήσεων στο τρίτο μέρος:</u>
Ερώτηση 1: 8,3
Ερώτηση 2: 8,4
Ερώτηση 3: 8,2
Ερώτηση 4: 8,4
Ερώτηση 5: 7,7
Ερώτηση 6: 6,1
Ερώτηση 7: 9

Μέρος τέταρτο – Εκμάθηση χρήσης:

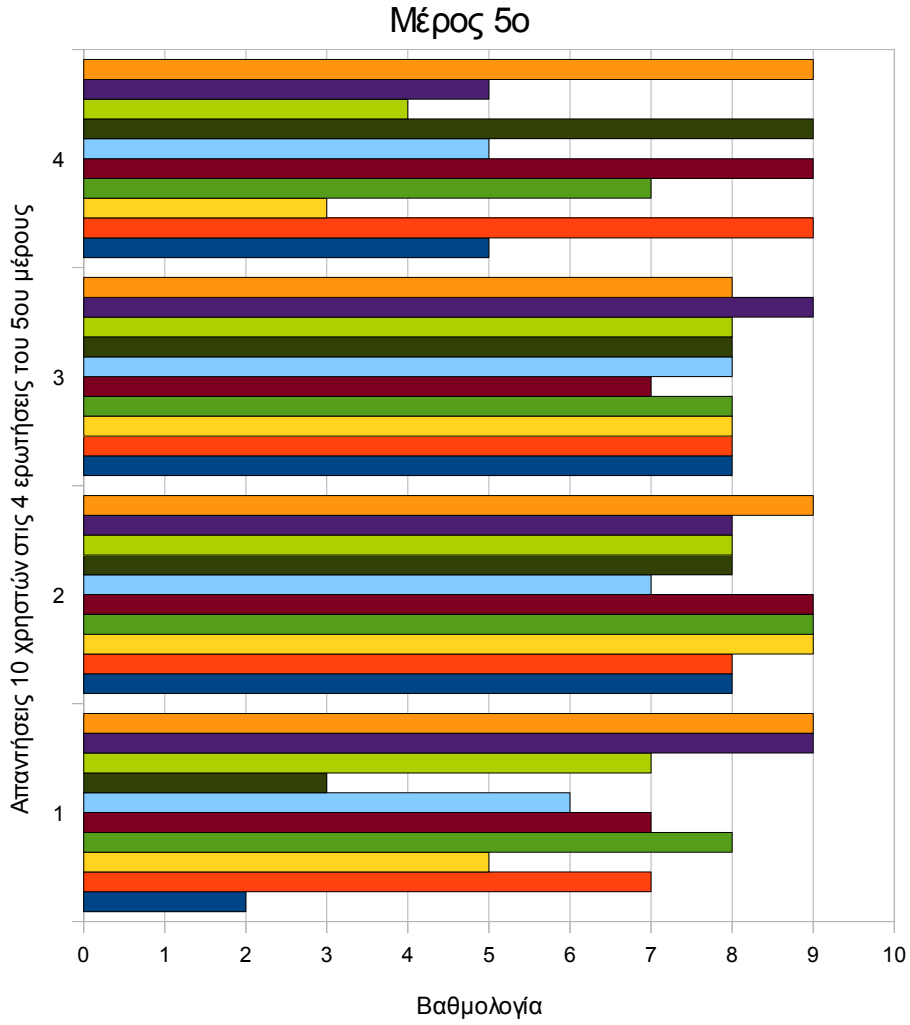


Εικόνα 39: Μέρος τέταρτο – Εκμάθηση χρήσης

Σχόλια χρηστών:
"Η εκμάθηση είναι εύκολη και γρήγορη."

Μέσοι όροι απαντήσεων στο τέταρτο μέρος:
Ερώτηση 1: 8,2
Ερώτηση 2: 2,3
Ερώτηση 3: 7,9
Ερώτηση 4: 8,4
Ερώτηση 5: 8,3
Ερώτηση 6: 8,6

Μέρος πέμπτο – Δυνατότητες του συστήματος:

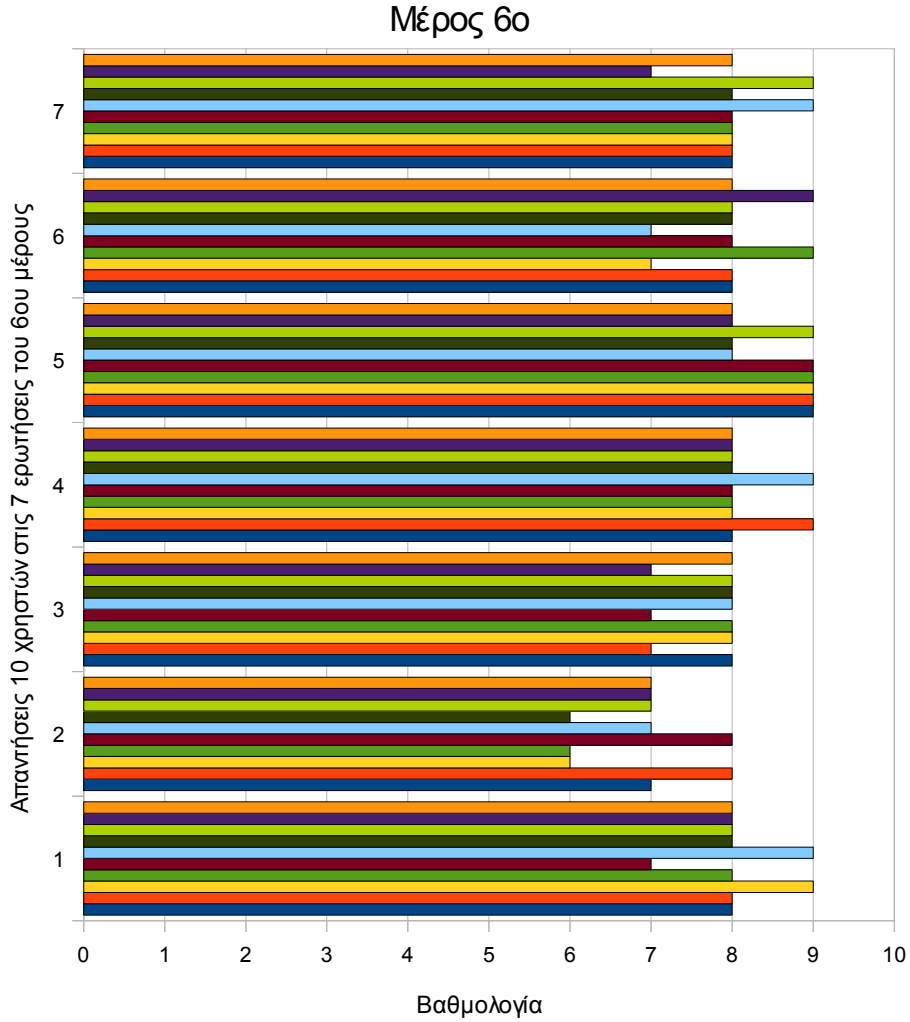


Εικόνα 40: Μέρος πέμπτο – Δυνατότητες του συστήματος

Σχόλια χρηστών:
"Η στόχευση με το όπλο μάλλον εξαρτάται από την εμπειρία του χρήστη."
"Το παιχνίδι έχει αργή ταχύτητα."

Μέσοι όροι απαντήσεων στο πέμπτο μέρος:
Ερώτηση 1: 6,3
Ερώτηση 2: 8,3
Ερώτηση 3: 8
Ερώτηση 4: 6,5

Μέρος έκτο – Πολυμέσα:

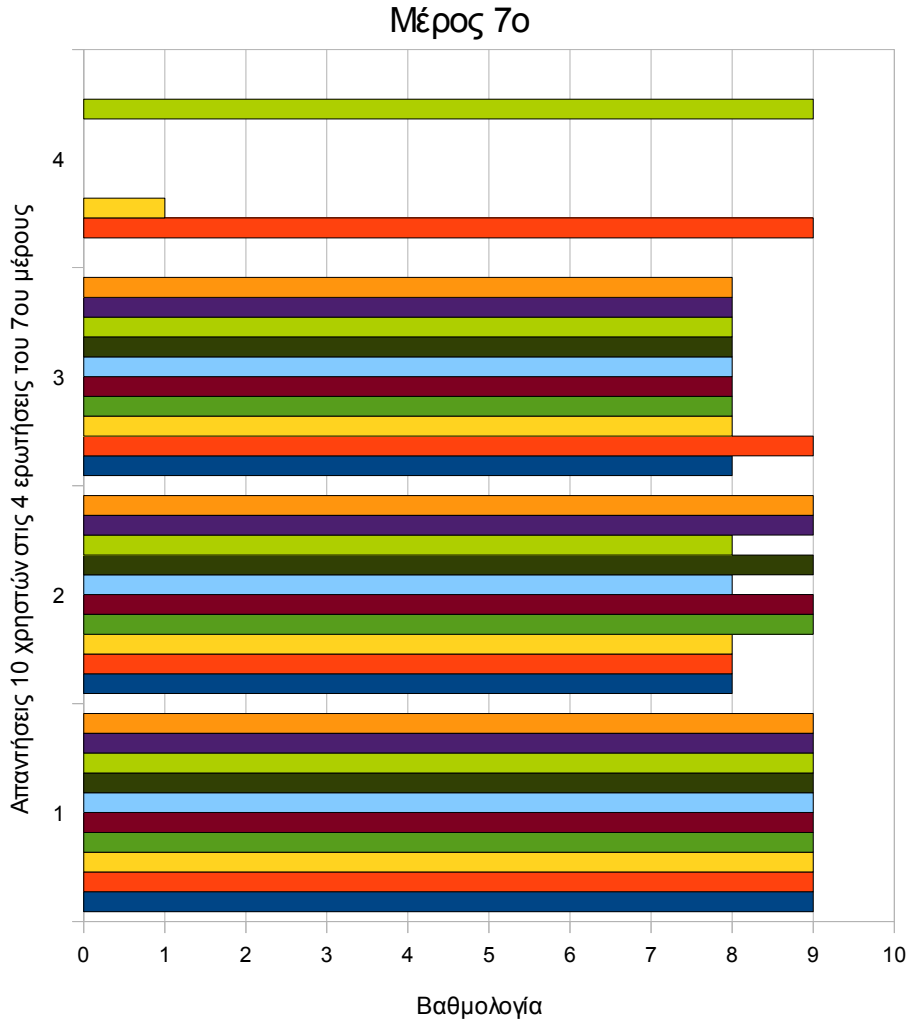


Εικόνα 41: Μέρος έκτο – Πολυμέσα

Σχόλια χρηστών:
"Οι υφές των αντικειμένων θα μπορούσαν να είναι πιο ξεκάθαρες"
"Η ποιότητα των εικόνων είναι καλή."

Μέσοι όροι απαντήσεων στο έκτο μέρος:
Ερώτηση 1: 8,1
Ερώτηση 2: 6,9
Ερώτηση 3: 7,7
Ερώτηση 4: 8,2
Ερώτηση 5: 8,6
Ερώτηση 6: 8
Ερώτηση 7: 8,1

Μέρος έβδομο – Εγκατάσταση του συστήματος:



Εικόνα 42: Μέρος έβδομο – Εγκατάσταση του συστήματος

Σχόλια χρηστών:
"Δεν χρειάζεται εγκατάσταση."
"Είναι άμεσα εκτελέσιμη εφαρμογή."

Μέσοι όροι απαντήσεων στο έβδομο μέρος:
Ερώτηση 1: 9
Ερώτηση 2: 8,5
Ερώτηση 3: 8,1
Ερώτηση 4: 6,33

7.3 Συμπεράσματα

Αναλύουμε, τα συμπεράσματα των ερωτηματολογίων, ανά μέρη και βάσει των μέσων όρων κάθε απάντησης που συλλέξαμε:

Μέρος πρώτο – Γενική εντύπωση του χρήστη:

Οι χρήστες ήταν γενικά ικανοποιημένοι από τη γενική αντίδραση του συστήματος, και γενικότερα τους άφησε ευχάριστες εντυπώσεις. Παρόλα αυτά, από τις απαντήσεις της τέταρτης ερώτησης και τα σχόλια, παρατηρούμε ότι μερικοί χρήστες χαρακτήρισαν το παιχνίδι κάπως δύσκολο.

Μέρος δεύτερο – Οθόνη:

Ο σχεδιασμός της οθόνης άφησε θετικές εντυπώσεις, και η δόμηση των πληροφοριών χαρακτηρίστηκε πολύ οργανωμένη. Το μειονέκτημα που εντόπισαν κάποιοι χρήστες, όσον αφορά την οθόνη, είναι η ποσότητα πληροφορίας, καθώς αρκετοί είχαν την εντύπωση ότι δεν ήταν απόλυτα επαρκής.

Μέρος τρίτο – Ορολογία και επικοινωνία με το σύστημα:

Από τις απαντήσεις των χρηστών συμπεραίνουμε, ότι τα μηνύματα που εμφανίζονται στην οθόνη, έχουν συνέπεια και σαφήνεια. Επίσης υπάρχει πάντα ενημέρωση από το σύστημα και το αποτέλεσμα μιας κίνησης είναι προβλέψιμο. Από την άλλη, οι καθυστερήσεις του συστήματος δεν ήταν παραδεκτές, και σχόλια χρηστών χαρακτηρίζουν την εφαρμογή κάπως αργή.

Μέρος τέταρτο – Εκμάθηση χρήσης:

Στο τέταρτο μέρος οι απαντήσεις των χρηστών καταλήγουν, στο ότι η εκμάθηση του παιχνιδιού είναι πολύ εύκολη, και δεν απαιτεί χρόνο. Κατά τα άλλα, τα βήματα των εργασιών και οι εργασίες έχουν λογική αλληλουχία και σειρά.

Μέρος πέμπτο – Δυνατότητες του συστήματος:

Στο μέρος που αφορούσε τις δυνατότητες του συστήματος, καταγράφουμε ξανά το γεγονός ότι η ταχύτητα του συστήματος είναι αργή, αν και υπήρχαν και κάποιες απαντήσεις που είναι αντίθετες. Αυτό ίσως να οφείλεται σε διαφορετικά συστήματα δοκιμής της εφαρμογής. Κατά τα άλλα, οι χρήστες χαρακτήρισαν το σύστημα πολύ σταθερό και αξιόπιστο. Η ευκολία χειρισμού ήταν ικανοποιητική, όμως κάποιοι χρήστες θεωρούν ότι εξαρτάται από την εμπειρία του χρήστη. Η τελευταία παρατήρηση, ίσως να οφείλεται στις διαφορετικές διακυμάνσεις εμπειρίας των χρηστών, όσον αφορά τα παιχνίδια στόχευσης.

Μέρος έκτο – Πολυμέσα:

Από τις απαντήσεις των χρηστών συμπεραίνουμε, ότι η ποιότητα των εικόνων τους φάνηκε καλή, όμως κάπως θολή. Επίσης η φωτεινότητα, των εικόνων, ήταν ικανοποιητική. Η ηχητική απόδοση επίσης ικανοποίησε τους χρήστες, οι οποίοι την χαρακτήρισαν εύρυθμη και ξεκάθαρη. Τα χρώματα που χρησιμοποιούνται έδωσαν φυσική εντύπωση, και η ποσότητα, τους φάνηκε επαρκής.

Μέρος έβδομο – Εγκατάσταση του συστήματος:

Τέλος στο έβδομο μέρος, όλοι οι χρήστες ήταν ικανοποιημένοι από την ταχύτητα της εγκατάστασης, το οποίο είναι φυσικό καθότι η εφαρμογή μας, δεν απαιτεί εγκατάσταση. Και στις υπόλοιπες ερωτήσεις αυτού του μέρους, οι χρήστες φάνηκαν ικανοποιημένοι, χωρίς να εντοπίσουν καμία ενόχληση.

Κεφάλαιο 8

Αποτελέσματα και μελλοντική εργασία

8.1 Αποτελέσματα

Στόχος αυτής της διπλωματικής εργασίας ήταν η ανάπτυξη ενός παιχνιδιού τριών διαστάσεων για ηλεκτρονικούς υπολογιστές, κάνοντας χρήση μηχανής απόδοσης τρισδιάστατων γραφικών (rendering engine) αλλά και λογισμικού χρήσιμου για ανάπτυξη παιχνιδιών.

Ο στόχος επιτεύχθηκε, και το όνομα αυτού **"The Missions"**! Καταφέραμε να υλοποιήσουμε τρισδιάστατα γραφικά, με οπτικά τεχνάσματα που προσδίδουν ρεαλισμό, εξομοίωση νόμων της φυσικής σε εικονικό περιβάλλον, τεχνητή νοημοσύνη για τους εικονικούς χαρακτήρες, και επεκτασιμότητα, μέσω ενός συστήματος διαχείρισης περιεχομένου των επιπέδων.

Σημαντικότερα επιτεύγματα της εργασίας μας είναι:

Η δημιουργία ενός **ολοκληρωμένου** και ευχάριστου τρισδιάστατου παιχνιδιού. Κάθε αποστολή έχει αρχή, μέση και τέλος. Στο τομέα των γραφικών, υλοποιήσαμε **πολλά σύγχρονα οπτικά εφέ** όπως, σωματιδιακά τεχνάσματα για νερό, καπνό και φωτιά, κινούμενες υφές, για εφέ θάλασσας και άλλα, ημιδιαφανής σκιαστής για εφέ συρματοπλέγματος, δυναμικές σκιές, δυναμικό φωτισμό, εναλλαγή υφών, εφέ πυροβολισμού, κινούμενες μπάλες φωτιάς που ακολουθούν αντικείμενα, morph animation για τους χαρακτήρες, γραφικές πινακίδες και τέλος διαχωρισμό οθόνης (split screen).

Αναπτύξαμε πολυσύνθετη λογική κόσμου, που περιλαμβάνει χαρακτήρες με **διαφορετικούς ρόλους και τεχνητή νοημοσύνη**. Υπάρχουν χαρακτήρες φιλικοί, εχθρικοί και επίσης χαρακτήρες που απλώς περιφέρονται στον κόσμο. Στον τομέα της τεχνητής νοημοσύνης, υλοποιήσαμε εύρεση **συντομότερης διαδρομής** προς κάποιο σημείο **με αναζήτηση A***, και κατόπιν ακολούθηση αυτού του μονοπατιού. Προσθέσαμε κίνηση και αλληλεπίδραση **περαστικών και αυτοκινήτων**.

Το **σενάριο και η πλοκή**, γίνονται εύκολα αντιληπτά και κατανοητά στο χρήστη μέσω της σελίδας οδηγιών που μπορεί να ενεργοποιήσει ο χρήστης κατά τη διάρκεια του παιχνιδιού, και επίσης υποστηρίζονται από τους σύντομους διαλόγους. Τα **αποτελέσματα των επιπέδων καταγράφονται** και οι χρήστες μπορούν να συγκρίνουν το χρόνο που χρειάστηκε στον καθένα, για την ολοκλήρωση της ίδιας αποστολής.

Από τα αποτελέσματα των ερωτηματολογίων, παρατηρήσαμε ότι οι χρήστες ήταν ιδιαίτερα ικανοποιημένοι, από την **οργάνωση και το σχεδιασμό** της οθόνης και την **αλληλουχία** των οθονών. Επίσης ευχαριστημένοι δήλωσαν για **τα μηνύματα που εμφανίζονται** στην οθόνη, πράγμα στο οποίο δόθηκε ιδιαίτερο βάρος στην υλοποίηση, όπως και από την ευκολία και την ταχύτητα, εκμάθησης του συστήματος.

Η εφαρμογή εκτελέσιμη **στις περισσότερες εκδόσεις Microsoft Windows, και εύκολα μεταφέρσιμη** και σε άλλα λειτουργικά συστήματα καθώς το λογισμικό που χρησιμοποιήσαμε είναι cross-platform. **Κανένα ελάττωμα (bug)** ή δυσλειτουργία, δεν παρουσιάστηκε στις δοκιμές, παρότι χρησιμοποιήσαμε αρκετές διαφορετικές ενότητες λογισμικού, και η υλοποίηση τελικά απαίτησε χιλιάδες γραμμές κώδικα. Επίσης οι χρήστες απάντησαν στα ερωτηματολόγια ότι το σύστημα

είναι 'πάντα' **σταθερό**. Εξασφαλίσαμε σημαντική **επεκτασιμότητα ως προς τα επίπεδα**, με το σύστημα διαχείρισης περιεχομένου επιπέδων. Μπορούν εύκολα να προστεθούν επίπεδα, χωρίς καμία αλλαγή στον πηγαίο κώδικα του παιχνιδιού. Τέλος, θετικό στοιχείο αποτελεί ότι η εφαρμογή μας **δεν απαιτεί εγκατάσταση**.

Όσον αφορά τη μηχανή απόδοσης Irrlicht, θα κάνουμε τις παρακάτω παρατηρήσεις, οι οποίες προέκυψαν, από την αξιολόγηση και τις δοκιμές του συστήματος:

Η "**Irrlicht**", προσφέρει ένα ολοκληρωμένο, αξιόπιστο και μεταφέρσιμο σύστημα απεικόνισης γραφικών. Επίσης είναι κατάλληλη για εφαρμογές πέρα του τομέα της ψυχαγωγίας. Προσφέρει έλεγχο χαμηλού επιπέδου, στον δημιουργό. Από την άλλη όμως, μια μηχανή παιχνιδιών, παρέχει περισσότερες έτοιμες λειτουργίες για παιχνίδια, πχ. "Αυτόματη κίνηση κάμερας", "σύστημα πυροβολισμών", περισσότερα οπτικά τεχνάσματα. Παρέχει αρκετά καλή ταχύτητα απόδοσης γραφικών, όμως το σύστημα φυσικής που περιλαμβάνει δημιουργεί καθυστερήσεις. Οπότε είναι προτιμότερο, να ενσωματωθεί ξεχωριστή ενότητα φυσικής. Θετικό στοιχείο αποτελεί, η δομή της. Τα στοιχεία της είναι διακριτά και συνοδεύονται από λεπτομερή τεκμηρίωση. Πραγματοποιείται συνεχής ανάπτυξη και βελτίωση της μηχανής. Όταν ξεκινήσαμε την εργασία, ήταν στην έκδοση 1.4, η εργασία υλοποιήθηκε με την 1.5.1, ενώ πρόσφατα εκδόθηκε η έκδοση 1.6.1. Η κοινότητα της είναι ιδιαίτερα ενεργή και παραγωγική.

8.2 Μελλοντικές επεκτάσεις

Η εφαρμογή που δημιουργήσαμε είναι ολοκληρωμένη και αυτόνομη. Παρόλα αυτά, με τη μέθοδο των ερωτηματολογίων, εντοπίσαμε κάποιες μελλοντικές βελτιώσεις που μπορούν να γίνουν:

Η δυσκολία του παιχνιδιού μπορεί να γίνει πιο εύκολη. Θα βοηθούσε, εάν τοποθετούσαμε στους χάρτες περισσότερα αντικείμενα πανοπλιών, ή εάν δημιουργούσαμε αντικείμενα που δίνουν πόντους ζωής στους παίχτες.

Αρκετοί χρήστες δεν χαρακτήρισαν την ποσότητα πληροφορίας στην οθόνη επαρκή. Θα μπορούσαμε να τοποθετήσουμε, επιπλέον μόνιμες ενδείξεις στην οθόνη κατά τη διάρκεια του παιχνιδιού, όπως κατεύθυνση του στόχου.

Σημαντική βελτίωση θα αποτελούσε η εξάλειψη των καθυστερήσεων που παρουσιάζει το παιχνίδι, οι οποίες δυσαρέστησαν τους χρήστες. Για μείωση των καθυστερήσεων και αύξηση των καρέ ανά δευτερόλεπτο, πρέπει να δημιουργηθούν βελτιστοποιημένες σκηνές, με πολύ λιγότερα πολύγωνα. Επίσης κάποιες εικόνες χαρακτηρίστηκαν θολές, άρα χρησιμοποιώντας υφές μεγαλύτερης ευκρίνειας ή μεγέθους, θα εξαλειφθεί αυτό το γεγονός.

Επίσης υπάρχουν βελτιώσεις που μπορούν να γίνουν, τις οποίες παρατηρήσαμε κατά την χρήση και δημιουργία της εφαρμογής εμείς. Σημαντική βελτίωση θα αποτελούσε, η προσθήκη πιο σύγχρονης μορφής χαρακτήρων και σκελετικής κίνησης. Οι τύποι πλέγματος md2 είναι πια ξεπερασμένοι, και η σκελετική κίνηση είναι πιο ρεαλιστική. Θα μπορούσαν να ενσωματωθούν ολοκληρωμένη και συνεχιζόμενη ιστορία και επίπεδα. Επίσης μπορούν να πραγματοποιηθούν βελτιώσεις στον τομέα της τεχνητής νοημοσύνης όλων των χαρακτήρων. Η ανάπτυξη συστήματος παιχνιδιού, πολλών χρηστών (multiplayer) είναι μια προσθήκη που βελτιώνει την εμπειρία παιχνιδιού που προσφέρεται στον χρήστη. Η ενσωμάτωση συστήματος φυσικής με περισσότερες δυνατότητες και μικρότερο κόστος στην απόδοση. Αντικατάσταση με φωνητικά, τω γραπτών διαλόγων, και επίσης καλύτερη σκηνοθεσία και φωτογραφία, θα προσέθετε ρεαλισμό.

Επίσης υπάρχει δυνατότητα τροποποίησης της μηχανής Irrlicht και δημιουργία ολοκληρωμένης μηχανής παιχνιδιών συγκεκριμένου τύπου. Τέλος, καλύτερα γραφικά αποτελέσματα μπορούν να επιτευχθούν, σε συνεργασία με ομάδα καλών τεχνών. Η τρισδιάστατη μοντελοποίηση αποτελεί ξεχωριστή επιστήμη, που αλλάζει δραματικά το αισθητικό αποτέλεσμα.

8.3 Επίλογος

Αυτή η διπλωματική εργασία ήταν από την αρχή, για εμένα, μια πρόκληση. Τολμώ να πω ότι στο παρελθόν είχα ξαναπροσπαθήσει να κάνω κάτι παρόμοιο, όμως δίχως επιτυχία. Η δημιουργία ενός παιχνιδιού τριών διαστάσεων αποτελούσε για εμένα παιδικό και εφηβικό όνειρο που τώρα πραγματοποιήθηκε. Η διαδικασία ήταν επίπονη και χρονοβόρα, όμως η πορεία και το αποτέλεσμα συναρπαστικά.

Παράλληλα είναι πολύ σημαντικό ότι η ανάπτυξη μιας τόσο πολύπλευρης διαδραστικής εφαρμογής απαίτησε μεγάλο μέρος των γνώσεων που απέκτησα από τα χρόνια σπουδών μου και τελικά ο συνδυασμός διαφορετικών γνωστικών τομέων οδήγησε στην ανάπτυξη μιας ευχάριστης και ολοκληρωμένης εφαρμογής. Επίσης, μου δόθηκε η ευκαιρία να μάθω πολλά και σε διαφορετικούς τομείς όπως η παραγωγή γραφικών, οπτικών τεχνασμάτων και τρισδιάστατων σκηνών, η λογική παιχνιδιών και πολλά άλλα.

Η βιομηχανία παραγωγής διαδραστικών εφαρμογών ψυχαγωγίας, είναι μια από τις πιο κερδοφόρες και αναπτυσσόμενες. Η επιστημονική κοινότητα διαρκώς ανακαλύπτει νέους τρόπους για πιο ρεαλιστική αναπαράσταση εικονικών περιβαλλόντων και βελτιωμένη εμπειρία χρήστη. Είναι πολύ πιθανό, κατά την άποψη μου, και στη χώρα μας η ανάπτυξη παιχνιδιών να ανθίσει τα επόμενα χρόνια και να απασχολήσει εντονότερα τους επιστήμονες πληροφορικής και τους μηχανικούς υπολογιστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

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

[2] Bob Gates. “Game Design” Second Edition σελίδα 250. 2004

[3] “Wikipedia, the free encyclopedia” http://en.wikipedia.org/wiki

[4] Irrlicht engine http://irrlicht.sourceforge.net/

[5] Julian Gold. “Object Oriented Game Development” σελίδα 135. 2004

[6] Irrlicht community forum http://irrlicht.sourceforge.net/phpBB2/index.php

[7] Crystal space http://www.crystalspace3d.org/

[8] Ogre 3D rendering engine http://www.ogre3d.org/

[9] Ιστοσελίδα Κώστα Αναγνώστου, Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο http://videogamesbook.wordpress.com/

[10] Διαφάνειες Κώστα Αναγνώστου, Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο

[11] Polycount, τρισδιάστατα πλέγματα http://www.polycount.com/models/quake2/

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

[13] Questionnaire For User Interaction Satisfaction, http://lap.umd.edu/quis/

[14] Διαφάνειες Τεχνητής Νοημοσύνης, Μ.Γ. Λαγουδάκης, Πολυτεχνείο Κρήτης

[15] Fuchs, H., Bishop, et al. (1992)

[16] Διαφάνειες Ανάλυση Συστημάτων, Αφροδίτη Τσαλγατίδου , Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

[17] Pro OGRE 3D Programming, Gregory Junker, Apress

[18] Τεχνητή νοημοσύνη, Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελαρίου